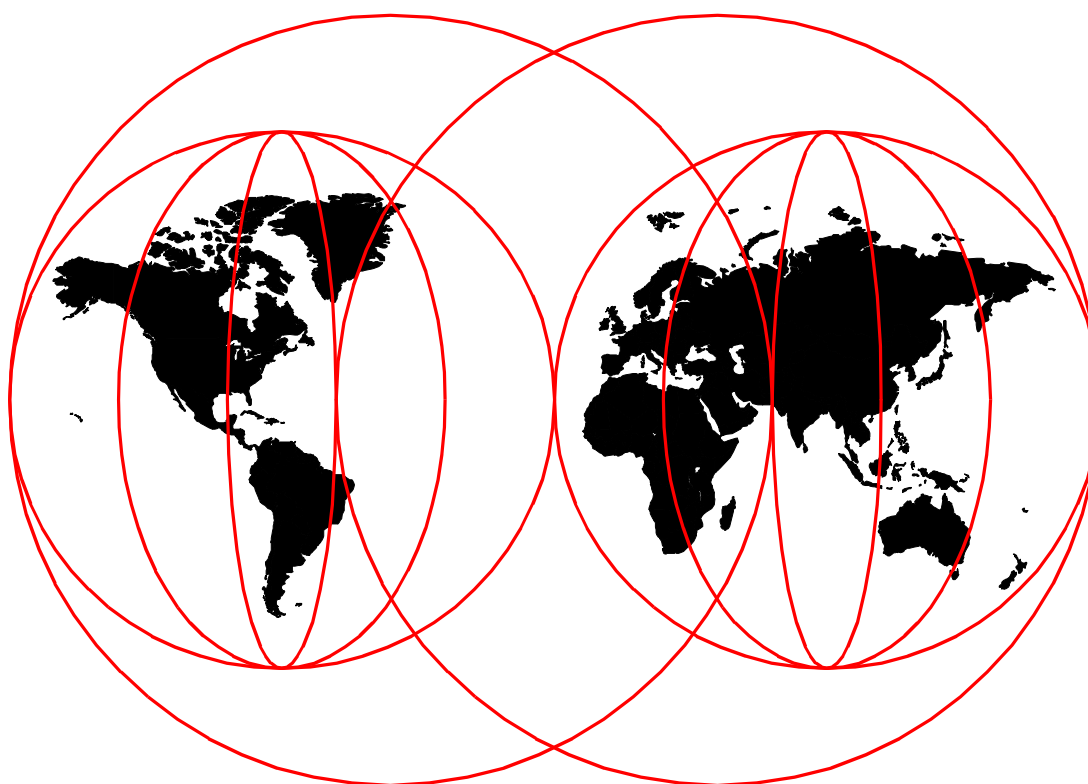


# Getting Started with TCP/IP for VSE/ESA 1.4

*Annegret Ackel, John Lawson, Charles J. McMurry*



**International Technical Support Organization**

[www.redbooks.ibm.com](http://www.redbooks.ibm.com)





International Technical Support Organization

SG24-5626-00

## **Getting Started with TCP/IP for VSE/ESA 1.4**

May 2000

**Take note!**

Before using this information and the product it supports, be sure to read the general information in Appendix A, "Special notices" on page 219.

**First Edition (May 2000)**

This edition applies to IBM TCP/IP for VSE/ESA Version 1 Release 4, Programm Number 5686-A04.

Comments may be addressed to:  
IBM Corporation, International Technical Support Organization  
Dept. HYJ Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

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

---

# Contents

|   |      |
|---|------|
| <b>Figures</b> .....  | vii  |
| <b>Tables</b> .....   | xiii |
| <b>Preface</b> .....  | xv   |
| The team that wrote this redbook .....  | xv   |
| Comments welcome .....  | xvi  |
| <b>Chapter 1. Introduction to TCP/IP</b> .....                                      | 1    |
| 1.1 Why TCP/IP .....  | 1    |
| 1.2 The growth of TCP/IP .....  | 1    |
| 1.3 Internet standards and Request for Comments (RFC) .....                         | 2    |
| 1.4 TCP/IP architecture .....   | 3    |
| 1.5 The TCP/IP network interface layer .....  | 5    |
| 1.5.1 Network interface .....   | 5    |
| 1.5.2 Interfacing with the network layer .....                                      | 6    |
| 1.6 TCP/IP internetwork layer protocols .....                                       | 8    |
| 1.6.1 Internet Protocol (IP) .....  | 8    |
| 1.6.2 Internet Control Message Protocol (ICMP) .....                                | 14   |
| 1.7 TCP/IP connectivity bridging, switching, and routing .....                      | 14   |
| 1.7.1 Local area network (LAN) .....  | 15   |
| 1.7.2 Wide area network (WAN) .....   | 17   |
| 1.7.3 Bridging and switching .....  | 18   |
| 1.7.4 Routing .....   | 21   |
| 1.8 TCP/IP transport layer protocols and interfaces .....                           | 24   |
| 1.8.1 Ports and sockets .....   | 24   |
| 1.8.2 The socket application programming interface .....                            | 25   |
| 1.8.3 User Datagram Protocol (UDP) .....  | 25   |
| 1.8.4 Transmission Control Protocol (TCP) .....                                     | 26   |
| 1.9 TCP/IP application protocols .....  | 27   |
| 1.9.1 Remote login and terminal emulation: Telnet .....                             | 27   |
| 1.9.2 File Transfer Protocol: FTP .....   | 28   |
| 1.9.3 Remote printing: LPR and LPD .....  | 28   |
| 1.9.4 Domain Name System (DNS) .....  | 28   |
| 1.9.5 Network File System (NFS) .....   | 31   |
| 1.10 Internet user applications and protocols .....                                 | 31   |
| 1.10.1 Gopher .....   | 31   |
| 1.10.2 The World Wide Web (WWW) .....   | 31   |
| 1.10.3 Hypertext Transfer Protocol (HTTP) .....                                     | 32   |
| 1.10.4 The advent of Java .....   | 33   |
| 1.11 TCP/IP and Internet security .....   | 34   |
| 1.11.1 Secure Sockets Layer (SSL) .....   | 35   |
| 1.11.2 Firewalls .....  | 36   |
| 1.12 TCP/IP and Internet publications .....   | 37   |
| <b>Chapter 2. Installation and implementation of TCP/IP for VSE/ESA</b> .....       | 39   |
| 2.1 Obtain TCP/IP for VSE/ESA 1.4 documentation .....                               | 39   |
| 2.2 Obtain and install TCP/IP for VSE/ESA Version 1.4 .....                         | 40   |
| 2.3 Create VSE library for TCP/IP for VSE/ESA installation-unique definitions ..... | 40   |
| 2.4 Obtain TCP/IP product keys from IBM and define to TCP/IP for VSE/ESA .....      | 42   |
| 2.5 Determine the TCP/IP for VSE/ESA applications to be used .....                  | 43   |

|                   |   |            |
|-------------------|---|------------|
| 2.6               | Configure the hardware the VSE system is running on . . . . .             | 43         |
| 2.7               | Configure the network to which the VSE system will be connected . . . . . | 44         |
| 2.8               | Define the hardware and network environment to the S/390 hosts. . . . .   | 44         |
| 2.8.1             | Hardware and network configuration used in this project . . . . .         | 44         |
| 2.8.2             | Hardware and software details. . . . .                                    | 44         |
| 2.8.3             | VM/ESA definitions . . . . .  | 46         |
| 2.8.4             | TCP/IP for VM/ESA guest machine (TCPIP) definitions . . . . .             | 46         |
| 2.8.5             | VSE/ESA 2.4 guest machine (VSE240) definitions. . . . .                   | 48         |
| 2.8.6             | VSE/ESA 2.3.1 guest machine (VSE231) definitions . . . . .                | 74         |
| 2.9               | Install and/or customize TCP/IP applications on other systems . . . . .   | 75         |
| 2.9.1             | Windows 98. . . . .   | 75         |
| 2.9.2             | OS/2 . . . . .  | 76         |
| 2.10              | Verifying system connectivity. . . . .                                    | 76         |
| 2.10.1            | The PING command . . . . .  | 76         |
| 2.10.2            | The TRACERT (Traceroute) command in Windows 98. . . . .                   | 78         |
| 2.11              | Operating TCP/IP for VSE/ESA . . . . .                                    | 79         |
| 2.11.1            | Command entry through the VSE console . . . . .                           | 79         |
| 2.11.2            | Command entry using the batch utility program IPNETCMD . . . . .          | 80         |
| 2.11.3            | Shutting down TCP/IP for VSE/ESA. . . . .                                 | 81         |
| 2.11.4            | Restarting TCP/IP for VSE/ESA following an abnormal termination . . . . . | 81         |
| 2.11.5            | Limiting TCP/IP for VSE/ESA messages on the console . . . . .             | 82         |
| <b>Chapter 3.</b> | <b>Using FTP . . . . .</b>  | <b>83</b>  |
| 3.1               | Definitions for FTP. . . . .  | 83         |
| 3.1.1             | Define FTP daemons . . . . .  | 83         |
| 3.1.2             | Define security for FTP . . . . .   | 84         |
| 3.1.3             | Defining the VSE file system . . . . .                                    | 85         |
| 3.1.4             | Define automatic FTP support . . . . .                                    | 86         |
| 3.1.5             | Define CICS transaction FTP. . . . .                                      | 86         |
| 3.2               | Configuring non-VSE remote hosts . . . . .                                | 87         |
| 3.2.1             | Configuring Windows FTP clients and daemons . . . . .                     | 87         |
| 3.2.2             | Configuring OS/2 FTP clients and daemons. . . . .                         | 92         |
| 3.3               | Referencing files in the VSE file system . . . . .                        | 94         |
| 3.4               | Data translation during file transfer . . . . .                           | 95         |
| 3.5               | VSE as an FTP server. . . . .   | 98         |
| 3.5.1             | FTP with VSE libraries. . . . .   | 98         |
| 3.5.2             | FTP with POWER queues . . . . .   | 101        |
| 3.5.3             | FTP from the ICCF library . . . . .                                       | 104        |
| 3.5.4             | FTP with VSAM files . . . . .   | 104        |
| 3.5.5             | FTP using the VSAM catalog . . . . .                                      | 108        |
| 3.6               | Using a graphical client on the workstation . . . . .                     | 108        |
| 3.7               | VSE as an FTP client . . . . .  | 111        |
| 3.7.1             | CICS FTP client transaction. . . . .                                      | 113        |
| 3.7.2             | The VSE batch FTP clients . . . . .                                       | 120        |
| 3.7.3             | Managing FTP daemons . . . . .  | 130        |
| <b>Chapter 4.</b> | <b>Using Telnet. . . . .</b>  | <b>133</b> |
| 4.1               | Telnet implementation . . . . .   | 133        |
| 4.1.1             | Telnet daemon definitions . . . . .                                       | 133        |
| 4.1.2             | VTAM APPL definitions . . . . .   | 134        |
| 4.1.3             | Telnet menu definition . . . . .  | 135        |
| 4.1.4             | CICS Telnet client definition. . . . .                                    | 137        |
| 4.1.5             | Configuring a TN3270 client. . . . .                                      | 138        |

|   |            |
|---|------------|
| 4.2 Using a TN3270 client . . . . .                                 | 139        |
| 4.3 Using the Telnet CICS client . . . . .                          | 141        |
| 4.3.1 Connecting in 3270 mode . . . . .                             | 141        |
| 4.3.2 Connecting in line mode to OS/2 . . . . .                     | 142        |
| 4.4 Using the Telnet batch client . . . . .                         | 144        |
| 4.5 Managing Telnet daemons . . . . .                               | 146        |
| 4.5.1 QUERY TELNETDS command . . . . .                              | 146        |
| 4.5.2 DELETE TELNETD command . . . . .                              | 147        |
| <b>Chapter 5. Network printing using LPD/LPR . . . . .</b>          | <b>149</b> |
| 5.1 Defining the Line Printer Daemon on VSE . . . . .               | 149        |
| 5.1.1 LPD operation commands . . . . .                              | 151        |
| 5.2 Access the VSE Line Printer Daemon from a workstation . . . . . | 152        |
| 5.3 Using the Line Printer Requester on VSE . . . . .               | 154        |
| 5.3.1 Using the CICS transaction LPR . . . . .                      | 154        |
| 5.4 The automatic LPR client . . . . .                              | 160        |
| 5.4.1 Defining events . . . . .                                     | 160        |
| 5.4.2 Using the automatic LPR client . . . . .                      | 161        |
| 5.4.3 Using scripts for the automatic LPR client . . . . .          | 162        |
| 5.5 Using the INSERTS phase to control printers . . . . .           | 163        |
| 5.5.1 Creating the INSERTS phase . . . . .                          | 164        |
| 5.5.2 Invoking the INSERTS phase . . . . .                          | 166        |
| 5.6 Using the batch LPR client . . . . .                            | 166        |
| 5.6.1 Return codes of batch LPR . . . . .                           | 167        |
| 5.7 Diagnosing LPR problems . . . . .                               | 168        |
| <b>Chapter 6. 3270 printing using GPS . . . . .</b>                 | <b>171</b> |
| 6.1 Installing GPS . . . . .  | 171        |
| 6.2 GPS definitions . . . . .                                       | 171        |
| 6.2.1 VTAM definitions for GPS . . . . .                            | 171        |
| 6.2.2 Defining GPS 3270 printers to CICS . . . . .                  | 172        |
| 6.2.3 Defining the GPS daemon on VSE . . . . .                      | 175        |
| 6.3 Operating GPS . . . . .   | 177        |
| 6.3.1 The DELETE GPSD command . . . . .                             | 178        |
| 6.3.2 The QUERY GPSD command . . . . .                              | 178        |
| 6.3.3 Testing GPS setup . . . . .                                   | 179        |
| <b>Chapter 7. Accessing VSE files using NFS . . . . .</b>           | <b>181</b> |
| 7.1 Installing NFS . . . . .  | 181        |
| 7.2 Configuring NFS . . . . .                                       | 182        |
| 7.2.1 Define NFS mount points . . . . .                             | 182        |
| 7.2.2 NFS configuration files . . . . .                             | 183        |
| 7.3 Operating NFS . . . . .   | 188        |
| 7.3.1 Starting NFS . . . . .  | 188        |
| 7.3.2 Terminating NFS . . . . .                                     | 188        |
| 7.3.3 Query and control commands . . . . .                          | 189        |
| 7.4 Installing and using an NFS client . . . . .                    | 191        |
| 7.4.1 Configuring the NFS client . . . . .                          | 191        |
| 7.4.2 Using the NFS client . . . . .                                | 195        |
| 7.4.3 Other NFS considerations . . . . .                            | 202        |
| <b>Chapter 8. Using VSE as a Web server . . . . .</b>               | <b>203</b> |
| 8.1 Defining the HTTP daemon . . . . .                              | 204        |
| 8.1.1 Managing HTTPD definitions . . . . .                          | 205        |

|  |            |
|--|------------|
| 8.2 Creating documents for your Web site . . . . .   | 206        |
| 8.2.1 HTTP file types . . . . .                      | 206        |
| 8.2.2 Documents on our Web site . . . . .            | 207        |
| 8.3 Creating a secured Web site . . . . .            | 209        |
| 8.3.1 Implementing a secured HTTP daemon . . . . .   | 209        |
| 8.3.2 Logging on to the secured Web server . . . . . | 210        |
| 8.4 Processing data on a VSE Web site . . . . .      | 212        |
| 8.4.1 Creating forms . . . . .                       | 212        |
| 8.4.2 Using CGI programs . . . . .                   | 214        |
| 8.4.3 Our sample REXX CGI program . . . . .          | 216        |
| <b>Appendix A. Special notices . . . . .</b>         | <b>219</b> |
| <b>Appendix B. Related publications . . . . .</b>    | <b>221</b> |
| B.1 IBM Redbooks . . . . .                           | 221        |
| B.2 IBM Redbooks collection . . . . .                | 221        |
| B.3 Other resources . . . . .                        | 221        |
| B.4 Referenced Web sites . . . . .                   | 222        |
| <b>How to get IBM Redbooks . . . . .</b>             | <b>223</b> |
| IBM Redbooks fax order form . . . . .                | 224        |
| <b>Abbreviations and acronyms . . . . .</b>          | <b>225</b> |
| <b>Index . . . . .</b>                               | <b>229</b> |
| <b>IBM Redbooks review . . . . .</b>                 | <b>235</b> |



---

## Figures

|   |    |
|---|----|
| 1. TCP/IP architecture model - layers and protocols . . . . .                     | 4  |
| 2. IP - assigned classes of IP addresses . . . . .                                | 9  |
| 3. IP - class A address without subnets . . . . .                                 | 10 |
| 4. IP - class A address with subnet address . . . . .                             | 11 |
| 5. IP - format of an IP datagram header . . . . .                                 | 13 |
| 6. Internetwork connectivity . . . . .  | 15 |
| 7. Three most commonly used LAN implementations . . . . .                         | 16 |
| 8. WAN point-to-point link . . . . .  | 17 |
| 9. LAN local and remote bridges . . . . .   | 19 |
| 10. ATM switch with multiple LAN connections . . . . .                            | 20 |
| 11. LAN switch Ethernet . . . . .   | 21 |
| 12. Routers packet transfer . . . . .   | 22 |
| 13. UDP - demultiplexing based on ports . . . . .                                 | 26 |
| 14. TCP - connection between processes . . . . .                                  | 26 |
| 15. Hierarchical namespace (chain of authority in assigning domain names) . . . . | 29 |
| 16. DNS - resolver and domain name server . . . . .                               | 30 |
| 17. Define VSAM catalog and space for library TCPIP . . . . .                     | 41 |
| 18. Define the TCPIP library and sublibraries . . . . .                           | 42 |
| 19. Installing the product key and customer identification . . . . .              | 43 |
| 20. Our project network . . . . .   | 44 |
| 21. Software layout on the IBM Integrated Server . . . . .                        | 45 |
| 22. VM/ESA directory entry for TCPIP guest machine . . . . .                      | 47 |
| 23. SYS1 TCPIP entries for TCP/IP for VM . . . . .                                | 47 |
| 24. SYSTEM DTCPARMS entries for TCP/IP for VM . . . . .                           | 48 |
| 25. VM/ESA directory entry for VSE240 guest machine . . . . .                     | 48 |
| 26. ASI master procedure \$ASIPROC.PROC . . . . .                                 | 49 |
| 27. Additions to the VSE240 IPL proc \$IPLESA4.PROC . . . . .                     | 49 |
| 28. Additions to the VSE240 JCL proc \$0JCL4.PROC for TCP/IP for VSE/ESA . . . .  | 50 |
| 29. Additions to USERBG4.PROC for TCP/IP for VSE/ESA . . . . .                    | 50 |
| 30. Startup job for primary TCP/IP partition Z1 . . . . .                         | 51 |
| 31. Startup job for secondary TCP/IP partition F7 . . . . .                       | 52 |
| 32. Standard IPINIT00.L delivered with TCP/IP for VSE/ESA (part 1 of 3) . . . . . | 53 |
| 33. Standard IPINIT00.L delivered with TCP/IP for VSE/ESA (part 2 of 3) . . . . . | 54 |
| 34. Standard IPINIT00.L delivered with TCP/IP for VSE/ESA (part 3 of 3) . . . . . | 55 |
| 35. IPINIT04.L as used in our environment (part 1 of 5) . . . . .                 | 57 |
| 36. IPINIT04.L as used in our environment (part 2 of 5) . . . . .                 | 58 |
| 37. IPINIT04.L as used in our environment (part 3 of 5) . . . . .                 | 59 |
| 38. IPINIT04.L as used in our environment (part 4 of 5) . . . . .                 | 60 |
| 39. IPINIT04.L as used in our environment (part 5 of 5) . . . . .                 | 61 |
| 40. Our NETWORK.L member . . . . .  | 69 |
| 41. IPINIT07.L as used in our environment . . . . .                               | 70 |
| 42. VTAM application definitions for the Telnet daemons . . . . .                 | 71 |
| 43. Member IPNCSDUP.Z, shipped with TCP/IP for VSE/ESA . . . . .                  | 72 |
| 44. Member IPNCSD.Z, shipped with TCP/IP for VSE/ESA . . . . .                    | 73 |
| 45. Job to generate DFHCSD entries for TCP/IP for VSE/ESA on CICS/VSE . . . .     | 74 |
| 46. VM/ESA directory entry for the VSE231 guest machine . . . . .                 | 74 |
| 47. VSE231 IPL procedure entries for TCP/IP for VSE . . . . .                     | 74 |
| 48. VSE231 JCL procedure entries for TCP/IP for VSE . . . . .                     | 75 |
| 49. TCP/IP for VSE entries for the VSE231 guest machine . . . . .                 | 75 |
| 50. HOSTS file used on Windows 98 . . . . .                                       | 76 |

|   |     |
|---|-----|
| 51. Issuing the PING command in Windows 98 . . . . .                            | 77  |
| 52. Issuing the PING command from the VSE console . . . . .                     | 77  |
| 53. CICS PING transaction . . . . .   | 78  |
| 54. Using the TRACERT command in Windows 98 . . . . .                           | 78  |
| 55. Cataloging a command file to be processed by the EXECUTE command . . . .    | 79  |
| 56. Using the EXECUTE command . . . . .   | 80  |
| 57. Job stream for IPNETCMD . . . . .   | 80  |
| 58. Using the IPNETCMD utility program . . . . .                                | 81  |
| 59. FTP daemon definition . . . . .   | 83  |
| 60. FTP security parameters. . . . .  | 84  |
| 61. FTP file system definitions . . . . .                                       | 85  |
| 62. DLBLs for each file defined in the file system . . . . .                    | 86  |
| 63. FTP DEFINE EVENT specification . . . . .                                    | 86  |
| 64. List of Windows 98 FTP commands . . . . .                                   | 87  |
| 65. Workstation batch file for FTP. . . . .                                     | 87  |
| 66. Workstation script for an FTP batch file . . . . .                          | 88  |
| 67. BlueZone FTP initial window. . . . .  | 88  |
| 68. Adding or updating an FTP configuration in BlueZone FTP . . . . .           | 89  |
| 69. VSE240 configuration in BlueZone FTP . . . . .                              | 89  |
| 70. Using the BlueZone FTP graphical client . . . . .                           | 90  |
| 71. SmartFTP-D initial window . . . . .   | 91  |
| 72. SmartFTP-D user definition. . . . .   | 91  |
| 73. SmartFTP-D settings. . . . .  | 92  |
| 74. List of OS/2 FTP commands. . . . .  | 92  |
| 75. Log on to remote host using OS/2 FTP-PM . . . . .                           | 93  |
| 76. OS/2 FTP-PM main window . . . . .   | 93  |
| 77. Project file system. . . . .  | 94  |
| 78. File system under the TCPIP entry . . . . .                                 | 94  |
| 79. File system under the TCPIP.CONFIG entry . . . . .                          | 95  |
| 80. Default entries in the EXTYPES.L table . . . . .                            | 97  |
| 81. Transfer a VSE library member to a remote workstation . . . . .             | 99  |
| 82. FTP explicitly specifying the file position. . . . .                        | 100 |
| 83. Transferring a .PHASE member with FTP . . . . .                             | 100 |
| 84. VSE console output during transfer of the .PHASE member . . . . .           | 101 |
| 85. Transfer a remote workstation file into a VSE library . . . . .             | 101 |
| 86. Transfer a VSE/POWER LST queue entry to our workstation . . . . .           | 102 |
| 87. Transfer a workstation file into the POWER LST queue. . . . .               | 103 |
| 88. Transfer a workstation file to the VSE/POWER RDR queue for execution. . .   | 103 |
| 89. Job submitted to the VSE/POWER RDR queue . . . . .                          | 104 |
| 90. Get VSE/ICCF library member to the workstation. . . . .                     | 104 |
| 91. Transfer a VSAM KSDS file to the workstation . . . . .                      | 105 |
| 92. Transferring a workstation file into a VSAM file . . . . .                  | 105 |
| 93. VSE console output from failed transfer of KSDS. . . . .                    | 106 |
| 94. VSE console output from successful transfer of KSDS. . . . .                | 106 |
| 95. Transfer an ESDS VSAM file to the workstation. . . . .                      | 106 |
| 96. Transfer workstation file to VSAM ESDS using APPEND. . . . .                | 107 |
| 97. VSE console after successful ESDS file transfer using APPEND. . . . .       | 107 |
| 98. Transfer workstation file to VSAM ESDS using PUT . . . . .                  | 107 |
| 99. Transfer a file from a VSAM catalog to the workstation . . . . .            | 108 |
| 100. Initial display from a graphical FTP client. . . . .                       | 109 |
| 101. Graphical client listing of sublibraries in the VSE library TCPIP. . . . . | 109 |
| 102. Graphical client listing of members in VSE sublibrary TCPIP.CONFIG . . .   | 110 |
| 103. Following FTP of member from VSE library using graphical client. . . . .   | 110 |

|   |     |
|---|-----|
| 104.Displaying the transferred member using Notepad . . . . .                   | 111 |
| 105.VSE FTP client HELP output (part 1 of 2) . . . . .                          | 112 |
| 106.VSE FTP client HELP output (part 2 of 2) . . . . .                          | 113 |
| 107.Logging on to a remote host using the VSE FTP client. . . . .               | 114 |
| 108.Transfer a file from a workstation using the CICS FTP client . . . . .      | 115 |
| 109.Transfer a file to a remote workstation using the CICS FTP client . . . . . | 115 |
| 110.Get VSE library member from a remote VSE system to a local VSE . . . . .    | 116 |
| 111.Transfer a local VSE library member to a remote VSE system. . . . .         | 117 |
| 112.Get a remote POWER LST queue entry into a local POWER LST queue . .         | 118 |
| 113.Put a POWER LST queue entry into a remote VSE POWER LST queue. . .          | 119 |
| 114.A typical FTP client batch job . . . . .                                    | 121 |
| 115.A typical FTP batch job with USER and PASS commands. . . . .                | 122 |
| 116.An FTP batch job transferring files among multiple systems. . . . .         | 122 |
| 117.List output of our FTP client batch job FTP3 (part 1 of 2) . . . . .        | 124 |
| 118.List output of our FTP client batch job FTP3 (part 2 of 2) . . . . .        | 125 |
| 119.FTPBATCH job transferring a VSE sequential file to a workstation . . . . .  | 126 |
| 120.FTPBATCH job transferring a workstation file to a VSAM-managed SAM file     | 126 |
| 121.FTPBATCH job transferring a workstation file to a VSE magnetic tape . . .   | 127 |
| 122.Basic script for Automatic FTP . . . . .                                    | 127 |
| 123.Job using Automatic FTP script. . . . .                                     | 128 |
| 124.Console output from processing an Automatic FTP job . . . . .               | 128 |
| 125.Advanced script with variables for Automatic FTP . . . . .                  | 129 |
| 126.Console output from Automatic FTP script using variables. . . . .           | 130 |
| 127.Display status of all FTP daemons . . . . .                                 | 130 |
| 128.Display status of individual FTP daemon. . . . .                            | 131 |
| 129.Delete an FTP daemon . . . . .  | 131 |
| 130.TELNETD definitions in the IPINIT04.L member. . . . .                       | 134 |
| 131.Define VTAM APPLs for the Telnet daemons . . . . .                          | 134 |
| 132.Source for the VTAM1.L menu . . . . .                                       | 136 |
| 133.Our Telnet menu . . . . .   | 137 |
| 134.Initial window from TN3270 Plus . . . . .                                   | 138 |
| 135.Configuring TN3270 Plus. . . . .  | 139 |
| 136.TN3270 client application selection menu . . . . .                          | 140 |
| 137.Connecting to CICSICCF without a menu . . . . .                             | 140 |
| 138.Starting the Telnet CICS client to VM/ESA . . . . .                         | 141 |
| 139.Connecting to VM/ESA in 3270 mode. . . . .                                  | 142 |
| 140.Telnet CICS client connection to OS/2 . . . . .                             | 143 |
| 141.Batch Telnet client job . . . . .   | 144 |
| 142.Telnet batch job output (part 1 of 2). . . . .                              | 145 |
| 143.Telnet batch job output (part 2 of 2). . . . .                              | 146 |
| 144.Display the status of all Telnet daemons. . . . .                           | 147 |
| 145.Display the status of individual Telnet daemon . . . . .                    | 147 |
| 146.Delete a Telnet daemon . . . . .  | 148 |
| 147.Line Printer Daemon definitions in our IPINIT04 file . . . . .              | 150 |
| 148.Example of the QUERY LPDS command . . . . .                                 | 152 |
| 149.After issuing the LPR command . . . . .                                     | 153 |
| 150.Display the POWER list queue after LPR . . . . .                            | 153 |
| 151.Invoking the CICS transaction LPR . . . . .                                 | 155 |
| 152.SET HOST and PRINTER. . . . .   | 155 |
| 153.Navigate to the POWER list queue . . . . .                                  | 156 |
| 154.Print to the remote LPD on VSE system P330VSE. . . . .                      | 156 |
| 155.Get to the TCP/IP for VSE/ESA initialization file . . . . .                 | 157 |
| 156.SET CC off, and QUERY OPT commands . . . . .                                | 157 |

|      |  |     |
|------|--|-----|
| 157. | Print the IPINIT04.L sublibrary member . . . . .                         | 158 |
| 158. | Print a file on our network printer . . . . .                            | 158 |
| 159. | The VSE LPR command LONG . . . . .                                       | 159 |
| 160. | Report of the SHORT command from the LPR client. . . . .                 | 159 |
| 161. | Leave the CICS LPR client transaction . . . . .                          | 160 |
| 162. | DEFINE EVENT and SCRIPT commands in initialization file IPINIT04.L . . . | 160 |
| 163. | The QUERY EVENTS command . . . . .                                       | 161 |
| 164. | Syntax of the POWER LST statement . . . . .                              | 161 |
| 165. | Job using the automatic LPR client . . . . .                             | 162 |
| 166. | VSE console output for the automatic LPR event . . . . .                 | 162 |
| 167. | Job stream to catalog our script file LPRSCR01.L . . . . .               | 163 |
| 168. | Job using the automatic LPR client with a script file. . . . .           | 163 |
| 169. | INSERTS macro format . . . . .   | 164 |
| 170. | Example of INSERTS phase HP4LAND. . . . .                                | 164 |
| 171. | Job stream to create INSERTS phase GPS1LEGL . . . . .                    | 165 |
| 172. | Job stream to create INSERTS phase GPS1LETR . . . . .                    | 165 |
| 173. | Our VSE batch job to invoke the LPR . . . . .                            | 166 |
| 174. | Check the batch job entry in the POWER list queue . . . . .              | 167 |
| 175. | VSE console message for unsuccessful LPR . . . . .                       | 168 |
| 176. | DIAGNOSE LPR command with console output . . . . .                       | 169 |
| 177. | VTAM APPL definitions for GPS . . . . .                                  | 172 |
| 178. | CICS RDO TERMINAL definition for the GPS printer . . . . .               | 173 |
| 179. | CICS RDO TYPETERM definition for the GPS printer (part 1 of 2) . . . . . | 174 |
| 180. | CICS RDO TYPETERM definition for the GPS printer (part 2 of 2) . . . . . | 175 |
| 181. | TCP/IP GPS daemon definitions . . . . .                                  | 176 |
| 182. | DEFINE GPS command file for the EXECUTE command . . . . .                | 177 |
| 183. | Use of the EXECUTE command to define the GPS daemon. . . . .             | 178 |
| 184. | DELETE GPS daemon command . . . . .                                      | 178 |
| 185. | QUERY GPSD command. . . . .  | 179 |
| 186. | VSE console messages for a GPS retry of an LPR request. . . . .          | 180 |
| 187. | DEFINE FILE and DEFINE NFSD entries . . . . .                            | 183 |
| 188. | NFSCFG01.L configuration file. . . . .                                   | 184 |
| 189. | NFSTYPES.L configuration file (part 1 of 2). . . . .                     | 185 |
| 190. | NFSTYPES.L configuration file (part 2 of 2). . . . .                     | 186 |
| 191. | NFSMODEL.L configuration file . . . . .                                  | 187 |
| 192. | NFS startup console messages . . . . .                                   | 188 |
| 193. | NFS termination console messages. . . . .                                | 189 |
| 194. | NFS QUERY CONFIG command. . . . .  | 189 |
| 195. | NFS QUERY PATHS command. . . . .   | 190 |
| 196. | NFS QUERY USERS command . . . . .  | 190 |
| 197. | NFS client configuration window (1 of 4) . . . . .                       | 192 |
| 198. | NFS client configuration window (2 of 4) . . . . .                       | 193 |
| 199. | NFS client configuration window (3 of 4) . . . . .                       | 194 |
| 200. | NFS client configuration window (4 of 4) . . . . .                       | 195 |
| 201. | NFS server file system window . . . . .                                  | 196 |
| 202. | Mapping the NFS server to a network drive (1 of 2). . . . .              | 197 |
| 203. | Mapping the NFS server to a network drive (2 of 2). . . . .              | 197 |
| 204. | Mapping the POWER queue as a network drive (1 of 2) . . . . .            | 198 |
| 205. | Mapping the POWER queue as a network drive (2 of 2) . . . . .            | 198 |
| 206. | My Computer display of POWER queues . . . . .                            | 199 |
| 207. | Windows Explorer access to VSE file system entries . . . . .             | 200 |
| 208. | Windows WordPad display of POWER LST queue entry. . . . .                | 201 |
| 209. | Windows Explorer display of VSAM catalog. . . . .                        | 202 |

|      |  |     |
|------|--|-----|
| 210. | Define HTTP and DEFINE CGI statements in the IPINIT04.L file . . . . . | 204 |
| 211. | Check the status of the HTTP daemons . . . . .                         | 205 |
| 212. | Default page INDEX.HTML in sublibrary TCPIP.HTML. . . . .              | 207 |
| 213. | Web browser display of VSE INDEX.HTML Web page. . . . .                | 209 |
| 214. | PASSWORD.HTML document from our secured HTTP daemon . . . . .          | 211 |
| 215. | VIOLATED.HTML Web page from our secured HTTP daemon . . . . .          | 211 |
| 216. | Form page ITSOFORM.HTML. . . . .                                       | 213 |
| 217. | Web browser display of ITSOFORM.HTML Web page . . . . .                | 214 |
| 218. | Console output from the QUERY CGIS command . . . . .                   | 216 |
| 219. | Source code for our sample REXX CGI program . . . . .                  | 217 |
| 220. | Web page created by our REXX CGI program . . . . .                     | 218 |



---

## Tables

|   |     |
|---|-----|
| 1. Internet growth . . . . .                              | 2   |
| 2. IP address . . . . .                                   | 8   |
| 3. IP - class A address concept without subnets . . . . . | 10  |
| 4. IP - class A subnet mask . . . . .                     | 11  |
| 5. IP - class A subnet host address . . . . .             | 11  |
| 6. IP - class A subnet number . . . . .                   | 11  |
| 7. DNS - some top-level Internet domains . . . . .        | 29  |
| 8. Growth of the World Wide Web . . . . .                 | 31  |
| 9. Member types recognized by the HTTP daemon. . . . .    | 207 |





---

## Preface

Transmission Control Protocol/Internet Protocol (TCP/IP) is the most important protocol in the Internet networking environment.

This redbook describes how VSE/ESA users can enable their VSE/ESA system to participate in the world of TCP/IP networks using TCP/IP for VSE/ESA. It describes in a detailed and comprehensive way all the steps that are required to install and customize TCP/IP for VSE/ESA on the VSE/ESA system.

Several examples show how to make use of the TCP/IP applications that are supported by TCP/IP for VSE/ESA. This includes functions such as Telnet/TN3270, FTP, LPD/LPR, GPS, or NFS. We also provide guidance to help you set up your VSE/ESA system as an HTTP server on the Internet. TCP/IP for VSE/ESA provides a socket application programming interface as described in *TCP/IP for VSE Programmer's Reference, Release 1.4*, but its discussion is beyond the scope of this redbook.

This publication is intended for system engineers responsible for implementing VSE/ESA host integration into TCP/IP networks.

This redbook replaces the redbook *The Native TCP/IP Solution for VSE*, SG24-2041.

---

## The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working for the International Technical Support Organization, Poughkeepsie Center. The project was run at the home location of IntelliWare Systems, Inc., Arlington, Texas.

**Annegret Ackel**, from the IBM Development Laboratory Boeblingen, Germany, was the project leader.

**John Lawson**, from IntelliWare Systems, Inc., Arlington, Texas, U.S.

**Charles J. McMurry**, from IntelliWare Systems, Inc., Arlington, Texas, U.S.

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

Billy Bingham  
IntelliWare Systems, Inc, Arlington, Texas, U.S.

Dagmar Kruse  
IBM Germany

Cyrus Mead IV  
IntelliWare Systems, Inc., Arlington, Texas, U.S.

Marc Schare  
Connectivity Systems, Inc., Columbus, Ohio, U.S.

Hanns-Joachim Uhl  
IBM Development Laboratory Boeblingen, Germany

---

## Comments welcome

### **Your comments are important to us!**

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

- Fax the evaluation form found in “IBM Redbooks review” on page 235 to the fax number shown on the form.
- Use the online evaluation form found at <http://www.redbooks.ibm.com/>.
- Send your comments in an Internet note to [redbook@us.ibm.com](mailto:redbook@us.ibm.com).

---

## Chapter 1. Introduction to TCP/IP

Many excellent publications have been written on the topic of TCP/IP and the Internet. The aim of this chapter, therefore, is to provide only a short overview for the benefit of those readers who may not be familiar with the topic or who may desire a quick review.

Note that this chapter provides an overview of the capabilities of the TCP/IP protocol, but not all of these capabilities are available with TCP/IP for VSE/ESA. Please refer to the product documentation for TCP/IP for VSE/ESA for a description of available functions and applications (see Appendix B, “Related publications” on page 221).

We have included a selection of publications on advanced TCP/IP and Internet topics for your reference in 1.12, “TCP/IP and Internet publications” on page 37.

---

### 1.1 Why TCP/IP

The need to interconnect networks that are based on different protocols and platforms was recognized in the early 1970s, during a period when the use and development of networking technology was increasing. The rapid growth in networking over the past three decades has allowed users much greater access to resources and information, but it has caused significant problems when merging, or interconnecting, different types of networks. Open protocols and common applications were required; this led to the development of a protocol suite known as Transmission Control Protocol/Internet Protocol (TCP/IP) which originated with the U.S. Department of Defense (DoD) in the mid-1960s and took its current form around 1978. An interesting article about the history of the Internet can be found at the following URL:

<http://www.isoc.org/internet-history/>

---

### 1.2 The growth of TCP/IP

In the early 1980s TCP/IP became the backbone protocol in multivendor networks such as ARPANET, NFSNET, and regional networks. The protocol suite was integrated into the University of California at Berkeley's UNIX operating system and became available to the public for a nominal fee. The inexpensive availability of TCP/IP in UNIX, combined with its spread to other operating systems, resulted in its increasing use in both local area network (LAN) and wide area network (WAN) environments. Today, TCP/IP provides corporations with the ability to merge differing physical networks while giving users a common suite of functions. It allows interoperability between equipment supplied by multiple vendors on multiple platforms, and it provides access to the Internet.

In fact, the Internet, which has become the largest computer network in the world, is based on the TCP/IP protocol suite. The Internet consists of large international, national, and regional backbone networks that allow local and campus networks and individuals access to global resources. Use of the Internet has grown rapidly

over the last few years, as illustrated in Table 1. The most recent estimate suggests in excess of 29 million hosts on the Internet today.

Table 1. Internet growth

| Date      | Hosts      | Networks | Domains   |
|-----------|------------|----------|-----------|
| July 1989 | 130,000    | 650      | 3,900     |
| July 1992 | 992,000    | 6,569    | 16,300    |
| July 1993 | 1,776,000  | 13,767   | 26,000    |
| July 1995 | 6,642,000  | 61,538   | 120,000   |
| July 1996 | 12,881,000 | 134,365  | 488,000   |
| July 1997 | 26,053,000 |          | 1,301,000 |
| July 1998 | 36,739,000 |          |           |
| July 1999 | 56,218,000 |          |           |

In contrast to the Internet, the term *intranet* has evolved recently to describe TCP/IP networks that are entirely under the control of a private authority or company. These intranets may or may not have connections to other independent intranets (which would then be referred to as extranets) or the Internet. They may or may not be fully or partially visible to the outside, depending on the implementation.

TCP/IP also provides for the routing of multiple protocols to and from diverse networks. For example, a requirement to connect isolated networks using IPX, AppleTalk, and TCP/IP protocols using a single physical connection can be accomplished by using routers utilizing TCP/IP protocols.

One further reason for the growth of TCP/IP is the popularity of the socket programming interface, which is the programming interface between the TCP/IP transport protocol layer and TCP/IP applications. A large number of applications today have been written for the TCP/IP socket interface.

---

### 1.3 Internet standards and Request for Comments (RFC)

We mentioned in the previous section that the Internet is a large multinational, multivendor, multiplatform network. This raises some questions, such as:

- Are there any standards for such a diverse network?
- Who establishes and reviews them?
- Who assigns network addresses?
- Who manages the Internet?

The Internet Society (ISOC), formerly known as the Internet Activities Board (IAB), is the nonprofit coordinating committee for Internet design, engineering, and management. The ISOC members are committed to making the Internet function effectively and evolve to meet a large-scale, high-speed future. The ISOC has established several bodies for administering, standardizing, and researching for the Internet:

- The Internet Architecture Board (IAB)

- The Internet Engineering Task Force (IETF)
- The Internet Research Task Force (IRTF)
- The Internet Assigned Numbers Authority (IANA)

While the IAB oversees and manages the Request for Comments (RFC) publication process, the IETF actually defines the standards through a number of subcommittees or task forces, and the IRTF engages in Internet-related research projects.

RFC is the mechanism through which the Internet protocol suite has been evolving. For example, an Internet protocol can have one of six states: standard, draft standard, proposed standard, experimental, informational, and historic. In addition, an Internet protocol has one of five statuses: required, recommended, elective, limited use, and not recommended. By communicating using the RFC, new protocols are being designed and implemented by researchers from both academic institutions and commercial corporations. At the same time, some old protocols are being superseded by new ones.

RFCs can be viewed or obtained online from the IETF Web page using the following URL:

<http://www.ietf.org/rfc.html>

The RFC standards are described in the Internet Official Protocol Standards RFC, currently RFC 2500.

The task of coordinating the assignment of values to the parameters of protocols is delegated to the IANA. These protocol parameters include op-codes, type fields, terminal types, system names, object identifiers, and so forth. The Assigned Numbers RFC, currently RFC 1700, documents these protocol parameters.

To obtain registered IP addresses (see 1.6.1.1, “IP addressing” on page 8) and domain names (see 1.9.4, “Domain Name System (DNS)” on page 28), you need to contact the Internet Network Information Center (InterNIC), the administrative body for the Internet.

Registration is available online at the NIC Web site using the following URL:

<http://rs.internic.net/>

---

## 1.4 TCP/IP architecture

TCP/IP, as a set of communications protocols, is based on layers. Unlike SNA or OSI, which distinguish seven layers of communication, there are only four layers in the TCP/IP model. They enable heterogeneous systems to communicate by performing network-related processing such as message routing, network control, error detection, and correction.

The layering model of TCP/IP is shown in Figure 1, with an explanation of each layer following thereafter:

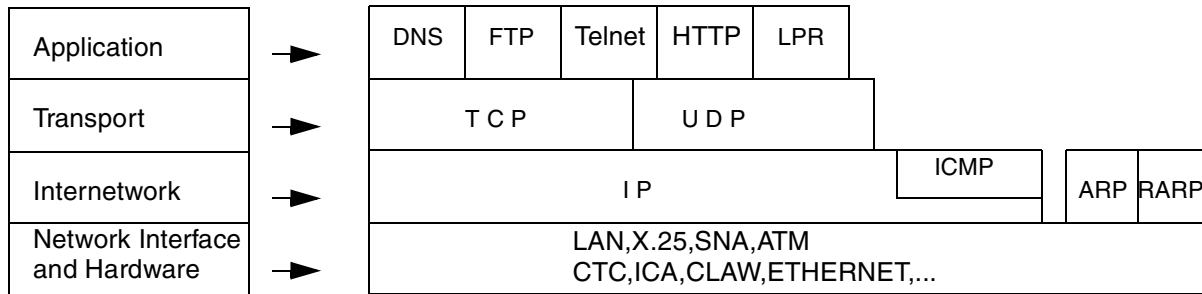


Figure 1. TCP/IP architecture model - layers and protocols

### **Application layer**

The application layer is provided by the program that uses TCP/IP for communication. Examples of applications are:

- Domain Name System (DNS)
- File Transfer Product (FTP)
- Teletypewriter Network (Telnet)
- Web server (HTTP)
- Line printer request (LPR)

The interface between the application and transport layers is defined by port numbers and sockets, which are described in more detail in 1.8.1, “Ports and sockets” on page 24.

### **Transport layer**

The transport layer provides communication between application programs. The applications may be on the same host or on different hosts. Multiple applications can be supported simultaneously. The transport layer is responsible for providing a reliable exchange of information. The main transport layer protocol is TCP. Another is User Datagram Protocol (UDP), which provides a connectionless service, in contrast to TCP, which provides a connection-oriented service. That means that applications using UDP as the transport protocol have to provide their own end-to-end flow control. Usually, UDP is used by applications that need a fast transport mechanism.

### **Internetwork layer**

The internetwork layer provides communication between computers. Part of communicating messages between computers is a routing function that ensures that messages will be correctly delivered to their destination. The Internet Protocol (IP) provides this routing function. Examples of internetwork layer protocols are IP, ICMP, ARP, and RARP.

### **Network interface layer**

The network interface layer, sometimes also referred to as link layer, data link layer, or network layer, is implemented by the physical network that connects the computers. Examples are LAN (token-ring = IEEE 802.5 and Ethernet = IEEE 802.3 standards), Ethernet, X.25, ISDN, ATM, SNA, CTC, CLAW or ICA.

---

## 1.5 The TCP/IP network interface layer

Network protocols define how data is transported over a physical network. After a TCP/IP packet is created, the network protocol adds a transport-dependent network header before the packet is sent out to the network.

### 1.5.1 Network interface

TCP/IP requires a network processor and associated components for attachment to the teleprocessing network. This section describes possible network attachments.

#### ***IBM 9221 Integrated Communication Processor***

The IBM 9221 Information System can be configured for Token-Ring LANs and Ethernet, depending on the communication processor installed. The IBM 9221 Communications Processor requires one of the following LAN adapters:

- IBM 9221 802.3 Ethernet LAN
- IBM 9221 Token-Ring LAN (16/4 Mbps or 4 Mbps)

#### ***IBM 2216 Nways Multiaccess Connector***

The IBM 2216 Nways Multiaccess Connector delivers wide area network access, S/390 host access and remote site concentration. Combined with the Nways Multiprotocol Access Service (MAS), the IBM 2216 helps to increase network performance and operations efficiencies.

The IBM 2216 Multiaccess Connector supports the following types of network interfaces:

- IBM Ethernet (10/100 Mbps)
- IBM Token-Ring (4/16 Mbps)
- IBM FDDI LAN (100 Mbps)
- ATM LAN Emulation Mode
- ATM native
- Multi-Path Channel (MPC+) support
- High Performance Data Transfer for TCP/IP and SNA

#### ***IBM Open Systems Adapter 2 (OSA-2)***

An IBM System/390 (S/390) Open Systems Adapter is an integrated hardware feature that lets its S/390 host platform provide connectivity to clients on directly-attached local area networks or, via attachment to an asynchronous transfer mode (ATM) switch, to clients in an ATM-based network. An OSA-2 therefore positions its host platform to be an open systems platform by bringing S/390 resources directly to networks. Each OSA-2 feature supports:

- ENTR (Ethernet/Token-Ring) feature with two ports. Each port can be configured as Ethernet or Token-Ring. Each port supports attachment to a 10 Mbps Ethernet LAN and/or a 4 or 16 Mbps Token-Ring LAN.
- FDDI feature has one LAN port and supports attachment to a 100 Mbps FDDI LAN. One single ring or dual ring station is supported, as well as attachment to an optical bypass switch.
- Fast Ethernet feature with one port and attachment to either a 10 Mbps or 100 Mbps Ethernet LAN.

The OSA-2 features are supported by the Open Systems Adapter Support Facility (OSA/SF) in the VSE/ESA (TCP/IP and SNA/APPN) environment. The ability to configure OSA-2 to allow sharing among logical partitions (LPAR support) is a key advantage offered by OSA/SF. When the S/390 server is running in LPAR mode, TCP/IP and SNA/APPN applications can share access to OSA-2 on the same LAN port.

#### ***Common Link Access to Workstation (CLAW)***

The Block Multiplexer Channel Adapter connectivity allows the system unit (RISC SYSTEM/6000) to communicate with an S/390 host. The support of TCP/IP for VSE/ESA is achieved via the high-performance CLAW protocol. This protocol improves the performance on the S/390 processor by reducing the number of I/O interrupts to the CLAW host. This connectivity allows the system unit to be used as a gateway between the VSE/ESA host and the downstream networks that consist of LANs or WANs. AIX Version 3.2 or later of the operating system is required for the support.

The system unit supports at most two Multiplexer Channel Adapters, allowing the attachment of two channel interfaces. The two channel interfaces may be attached to the same host or to two different hosts.

The various configurations are:

- Normal mode connectivity between S/390 host and a system unit. The protocol is similar to CTC.
- Dual Attachment configurations: a system unit with two Block Multiplexer Channel Adapters may be connected to a single host or to two different hosts.
- Direct channel attachment between S/390 hosts and a system unit: the system unit connected to the channel is the machine used to communicate with the host. The user can be connected to the system unit by means of an ASCII terminal, an X-station on a local area network, or remotely logged in from a LAN.
- Direct channel attachment between S/390 hosts and a system unit using the 3044 Channel Extender: the only difference with the previously described configuration is that the 3044-Fiber Optic Channel Extender units allow the channel distance to extend to a maximum of 3 km.
- Attachment to ESCON using the 9034 converter: if a 9034 is installed, it requires a dedicated ESCON channel. No other ESCON control unit may be attached; however, other parallel channel control units may be added through the same 9034.

#### ***Channel-to-Channel support (CTC)***

VSE/ESA supports the IBM 3088 CTC for S/390 host interconnection.

A Virtual-Channel-to-Channel device (VCTC) is normally used to connect two “virtual VSE machines” if running under VM/ESA.

The TCP/IP device driver CTC is used to transport IP packets.

### **1.5.2 Interfacing with the network layer**

Though the network interface layer itself is not covered by the TCP/IP standards, the RFCs do specify certain methods to access that layer from the higher layers. Before we describe some of the protocols that interface with the network layer,



we need to distinguish between different types of networks with which the internetwork layer can be connected.

### ***Multiaccess broadcast networks***

In a network of this type, any system (TCP/IP host) can have multiple connections to other hosts simultaneously, and it can also send information to all other hosts on the same network with a single, special kind of message (broadcast). Local area networks typically represent this type of network. Protocols such as ARP, ProxyARP, RARP, BootP, and DHCP are used with this type of network. We will briefly describe some of them in this and following sections.

### ***Multiaccess nonbroadcast networks***

In a network of this type, any host can have multiple connections to other hosts simultaneously, but there are no broadcast mechanisms in place.

### ***Point-to-point networks***

In a network of this type, a host can only have one connection to one other host at any time, and there are no broadcast mechanisms in place. Examples of this type of network are CTC connections.

#### **Notes:**

- The term *connection* in the three paragraphs above applies to any single IP interface of a host in any of the network types mentioned. For instance, a host could have multiple point-to-point interfaces and thus more than one connection at a time, but still only one per interface.
- Some publications only distinguish between broadcast and nonbroadcast networks.

#### **1.5.2.1 Hardware address resolution (ARP and RARP)**

The Address Resolution Protocol (ARP) maps Internet addresses to hardware addresses. When an application attempts to send data over a TCP/IP network capable of broadcasting, IP requests the appropriate hardware address mapping using ARP. If the mapping is not in the mapping table (ARP cache), an ARP broadcast packet is sent to all the hosts on the network requesting the physical hardware address for the host. For more information about ARP, see RFC 826.

An exception to the rule constitutes the asynchronous transfer mode (ATM) technology where ARP cannot be implemented in the physical layer as described above. Therefore, an ARP server is used, with which every host has to register upon initialization to be able to resolve IP addresses to hardware addresses.

Some network hosts do not know their IP addresses when they are initialized. This can be true especially in the case of a host needing to be booted from a diskette. Reverse ARP (RARP) can be used by, for example, a diskless workstation to determine its own IP address. In this case the workstation would already know its hardware address (discovered at initialization) and would broadcast a request to a RARP server to map the addresses. It is necessary to have a RARP server in your network in order to implement RARP.

---

## 1.6 TCP/IP internetwork layer protocols

This section provides a short overview of the most important and common protocols of the TCP/IP internetwork layer.

### 1.6.1 Internet Protocol (IP)

IP is the layer that hides the underlying physical network from the upper-layer protocols. It is an unreliable, best-effort, and connectionless packet delivery protocol. Note that best-effort means that the packets sent by IP may be lost, out of order, or even duplicated, but IP will not handle these situations. It is up to the higher-layer protocols to deal with these situations.

One of the reasons for using a connectionless network protocol was to minimize the dependency on specific computing centers that used hierarchical connection-oriented networks. The Department of Defense (DoD) intended to deploy a network that would still be operational if parts of the country were destroyed. During earthquakes, this has been proved to be true for the Internet.

#### 1.6.1.1 IP addressing

IP uses *IP addresses* to specify source and target hosts on the Internet. (For example, we can contrast an IP address in TCP/IP with a fully qualified NETID.LUNAME in SNA). An IP address consists of 32 bits, which is usually represented in the form of four decimal numbers, one decimal number for each byte (or octet). For example:

Table 2. IP address

|          |          |          |          |                              |
|----------|----------|----------|----------|------------------------------|
| 00001001 | 01000011 | 00100110 | 00000001 | A 32-bit IP address          |
| 9        | 67       | 38       | 1        | Decimal notation (9.67.38.1) |

An IP address consists of two logical parts: a *network address* and a *host address*. An IP address belongs to one of four classes depending on the value of its first four bits. (A fifth class, class E, is not commonly used.) This is shown in Figure 2 on page 9.

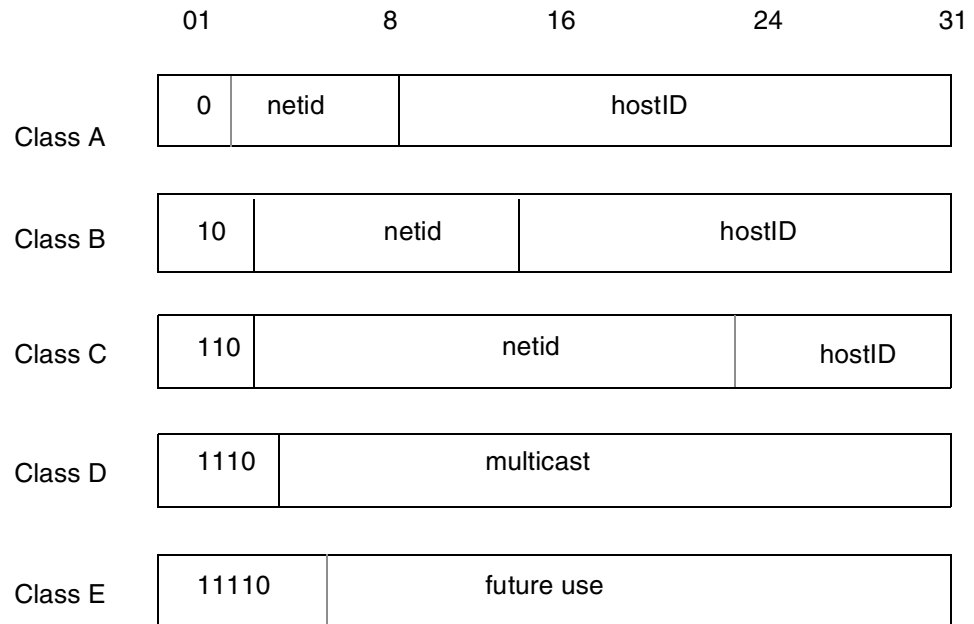


Figure 2. IP - assigned classes of IP addresses

- Class A addresses use 7 bits for the *network* and 24 bits for the *host* portion of the IP address. That allows for 126 ( $2^{**7}-2$ ) networks with 16777214 ( $2^{**24}-2$ ) hosts each; a total of over 2 billion addresses.
- Class B addresses use 14 bits for the *network* and 16 bits for the *host* portion of the IP address. That allows for 16382 ( $2^{**14}-2$ ) networks with 65534 ( $2^{**16}-2$ ) hosts each; a total of over 1 billion addresses.
- Class C addresses use 21 bits for the *network* and 8 bits for the *host* portion of the IP address. That allows for 2097150 ( $2^{**21}-2$ ) networks with 254 ( $2^{**8}-2$ ) hosts each; a total of over half a billion addresses.
- Class D addresses are reserved for multicasting (a sort of broadcasting, but in a limited area, and only to hosts using the same class D address).
- Class E addresses are reserved for future use.

Why always -2? Some values for these host IDs and network IDs are preassigned and cannot be used for actual network or host addressing:

*all bits 0*

Stands for this: this host (IP address with *host address*=0) or this network (IP address with *network address*=0). When a host wants to communicate over a network but does not yet know the network IP address, it may send packets with *network address*=0. Other hosts on the network will interpret the address as meaning "this network". Their reply will contain the fully qualified network address, which the sender will record for future use.

*all bits 1*

Stands for all: all networks or all hosts. For example:

128.2.255.255

means all hosts on network 128.2 (class B address).

This is called a directed broadcast address because it contains both a valid *network address* and a broadcast *host address*.

### Loopback

The class A network 127.0.0.0 is defined as the loopback network. Addresses from that network are assigned to interfaces that process data inside the local system and never access a physical network (loopback interfaces).

### 1.6.1.2 IP subnets

Due to the explosive growth of the Internet, the principle of assigned IP addresses became too inflexible to allow easy changes to local network configurations. Those changes might occur when:

- A new type of physical network is installed at a location.
- Growth of the number of hosts requires splitting the local network into two or more separate networks.
- Growing distances require splitting a network into smaller networks, with gateways between them.

To avoid having to request additional IP network addresses in these cases, the concept of subnets was introduced. The assignment of subnets can be done locally, as the whole network still appears to be one IP network to the outside world.

Recall that an IP address consists of a network address and a host address. For example, let us take a class A network; the address format is shown in Figure 3:

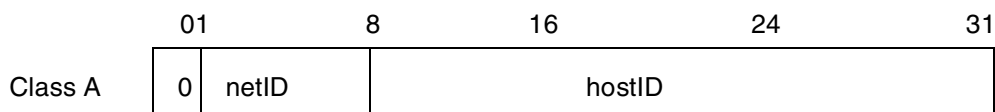


Figure 3. IP - class A address without subnets

Let us use the following IP address:

Table 3. IP - class A address concept without subnets

|          |          |          |          |   |
|----------|----------|----------|----------|---|
| 00001001 | 01000011 | 00100110 | 00000001 | A 32-bit IP address 9.67.38.1                           |
| 9        | 67       | 38       | 1        | Decimal notation (9.67.38.1) is an IP address (class A) |
| 9        |          |          |          | The network address                                     |
|          | 67       | 38       | 1        | The host address  |

Subnets form an extension to this by considering a part of the host address to be a subnetwork address. IP addresses are then interpreted as *network address-subnetwork address-host address*.

We may, for example, wish to choose the bits from 8 to 25 of a class A IP address to indicate the subnet addresses, and the bits from 26 to 31 to indicate the actual host addresses. Figure 4 on page 11 shows the format of a subnetted address that is thus derived from the original class A address.

|                |    |       |                |        |    |
|----------------|----|-------|----------------|--------|----|
|                | 01 | 8     | 16             | 24     | 31 |
| Class A Subnet | 0  | netID | subnet address | hostID |    |

Figure 4. IP - class A address with subnet address

We normally use a bit mask, known as the subnet mask, to identify which bits of the original host address field indicate the subnet number. In the above example, the subnet mask is 255.255.255.192 in decimal notation (or 11111111 11111111 11111111 11000000 in bit notation). Note that by convention, the network address is part of the subnet mask as well.

For each of these subnet values, only  $(2^{18})-2$  addresses (from 1 to 262142) are valid because of the all-bits-0 and all-bits-1 number restrictions. This split will therefore give 262142 subnets each with a maximum of  $(2^6)-2$  or 62 hosts.

You will notice that the value applied to the subnet number takes the value of the full byte with non-significant bits being set to zero. For example, the hexadecimal value 11 in this subnet mask assumes an 8-bit value 11000000 and gives a subnet value of 192 and not 3 as it might seem.

Applying this mask to our sample class A address 9.67.38.1 would break the address down as follows:

Table 4. IP - class A subnet mask

|          |          |          |          |   |                               |
|----------|----------|----------|----------|---|-------------------------------|
| 00001001 | 01000011 | 00100110 | 00000001 | = | 9.67.38.1 (Class A address)   |
| 11111111 | 11111111 | 11111111 | 11=====  | = | 255.255.255.192 (Subnet mask) |
| =====    | =====    | =====    | =====    | = | Logical AND                   |
| 00001001 | 01000011 | 00100110 | 00=====  | = | 9.67.38 (Subnet base address) |

and leaves a host address of:

Table 5. IP - class A subnet host address

|       |       |       |          |   |                  |
|-------|-------|-------|----------|---|------------------|
| ===== | ===== | ===== | ==000001 | = | 1 (Host address) |
|-------|-------|-------|----------|---|------------------|

IP will recognize all host addresses as being on the local network for which the logical AND operation described above produces the same result. This is important for routing IP datagrams in subnet environments.

Note that the actual subnet number would be:

Table 6. IP - class A subnet number

|       |          |          |         |   |                       |
|-------|----------|----------|---------|---|-----------------------|
| ===== | 01000011 | 00100110 | 00===== | = | 68760 (Subnet number) |
|-------|----------|----------|---------|---|-----------------------|

You will notice that the subnet number shown above is a relative number, that is, it is the 68760th subnet of network 9 with the given subnet mask.

The division of the original *host address* part into *subnet* and *host* parts can be chosen freely by the local administrator, except that the values of all zeros and all ones in the *subnet* field are reserved for special addresses.

**Note:** Because the range of available IP addresses is decreasing rapidly, many routers do support the use of all zeros and all ones in the *subnet* field, though this is not consistent with the standards.

#### 1.6.1.3 Intranets (private IP addresses)

An approach to conservation of the IP address space is described in RFC 1918, Address Allocation for Private Internets. Briefly, it relaxes the rule that IP addresses are globally unique by reserving part of the address space for networks that are used exclusively within a single organization and that do not require IP connectivity to the Internet. There are three ranges of addresses that have been reserved by IANA for this purpose:

|                               |                                 |
|-------------------------------|---------------------------------|
| 10                            | A single class A network        |
| 172.16 through 172.31         | 16 contiguous class B networks  |
| 192.168.0 through 192.168.255 | 256 contiguous class C networks |

Any organization can use any addresses in these ranges without reference to any other organization. However, because these addresses are not globally unique, they cannot be referenced by hosts in another organization and they are not defined to any external routers.

Routers in networks not using private addresses, particularly those operated by Internet service providers, are expected to quietly discard all routing information regarding these addresses. Routers in an organization using private addresses are expected to limit all references to private addresses to internal links; they should neither advertise routes to private addresses to external routers nor forward IP datagrams containing private addresses to external routers.

Hosts having only a private IP address do not have IP layer connectivity to the Internet. This may be desirable and may even be a reason for using private addressing. All connectivity to external Internet hosts must be provided with application gateways.

#### 1.6.1.4 IP datagram

The unit of transfer of a data packet in TCP/IP is called an IP datagram. It is made up of a header, containing information for IP and data that is only relevant to the higher level protocols. IP can handle fragmentation and reassembly of IP datagrams. The maximum length of an IP datagram is 65,535 bytes (or octets). There is also a requirement for all TCP/IP hosts to support IP datagrams up to a size of 576 bytes without fragmentation.

The IP datagram header is a minimum of 20 bytes long, and is formatted as shown in Figure 5.

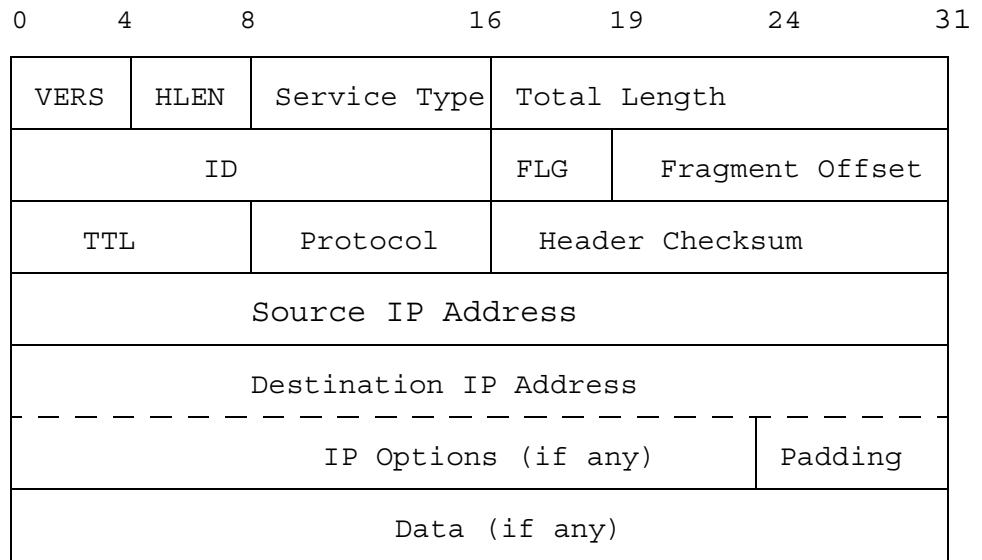


Figure 5. IP - format of an IP datagram header

We do not elaborate on the format of the IP datagram header. You can find this information in RFC 791, Internet Protocol. For information on accessing this document, see 1.12, “TCP/IP and Internet publications” on page 37.

#### 1.6.1.5 IP Routing

There are two types of IP routing: direct and indirect.

##### **Direct routing**

If the destination host is attached to a physical network to which the source host is also attached, an IP datagram can be sent directly, simply by encapsulating the IP datagram in the physical network frame. This is called direct delivery and is referred to as direct routing.

##### **Indirect routing**

Indirect routing occurs when the destination host is not on a network directly attached to the source host. The only way to reach the destination is via one or more IP gateways (note that in TCP/IP terminology, the terms gateway and router are used interchangeably for a system that actually performs the duties of a router). The address of the first of these gateways (the first hop) is called an indirect route in the context of the IP routing algorithm. The address of the first gateway is the only information needed by the source host.

##### **IP routing table**

The determination of available direct routes is derived from a list of local interfaces available to IP and is composed by IP automatically at initialization. A list of networks and associated gateways (indirect routes) needs to be configured to be used with IP routing if required. Each host keeps the set of mappings between the following:

- Destination IP network address(es)
- Route(s) to next gateway(s)

These are stored in a table called the IP routing table. Three types of mappings can be found in this table:

1. The direct routes, for locally attached networks
2. The indirect routes, for networks reachable via one or more gateways
3. The default route, which contains the (direct or indirect) route to be used in case the destination IP network is not found in the mappings of type 1 and 2 above

Additional information on routing can be found in 1.7.4, “Routing” on page 21.

### **1.6.2 Internet Control Message Protocol (ICMP)**

ICMP is shown in Figure 1 on page 4 as being in the same protocol layer as IP. It is actually an integral part of IP. ICMP is used for reporting errors in datagram delivery, such as “destination unreachable”, and it can assist in discovering routers and maximum transmission units (MTUs) along a path that an IP datagram eventually travels.

#### **1.6.2.1 Packet internet groper (PING)**

Perhaps one of the most useful commands available on all TCP/IP implementations is the PING application. PING uses ICMP to send an Echo datagram to a specified IP address and wait for it to return.

#### **1.6.2.2 Traceroute**

The Traceroute program can be useful for debugging purposes. Traceroute enables determination of the route that an IP datagram follows from host to host. Traceroute is based upon ICMP and UDP. It sends an IP datagram with a TTL of 1 to the destination host. The first router to see the datagram will decrement the TTL to 0 and return an ICMP Time Exceeded message as well as discarding the datagram. In this way, the first router in the path is identified. This process can be repeated with successively larger TTL values in order to identify the series of routers in the path to the destination host. Traceroute actually sends UDP datagrams to the destination host which reference a port number that is outside the normally used range. This enables Traceroute to determine when the destination host has been reached, that is, when an ICMP Port Unreachable message is received. This is very useful for debugging purposes and also for learning if a remote host can be reached from the local host.

ICMP is defined in RFC 792.

---

## **1.7 TCP/IP connectivity bridging, switching, and routing**

An internetwork is a collection of individual networks, connected by networking devices, that functions as a single large network. Figure 6 illustrates different kinds of network technologies that can be interconnected by routers and other networking devices.



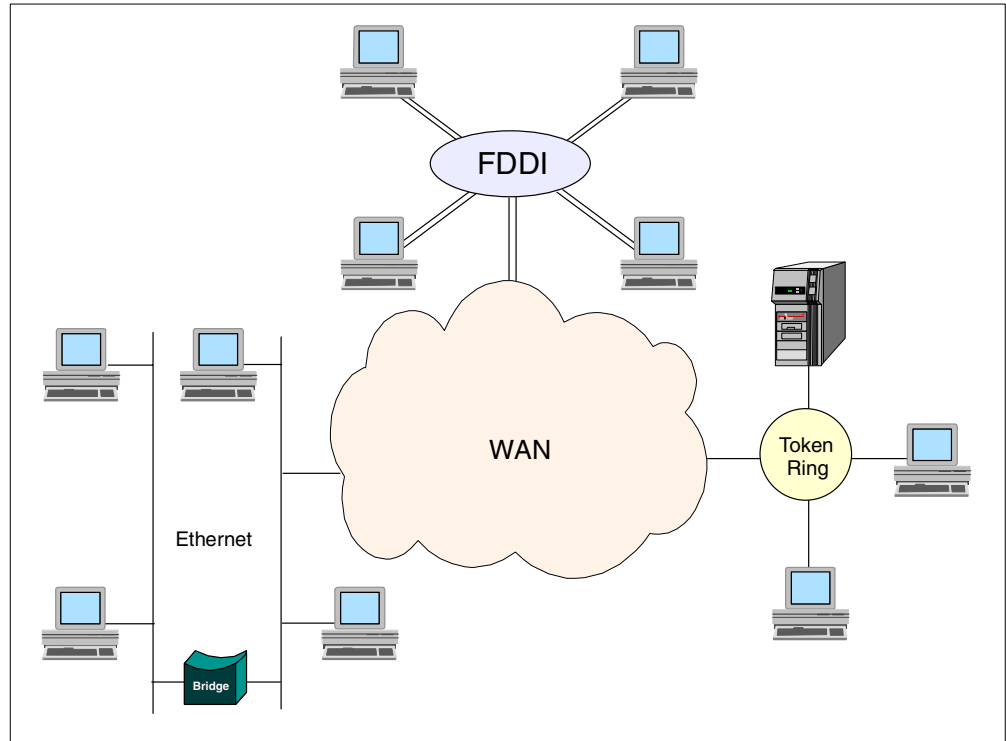


Figure 6. Internetwork connectivity

### 1.7.1 Local area network (LAN)

A LAN is a high-speed data network. It connects workstations, personal computers, printers, and other devices. LANs offer many advantages, including shared access to devices and applications, file exchange between connected users, and communication via electronic mail and other applications. LAN protocols function at the lowest two layers of the TCP/IP model between the physical layer and the data link layer, as discussed in 1.4, “TCP/IP architecture” on page 3.

#### 1.7.1.1 LAN media-access methods

LAN protocols use a couple of methods to access the physical network medium: carrier sense multiple access with collision detection (CSMA/CD) and token passing. An example of the CSMA/CD scheme is ETHERNET/IEEE 802.3; examples of the token passing scheme are token-ring/IEEE 802.5 and FDDI. See also Figure 7 for an illustration.

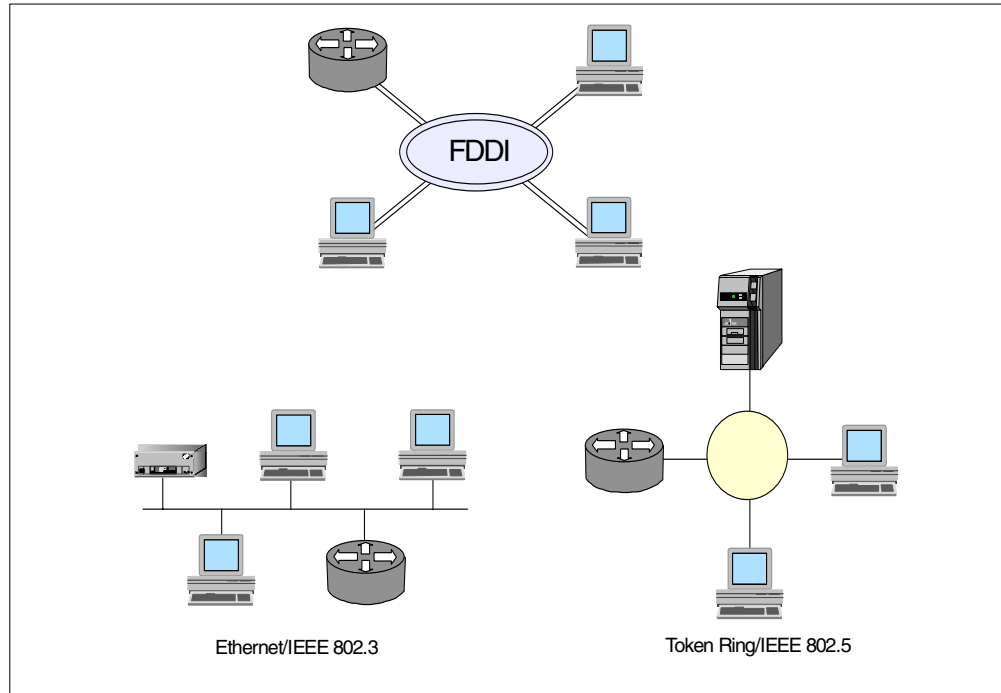


Figure 7. Three most commonly used LAN implementations

#### 1.7.1.2 LAN topology

LAN topologies define the manner in which network devices are organized. Four common LAN topologies exist: *bus*, *ring*, *star*, and *tree*.

A *bus* topology is a linear LAN architecture in which transmissions from network stations propagate the length of the medium and are received by all other stations. The most common LAN implementations are Ethernet/IEEE 802.3 networks, a bus topology.

A *ring* topology is a LAN architecture that consists of devices connected by unidirectional transmission links to form a single closed loop. Both token-ring/IEEE 802.5 and FDDI networks implement a ring topology.

A *star* topology is a LAN architecture in which the endpoints on a network are connected to a common central hub, or switch, by dedicated links. Logical bus and ring topologies are often implemented physically in a star topology.

A *tree* topology is a LAN architecture that is identical to the bus topology, except that branches with multiple nodes are possible in this case.

#### 1.7.1.3 LAN devices

Devices commonly used in LANs are repeaters, hubs, LAN extenders, bridges, LAN switches, and routers.

A *repeater* is a physical layer device used to interconnect an extended network. Repeaters receive signals from one network and retransmit those signals to another network. These actions prevent signal deterioration caused by long cable lengths and large numbers of connected devices. Repeaters are unable to perform complex filtering and other processing. In addition, all signals, including errors, are repeated and amplified.

A *hub* is a physical-layer device that connects multiple user stations via a dedicated cable. Hubs are used to create a physical star network while maintaining the logical bus or ring configuration of the LAN. In some respects, a hub functions as a multiport repeater.

A LAN *extender* is a remote-access multilayer switch that connects to a host router. LAN extenders forward traffic from all the standard network-layer protocols (such as IP, IPX, and AppleTalk), and filter traffic based on the MAC address or network-layer protocol type. LAN extenders filter out unwanted broadcasts and multicasts. LAN extenders are unable to segment traffic or create security firewalls. (See 1.7.3, “Bridging and switching” on page 18 and 1.7.4, “Routing” on page 21 for more discussion about bridges, switches, and routers.)

## 1.7.2 Wide area network (WAN)

A wide area network is a data communications network that covers a broad geographic area and often uses transmission facilities provided by common carriers, such as telephone companies. WAN technologies function at the lower three layers of the TCP/IP model: the physical layer, the data link layer, and the network layer.

### 1.7.2.1 Point-to-point links

A point-to-point link provides a single WAN communications path from a specific customer site, through a carrier network such as a telephone company, to a remote network. A point-to-point link is also known as a leased line because its path is permanent and fixed for each remote network reached through the carrier facilities. The carrier company reserves point-to-point links for the private use of the customer. CTC is an example of a point-to-point link.

Figure 8 shows a typical point-to-point link operating through a WAN to a remote network:

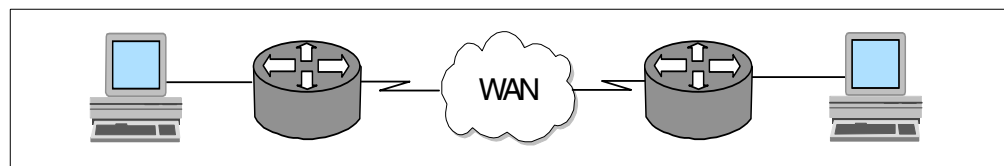


Figure 8. WAN point-to-point link

### 1.7.2.2 Circuit switching: ISDN

Circuit switching is a WAN switching method in which a dedicated physical circuit is established, maintained, and terminated through a carrier network for each communication session. Circuit switching operates much like a normal telephone call. Integrated services digital network (ISDN) is an example of a circuit-switched WAN technology.

### 1.7.2.3 Packet switching

Packet switching is a WAN switching method in which network devices share a single point-to-point link to transport packets from a source to a destination across a carrier network. Asynchronous Transfer Mode (ATM), frame relay, and X.25 are examples of packet-switched WAN technologies.

#### 1.7.2.4 WAN devices

Wide area networks use devices such as switches and access servers.

A WAN *switch* is a multiport internetworking device used in carrier networks. These devices typically switch such traffic as frame relay and X.25, and they operate at the data link layer of the TCP/IP architecture model.

An *access server* acts as a concentration point for dial-in and dial-out connections.

### 1.7.3 Bridging and switching

Bridges and switches are data communications devices that operate at layer-2 of the TCP/IP architecture model. They are referred to as data link layer devices.

Bridges became commercially available in the early 1980s. At the time of their introduction, bridges connected and enabled packet forwarding between networks. More recently, bridging between different networks has been defined and standardized. Several kinds of bridging have proven important as internetworking devices.

*Transparent bridging* is found primarily in Ethernet environments.

*Source-route bridging* occurs primarily in token-ring environments.

*Translational bridging* provides translation between the formats and transit principles of different media types (usually Ethernet and token-ring).

*Source-route transparent bridging* combines the algorithms of transparent bridging and source-route bridging to enable communication in mixed Ethernet/token-ring environments.

Bridging and switching occur at the link layer, which controls data flow, handles transmission errors, provides physical addressing, and manages access to the physical medium. Bridges provide these functions by using various link-layer protocols with specific flow control, error handling, addressing, and media-access algorithms. Examples of popular link-layer protocols include Ethernet, token-ring, and FDDI.

Bridges and switches analyze incoming frames, make forwarding decisions based on information contained in the frames, and forward the frames toward the destination. In some cases, such as source-route bridging, the entire path to the destination is contained in each frame. In other cases, such as transparent bridging, frames are forwarded one hop at a time toward the destination.

Upper-layer protocol transparency is a primary advantage of both bridging and switching. Because both device types operate at the link layer, they are not required to examine upper-layer information. This means that they can rapidly forward traffic representing any network-layer protocol. It is common for a bridge to move AppleTalk, DECnet, IP, XNS, NetBIOS, and other traffic between two or more networks.

Bridges are capable of filtering frames based on any layer-2 fields. Because link-layer information often includes a reference to an upper-layer protocol, bridges usually can filter on this parameter. Filters can be helpful in dealing with broadcast and multicast packets.

By dividing large networks into smaller units, called subnetworks, bridges, and switches provide several advantages. Because a certain percentage of traffic is forwarded, a bridge or switch will act as a firewall for some potentially damaging network errors, and can accommodate communication between a larger number of devices than would be supported on any single LAN connected to the bridge. Bridges and switches extend the effective length of a LAN.

One of the differences is that switches are significantly faster because they switch in hardware, while bridges switch in software. Switches also support higher port densities than bridges. Some switches support cut-through switching, which reduces latency and delays in the network, while bridges support only store-and-forward. Finally, switches reduce collisions on network segments because they provide dedicated bandwidth to each network segment.

### 1.7.3.1 Types of bridges

Bridges can be grouped into categories based on various characteristics. Using one classification scheme, bridges are either local or remote. Local bridges provide a direct connection between multiple LAN segments in the same area. Remote bridges connect multiple LAN segments in different areas, usually over telecommunications lines. Figure 9 illustrates these two configurations:

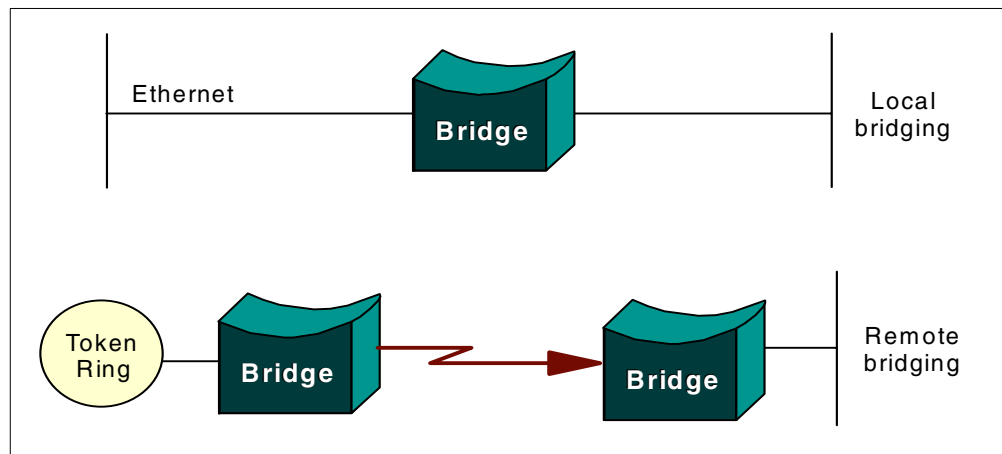


Figure 9. LAN local and remote bridges

Remote bridges cannot improve WAN speeds, but they compensate for speed discrepancies with a buffering capability. If a LAN device capable of a 3-Mbps transmission rate wants to communicate with a device on a remote LAN connected by a 64-kbps link, the local bridge must regulate the 3-Mbps data stream so that it does not overwhelm the 64-kbps serial link. This is done by storing the incoming data in on-board buffers and sending it over the serial link at a rate that the serial link can accommodate.

The Institute of Electrical and Electronic Engineers (IEEE) differentiates the OSI link layer into two separate sublayers: the media access control (MAC) sublayer and the logical link control (LLC) sublayer. The MAC sublayer permits media access, such as contention and token passing, while the LLC sublayer deals with framing, flow control, error control, and MAC-sublayer addressing.

Some bridges are MAC-layer bridges, which bridge between homogeneous networks (for example, IEEE 802.3 and IEEE 802.3), while other bridges can

translate between different link-layer protocols (for example, IEEE 802.3 and IEEE 802.5).

### 1.7.3.2 Types of switches

*Switches* are data link layer devices that, like bridges, enable multiple physical LAN segments to be interconnected into a single larger network. Similar to bridges, switches forward traffic based on MAC addresses. Because switching is performed in hardware instead of in software, it is significantly faster. Switches use either store-and-forward switching or cut-through switching when forwarding traffic. Many types of switches exist, including ATM switches, LAN switches, and various types of WAN switches.

### 1.7.3.3 ATM switch

Asynchronous transfer mode (ATM) switches provide high-speed switching for local and wide area applications. ATM switches support voice, video, and data applications and are designed to switch fixed-size information units called cells, which are used in ATM communications. Figure 10 illustrates an enterprise network comprised of multiple LANs interconnected across an ATM backbone:

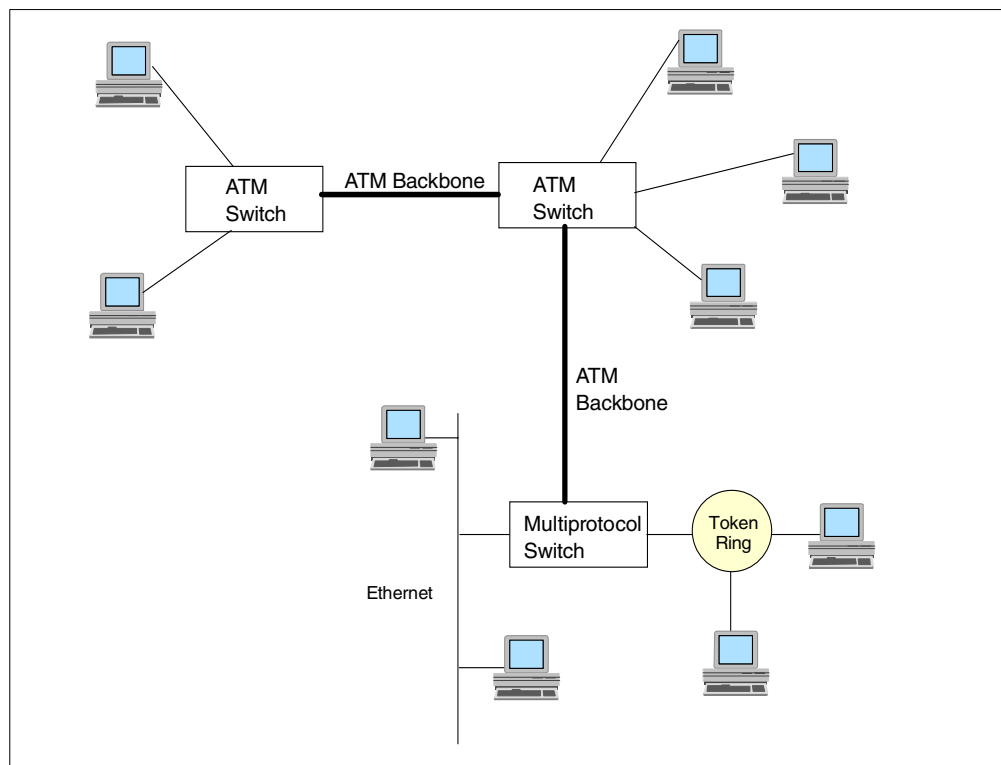


Figure 10. ATM switch with multiple LAN connections

### 1.7.3.4 LAN switch

LAN switches are used to interconnect multiple LAN networks. LAN switching provides dedicated, collision-free communication between network devices, with support for multiple simultaneous conversations. LAN switches are designed to switch data frames at high speeds. Figure 11 illustrates a simple network in which a LAN switch interconnects a 10-Mbps and a 100-Mbps Ethernet LAN:

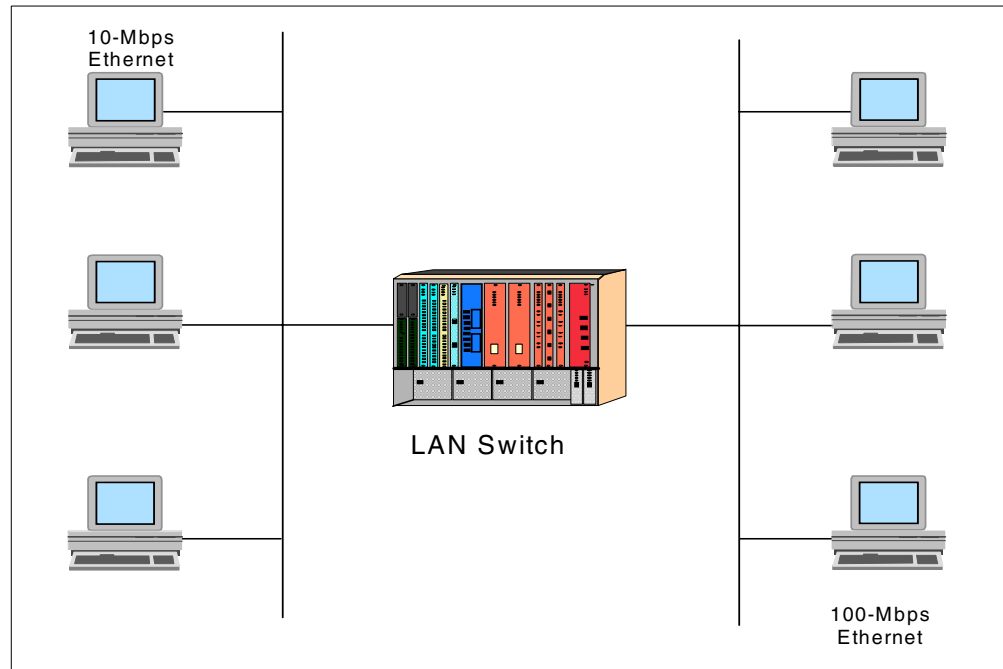


Figure 11. LAN switch Ethernet

### Switching

Switching algorithms are relatively simple and are basically the same for most routing protocols. In most cases, a host determines that it must send a packet to another host. Having acquired a router's address, the source host sends a packet addressed specifically to a router's physical (media access control, or MAC-layer) address, with the protocol (network layer) address of the destination host.

The router determines how to forward the packet to the next hop. If the router does not know how to forward the packet, it typically drops the packet. If the router knows how to forward the packet, it changes the destination physical address to that of the next hop and transmits the packet.

The next hop may be the ultimate destination host. If not, the next hop is usually another router, which executes the same switching decision process. As the packet moves through the internetwork, its physical address changes but its protocol address remains constant, as illustrated in Figure 12 on page 22.

The preceding discussion describes switching between a source and a destination system. The International Organization for Standardization (ISO) has developed a hierarchical terminology that describes this process.

## 1.7.4 Routing

Routing is the act of moving information across an internetwork from a source to a destination. Along the way, at least one intermediate node typically is encountered. Routing is often contrasted with bridging, which might seem to be the same thing to the casual observer. The primary difference between the two is that bridging occurs at layer-2 (the link layer), and routing occurs at layer-3 (the network layer). Routing and bridging use different protocols in the process of moving data from source to destination.

Routing involves two basic activities: determining optimal routing paths and transporting information groups (typically called packets) through connected subnetworks.

#### 1.7.4.1 Path determination

To aid the process of path determination, routers maintain information about the topology of the network.

They accomplish this by sending each other their routing tables through the transmission of messages. The routing update message generally consists of all or a portion of a routing table. By analyzing routing updates from all other routers, a router can build a detailed picture of network topology. A link-state advertisement, another example of a message sent between routers, informs other routers of the state of the sender's links. Link information also can be used to build a complete picture of topology to enable routers to determine optimal routes to network destinations.

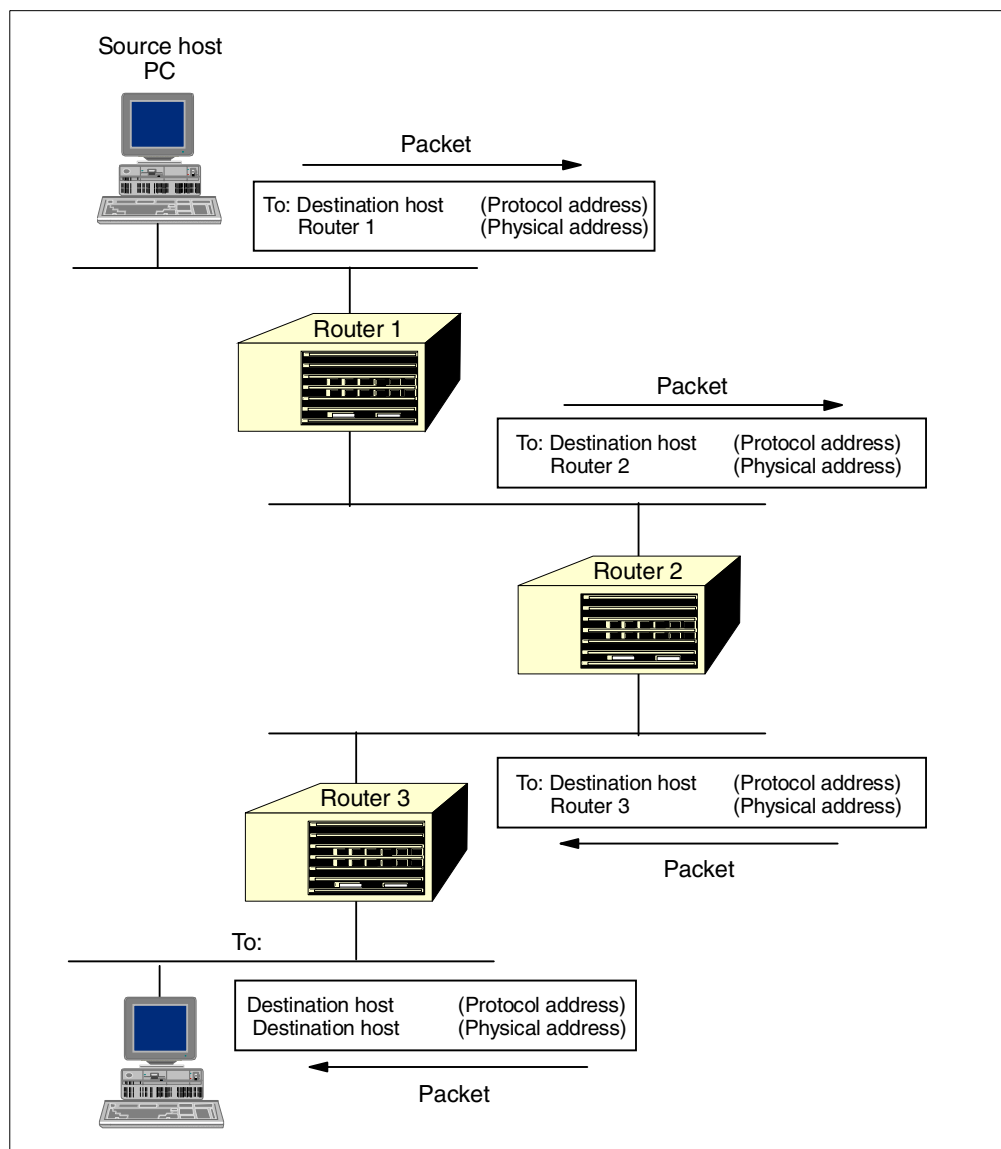


Figure 12. Routers packet transfer



Figure 12 illustrates a system in which numerous routers come into play during the switching process.

#### **1.7.4.2 Routing algorithms**

The following sections analyze some routing algorithm types with different key characteristics.

##### ***Static and dynamic routing***

Static routing requires that routes be configured manually by the network administrator. Because static routing systems cannot react to network changes, they are considered unsuitable for today's large, changing networks. Most of the dominant routing algorithms in the 1990s were dynamic routing algorithms, which adjusted to changing network circumstances by analyzing incoming routing update messages. If the messages indicated that a network change had occurred, the routing software recalculated routes and sent out new routing update messages. These messages changed the routing tables accordingly.

##### ***Single-path and multipath routing***

Some routing protocols support multiple paths to the same destination. Unlike single-path algorithms, these multipath algorithms permit traffic multiplexing over multiple lines. The advantages of multipath algorithms are obvious: they can provide better throughput and reliability.

#### **1.7.4.3 Routing metrics**

Routing tables contain information used by switching software to select the best route. But how are routing tables built? What is the information they contain? How do routing algorithms determine that one route is preferable to others?

Routing algorithms use many different metrics to determine the best route.

Sophisticated routing algorithms can base route selection on multiple metrics, combining them in a single (hybrid) metric. All the following metrics can be used:

- **Path length**

Path length is the most common routing metric. Some routing protocols allow network administrators to assign arbitrary costs to each network link. In this case, path length is the sum of the costs associated with each link traversed. Other routing protocols define hop count, a metric that specifies the number of passes through internetworking products, such as routers, that a packet must take en route from a source to a destination.

- **Reliability**

Reliability, in the context of routing algorithms, refers to the dependability (usually described in terms of the bit-error rate) of each network link. Some network links might go down more often than others. After a network fails, certain network links might be repaired more easily or more quickly than other links. Any reliability factors can be taken into account in the assignment of the reliability ratings, which are arbitrary numeric values usually assigned to network links by network administrators.

- **Delay**

Routing delay refers to the length of time required to move a packet from source to destination through the internetwork. Delay depends on many factors, including the bandwidth of intermediate network links, the port queues at each router along the way, network congestion on all intermediate network

links, and the physical distance to be traveled. Because delay is a conglomeration of several important variables, it is a common and useful metric.

- **Bandwidth**

Bandwidth refers to the available traffic capacity of a link. All other things being equal, a 10-Mbps Ethernet link would be preferable to a 64-kbps leased line. Although bandwidth is a rating of the maximum attainable throughput on a link, routes through links with greater bandwidth do not necessarily provide better routes than routes through slower links. If, for example, a faster link is busier, the actual time required to send a packet to the destination could be greater.

- **Load**

Load refers to the degree to which a network resource, such as a router, is busy. Load can be calculated in a variety of ways, including CPU utilization and packets processed per second. Monitoring these parameters on a continual basis can itself be a resource-intensive activity.

- **Communication cost**

Communication cost is another important metric, especially because some companies may not care about performance as much as they care about operating expenditures. Even though line delay may be longer, they will send packets over their own lines rather than through the public lines that cost money for usage time.

#### **1.7.4.4 Network protocols**

Routed protocols are transported by routers across an internetwork. In general, routed protocols in this context also are referred to as network protocols. These network protocols perform a variety of functions required for communication between user applications in source and destination devices, and these functions can differ widely among protocol suites. Network protocols occur at the upper four layers of the OSI reference model: the transport layer, the session layer, the presentation layer, and the application layer.

---

## **1.8 TCP/IP transport layer protocols and interfaces**

This section provides a brief overview of the protocols of the TCP/IP transport layer.

### **1.8.1 Ports and sockets**

Each process that wants to communicate with another process identifies itself to the TCP/IP protocol suite by one or more ports. A port is a 16-bit number used by the host-to-host protocol to identify to which higher-level protocol or application program (process) it must deliver incoming messages.

As some higher-level programs are themselves protocols, standardized in the TCP/IP protocol suite, such as Telnet and FTP, they use the same port number in all TCP/IP implementations. (Port 23 is used by a Telnet server; ports 20 and 21 are used by an FTP server.) Those assigned port numbers are called well-known ports and the standard applications are called well-known services.

The well-known ports are controlled and assigned by the IANA. On most systems these ports can only be used by system processes or by programs executed by privileged users. The assigned well-known ports occupy port numbers in the range 0 to 1023. The ports with numbers in the range 1024-65535 are not controlled by the Internet central authority and on most systems can be used by ordinary user-developed programs.

Confusion caused by two different applications trying to use the same port number on one host is avoided by writing those applications to request an available port from TCP/IP. Because this port number is dynamically assigned, it may differ from one invocation of an application to the next.

UDP, TCP, and ISO TP-4 all use the same port principle. To the extent possible, the same port numbers are used for the same services on top of UDP, TCP, and ISO TP-4.

A socket is a special type of file handle that is used by a process to request network services from the operating system. A socket address (also known as a transport address) is made up of three parts: {protocol, local address, local port number} or {protocol, foreign address, foreign port number}. An example of a socket in the TCP/IP suite is {TCP, 193.44.234.3, 12345}.

In other words, a socket is an endpoint for communication that can be named and addressed in a network.

### **1.8.2 The socket application programming interface**

The socket interface is one of several application programming interfaces (APIs) to the communication protocols. Designed to be a generic communication programming interface, it was first introduced by the 4.2BSD UNIX system. Although it has not been standardized, it has become a de facto industry standard.

The socket interface is differentiated by the services that are provided to applications: stream sockets (connection oriented), datagram sockets (connectionless), and raw sockets (direct access to lower-layer protocols) services.

A variation of the BSD socket interface is provided by the Winsock interface developed by Microsoft and other vendors to support TCP/IP applications on Windows operating systems. Winsock 2.0, the latest version, provides a more generalized interface allowing applications to communicate with any available transport layer protocol and underlying network services, including, but no longer limited to, TCP/IP.

### **1.8.3 User Datagram Protocol (UDP)**

User Datagram Protocol is an application interface to IP. It provides no additional reliability, flow control, or error recovery. It simply serves as a multiplexer/demultiplexer for sending/receiving IP datagrams, using ports to direct the datagram. As a result of this, the maximum amount of data that can be transferred by UDP at any time is what fits within a single IP datagram, less the size of the UDP header. This typically makes UDP a good choice for messaging and querying applications that do not rely on immediate acknowledgments, such

as SMTP, DNS, and SNMP. See Figure 13 on page 26 for a graphic representation of this interface.

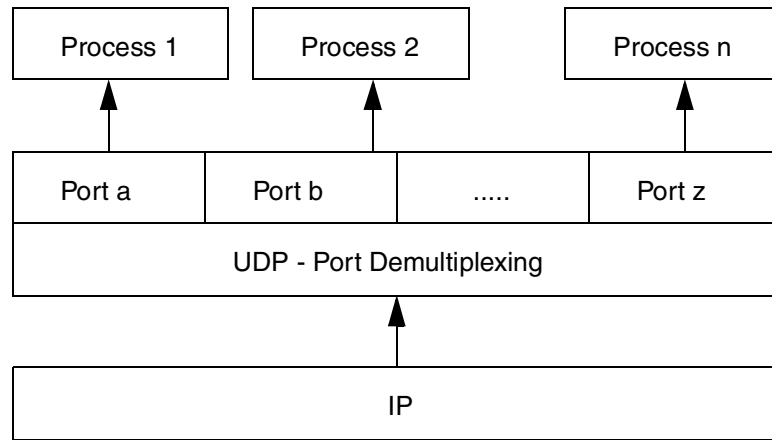


Figure 13. UDP - demultiplexing based on ports

#### 1.8.4 Transmission Control Protocol (TCP)

We have previously discussed IP, the unreliable connectionless packet (datagram) delivery mechanism that forms the basis of the TCP/IP protocol suite. Transmission Control Protocol, or TCP, is the higher-level protocol that provides reliability, flow control, and some error recovery. Many of the TCP/IP application protocols, such as Telnet and FTP, use TCP as the underlying protocol.

TCP is a connection-oriented, end-to-end reliable protocol providing logical connections between pairs of processes. Within TCP, a connection is uniquely defined by a pair of sockets (that is, by a pair of processes, on the same or different systems, that are exchanging information).

The two processes communicate with each other over a TCP connection (interprocess communication - IPC), as shown in Figure 14:

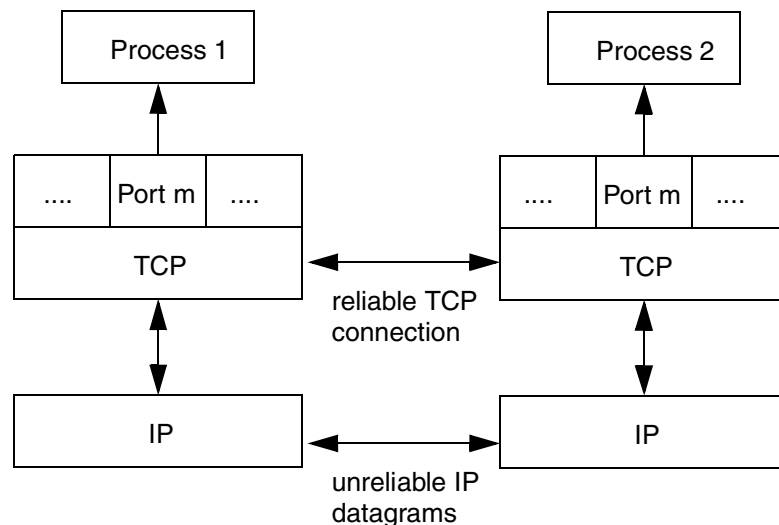


Figure 14. TCP - connection between processes

In this example, processes 1 and 2 communicate over a TCP connection carried by an IP datagram.

TCP does not recognize any application's data patterns but treats data as a stream of bytes that the sending application writes to a buffer known as the TCP window. The following steps explain the TCP operation in a simplified way:

- During connection establishment, the two hosts agree on an initial window size for that connection. TCP on the sending system then fills up the window with application data, chops its content into packets that conveniently fit into IP datagrams, and passes those on to IP to be submitted to the receiver.
- TCP on the receiving system puts the packets back in order, sends an acknowledgment for every two packets received in order, and fills up a receive buffer (window) from which the receiving application retrieves the incoming data stream.
- TCP on the sending system advances the window for as many packets as have been acknowledged (in order) and continues to transmit packages until the end of the data stream is reached. If packets are not acknowledged, TCP will wait and retransmit them before notifying the application of a communications problem. While TCP awaits acknowledgments, the window cannot be advanced and packet transfer will slow down.
- TCP provides flow control during an ongoing communication by varying send and receive window sizes at each transfer as required, which is important if a system runs out of memory for buffer space, or when a very fast system is communicating with a very slow system.
- TCP also provides for urgent data (interrupt signaling) to travel ahead of communication traffic on the current connection.
- When the sending application terminates, normally or abnormally, TCP will try to gracefully shut down the connection.

---

## 1.9 TCP/IP application protocols

One of the reasons why TCP/IP is so popular is that there are many simple and useful standard applications available. We summarize several common TCP/IP applications in this section.

### 1.9.1 Remote login and terminal emulation: Telnet

Teletypewriter network, or Telnet, is the virtual terminal protocol in TCP/IP. It allows users of one host to log in to a remote host and interact as normal terminal users of that host. Telnet is a line-oriented protocol using the ASCII character set. Though initially designed for terminal emulation, Telnet is also used as the underlying protocol for file transfer (FTP control sessions).

For readers who are familiar with SNA, we can relate Telnet in TCP/IP to the terminal emulators (3270 or 5250 types) in SNA. In fact, all of the IBM TCP/IP product implementations provide Telnet support of 3270 terminal emulation in addition to the many other terminal emulation protocols, such as the widely used DEC VT terminal emulation types.

The 3270 terminal emulation differs from normal Telnet operation in the following ways:

- It uses block mode rather than line mode.
- It uses EBCDIC character set rather than ASCII character set.
- It uses special key functions such as ATTN and SYSREQ.

A 3270 Telnet (TN3270) server must support those characteristics during initial client/server session negotiations. TN3270 sessions can represent either display or printer devices.

### **1.9.2 File Transfer Protocol: FTP**

File Transfer Protocol (FTP) provides the functions of transferring files between two TCP/IP hosts. Since FTP is built on the services of TCP in the transport layer, it provides a reliable and end-to-end connection during the file transfer operation. Security is provided by the normal user ID and password authentication.

### **1.9.3 Remote printing: LPR and LPD**

The Line Printer Requester (LPR) allows access to printers on other computers running the Line Printer Daemon (LPD) as though they were on your computer. The clients provided (LPR, LPQ, LPRM or LPRMON or LPRPORTD) allow the user to send files or redirect printer output to a remote host running a remote print server (LPD). Some of these clients can also be used to query the status of a job, as well as to delegate a job. For more information about remote printing, see RFC 1179.

### **1.9.4 Domain Name System (DNS)**

Recall that TCP/IP hosts are addressed by 32-bit IP addresses that are represented in decimal notation. For example, from Telnet to a remote host with an IP address of 9.67.38.1, the user would typically enter Telnet 9.67.38.1. This approach can be cumbersome and error prone, so a method was developed to use symbolic high-level machine names that are more meaningful to users than IP addresses.

This introduced the problem of maintaining the mappings between IP addresses and high-level machine names in a coordinated and centralized way. Initially, host names to address mappings were maintained by the Internet Network Information Center (NIC) in a single file (HOSTS.TXT), which was reached by all hosts using FTP. Most hosts would have a copy of that file, which may or may not have been current or correct.

Due to the explosive growth in the number of hosts, this mechanism became too complicated and time-consuming, and was replaced by a new concept: the Domain Name System (DNS).

The domain concept lies in decentralizing the naming mechanism by distributing responsibility (and authority) for mapping between names and addresses. For example, consider the internal structure of a large organization. As the chief executive cannot do everything, the organization will probably be partitioned into divisions, each of them having autonomy within certain limits. Specifically, the executive in charge of a division has authority to make direct decisions, without permission from his or her chief executive.

Domain names are formed in a similar way and will often reflect the hierarchical delegation of authority used to assign them. For example, consider the name:

small.itso.raleigh.ibm.com

Here, itso.raleigh.ibm.com is the lowest-level domain name, a subdomain of raleigh.ibm.com, which again is a subdomain of ibm.com, a subdomain of com. We can also represent this naming concept by a hierarchical tree (see Figure 15):

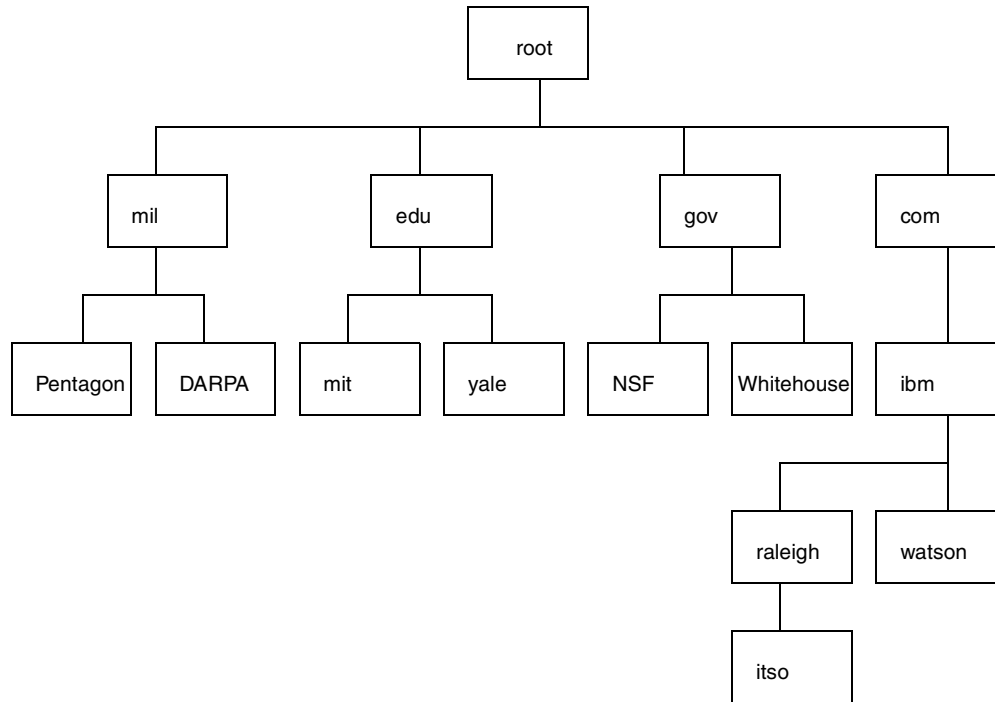


Figure 15. Hierarchical namespace (chain of authority in assigning domain names)

Table 7 shows some of the top-level domains of today's Internet domain namespace:

Table 7. DNS - some top-level Internet domains

| Domain name  | Meaning   |
|--------------|---|
| com          | Commercial organizations  |
| edu          | Educational institutions  |
| gov          | Government Institutions   |
| mil          | U.S. Military   |
| net          | Major network support centers                                   |
| org          | Nonprofit organizations   |
| country code | ISO standard two-letter identifier for country-specific domains |

#### 1.9.4.1 Mapping domain names to IP addresses

The mapping of names to addresses consists of independent, cooperative systems called name servers. A name server is a server program that holds a master or a copy of a name-to-address mapping database, or otherwise points to

a server that does, and that answers requests from the client software, called a name resolver.

Conceptually, all Internet domain servers are arranged in a tree structure that corresponds to the naming hierarchy in Figure 15 on page 29. Each leaf represents a name server that handles names for a single subdomain. Links in the conceptual tree do not indicate physical connections. Instead, they show which other name server a given server can contact.

Figure 16 graphically depicts the domain name resolution process that is summarized in the following steps:

- A user program issues a request such as the `gethostbyname()` sockets call. (This particular call is used to ask for the IP address of a host by passing the host name.)
- The resolver formulates a query to the name server. (Full resolvers have a local name cache to consult first, stub resolvers do not.)
- The name server checks to see if the answer is in its local authoritative database or cache, and if so, returns it to the client. Otherwise, it will query other available name server(s), starting down from the root of the DNS tree or as high up the tree as possible.
- The user program will finally be given a corresponding IP address (or host name, depending on the query) or an error if the query could not be answered. Normally, the program will not be given a list of all the name servers that have been consulted to process the query.

The query/reply messages are transported by either UDP or TCP. DNS is conceptually defined in RFC 1034 and RFC 1035.

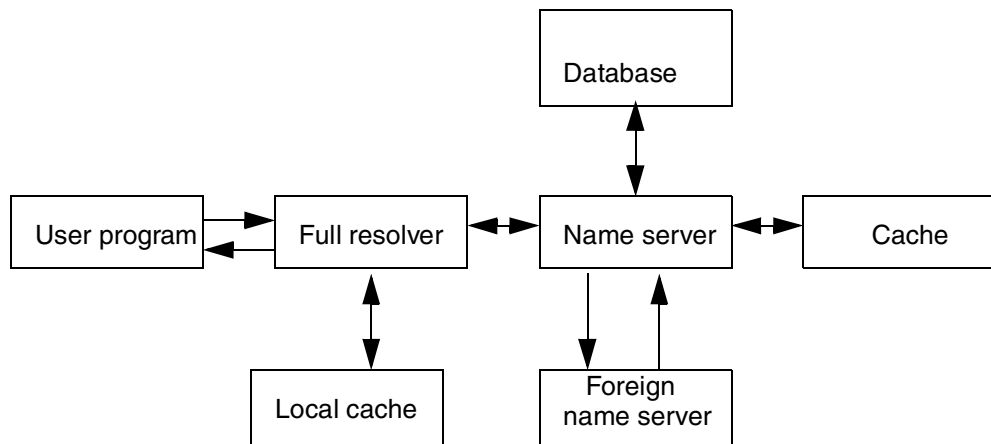


Figure 16. DNS - resolver and domain name server

#### 1.9.4.2 Reverse mapping

In some cases, it may be necessary to find a host name for a given IP address. This is called reverse mapping and is a standard function of most DNS servers available today. A special domain, `in-addr.arpa`, is being used for inverse name queries.



### 1.9.5 Network File System (NFS)

The Network File System (NFS) enables machines to share file systems across a network. It allows authorized users to access files located on remote systems as if they were local. It is designed to be machine independent, operating system independent, and transport protocol independent.

---

## 1.10 Internet user applications and protocols

This section provides a brief overview of some of the protocols and applications that have made the task of using the Internet both easier and very popular over the past couple of years.

### 1.10.1 Gopher

Gopher is a client/server protocol designed for information location and retrieval. The client function provides a menu-driven interface to access the files stored on a Gopher server. The server function allows descriptive names to be assigned to the files, making it easier to identify the content of each file. Gopher was designed at the University of Minnesota. For more information about Gopher, see RFC 1436.

### 1.10.2 The World Wide Web (WWW)

The World Wide Web is a global hypertext system that was initially developed in 1989 by Tim Berners Lee at the European Laboratory for Particle Physics, CERN in Switzerland to provide an easy way of sharing and editing research documents among a geographically dispersed group of scientists.

In 1993 the Web started to grow rapidly, which was mainly due to the National Center for Supercomputing Applications (NCSA) developing a Web browser program called Mosaic, an X Windows-based application. This application provided the first graphical user interface to the Web and made browsing more convenient.

Today there are Web browsers and servers available for nearly all platforms. You can get them either from an FTP site for free or buy a licensed copy. The rapid growth in popularity of the Web is due in part to the flexible way people can navigate through world-wide resources on the Internet and retrieve them. Table 8 presents some statistics about the growth of the Web.

The number of Web servers is also increasing rapidly and the traffic over port 80, which is the well-known port for HTTP Web servers on the NSF backbone, has had a phenomenal rate of growth, too. The NSFNET was converted back to a private research network in 1995; therefore, comprehensive statistics of backbone traffic are not as easily available anymore, if they are at all.

*Table 8. Growth of the World Wide Web*

| Date       | Web sites | Web traffic (%) | FTP traffic (%) | E-mail traffic (%) |
|------------|-----------|-----------------|-----------------|--------------------|
| June 1993  | 130       | 0.5             | 42.9            | 6.4                |
| June 1994  | 2,738     | 6.1             | 35.2            | 6.4                |
| March 1995 | --        | 23.9            | 24.2            | 4.9                |

| Date          | Web sites | Web traffic (%) | FTP traffic (%) | E-mail traffic (%) |
|---------------|-----------|-----------------|-----------------|--------------------|
| June 1995     | 23,000    | --              | --              | --                 |
| June 1996     | 230,000   | --              | --              | --                 |
| June 1997     | 1,203,096 | --              | --              | --                 |
| June 1998     | 2,594,622 | --              | --              | --                 |
| June 1999     | 6,177,453 | --              | --              | --                 |
| December 1999 | 9,560,866 | --              | --              | --                 |

### 1.10.3 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is a protocol designed to allow the transfer of Hypertext Markup Language (HTML) documents. HTML is a tag language used to create hypertext documents. Hypertext documents include links to other documents that contain additional information about the highlighted term or subject. Such documents may contain other elements apart from text, such as graphics, audio, and video clips, and even virtual reality worlds (which are described in VRML, a scripting language for that kind of element).

HTTP is based on request-response activity. A client, running an application called a browser, establishes a connection with a server and sends a request to the server in the form of a request method. The server responds with a status line including the message's protocol version and a success or error code, followed by a message containing server information, entity information, and possible body content.

An HTTP transaction is divided into four steps:

- The browser opens a connection.
- The browser sends a request to the server.
- The server sends a response to the browser.
- The connection is closed.

On the Internet, HTTP communication generally takes place over TCP connections. The default port is TCP 80, but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used, but the mapping of the HTTP request and response structures onto the transport data units of the protocol in question is outside the scope of this document.

Except for experimental applications, current practice requires that the connection be established by the client prior to each request and closed by the server after sending the response. Both clients and servers should be aware that either party may close the connection prematurely, due to user action, automated timeout, or program failure, and should handle such closing in a predictable fashion. In any case, the closing of the connection by either or both parties always terminates the current request, regardless of its status.

What we have just described means that, in simple terms, HTTP is a stateless protocol because it keeps no track of the connections. To load a page including two graphics for example, a graphic-enabled browser will open three TCP connections: one for the page, and two for the graphics. Most browsers, however, are able to handle several of these connections simultaneously.

This behavior can be resource-intensive if one page consists of a lot of elements as quite a number of Web pages nowadays do. HTTP 1.1, as defined in RFC 2068, alleviates this problem to the extent that one TCP connection will be established per type of element on a page, and all elements of that kind will be transferred over the same connection.

However, if a request depends on the information exchanged during a previous connection, then this information has to be kept outside the protocol. One way of tracking such persistent information is the use of cookies. A cookie is a file that is sent to a client by a server to be filled in with information about the client and/or the user at the client. It can be retrieved and checked by the server at any subsequent connection. Because cookies are regarded as a potential security hole, a Web browser should allow the user to decide whether to accept cookies or not. A more secure way of client and server authentication is provided by the Secure Sockets Layer (SSL), which is described in 1.11.1, “Secure Sockets Layer (SSL)” on page 35.

#### **1.10.4 The advent of Java**

Java is an important new technology in the world of the Internet. In summary, it is a simple, robust, object-oriented, platform-independent, multithreaded, dynamic general-purpose programming environment for creating applications for the Internet and intranet. Java includes the following components:

##### **1.10.4.1 Java language**

Java is a programming language developed by Sun Microsystems, which is object oriented, distributed, interpreted, architecture neutral, and portable. Java can be used to create downloadable program fragments, so-called applets, that augment the functionality of a Java-capable browser such as HotJava or Netscape Navigator.

##### **1.10.4.2 Java Virtual Machine**

The Java Virtual Machine (JVM) is an abstract computer that runs compiled Java programs (or precisely, that interprets Java byte-code that has been produced by a Java compiler). JVM is virtual because it is generally implemented in software on top of a real hardware platform and operating system. In this way, it is architecture neutral and platform independent. All Java programs should be compiled to run in a JVM.

##### **1.10.4.3 Programs, applets, and servlets**

When a Java program is started from inside an HTML (Web) page, it is called a Java applet as opposed to a Java program that is executed from the command line or otherwise on the local system. Applets are downloaded via the Web browser from a server and, by definition, are somewhat limited in the way they can use resources of the local system.

Originally, no Java applet was supposed to touch anything locally outside of its JVM and could only communicate back to the server from where it was

downloaded. With Java 1.1, applets can be signed with security keys and certificates and can therefore be authenticated. Thus, an applet can be authorized to access local resources, such as file systems, and it may communicate with other systems.

To spare resources on clients and networks, Java applets can be executed on the server rather than downloaded and started at the client. Such programs are then referred to as servlets. Though that method requires a significantly more powerful server, it is highly suitable for environments with medialess systems, such as network computers.

#### **1.10.4.4 HotJava**

HotJava is a Java-enabled Web browser, developed by Sun Microsystems, which lets you view Java applets.

#### **1.10.4.5 JavaOS**

JavaOS is a highly compact operating system, developed by JavaSoft, which is designed to run Java applications directly on microprocessors in anything from personal computers to pagers. JavaOS will run equally well on a network computer, a PDA, a printer, a game machine, or countless other devices that require a very compact OS and the ability to run Java applications.

#### **1.10.4.6 JavaBeans**

An initiative called JavaBeans is brewing a similar set of APIs that will make it easy to create Java applications from reusable components. JavaBeans will be used in a wide range of applications, from simple widgets to full-scale, mission-critical applications. Many software vendors including IBM have announced plans to support it.

#### **1.10.4.7 JavaScript**

JavaScript is an HTML extension and programming language, developed by Netscape, which is a simple object-based language compatible with Java. JavaScript programs are embedded as source directly in an HTML document. They can control the behavior of forms, buttons, and text elements. JavaScript is used to create dynamic behavior in the elements of a Web page. In addition, it can be used to create forms whose fields have built-in error checking routines. For more information about Java, check out the following URLs:

<http://www.ibm.com/javainfo/>

<http://www.java.sun.com/>

---

## **1.11 TCP/IP and Internet security**

In this section we briefly introduce some concepts and protocols that allow you to establish various degrees of security in TCP/IP networks.

You can say that the Internet is great because there is so much information that can be accessed very easily and quickly. Electronic communication has become a lot easier because of the Internet, but it can also be a dangerous at times.

Imagine that someone accesses your system and destroys data at random just because you forgot to implement security that could have prevented it. Or worse,

imagine someone taps into your communication, learns your passwords, and then uses your account information to electronically shop.

One of the major concerns when providing commercial services on the Internet is providing for transaction security and communications security.

Information exchanges are secure if all the following are true:

- Messages are confidential.
- The information exchange has integrity.
- Both sender and receiver are accountable.
- You can authenticate both parties in the exchange.

There are certainly other ways of compromising information on the Internet, so how should you go ahead to protect yourself against them?

### 1.11.1 Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) is a security protocol that was developed by Netscape Communications Corporation, along with RSA Data Security, Inc. The primary goal of the SSL protocol is to provide a private channel between communicating applications that ensures privacy of data, authentication of the partners, and integrity.

SSL provides an alternative to the standard TCP/IP socket API, which has security implemented within it. Hence, in theory it is possible to run any TCP/IP application in a secure way without changing it. In practice, SSL is so far only implemented for HTTP connections.

In fact, the protocol is composed of two layers:

- At the lower layer is the SSL Record protocol. It is used for data encapsulation.
- At the upper layer is the SSL Handshake protocol used for initial authentication and the transfer of encryption keys.

The SSL protocol addresses the following security issues:

#### ***Privacy***

After the symmetric key is established in the initial handshake, the messages are encrypted using this key.

#### ***Integrity***

Messages contain a message authentication code ensuring message integrity.

#### ***Authentication***

During the handshake, the client authenticates the server using an asymmetric or public key.

SSL requires each message to be encrypted and decrypted and therefore, has a high performance and resource impact. In addition, since only the server is authenticated, SSL is not suitable for applications, such as electronic banking, which require that the server authenticates its clients.

## 1.11.2 Firewalls

One way to deal with network security is to install a specialized server, a so-called firewall. Firewalls tend to be seen as a protection between the Internet and a private network. But generally speaking a firewall should be considered as a means to divide the world into two or more networks: one or more secure networks and one or more nonsecure networks.

Imagine a company where all the departments are connected to the internal network, including sales, accounts, development, and human resources departments. The administrator would like to be able to restrict access from the development department machines to the human resources department machines and from the sales department to the development department.

To provide maximum security, a good firewall design is paramount. The following factors should be considered in the firewall design process:

- Anything not explicitly permitted should default to denied.
- Increasing complexity leads to bugs, which lead to opportunities.
- The server should be kept in a physically secure environment.
- Provide extensive logging.
- Turn off known problems and nonessential daemons (applications and services).

Most of the firewalls available today offer one or more of the following services, some of which we briefly describe in the remainder of this section:

- Filtering gateways
- Proxy application layer gateways
- Circuit layer gateways (SOCKS servers)
- Domain name server hiding
- Mail handling
- Auditing and logging

Multiple technologies are needed to provide capabilities and protection. The IBM eNetwork Firewall, for instance, is based on IBM technology and has been used for more than 10 years to protect internal IBM IP networks.

### 1.11.2.1 Screening filters

The screening filter looks at each IP packet flowing through it, controlling access to machines and/or ports in the private network and possibly limiting access from the private network to the Internet. Screening filters operate at the IP layer and cannot control access at the application layer.

### 1.11.2.2 Proxy servers

Proxy servers are used to control access to or from the private network relaying only acceptable communications from known users.

Users in the private network can access an application, such as FTP, in the proxy server using their usual utilities (clients). Users authenticate themselves to the proxy server and can then access the application on the desired machine in the public network. Proxy servers can also be used from the public network to access

applications in the private network, but this exposes login names and passwords to attackers in the public network.

### 1.11.2.3 SOCKS servers

SOCKS servers are like proxy servers without the requirement for double connections. With SOCKS, users can benefit from secure communication without needing to be aware that it is happening.

Users have to use new versions of applications called SOCKSified clients. The SOCKSified client code directs its requests to the SOCKS port on the firewall. Sessions are broken at the firewall, as they are with proxy servers. With SOCKS, however, the connection to the destination application is created automatically once the user is validated.

Both the client and the SOCKS server need to have SOCKS code. The SOCKS server acts as an application-level router between the client and the real application server. SOCKS V4 is for outbound TCP sessions only. It is simpler for the private network user but does not have secure password delivery, so it is not intended for sessions between public network users and private network applications. SOCKS V5 provides for several authentication methods and can therefore be used for inbound connections as well, but you should be cautious. SOCKS V5 also supports UDP-based applications and protocols.

The majority of Web browsers are SOCKSified and you can get SOCKSified TCP/IP stacks for many platforms. For additional information, refer to RFC 1928, 1929, 1961, and the following URL:

<http://www.socks.nec.com>

---

## 1.12 TCP/IP and Internet publications

For more detailed or advanced knowledge of TCP/IP, please refer to the following selected publications:

- *Internetworking with TCP/IP, Volume I, Principles, Protocols and Architecture*, Third Edition, Prentice-Hall, Inc., 1995, by Douglas E. Comer; ISBN 0-13-216987-8.
- *TCP/IP Tutorial and Technical Overview*, Sixth Edition, IBM Corp., 1998, GG24-3376-05, and Prentice-Hall, Inc., 1998, by Martin W. Murhammer, Orcun Atakan, Stefan Bretz, Larry R. Pugh, Kazunari Suzuki, David H. Wood; ISBN 0-13-020130-8.
- *IPng and the TCP/IP Protocols*, John Wiley & Sons, Inc., 1996, by Stephen A. Thomas; ISBN 0-471-13088-5.
- *IP Multicasting*, McGraw-Hill, 1999, by Marcus Goncalves and Kitty Niles; ISBN 0-07-913791-1.
- *The Request for Comments (RFCs)*

There are more than 2600 RFCs today. For those readers who want to keep up to date with the latest advances and research activities in TCP/IP, the ever-increasing number of RFCs and Internet drafts (IDs) is the best source of

this information. (See 1.3, "Internet standards and Request for Comments (RFC)" on page 2 for more details on RFCs.)

RFCs can be viewed or obtained online from the IETF Web page using the following URL:

<http://www.ietf.org/rfc.html>



---

## Chapter 2. Installation and implementation of TCP/IP for VSE/ESA

This book is written to assist you in implementing the IBM version of TCP/IP for VSE/ESA Version 1.4 on VSE/ESA 2.3 or 2.4. The chapters on using the various applications also apply to using TCP/IP for VSE 1.4 obtained directly from Connectivity Systems, Inc.

TCP/IP for VSE/ESA Version 1.3 comes already installed on VSE/ESA 2.3 and VSE/ESA 2.4; it is in the PRD1.BASE sublibrary. To upgrade to Release 1.4 and to put TCP/IP for VSE/ESA into production a number of steps must be accomplished; these are described in detail in this chapter.

---

### 2.1 Obtain TCP/IP for VSE/ESA 1.4 documentation

No hardcopy documentation is available from IBM for the TCP/IP for VSE/ESA product. However, you can view or print the manuals from several sources:

- IBM CD-ROM (SK2T-1336)

This CD-ROM will be mailed to every VSE/ESA Version 2 customer automatically.

- IBM VSE home page

<http://www.s390.ibm.com/vse>

At these sources you will find six books with the original program description of TCP/IP for VSE 1.4 from Connectivity Systems, Inc., the provider of the TCP/IP for VSE program, plus one manual describing the setup of the TCP/IP for VSE/ESA program IBM is providing.

The books describing TCP/IP for VSE/ESA 1.4 are as follows:

- *TCP/IP for VSE/ESA - IBM Program Setup and Supplementary Information*
- *TCP/IP for VSE Installation Guide, Release 1.4, Release 1.4*
- *TCP/IP for VSE User's Guide, Release 1.4, Release 1.4*
- *TCP/IP for VSE Commands, Release 1.4, Release 1.4*
- *TCP/IP for VSE Programmer's Reference, Release 1.4*
- *TCP/IP for VSE Messages and Codes, Release 1.4*
- *TCP/IP for VSE Optional Products, Release 1.4*

The book *TCP/IP for VSE/ESA - IBM Program Setup and Supplementary Information*, SC33-6601 replaces the former *TCP/IP for VSE/ESA User's Guide*, SC33-6601. All the available documentation for the TCP/IP for VSE/ESA 1.4 program is described in Chapter 1 of that publication.

The documentation is available in PDF format only. You can use the Adobe Acrobat Reader to view and print the documentation. If you do not already have an Acrobat Reader installed, or if you need information on installing and using an Acrobat Reader, see the Adobe Web site at:

<http://www.adobe.com>

Note that the TCP/IP for VSE/ESA product from IBM (product number 5686-A04) is in general the same as the product TCP/IP for VSE from Connectivity Systems, Inc. The differences and additional functions exploiting TCP/IP for VSE/ESA are described in *TCP/IP for VSE/ESA - IBM Program Setup and Supplementary Information*, SC33-6601. A detailed list of comparable service packs from CSI and APARs/PTFs from IBM can be found in Appendix A of SC33-6601.

---

## 2.2 Obtain and install TCP/IP for VSE/ESA Version 1.4

TCP/IP for VSE/ESA Version 1.4 is available from IBM as APAR PQ29053. The related PTF can be applied to any VSE/ESA 2.3 or 2.4 system. IBM TCP/IP for VSE/ESA Version 1.3 will automatically be replaced during installation of the PTF. If you already have TCP/IP from Connectivity Systems, Inc. installed, you must remove any sublibraries containing that product or its customization members from all of your LIBDEF chains; this will ensure you use only the IBM version of TCP/IP for VSE/ESA.

You should use the Apply PTFs dialog of the VSE/ESA Interactive Interface to generate the PTF installation job stream.

---

## 2.3 Create VSE library for TCP/IP for VSE/ESA installation-unique definitions

We believe it is a good idea to store all TCP/IP for VSE/ESA configuration definitions, temporary fixes, Web pages, and any other TCP/IP for VSE/ESA members, in a library separate from any IBM-supplied libraries. Using a separate VSE library, rather than system-provided or user-defined sublibraries in an IBM-supplied library, separates user data from IBM data. It also facilitates easy migration from one VSE release or version to another, such as VSE/ESA 2.3 to VSE/ESA 2.4, where Fast Service Upgrade (FSU) is not supported.

We created a separate VSE library (TCPIP) on a separate volume to hold all of the installation-unique members created during this project. We used a 100 cylinder simulated 3390 for the SYSWK2 volume.

Figure 17 on page 41 shows the job stream we used to define the VSAM catalog and space to hold the library for the TCP/IP for VSE/ESA specific members. Note that with VSE/ESA 2.3 and 2.4 the TCP/IP for VSE/ESA product itself is already preinstalled in library PRD1.BASE.

```

* $$ JOB JNM=TCPCAT,CLASS=0,DISP=D,NTFY=YES
// JOB TCPCAT DEFINE USER CATALOG
// EXEC IDCAMS,SIZE=AUTO
DEFINE USERCATALOG ( -
    NAME (TCPIP.CATALOG ) -
    TRACKS (30 ) -
    ORIGIN (15 ) -
    VOLUME (SYSWK2))
IF LASTCC NE 0 THEN CANCEL JOB
DEFINE SPACE ( -
    VOLUME (SYSWK2) -
    TRACKS (720)) -
    CATALOG (TCPIP.CATALOG )
DEFINE CLUSTER ( -
    NAME (VSAM.COMPRESS.CONTROL) -
    RECORDS (200 100) -
    SHAREOPTIONS (4 4) -
    RECORDSIZE (128 500) -
    VOLUMES (SYSWK2) -
    NOREUSE -
    KEYS (44 0) -
    TO (99366)) -
    DATA (NAME (VSAM.COMPRESS.CONTROL.@D@) -
        INDEX (NAME (VSAM.COMPRESS.CONTROL.@I@) -
            CATALOG (TCPIP.CATALOG )
        IF LASTCC NE 0 THEN CANCEL JOB
    /*
// OPTION STDLABEL=ADD
// DLBL TCPCAT, 'TCPIP.CATALOG', , VSAM
/*
// EXEC IESVCLUP,SIZE=AUTO
A TCPIP.CATALOG TCPCAT
/*
/&
* $$ EOJ

```

Figure 17. Define VSAM catalog and space for library TCPIP

Figure 18 on page 42 shows the job stream we used to create the VSE library and sublibraries in that VSAM space. We created the following sublibraries:

- TCPIP.CONFIG, to contain unique configuration members, such as the initialization member and the Telnet menu
- TCPIP.HTML, to contain sample HTML pages
- TCPIP.FIXES, to contain any temporary fixes that might be needed
- TCPIP.PRINT, used to permanently hold incoming print data for an LPD
- TCPIP.TEMP, used to temporarily hold incoming print data for an LPD and GPS
- TCPIP.XFER, used for FTP into and out of a VSE library

```

* $$ JOB JNM=TCPIPLIB,CLASS=0,DISP=D,NTFY=YES
// JOB TCPIPLIB DEFINE VSE LIBRARY FOR TCP/IP
// EXEC IDCAMS,SIZE=AUTO
DEFINE CLUSTER ( -
    NAME (VSE.TCPIP.LIBRARY ) -
    TRACKS (00000092 00000031 ) -
    SHAREOPTIONS (3) -
    RECORDFORMAT (NOCIFORMAT ) -
    VOLUMES (SYSWK2 ) -
    NOREUSE -
    NONINDEXED -
    TO (99366 ) ) -
    DATA (NAME (VSE.TCPIP.LIBRARY.@D@ ) ) -
    CATALOG (TCPIP.CATALOG ) ) -
IF LASTCC NE 0 THEN CANCEL JOB
/*
// OPTION STDLABEL=ADD
// DLBL TCPIP, 'VSE.TCPIP.LIBRARY', , VSAM, X
    CAT=TCPCAT, DISP=(OLD, KEEP)
/*
// EXEC IESVCLUP, SIZE=AUTO
A VSE.TCPIP.LIBRARY TCPIP TCPCAT OLD KEEP
/*
// EXEC LIBR, PARM='MSHP'
    DEFINE LIB=TCPIP REPLACE=YES
    DEFINE SUBL=TCPIP.CONFIG
    DEFINE SUBL=TCPIP.HTML
    DEFINE SUBL=TCPIP.FIXES
    DEFINE SUBL=TCPIP.TEMP
    DEFINE SUBL=TCPIP.XFER
    DEFINE SUBL=TCPIP.PRINT
/*
/&
* $$ EOJ

```

Figure 18. Define the TCPIP library and sublibraries

## 2.4 Obtain TCP/IP product keys from IBM and define to TCP/IP for VSE/ESA

An IBM product key is required for each of the following components of TCP/IP for VSE/ESA that you plan to use:

- Base Pak or Application Pak
- Network File System
- General Print Server

You must contact IBM to obtain the keys. The different keys for the Application Pak or Base Pak, NFS or GPS, will be delivered to the customer when the product is licensed. To license the TCP/IP for VSE/ESA product and its features, you have to use the normal IBM ordering process, for example, CFSW (the Software Configurator).

Figure 19 on page 43 shows the job stream used to catalog the product keys and customer information:

```

* $$ JOB JNM=TCPKEY,DISP=D,CLASS=0
// JOB TCPKEY      GENERATE TCP/IP PRODUCT KEY AND CUSTOMER INFO PHASES
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF PHASE,CATALOG=TCPIP.CONFIG
// OPTION CATAL
// EXEC ASMA90,SIZE=(ASMA90,50K)
        PRODKEY nnnn-nnnn-nnnn-nnnn-nnnn      BASE PAK OR APPLICATION PAK
        PRODKEY nnnn-nnnn-nnnn-nnnn-nnnn      NFS
        PRODKEY nnnn-nnnn-nnnn-nnnn-nnnn      GPS
        END

/*
// EXEC LNKEDT
// EXEC ASMA90,SIZE=(ASMA90,50K)
        CUSTDEF DEFINE,                                X
                NAME='user name',                        X
                NUMBER=Chnn-nnn-nnn
        END

/*
// EXEC LNKEDT
/&
* $$ EOJ

```

Figure 19. Installing the product key and customer identification

## 2.5 Determine the TCP/IP for VSE/ESA applications to be used

TCP/IP for VSE/ESA includes a number of applications:

- File Transfer Protocol (FTP)
- Telnet, including TN3270
- Line Printer Daemon/Line Printer Requestor (LPD/LPR)
- General Print Server (GPS)
- Network File System (NFS)
- Web Server Support (HTTP)
- Sockets Application Programming Interface (API)

You may be installing TCP/IP for VSE/ESA for only one or two applications, such as FTP and LPR; if so, you should only make definitions for what you need.

We will provide definitions for all of the applications, and give examples of using each.

## 2.6 Configure the hardware the VSE system is running on

This task generally will involve updating the processor IOCP to reflect any new adapters or control units attached to the system for TCP/IP for VSE/ESA to use. Examples would be the OSA-2, 3172, Cisco Router, or Bus-Tech NetShuttle.

If an IBM Integrated Server or an IBM PC Server (P/390) is used, as we did, the P/390 Configurator must be used to define an even/odd pair of addresses with a

device type of 3088 and using the LCS3172 device manager. You must also use MPTS under OS/2 to configure the LAN adapter, and if you are using Ethernet be sure to change the “Type of Ethernet driver support” from I to D under the 802.2 protocol.

The *TCP/IP for VSE Installation Guide, Release 1.4* has good descriptions of the setup required for all of the supported LAN attachments.

---

## 2.7 Configure the network to which the VSE system will be connected

This activity could take many forms, depending on the network, but generally it will consist of updating router tables to include the IP address for TCP/IP for VSE/ESA, and customizing clients on workstations and other remote hosts to include the new IP address.

---

## 2.8 Define the hardware and network environment to the S/390 hosts

Our project environment included VM/ESA and multiple VSE/ESA systems. Each system required definitions for TCP/IP.

### 2.8.1 Hardware and network configuration used in this project

The project environment used for developing this book is shown in Figure 20:

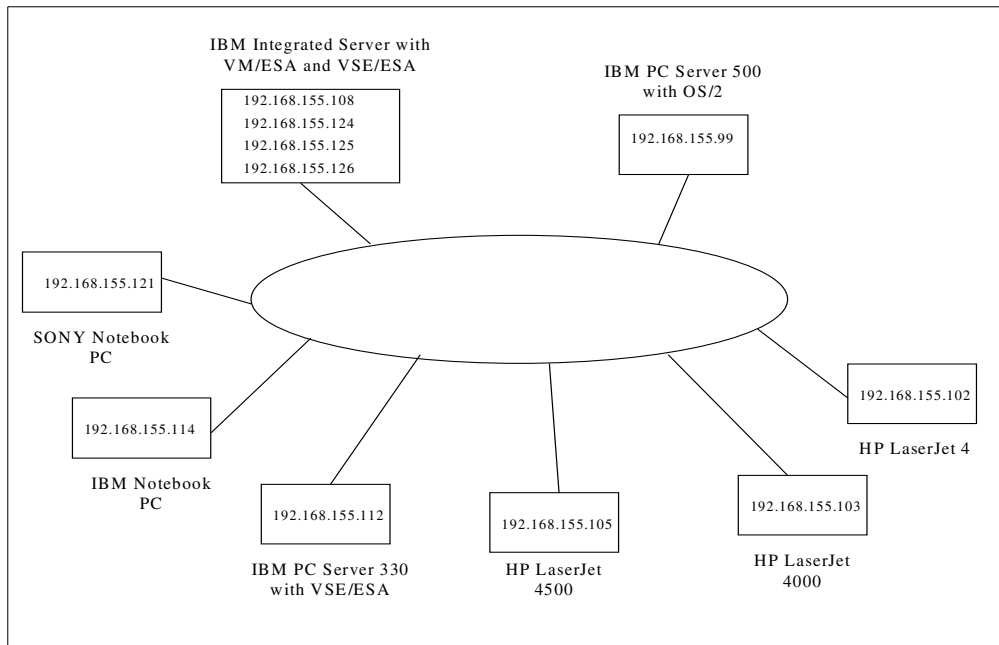


Figure 20. Our project network

### 2.8.2 Hardware and software details

The following section describes details of the hardware and software that we used in our project environment:

An Integrated Server with the following software:

- An Ethernet 10/100 Mb adapter was dedicated to the VSE/ESA 2.4 system, and virtual CTCAs were defined between the VSE240 guest and VM/ESA, and between the VSE240 guest and the VSE231 guest. Figure 21 shows the layout.

The diagram illustrates the architecture of a VM/ESA 2.3 system. At the top is the **VM/ESA 2.3** container. Below it, the system is divided into three main sections:

- VSE/ESA 2.3**: Contains **TCP/IP ID=00** with IP address **192.168.155.124**. It connects to the central VSE/ESA 2.4 via **CTCA** (Control Transfer Channel Adapter) with ports **600/601** and **403/402**.
- VSE/ESA 2.4**: The central component, containing:
  - TCP/IP ID=00 Z1** with IP address **192.168.155.108** and range **600 - 601**.
  - TCP/IP ID=07 F7** with IP address **192.168.155.126**.
  - IPNET** connecting the two TCP/IP stacks.
- TCP/IP for VM**: Contains IP address **192.168.155.125**. It connects to the central VSE/ESA 2.4 via **CTCA** with ports **401/400** and **610/611**.

At the bottom, an **Ethernet LAN adapter** is connected to the VSE/ESA 2.4 section via a **3172** interface.

Because this book is about using TCP/IP for VSE/ESA, we chose to dedicate the Ethernet adapter to VSE; in a normal VM/VSE environment, you would probably have TCP/IP for VM/ESA own the adapter.

A P/390 system with VSE/ESA 2.3 and TCP/IP for VSE 1.3 from Connectivity Systems. This system was used to demonstrate FTP, LPD/LPR, and Telnet between VSE systems.

A PC Server 500 with OS/2 WARP 4 Server Advanced and TCP/IP for OS/2. This system was used to demonstrate FTP and Telnet with an OS/2 client. The

VM/ESA and various VSE/ESA systems installed on this machine were not used for this project.

### **Windows 98 workstations**

Sony and IBM notebook PCs with Windows 98. The following additional software was installed:

- BlueZone FTP, a free graphical Windows FTP client from Renex Corporation
- SmartFTP-Daemon, a free graphical Windows FTP daemon by Jochen Letsch
- TN3270 Plus, a TN3270 client from SDI Bermuda LTD
- InterDrive Client (NFS for Windows 95 or Windows 98), an NFS client from FTP Software, Inc.

These were used to demonstrate FTP, Telnet, and NFS with Windows clients and servers.

### **HP LaserJet printers**

Three printers (HP LaserJet 4, HP LaserJet 4000, HP LaserJet 4500) with JetDirect cards. They were used to demonstrate LPR and auto-LPR.

## **2.8.3 VM/ESA definitions**

VM/ESA was installed from the starter CD-ROM that is available with an Integrated Server. Also installed from the CD-ROM was TCP/IP for VM/ESA. VM/ESA was customized for the Integrated Server environment, but no unique definitions were made for this project.

## **2.8.4 TCP/IP for VM/ESA guest machine (TCPIP) definitions**

Because we were running on an Integrated Server, we used the sample TCP/IP definitions that came with the VM/ESA starter system and customized them. The starter system includes:

- A sample directory entry for guest TCPIP
- Member SYS1 TCPIP D, which contains all of the environmental definitions for TCP/IP for VM/ESA
- Member SYSTEM DTCPARMS D, which defines the adapter interfaces to be used by TCP/IP for VM/ESA

### **2.8.4.1 Directory entry for TCP/IP for VM/ESA**

Figure 22 on page 47 shows our directory entry for the TCPIP guest machine. The only changes we made were to add the SPECIALs for the CTCAs.



```

USER TCPIP      TCPIP      32M      128M      ABG
* 5735-FAL TCP/IP TCPIP userid
  INCLUDE USERPROF
  ACCOUNT 55 P390
  OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
  SHARE RELATIVE 3000
  IUCV ALLOW
  IUCV ANY PRIORITY
  IUCV *CCS PRIORITY MSGLIMIT 255
  CONSOLE 009 3215 T OPERATOR
SPECIAL 610 CTCA
SPECIAL 611 CTCA
LINK TCPMAINT 198 198 RR
LINK TCPMAINT 591 591 RR
LINK TCPMAINT 592 592 RR
MDISK 191  FB-512 121216 6000      TCPIP  MR ALL

```

Figure 22. VM/ESA directory entry for TCPIP guest machine

#### 2.8.4.2 TCP/IP for VM/ESA definitions

The sample TCP/IP for VM/ESA member SYS1 TCPIP was updated to specify the virtual CTCA used to communicate with TCP/IP for VSE/ESA, and the IP address for TCP/IP for VM/ESA.

The highlighted entries in Figure 23 show the changes made to SYS1 TCPIP:

```

DEVICE VSECTCA CTC          610
LINK VSELINK CTC    0 VSECTCA
.
.
HOME
192.168.155.125 VSELINK
.
.
GATEWAY
192.168.155.108 =          VSELINK  4096          HOST
DEFAULTNET 192.168.155.108 VSELINK  DEFAULTSIZE  0
.
.
START VSECTCA

```

Figure 23. SYS1 TCPIP entries for TCP/IP for VM

The changes we made to SYSTEM DTCPARMS are highlighted in Figure 24 on page 48.

```

.
:nick.tcpiip :Type.server :Class.stack
               :Vctc.610 VSE240 401, 611 VSE240 400
.

```

Figure 24. SYSTEM DTCPARMS entries for TCP/IP for VM

## 2.8.5 VSE/ESA 2.4 guest machine (VSE240) definitions

### 2.8.5.1 Directory entry for VSE240

The highlighted entries in Figure 25 show the definitions added to the directory for VSE240 to support TCP/IP for VSE/ESA.

```

USER VSE240    VSE240    32M    32M    ABCDEFG
*   VSE/ESA 2.4.0
  INCLUDE SYSPROF
  ACCOUNT 93 P390
  OPTION CPUID 000004
IPL 150
  DEDICATE 150 150
  DEDICATE 151 151
  DEDICATE 152 152
  DEDICATE 600 600
  DEDICATE 601 601
  CONSOLE 01F 3270
  SPECIAL 400 CTCA
  SPECIAL 401 CTCA
  SPECIAL 402 CTCA
  SPECIAL 403 CTCA
  SPECIAL 200 3270
  SPECIAL 201 3270
  SPECIAL 202 3270
.

```

Figure 25. VM/ESA directory entry for VSE240 guest machine

The **OPTION** statement sets the CPU serial number to 4, and we used that in conjunction with an ASI master procedure to select the correct IPL and JCL procedures.

The **DEDICATE** statement for 152 is the DASD volume containing the VSE library we built for TCP/IP for VSE/ESA to use.

The **DEDICATE** statements for 600 and 601 make the Ethernet LAN adapter available to the VSE240 guest.

The **SPECIAL** statements for 400-403 create virtual CTCAs that are used to communicate from this VSE system to TCP/IP for VM/ESA and the VSE231 guest machine.

### 2.8.5.2 VSE240 guest machine installation

The VSE/ESA 2.4 system on the IBM Integrated Server was installed onto a simulated 3390-1 DASD using the normal install process.

TCP/IP for VSE/ESA Version 1.4 was installed using APAR PQ29053.

A set of IPL/JCL procedures with a suffix of 4 was created to define the desired environment (\$IPLESA4.PROC, \$0JCL4.PROC, USERBG4, and so forth). An IPL master procedure (\$ASIPROC.PROC) was created to automate IPLing with these suffixed procedures. Figure 26 shows the IPL master procedure we used.

```
* $$ JOB JNM=ASIMAST,DISP=D,CLASS=0
// JOB ASIMAST      CATALOG ASI MASTER PROC
// EXEC LIBR,PARM='MSHP'
CONN S=IJSYSRS.SYSLIB : PRD2.SAVE
COPY $ASIPROC.PROC
A S=IJSYSRS.SYSLIB
CATALOG $ASIPROC.PROC      R=Y
CPU=FF0000047490,IPL=$IPLESA4,JCL=$$JCL4
/+
/*
/&
* $$ EOJ
```

Figure 26. ASI master procedure \$ASIPROC.PROC

The IPL procedure (\$IPLESA4.PROC) was updated as shown in Figure 27 to add the 3390 DASD for the TCP/IP library, the virtual CTCAs used for communication between TCP/IP systems, and the “real” CTCA used for the emulated 3172.

```
.
ADD 152,ECKD
.
ADD 400:401,CTCA,EML      VIRTUAL CTCA TO VM/ESA
ADD 402:403,CTCA,EML      VIRTUAL CTCA TO VSE/ESA 2.3
.
ADD 600:601,CTCA          I/S LCS3172 FOR TCP/IP FOR VSE
.
```

Figure 27. Additions to the VSE240 IPL proc \$IPLESA4.PROC

Figure 28 on page 50 shows the commands added at the front of \$0JCL4.PROC to prevent VM/ESA from erasing the VSE console whenever a dialed terminal is disconnected, and to automate the coupling between this VSE system and TCP/IP for VM/ESA and the VSE231 guest:

```

.
* CP TERM BREAKIN MIN
* CP COUPLE 400 TCP/IP 611
* CP COUPLE 401 TCP/IP 610
* CP COUPLE 402 VSE231 601
* CP COUPLE 403 VSE231 600
STDOP ACANCEL=NO,DECK=NO,DUMP=PART,SYSDUMP=YES,SXREF=YES
.
.

```

Figure 28. Additions to the VSE240 JCL proc \$0JCL4.PROC for TCP/IP for VSE/ESA

USERBG4.PROC was updated to automatically release the TCP/IP for VSE/ESA startup jobs (TCPSZ1 and TCPSF7), and to set the partition priorities so the TCP/IP partitions would be a higher priority than any CICS or batch partitions, as shown in Figure 29.

```

.
PRTY BG=FA=F9=F8=F6=F5=F4,F2,F7,Z,FB,F3,F1
// PWR PRELEASE RDR,VTAMSTRT          START VTAM IF REQUIRED
// EXEC IESWAIT,PARM='03'
// PWR PRELEASE RDR,TCPSZ1            START TCP/IP IN Z1
// EXEC IESWAIT,PARM='03'
// PWR PRELEASE RDR,TCPSF7            START TCP/IP IN F7
// EXEC IESWAIT,PARM='03'
/. NOVAM
// PWR PRELEASE RDR,CICSICCF          START CICS .

```

Figure 29. Additions to USERBG4.PROC for TCP/IP for VSE/ESA

Dynamic class Z in the IBM-supplied dynamic class table was updated to contain only one partition of 20 MB for TCP/IP for VSE/ESA.

Partition usage was as follows:

- F1 - VSE/POWER
- F2 - CICS Transaction Server with VSE/ICCF (CICSICCF)
- F3 - ACF/VTAM
- F7 - Additional TCP/IP for VSE/ESA partition (ID=07), connected to primary TCP/IP for VSE/ESA partition Z1 (ID=0) using IPNET support
- FB - Basic Security Manager
- Z1 - Primary TCP/IP for VSE/ESA partition (ID=0) controlling the emulated 3172

### 2.8.5.3 TCP/IP for VSE/ESA startup jobs

TCP/IP for VSE/ESA runs in either static or dynamic partitions on a VSE system. The required size of this partition depends on how many of the FTP, Telnet, or other daemons you want to run. TCP/IP for VSE/ESA will run in as little as 8 to 10 MB, but to exploit the 31-bit support the partition must cross the 16 MB line.

For our test environment, we started the primary TCP/IP in dynamic partition Z1 with a size of 20 MB, and a secondary TCP/IP in static partition F7 with a size of 13 MB. If you have a large number of TCP/IP clients that want to access the VSE system concurrently or maybe make intensive use of VSE as a Web server, you might have to increase the partition size accordingly.

Figure 30 shows the startup job for the primary TCP/IP partition Z1:

```
* $$ JOB JNM=TCPSZ1,CLASS=Z,DISP=L
* $$ LST CLASS=X,DISP=L
// JOB TCPSZ1 START UP TCPIP IN PARTITION Z1
// LIBDEF *,SEARCH=(TCPIP.CONFIG,TCPIP.HTML,TCPIP.TEMP,TCPIP.XFER,      X
TCPIP.PRINT,TCPIP.FIXES,PRD1.BASE,PRD2.SCEEBASE)
// DLBL SAMFILE, 'VSAM.SAM.FTP.FILE1', 99/365,VSAM,CAT=VSESPUC,      X
RECORDS=(100,100),RECSIZE=80
// DLBL KSDSFIL, 'VSAM.KSDS.FTP.FILE1', ,VSAM,CAT=VSESPUC
// DLBL ESDSFIL, 'VSAM.ESDS.FTP.FILE1', ,VSAM,CAT=VSESPUC
// DLBL KSDSPRT, 'VSAM.KSDS.PRINT.FILE1', ,VSAM,CAT=VSESPUC
// DLBL ESDSPRT, 'VSAM.ESDS.PRINT.FILE1', ,VSAM,CAT=VSESPUC
// EXEC PROC=DTRICCF
// SETPFIX LIMIT=400K
// EXEC IPNET,SIZE=IPNET,PARM='ID=00,INIT=IPINIT04',DSPACE=4M
/&
* $$ EOJ
```

Figure 30. Startup job for primary TCP/IP partition Z1

Here is some additional information on several of the JCL statements:

**// LIBDEF \*,SEARCH=(TCPIP.CONFIG,TCPIP.HTML,TCPIP.TEMP,TCPIP.XFER,TCPIP.PRINT,TCPIP.FIXES, PRD1.BASE,PRD2.SCEEBASE)**

Identifies the VSE sublibrary search chain where various TCP/IP for VSE/ESA related members are cataloged:

**TCPIP.CONFIG** contains the configuration member IPINIT04.L, as well as any other installation-modified or created members.

**TCPIP.HTML** is the root library for the HTTP daemon, and contains all of the various HTML pages.

**TCPIP.TEMP** is used to temporarily store print members.

**TCPIP.XFER** is used to transfer VSE library members into and out of the VSE system using FTP.

**TCPIP.PRINT** is used to permanently store print members.

**TCPIP.FIXES** is normally empty but could contain any fixes above the supported IBM PTF level.

**PRD1.BASE** contains the TCP/IP for VSE/ESA product itself.

**PRD2.SCEEBASE** contains the C Language run-time support, which is used by the product code validation routine.

**// EXEC PROC=DTRICCF**

If FTP from the VSE/ICCF library is to be used, this JCL statement is required to identify to TCP/IP for VSE/ESA where the ICCF library is

located. DTRICCF is an IBM-supplied procedure containing the `ASSGN` statement for the ICCF library.

#### // SETPFIX LIMIT=400K

TCP/IP for VSE/ESA requires some PFIXed storage for operation. This statement is required to allow TCP/IP for VSE/ESA to PFIX up to the amount of storage specified; 400K is usually a good number to use.

#### // EXEC IPNET,SIZE=IPNET,PARM='ID=00,INIT=IPINIT04',DSPACE=4M

**SIZE=IPNET** sets the partition program area to the size of the IPNET phase. This leaves the rest of the partition for GETVIS.

**ID=00** identifies a specific TCP/IP for VSE/ESA partition; 00 is the default value. Each TCP/IP for VSE/ESA partition in a single VSE system must have a unique ID, so that clients and servers running in other partitions can identify to which TCP/IP for VSE/ESA partition to connect.

There should be one partition in any VSE system started with ID=00, because that is the default that all VSE clients use.

**INIT=IPINIT04** identifies the library initialization member to be used for this startup. This library member contains configuration commands for TCP/IP to execute during initialization.

In 2.8.5.4, “Customizing the IPINIT file for TCP/IP for VSE/ESA” on page 53 you will find a detailed description of our initialization member.

**DSPACE=4M** defines the maximum size of the data space that will be used by ACF/VTAM when communicating with TCP/IP for VSE/ESA for TN3270 and GPS sessions. For our test system, we used a DSPACE of 4 MB.

Just as important is the DSPACE value you specify in the ACF/VTAM startup job. For our test environment, we set the DSPACE parameter of our VTAM startup job to 4 MB.

It is better to specify a larger value for DSPACE and not need all of it than to specify a smaller value (1 MB or 2 MB) and need more. VTAM only acquires what it needs.

Figure 31 shows the startup job for the secondary TCP/IP partition F7:

```
* $$ JOB JNM=TCPSF7,CLASS=7,DISP=L
* $$ LST CLASS=X,DISP=L
// JOB TCPSF7 START UP TCPIP IN F7 PARTITION
// LIBDEF *,SEARCH=(TCPIP.CONFIG,TCPIP.HTML,TCPIP.TEMP,TCPIP.XFER,      X
                      TCPIP.PRINT,TCPIP.FIXES,PRD1.BASE,PRD2.SCEEBASE)
// EXEC PROC=DTRICCF
// SETPFIX LIMIT=400K
// EXEC IPNET,SIZE=IPNET,PARM='ID=07,INIT=IPINIT07',DSPACE=4M
/&
* $$ EOJ
```

Figure 31. Startup job for secondary TCP/IP partition F7

The major difference in this job is that the TCP/IP initialization parameters are different (ID=07 and INIT=IPINIT07).

#### 2.8.5.4 Customizing the IPINIT file for TCP/IP for VSE/ESA

All startup definitions of TCP/IP for VSE/ESA such as the IP address, the adapters, and links to be used, the file system and what daemons are to be started should be specified in the initialization file. While most commands can be entered at any time from the VSE console or from an executed .L member, it is far less error-prone and more convenient to include all of the commands in the startup member.

IBM delivers, with the product, the sample initialization member IPINIT00.L in PRD1.BASE. Its contents are shown in Figure 32, Figure 33 on page 54, and Figure 34 on page 55. Note that many of the commands included are actually comments. We recommend that, once you have created your own entries, you remove all of the commented entries; this will make the member considerably shorter and reduce confusion.

```
CATALOG IPINIT00.L          EOD=YY          REPLACE=YES
*-----*
*                               *
*      Define the constants    *
*                               *
*-----*
SET IPADDR   = 100.100.010.010
SET MASK     = 000.255.000.000
*
SET ALL_BOUND      = 30000
SET WINDOW         = 4096
SET TRANSFER_BUFFERS = 20
SET TELNETD_BUFFERS = 20
SET RETRANSMIT     = 100
SET DISPATCH_TIME  = 30
SET REDISPATCH    = 10
*
SET GATEWAY        = ON
*-----*
*                               *
*      Wait for VTAM Startup   *
*                               *
*-----*
WAIT          VTAM
*-----*
*                               *
*      Define the Communication Links    *
*                               *
*-----*
*  DEFINE LINK,ID=ICA_ETHERNET,TYPE=ETHERNET,DEV=500,MTU=1500
*
*  DEFINE LINK,ID=ICA_TOKEN-RING,TYPE=TOKEN_RING,DEV=800,MTU=1500
*
*  DEFINE LINK,ID=IBM_3172,TYPE=3172,DEV=200,MTU=1500
```

Figure 32. Standard IPINIT00.L delivered with TCP/IP for VSE/ESA (part 1 of 3)

```

*
* DEFINE LINK, ID=IBM_3172, TYPE=3172, DEV=200, MTU=1500
*
* DEFINE LINK, ID=P390_3172, TYPE=3172, DEV=200, MTU=1480
* DEFINE ADAPTER, LINKID=P390_3172, NUMBER=0, TYPE=ETHERNET
* DEFINE ADAPTER, LINKID=P390_3172, NUMBER=1, TYPE=TOKEN_RING
* DEFINE ADAPTER, LINKID=P390_3172, NUMBER=2, TYPE=FDDI
*
* DEFINE LINK, ID=PARTITION, TYPE=IPNET, SYSID=01, MTU=4096
*
* DEFINE LINK, ID=VSE_PART_F8, TYPE=CTCA, DEV=300, MTU=4096
*
* DEFINE LINK, ID=VM_TCPIP, TYPE=CTCA, DEV=300, MTU=4096
*
* DEFINE LINK, ID=MVS_TCPIP, TYPE=CTCA, DEV=302, MTU=4096
*-----*
*                               *
*       Define Routine Information       *
*                               *
*-----*
* DEFINE ROUTE, ID=LOCAL_NET, LINKID=ICA_ETHERNET, IPADDR=100.100.0.0
* DEFINE ROUTE, ID=MACINTOSH, LINKID=ICA_ETHERNET, IPADDR=100.100.50.50
* DEFINE ROUTE, ID=IBM486, LINKID=ICA_ETHERNET, IPADDR=100.100.100.100
* DEFINE ROUTE, ID=VMESA, LINKID=ICA_ETHERNET, IPADDR=100.100.90.90
* DEFINE ROUTE, ID=VMESA, LINKID=VM_TCPIP, IPADDR=100.100.90.90
* DEFINE ROUTE, ID=MVSESA, LINKID=MVS_TCPIP, IPADDR=100.100.80.80
* DEFINE ROUTE, ID=VSETCPIP, LINKID=VSE_PART_F8, IPADDR=100.50.0.0
*-----*
*                               *
*       Define Telnet Daemons       *
*                               *
*-----*
DEFINE TELNETD, ID=LU, TERMNAME=TELNLU, TARGET=DBDCCICS, PORT=23, COUNT=2
DEFINE TELNETD, ID=LU21, TERMNAME=TELNLU21, TARGET=DBDCCICS, PORT=23
DEFINE TELNETD, ID=LU22, TERMNAME=TELNLU22, TARGET=DBDCCICS, PORT=23
*-----*
*                               *
*       Define FTP Daemons       *
*                               *
*-----*
DEFINE FTPD, ID=FTP, PORT=21, COUNT=2
DEFINE FTPD, ID=FTP11, PORT=21
DEFINE FTPD, ID=FTP12, PORT=21
*-----*
*                               *
*       Line Printer Daemons       *
*                               *
*-----*
DEFINE LPD, PRINTER=FAST, QUEUE='POWER.LST.A'
DEFINE LPD, PRINTER=FASTLIB, QUEUE='PRD2.SAVE'
DEFINE LPD, PRINTER=LOCAL, QUEUE='POWER.LST.A', LIB=PRD2, SUBLIB=SAVE

```

Figure 33. Standard IPINIT00.L delivered with TCP/IP for VSE/ESA (part 2 of 3)



```

*-----*
*
*      Automated Line Printer Client      *
*
*-----*
DEFINE EVENT, ID=LST_LISTEN, TYPE=POWER, CLASS=X, QUEUE=LST, ACTION=LPR
*-----*
*
*      Setup the File System              *
*
*-----*
DEFINE FILESYS, LOCATION=SYSTEM, TYPE=PERM
*
DEFINE FILE, PUBLIC='IJSYSRS', DLBL=IJSYSRS, TYPE=LIBRARY
DEFINE FILE, PUBLIC='PRD1', DLBL=PRD1, TYPE=LIBRARY
DEFINE FILE, PUBLIC='PRD2', DLBL=PRD2, TYPE=LIBRARY
DEFINE FILE, PUBLIC='POWER', DLBL=IJQFILE, TYPE=POWER
*
MODIFY FILE, PUBLIC='VSE.SYSRES.LIBRARY', TYPE=LIBRARY
MODIFY FILE, PUBLIC='VSE.PRD1.LIBRARY', TYPE=LIBRARY
MODIFY FILE, PUBLIC='VSE.PRD2.LIBRARY', TYPE=LIBRARY
MODIFY FILE, PUBLIC='VSE.DUMP.LIBRARY', TYPE=LIBRARY
MODIFY FILE, PUBLIC='VSE.PRIMARY.LIBRARY', TYPE=LIBRARY
*
MODIFY FILE, PUBLIC='ICCF.LIBRARY', TYPE=ICCF
MODIFY FILE, PUBLIC='VSE.POWER.QUEUE.FILE', TYPE=POWER
*-----*
*
*      Setup member NETWORK.L to          *
*      execute once the engine has        *
*      been activated                     *
*
*-----*
INCLUDE NETWORK, DELAY
YY
/*

```

Figure 34. Standard IPINIT00.L delivered with TCP/IP for VSE/ESA (part 3 of 3)

We created our own initialization files IPINIT04.L and IPINIT07.L in sublibrary TCPIP.CONFIG. Initialization files must have the member type .L and the member name must be specified as the INIT= parameter in the EXEC IPNET statement of the TCP/IP startup job (unless the default name of IPINIT00.L is used).

Our initialization members contain commands to define:

- The overall environment, such as IP address, buffer options, message display options, and so forth
- The hardware and software adapter and link connections (3172, CTCA, and so forth)
- Network routing information
- The Telnet daemons for TN3270 access to ACF/VTAM applications

- The FTP daemons for file transfer
- Files in the file system for file transfer
- Users who are authorized to access the system
- Symbolic names of various remote hosts that can be used instead of actual IP addresses
- LPR clients
- Automatic LPR events
- HTTP daemon and CGI programs used for Web serving

Figure 35 on page 57, Figure 36 on page 58, Figure 37 on page 59, Figure 38 on page 60, and Figure 39 on page 61 show the IPINIT04.L we used for the primary TCP/IP for VSE/ESA during this project.

```

* $$ JOB JNM=IPINIT04,DISP=D,CLASS=0
// JOB IPINIT04    CATALOG TCP/IP IPINIT04.L
// EXEC LIBR
A S=TCPIP.CONFIG
CATALOG IPINIT04.L                                REPLACE=YES
*-----*
*                                           *
*           Define the constants           *
*                                           *
*-----*
SET IPADDR   = 192.168.155.108
SET MASK     = 255.255.255.224
*
SET ALL_BOUND      = 30000
SET WINDOW         = 4096
SET TRANSFER_BUFFERS = 20
SET TELNETD_BUFFERS = 20
SET RETRANSMIT     = 100
*
SET GATEWAY        = ON
SET DOWNCHECK      = OFF
SET SECURITY        = ON
*-----*
*                                           *
*           Wait for VTAM Startup           *
*                                           *
*-----*
WAIT      VTAM
*-----*
*                                           *
*           Define the Communication Links   *
*                                           *
*-----*
*
* P/390 SIMULATED 3172
*
DEFINE LINK,ID=IS3172,TYPE=3172,DEV=600,MTU=1492
DEFINE ADAPTER,LINKID=IS3172,NUMBER=0,TYPE=ETHERNET
*
* VSE240 F7 SECONDARY TCP/IP PARTITION
*
DEFINE LINK,ID=PARTF7,TYPE=IPNET,SYSID=07,MTU=4096,STOPPED
*
* VSE231 GUEST MACHINE UNDER VM/ESA
*
DEFINE LINK,ID=VSE231,TYPE=CTCA,DEV=402,MTU=4096,STOPPED
*
* VM/ESA 2.3 SYSTEM
*
DEFINE LINK,ID=VMESA,TYPE=CTCA,DEV=400,MTU=4096
*

```

Figure 35. IPINIT04.L as used in our environment (part 1 of 5)

```

*-----*
*
*          Define Alternate IP addresses
*
*-----*
DEFINE ALTIP, ID=VSE231, IPADDR=192.168.155.124
DEFINE ALTIP, ID=VMESA, IPADDR=192.168.155.125
DEFINE ALTIP, ID=PARTF7, IPADDR=192.168.155.126
*-----*
*
*          Define Routing Information
*
*-----*
DEFINE ROUTE, ID=VSE231, LINKID=VSE231, IPADDR=192.168.155.124
DEFINE ROUTE, ID=VMESA, LINKID=VMESA, IPADDR=192.168.155.125
DEFINE ROUTE, ID=PARTF7, LINKID=PARTF7, IPADDR=192.168.155.126
DEFINE ROUTE, ID=LOCAL, LINKID=IS3172, IPADDR=0.0.0.0
*-----*
*
*          Define Telnet Daemons
*          and Menu
*
*-----*
DEFINE TELNETD, ID=TRMA, TERMNAME=TELNA, PORT=5023, COUNT=5, TARGET=DBDCCICS
DEFINE TELNETD, ID=TRMB, TERMNAME=TELNB, PORT=23, COUNT=5, MENU=VTAM1
DEFINE MENU, ID=VTAM1, MEMBER=VTAM1
*-----*
*
*          Define FTP Daemons
*
*-----*
DEFINE FTPD, ID=FTP, PORT=21, COUNT=5, UNIX=YES
*-----*
*
*          Automated FTP Daemon
*
*-----*
DEFINE EVENT, ID=FTP_N, TYPE=POWER, CLASS=N, QUEUE=LST, ACTION=FTP
*-----*
*
*          Line Printer Daemons
*
*-----*
DEFINE LPD, PRINTER=FAST, QUEUE='POWER.LST.A'
DEFINE LPD, PRINTER=FASTLIB, QUEUE='TCPIP.PRINT'
DEFINE LPD, PRINTER=LOCAL, QUEUE='POWER.LST.A', LIB=TCPIP, SUBLIB=TEMP
DEFINE LPD, PRINTER=KSDS, QUEUE='KSDSTCP.PRINT', LIB=TCPIP, SUBLIB=TEMP
DEFINE LPD, PRINTER=ESDS, QUEUE='ESDSTCP.PRINT', LIB=TCPIP, SUBLIB=TEMP

```

Figure 36. IPINIT04.L as used in our environment (part 2 of 5)

```

*-----*
*
*      Automated Line Printer Client      *
*
*-----*
DEFINE EVENT, ID=LST_LISTEN, TYPE=POWER, CLASS=M, QUEUE=LST, ACTION=LPR
*-----*
*
*      Define LPR Scripts                  *
*
*-----*
DEFINE NAME, NAME=LPRSCR01, SCRIPT=LPRSCR01
*-----*
*
*      Define Symbolic Names              *
*
*-----*
DEFINE NAME, NAME=P500OS2, IPADDR=192.168.155.99
DEFINE NAME, NAME=NTSERVER, IPADDR=192.168.155.101
DEFINE NAME, NAME=HP4, IPADDR=192.168.155.102
DEFINE NAME, NAME=HP4000, IPADDR=192.168.155.103
DEFINE NAME, NAME=HP4500, IPADDR=192.168.155.105
DEFINE NAME, NAME=VSE240, IPADDR=192.168.155.108
DEFINE NAME, NAME=P330VSE, IPADDR=192.168.155.112
DEFINE NAME, NAME=PC1, IPADDR=192.168.155.114
DEFINE NAME, NAME=PC2, IPADDR=192.168.155.121
DEFINE NAME, NAME=VSE231, IPADDR=192.168.155.124
DEFINE NAME, NAME=VMESA, IPADDR=192.168.155.125
DEFINE NAME, NAME=PARTF7, IPADDR=192.168.155.126
*-----*
*
*      Define Users and Passwords         *
*
*-----*
DEFINE USER, ID=TCPA, -
+      PASSWORD=TWINKIE
DEFINE USER, ID=JUST, -
+      PASSWORD=TWINKIE
DEFINE USER, ID=JOHN, -
+      PASSWORD=TWINKIE
DEFINE USER, ID=SYSA, -
+      PASSWORD=TWINKIE
DEFINE USER, ID=SYSB, -
+      PASSWORD=TWINKIE
DEFINE USER, ID=SYSC, -
+      PASSWORD=TWINKIE
DEFINE USER, ID=SYSD, -
+      PASSWORD=TWINKIE

```

Figure 37. IPINIT04.L as used in our environment (part 3 of 5)

```

*-----*
*
*      Setup the File System
*
*-----*
DEFINE FILE, PUBLIC='TCPIP',          DLBL=TCPIP,   TYPE=LIBRARY
DEFINE FILE, PUBLIC='PRD2',          DLBL=PRD2,    TYPE=LIBRARY, -
      READONLY=YES
DEFINE FILE, PUBLIC='POWER',        DLBL=IJQFILE,  TYPE=POWER
DEFINE FILE, PUBLIC='ICCF',          DLBL=ICCF,    TYPE=ICCF
DEFINE FILE, PUBLIC='KSDSTCP.PRINT', DLBL=KSDSPRT, TYPE=KSDS
DEFINE FILE, PUBLIC='ESDSTCP.PRINT', DLBL=ESDSPRT, TYPE=ESDS
DEFINE FILE, PUBLIC='KSDSTCP.FILE',  DLBL=KSDSFIL, TYPE=KSDS
DEFINE FILE, PUBLIC='ESDSTCP.FILE',  DLBL=ESDSFIL, TYPE=ESDS
DEFINE FILE, PUBLIC='SAMTCP.FILE',   DLBL=SAMFILE, TYPE=SAM
DEFINE FILE, PUBLIC='IJSYSCT',       DLBL=IJSYSCT, TYPE=VSAMCAT, -
      READONLY=YES
DEFINE FILE, PUBLIC='VSESPUC',       DLBL=VSESPUC, TYPE=VSAMCAT, -
      READONLY=YES
DEFINE FILE, PUBLIC='TCPCAT',        DLBL=TCPCAT,  TYPE=VSAMCAT, -
      READONLY=YES
DEFINE FILE, PUBLIC='UCATF01',       DLBL=UCATF01, TYPE=VSAMCAT, -
      READONLY=YES
*-----*
*
*      Define NFS Daemon
*
*-----*
DEFINE NFSD, ID=NFSD, CONFIG=NFSCFG01
*-----*
*
*      Define GPS Daemon
*
*-----*
DEFINE GPSD, ID=GPS1, PRINTER=TEXT, IPADDR=HP4000, INSERTS=HP4LAND, -
      TERMNAME=GPSPT01, TARGET=DBCICCS, INSESS=YES, LOGMODE=DSC2K, -
      QUEUING=MEMORY, STORAGE='TCPIP.TEMP', RETRY_COUNT=3
DEFINE GPSD, ID=GPS2, PRINTER=TEXT, IPADDR=HP4500, INSERTS=HP4LAND, -
      TERMNAME=GPSPT02, TARGET=DBCICCS, INSESS=YES, LOGMODE=DSC2K, -
      QUEUING=MEMORY, STORAGE='TCPIP.TEMP', RETRY_COUNT=3
*-----*
*
*      Define HTTP Daemons
*
*-----*
DEFINE HTTPD, ID=HTTP1, ROOT='TCPIP.HTML', SECURE=NO, CONFINES=YES
DEFINE HTTPD, ID=HTTP2, ROOT='TCPIP.HTML', PORT=5080, SECURE=YES, CONFINES=NO

```

Figure 38. IPINIT04.L as used in our environment (part 4 of 5)

```

*-----*
*                                           *
*           Define CGI Programs           *
*                                           *
*-----*
DEFINE CGI,PUBLIC='ASMCGI1',TYPE=CGI
DEFINE CGI,PUBLIC='ASMCGI2',TYPE=CGI-BAL
DEFINE CGI,PUBLIC='REXXCGI1',TYPE=CGI-REXX
*-----*
*                                           *
*           Setup member NETWORK.L to     *
*           execute once the engine has   *
*           been activated                 *
*                                           *
*-----*
INCLUDE NETWORK,DELAY
/+
/*
/&
* $$ EOJ

```

Figure 39. IPINIT04.L as used in our environment (part 5 of 5)

Following are brief descriptions of the commands and parameters we used in our IPINIT04.L test environment. The full description of all the definitions in the IPINIT file can be found in the manual *TCP/IP for VSE Commands, Release 1.4*.

#### **SET IPADDR = 192.168.155.128**

This is the IP address of our primary TCP/IP for VSE/ESA partition (ID=00). If you work only on a local network you can select this IP address yourself, but when you want to connect outside to the Internet you need to register your network with the Internet Network Information Center (NIC) and be assigned your unique network ID number.

The addresses we used for this project are private class C addresses.

#### **SET MASK = 255.255.255.224**

The `SET MASK` command identifies what portion of the host number in a network address will be taken to identify a subnetwork. In our case 224 is B'11100000', indicating that the first three bits are the subnetwork number, and the last five bits are the host number.

During startup, TCP/IP for VSE/ESA will display the subnetwork and host number for this partition.

#### **SET ALL\_BOUND = 30000**

This command sets the maximum time that TCP/IP for VSE/ESA releases control of the CPU when it has no work to process.

We used 30000, which is the value specified in the sample IPINIT00 member. This value has little or no effect if the TCP/IP partition is active. A low value will cause excessive CPU usage, and a high value may cause delays if the partition activity is low. The default value is 9000 (30 seconds).

#### **SET WINDOW = 4096**

This command controls the amount of data that can be received from a remote host before it must wait for an acknowledgment.

We used 4096, which is the value specified in the sample IPINIT00 member. The value can range from 1500 through 65535, and the default is 8192.

#### **SET TRANSFER\_BUFFERS = 20**

This command sets the size of the buffer pool shared by all FTP daemons. The buffers are allocated in 31-bit partition GETVIS space, if available.

This parameter has a major effect on FTP performance, and thus on CPU consumption and I/O activity. A low value will lower CPU consumption and slow I/O activity, but file transfers will be much slower. A high value will allow file transfers to complete quickly, but they may consume most or all of the available CPU cycles.

We left this value unchanged. We were unable to do any performance measurements to see what effect changing this value would have.

#### **SET TELNETD\_BUFFERS = 20**

This command sets the size of the buffer pool used by Telnet daemons defined with POOL=YES. The buffers are allocated in 31-bit partition GETVIS space, if available. Each buffer is 16K.

We left this value unchanged.

#### **SET RETRANSMIT = 100**

This command sets the initial time interval for determining when a packet has been lost and must be retransmitted. If an acknowledgment of receipt is not received within the time interval, the packet is presumed lost and sent again; this can result in excessive network traffic. TCP/IP for VSE/ESA will dynamically adjust this value on each connection based on actual activity.

We left this value unchanged.

#### **SET GATEWAY = ON**

The `SET GATEWAY` command controls whether TCP/IP for VSE/ESA will forward transmissions between networks. We set this to ON because we were routing messages to a TCP/IP for VM/ESA system and a TCP/IP for VSE/ESA partition that were not connected to the physical network, and thus were acting as a gateway.

#### **SET DOWNCHECK = OFF**

The `SET DOWNCHECK` command controls whether TCP/IP for VSE/ESA shuts down immediately in response to the `SHUTDOWN` command, or whether it requires a confirmation message.

We used OFF because we were frequently starting and stopping TCP/IP during the project; for a production TCP/IP partition, we highly recommend ON to prevent accidental shutdown.

#### **SET SECURITY = ON**



The `SET SECURITY` command controls what security, if any, is provided for TCP/IP for VSE/ESA access.

ON means the remote user is prompted for a user ID and password when establishing a session. The user ID table (see the `DEFINE USER` command) is searched for a match, and access is allowed if they do. If a security exit is present (see the `DEFINE SECURITY` command), the data is also passed to the exit for verification, and the exit can overrule a match in the user ID table.

OFF is the default, which means that the remote user may be prompted for a user ID and password when establishing a session, but no attempt is made to validate the user ID or password.

## **WAIT VTAM**

This command causes TCP/IP for VSE/ESA startup to check that ACF/VTAM is active in the system and ready to accept logons, and wait if it is not.

We included this command because we have Telnet daemons defined that are communicating with VTAM, and did not want the daemons accepting input from remote clients without VTAM active.

**DEFINE LINK,ID=IS3172,TYPE=3172,DEV=600,MTU=1492**

**DEFINE ADAPTER,LINKID=IS3172,NUMBER=0,TYPE=ETHERNET**

**DEFINE LINK,ID=PARTF7,TYPE=IPNET,SYSID=07,MTU=4096,STOPPED**

**DEFINE LINK,ID=VSE231,TYPE=CTCA,DEV=402,MTU=4096,STOPPED**

**DEFINE LINK,ID=VMESA,TYPE=CTCA,DEV=400,MTU=4096**

The `DEFINE LINK` command is used to define a connection to a network connection device, such as an IBM 3172, OSA, or CTCA.

The `DEFINE ADAPTER` command identifies the specific adapter to be used on a 3172 or OSA. This command is valid only when it follows a previously issued `DEFINE LINK` command defining a 3172 or OSA.

**ID=IS3172:** This `DEFINE LINK/DEFINE ADAPTER` pair of commands defines the connection for TCP/IP for VSE/ESA via the emulated IBM 3172 Interconnect Controller of the Integrated Server using the subchannel pair 600/601. Our environment is an Ethernet connection; the maximum value for MTU for the Integrated Server is 1492 using Ethernet.

**ID=PARTF7:** This `DEFINE LINK` command defines the connection between the primary TCP/IP for VSE/ESA partition (ID=00) and the secondary partition (ID=07) using IPNET, the TCP/IP for VSE/ESA cross partition communication mechanism.

**ID=VMESA:** This `DEFINE LINK` command defines the connection of TCP/IP for VSE/ESA via a virtual channel-to-channel adapter (addresses 400 and 401) to TCP/IP for VM/ESA. The MTU size of 4096 is a good value to use for CTCA connections.

**ID=VSE231:** This `DEFINE LINK` command defines the connection of TCP/IP for VSE/ESA via a virtual channel-to-channel adapter (addresses 402 and 403) to TCP/IP for VSE in another VSE guest (VSE231) running under the same VM/ESA system.

We included the STOPPED option on several of the `DEFINE LINK` commands to demonstrate how to define a link but not actually start it. This can be useful if the other end of the link will frequently not be started at the same time as this end, because, if the link is started and fails to connect, TCP/IP for VSE/ESA will periodically retry the connection, and issue error messages on the VSE console for each failure.

**DEFINE ALTIP, ID=VSE231, IPADDR=192.168.155.124**

**DEFINE ALTIP, ID=VMESA, IPADDR=192.168.155.125**

**DEFINE ALTIP, ID=PARTF7, IPADDR=192.168.155.126**

These commands identify additional IP addresses that TCP/IP for VSE/ESA should listen for. In our case, since this TCP/IP partition owns the LAN adapter, we are the gateway for TCP/IP for VM/ESA, the TCP/IP for VSE/ESA running in partition F7, and the TCP/IP for VSE running in the VSE231 guest, and must listen for and process all messages addressed to their IP addresses as well as our own.

**DEFINE ROUTE, ID=VSE231, LINKID=VSE231, IPADDR=192.168.155.124**

**DEFINE ROUTE, ID=VMESA, LINKID=VMESA, IPADDR=192.168.155.125**

**DEFINE ROUTE, ID=PARTF7, LINKID=PARTF7, IPADDR=192.168.155.126**

**DEFINE ROUTE, ID=LOCAL, LINKID=IS3172, IPADDR=0.0.0.0**

The `DEFINE ROUTE` command is used to identify a route to an attached host. Although this information can be dynamically determined in many cases, specifying it will speed initialization and prevent potential routing failures. ID is a unique name that identifies this route.

Because we are listening for alternate IP addresses for TCP/IP for VM/ESA and two TCP/IP for VSE/ESA partitions, we need route statements to tell this TCP/IP partition what link to use in forwarding traffic destined for them.

The `ROUTE` command for ID=LOCAL is a generic route to specify that all traffic not destined for one of the preceding routes is to be sent on LINK IS3172; this will direct the traffic out onto the LAN.

**DEFINE TELNETD, ID=TRMA, TERMNAME=TELNA, PORT=5023, COUNT=5, TARGET=DBDCCICS**

**DEFINE TELNETD, ID=TRMB, TERMNAME=TELNB, PORT=23, COUNT=5, MENU=VTAM1**

The `DEFINE TELNETD` command defines and initializes one or more Telnet daemons (servers). Each TN3270 client that is to be concurrently supported requires a Telnet daemon.

These commands define two sets of Telnet daemons, one to connect directly to VTAM application DBDCCICS if traffic arrives for port 5023, and another to display a selection menu for traffic on port 23.

**DEFINE MENU, ID=VTAM1, MEMBER=VTAM1**

The `DEFINE MENU` command permits you to display a selection menu to Telnet users, allowing them to select to which VTAM application to connect.

We created a simple menu named VTAM1 to allow the selection of two VTAM applications, and to allow the entry of a user ID and password when security is turned on in the TCP/IP for VSE/ESA system. See 4.1.3, "Telnet menu definition" on page 135 for details on the menu.

#### **DEFINE FTPD,ID=FTP,PORT=21,COUNT=5,UNIX=YES**

The `DEFINE FTPD` command initiates one or more File Transfer Protocol daemons. You must provide one daemon for each concurrent FTP session.

We defined five daemons to allow five file transfers to occur simultaneously. We also started the daemons in UNIX mode so that any graphical FTP clients we ran would function correctly (graphical FTP clients all seem to be written expecting to communicate with TCP/IP on UNIX, and usually do not function correctly with the VSE FTP daemon unless `UNIX=YES` is specified).

#### **DEFINE EVENT,ID=FTP\_N,TYPE=POWER,CLASS=N,QUEUE=LST, ACTION=FTP**

The `DEFINE EVENT` command permits you to establish actions to be performed when specified events occur on your VSE system.

We defined an event that would perform an FTP transfer automatically whenever any class N output was put into the VSE/POWER LST queue. The script to be executed is determined by the `DEST=` parameter on the \*  
\$\$ LST statement of the job that puts the output into the queue.

#### **DEFINE LPD,PRINTER=FAST,QUEUE='POWER.LST.A'**

#### **DEFINE LPD,PRINTER=FASTLIB,QUEUE='TCPIP.PRINT'**

#### **DEFINE LPD,PRINTER=LOCAL,QUEUE='POWER.LST.A',LIB=TCPIP, SUBLIB=TEMP**

#### **DEFINE LPD,PRINTER=KSDS,QUEUE='KSDSTCP.PRINT',LIB=TCPIP, SUBLIB=TEMP**

#### **DEFINE LPD,PRINTER=ESDS,QUEUE='ESDSTCP.PRINT',LIB=TCPIP, SUBLIB=TEMP**

The `DEFINE LPD` command initiates a Line Printer Daemon (server). You must provide a definition for each virtual printer to be supported.

We defined several Line Printer Daemons to demonstrate receiving and storing print output from remote hosts directed to various destinations, such as a VSE/POWER queue, a VSE library, and VSAM files.

QUEUE is the location to receive the output routed to this daemon. This location must be a valid public name from the TCP/IP for VSE/ESA file system (see `DEFINE FILE`).

LIB and SUBLIB specify the library and sublibrary to be used for temporary storage of a file. If you have not defined these parameters, incoming data will be temporarily stored in memory. This means that the largest file that can be handled is limited to available memory.

#### **DEFINE EVENT,ID=LST\_LISTEN,TYPE=POWER,CLASS=M,QUEUE=LST, ACTION=LPR**

The `DEFINE EVENT` command permits you to establish actions to be performed when specified events occur on your VSE system.

We defined an event that would activate a Line Printer Requester to automatically print any class M output from the VSE/POWER LST queue.

#### **DEFINE NAME,NAME=LPRSCR01,SCRIPT=LPRSCR01**

This command allows you to define a script file in which you can specify LPR commands that are executed when printing a file with the automatic LPR. The name given in the NAME parameter references the name defined in the `POWER * $$ LST` statement as USER. The name specified in the SCRIPT parameter is the name of the VSE library member containing this script file.

We created script LPRSCR01.L to demonstrate using scripts. See 5.4.3, “Using scripts for the automatic LPR client” on page 162 for details.

**DEFINE NAME, NAME=P500OS2, IPADDR=192.168.155.99**

**DEFINE NAME, NAME=HP4, IPADDR=192.168.155.102**

**DEFINE NAME, NAME=HP4000, IPADDR=192.168.155.103**

**DEFINE NAME, NAME=HP4500, IPADDR=192.168.155.105**

**DEFINE NAME, NAME=VSE240, IPADDR=192.168.155.108**

**DEFINE NAME, NAME=P330VSE, IPADDR=192.168.155.112**

**DEFINE NAME, NAME=PC1, IPADDR=192.168.155.114**

**DEFINE NAME, NAME=PC2, IPADDR=192.168.155.121**

**DEFINE NAME, NAME=VSE231, IPADDR=192.168.155.124**

**DEFINE NAME, NAME=VMESA, IPADDR=192.168.155.125**

**DEFINE NAME, NAME=PARTF7, IPADDR=192.168.155.126**

The `DEFINE NAME` command can be used to give an IP address a symbolic name within TCP/IP for VSE/ESA. The specified name can be used in commands such as PING, FTP, and LPR.

We gave each of the TCP/IP hosts in our network a name, just to make it easier to refer to each in various commands.

**DEFINE USER,ID=TCPA, -  
+ PASSWORD=TWINKIE**

**DEFINE USER,ID=JUST, -  
+ PASSWORD=TWINKIE**

**DEFINE USER,ID=JOHN, -  
+ PASSWORD=TWINKIE**

**DEFINE USER,ID=SYSA, -  
+ PASSWORD=TWINKIE**

```
DEFINE USER,ID=SYSB, -  
+   PASSWORD=TWINKIE
```

```
DEFINE USER,ID=SYSC, -  
+   PASSWORD=TWINKIE
```

```
DEFINE USER,ID=SYSD, -  
+   PASSWORD=TWINKIE
```

The `DEFINE USER` command allows you to define who can access the TCP/IP for VSE/ESA system. The user ID and password are only checked if security has been turned on using the `SECURITY` command.

ID and PASSWORD can each be up to 16 characters.

The “+” at the beginning of each line containing PASSWORD causes TCP/IP for VSE/ESA not to print the contents of that line on SYSLST during startup; this provides some measure of security for the passwords.

We defined several users to demonstrate using security.

```
DEFINE FILE, PUBLIC='TCPIP',      DLBL=TCPIP,  TYPE=LIBRARY
```

```
DEFINE FILE, PUBLIC='PRD2',      DLBL=PRD2,  TYPE=LIBRARY, -  
      READONLY=YES
```

```
DEFINE FILE, PUBLIC='POWER',    DLBL=IJQFILE, TYPE=POWER
```

```
DEFINE FILE, PUBLIC='ICCF',      TYPE=ICCF
```

```
DEFINE FILE, PUBLIC='KSDSTCP.PRINT', DLBL=KSDSPRT, TYPE=KSDS
```

```
DEFINE FILE, PUBLIC='ESDSTCP.PRINT', DLBL=ESDSPRT, TYPE=ESDS
```

```
DEFINE FILE, PUBLIC='KSDSTCP.FILE', DLBL=KSDSFIL, TYPE=ESDS
```

```
DEFINE FILE, PUBLIC='ESDSTCP.FILE', DLBL=ESDSFIL, TYPE=ESDS
```

```
DEFINE FILE, PUBLIC='SAMTCP.FILE', DLBL=SAMFILE, TYPE=ESDS
```

```
DEFINE FILE, PUBLIC='IJSYSCT',    DLBL=IJSYSCT, TYPE=VSAMCAT, -  
      READONLY=YES
```

```
DEFINE FILE, PUBLIC='VSESPUC',    DLBL=VSESPUC, TYPE=VSAMCAT, -  
      READONLY=YES
```

```
DEFINE FILE, PUBLIC='TCPCAT',     DLBL=TCPCAT,  TYPE=VSAMCAT, -  
      READONLY=YES
```

```
DEFINE FILE, PUBLIC='UCATF01',    DLBL=UCATF01, TYPE=VSAMCAT, -  
      READONLY=YES
```

The `DEFINE FILE` command adds a file to the TCP/IP for VSE/ESA file system. The file can be a SAM file, a VSAM file or catalog, a VSE library, the POWER queues, or the ICCF library.

We defined a variety of files to the file system to use in demonstrating FTP and LPD.

Note that we did not use the `DEFINE FILESYS` command; we feel this command is a very dangerous command to use, particularly in a production environment, and strongly recommend it not be used.

#### **DEFINE NFSD,ID=NFSD,CONFIG=NFSCFG01**

This command defines the Network File System daemon. NFSCFG01 is the name of the configuration file.

```
DEFINE GPSD,ID=GPS1,PRINTER=TEXT,IPADDR=HP4000,  
    INSERTS=HP4LAND,TERMNAME=GPSVRT01,  
    TARGET=DBCDCICS,INSESS=YES,LOGMODE=DSC2K,  
    QUEUING=MEMORY,STORAGE='TCPIP.TEMP',  
    RETRY_COUNT=3
```

```
DEFINE GPSD,ID=GPS2,PRINTER=TEXT,IPADDR=HP4500,  
    INSERTS=HP4LAND,TERMNAME=GPSVRT02,  
    TARGET=DBCDCICS,INSESS=YES,LOGMODE=DSC2K,  
    QUEUING=MEMORY,STORAGE='TCPIP.TEMP',  
    RETRY_COUNT=3
```

These commands define the General Print Server daemons. There must be one daemon for each VTAM 3287-type printer you wish to emulate.

We created two daemons to support two printers.

```
DEFINE HTTPD,ID=HTTP1,ROOT='TCPIP.HTML',SECURE=NO,  
    CONFINE=YES
```

```
DEFINE HTTPD,ID=HTTP2,ROOT='TCPIP.HTML',PORT=5080,SECURE=YES,  
    CONFINE=NO
```

The `DEFINE HTTPD` command initiates an HTTP (Web) daemon (server).

We defined two daemons, one secured and one nonsecured, to demonstrate the difference.

```
DEFINE CGI,PUBLIC='ASMCGI1',TYPE=CGI
```

```
DEFINE CGI,PUBLIC='ASMCGI2',TYPE=CGI-BAL
```

```
DEFINE CGI,PUBLIC='REXXCGI1',TYPE=CGI-REXX
```

The `DEFINE CGI` command identifies the Common Gateway Interface (CGI) programs to be used by Web browsers.

We defined three programs to demonstrate using programs written in 24-bit assembler, 31-bit assembler, and REXX.

#### **INCLUDE NETWORK,DELAY**

The `INCLUDE` command causes TCP/IP to load a .L member from a VSE library and execute the contents as if they were part of the initialization member. The `DELAY` parameter causes the commands in the member not to be executed until TCP/IP for VSE/ESA initialization is complete.

We included the member NETWORK.L as an example. Our member contains a `WAIT` and a `PING` command; see Figure 40 on page 69.

```

* $$ JOB JNM=NETWORK,DISP=D,CLASS=0
// JOB NETWORK          TCP/IP NETWORK.L
// EXEC LIBR
A S=TCPIP.CONFIG
CATALOG NETWORK.L          REPLACE=YES
*-----*
*          Announce our presence on the network          *
*-----*
*
*          Wait for 20 seconds, to allow the system to complete startup
*
WAIT TIME 20
*
*          Issue a network ping
*
PING HP4000
/+
/*
/&
* $$ EOJ

```

Figure 40. Our NETWORK.L member

We think it is a very good idea to issue at least one PING to some remote host that is always on the network at the completion of TCP/IP for VSE/ESA startup. Successful completion of the PING provides a positive indication that we are talking to the network.

Figure 41 on page 70 shows part of the IPINIT07.L we used for the secondary TCP/IP during this project. We have included only the commands related to running as a secondary TCP/IP partition using the IPNET interface.

```

* $$ JOB JNM=IPINIT07,DISP=D,CLASS=0
// JOB IPINIT07    CATALOG TCP/IP IPINIT07.L
// EXEC LIBR
A S=TCPIP.CONFIG
CATALOG IPINIT07.L                                REPLACE=YES
.
SET IPADDR      = 192.168.155.126
SET MASK        = 255.255.255.224
.
SET GATEWAY      = OFF
.
DEFINE LINK,ID=PARTZ1,TYPE=IPNET,SYSID=00,MTU=4096
.
DEFINE ROUTE,ID=ALL,LINKID=PARTZ1,IPADDR=0.0.0.0
.
DEFINE TELNETD,ID=TRMC,TERMNAME=TELNC,PORT=23,COUNT=5,MENU=VTAM1
DEFINE MENU,ID=VTAM1,MEMBER=VTAM1
.
/+
/*
/&
* $$ EOJ

```

Figure 41. IPINIT07.L as used in our environment

The following commands in IPINIT07.L are significant for the secondary TCP/IP partition:

**SET IPADDR=192.168.155.126**

This command sets the IP address for the secondary partition.

**SET MASK=255.255.255.224**

Because we are on the same subnetwork, we used the same mask as for the primary partition.

**SET GATEWAY=OFF**

This partition is not being used as a gateway to another network, so we set gateway support to OFF.

**DEFINE LINK,ID=PARTZ1,TYPE=IPNET,SYSID=00,MTU=4096**

This command identifies the link back to the primary TCP/IP partition.

**DEFINE ROUTE,ID=ALL,LINKID=PARTZ1,IPADDR=0.0.0.0**

This command routes all traffic from this TCP/IP partition over the link to the primary TCP/IP partition, which will then route it to the correct remote host.

**DEFINE TELNETD,ID=TRMC,TERMNAME=TELNC,PORT=23,COUNT=5,  
MENU=VTAM1**

**DEFINE MENU,ID=VTAM1,MEMBER=VTAM1**



These commands define the Telnet daemons and the menu to be used in this partition. We changed the name of the Telnet daemon so it would use different VTAM APPL definitions from the ones used by the primary TCP/IP partition. The menu is the same one used in the other partition.

#### 2.8.5.5 ACF/VTAM definitions for TCP/IP for VSE/ESA

If you wish to access your VSE ACF/VTAM applications using TN3270 clients and Telnet, you must define one or more VTAM application IDs to be used as virtual terminal LU names. One ID (LUname) will be required for each concurrent Telnet session. Each application name will be referenced by a corresponding TCP/IP for VSE/ESA `DEFINE TELNETD` command. See 2.8.5.4, “Customizing the IPINIT file for TCP/IP for VSE/ESA” on page 53 and Figure 42 on page 71 to see how the names used must match.

We added TCPAPPL.B to our VTAM startup book, ATCCON00.B, to activate these definitions automatically at each VTAM startup.

```
* $$ JOB JNM=TCPAPPL,DISP=D,CLASS=0
// JOB TCPAPPL      CATALOG TCP/IP TCPAPPL.B
// EXEC LIBR
A S=PRD2.CONFIG
CATALOG TCPAPPL.B                      REPLACE=YES
TCPAPPL  VBUILD TYPE=APPL
TCPDIAG  APPL  AUTH=(ACQ)
TCPIP    APPL  AUTH=(ACQ)
TELNA01  APPL  AUTH=(ACQ)
TELNA02  APPL  AUTH=(ACQ)
TELNA03  APPL  AUTH=(ACQ)
TELNA04  APPL  AUTH=(ACQ)
TELNA05  APPL  AUTH=(ACQ)
TELNB01  APPL  AUTH=(ACQ)
TELNB02  APPL  AUTH=(ACQ)
TELNB03  APPL  AUTH=(ACQ)
TELNB04  APPL  AUTH=(ACQ)
TELNB05  APPL  AUTH=(ACQ)
TELNC01  APPL  AUTH=(ACQ)
TELNC02  APPL  AUTH=(ACQ)
TELNC03  APPL  AUTH=(ACQ)
TELNC04  APPL  AUTH=(ACQ)
TELNC05  APPL  AUTH=(ACQ)
/+
/*
/&
* $$ EOJ
```

Figure 42. VTAM application definitions for the Telnet daemons

To support using the GPS feature, we also created VTAM definitions in GPSAPPL.B for the printers; see 6.2.1, “VTAM definitions for GPS” on page 171 for details.

#### 2.8.5.6 CICS definitions for TCP/IP for VSE/ESA

TCP/IP for VSE/ESA includes several CICS-based clients. These clients provide CICS users with the ability to:

- Log on (from a CICS terminal) to other platforms and applications via Telnet, using the CICS transaction TELNET. For example, a user can log on to a UNIX system from CICS.
- Initiate a file transfer between the TCP/IP for VSE/ESA server and a remote FTP server using the CICS transaction FTP.
- Print on TCP/IP printers in the network using the Line Printer Requester CICS transaction LPR.

All required programs and CICS transactions come with TCP/IP for VSE/ESA.

To be able to use these CICS client transactions, all the required transactions and related programs must be defined to CICS:

- For VSE/ESA 2.4, IBM has included the definitions in the CICS CSD under the group name TCPIP. This group has also been included in VSELIST, the list used for CICSICCF startup. Thus, no setup is required unless you plan to use another list; in that case, be sure to add group TCPIP to your list.
- For VSE/ESA 2.3, the CICS CSD is not prepared to use the TCP/IP for VSE/ESA product. Therefore, you have to define the required programs and transactions to CICS/VSE as described in the following section.

Because CICS TS for VSE/ESA 1.1 (included in VSE/ESA 2.4) does not support CICS table definitions any longer a member IPNCSD.Z is shipped with TCP/IP for VSE/ESA, which uses CICS command definitions instead of CICS macro definitions. Additionally, a member IPNCSDUP.Z is available to use IPNCSD.Z for generating DFHCSD entries for TCP/IP for VSE/ESA on CICS/TS and CICS/VSE. Note that both jobs can also be used for CICS/VSE 2.3.

IPNCSDUP.Z looks like this:

```
* $$ JOB JNM=IPNCSDUP,CLASS=0,DISP=D
// JOB IPNCSDUP
* SHUT DOWN CICS FIRST
// PAUSE CLOSE DFHCSD FILE IF CICS IS UP : CEMT SE FI(DFHCSD) CLOSE
/*
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASE,PRD2.SCEEBASE)
// EXEC DFHCSDUP,SIZE=600K INIT AND LOAD CICS
DELETE GROUP(TCPIP)
* $$ SLI MEM=IPNCSD.Z,S=(PRD1.BASE)
ADD GROUP(TCPIP) LIST(VSELIST)
LIST ALL
/*
/&
* $$ EOJ
```

Figure 43. Member IPNCSDUP.Z, shipped with TCP/IP for VSE/ESA

Please note that the use of group TCPIP and list VSELIST is arbitrary. You may make any adjustments that your site requires.

The member IPNCSD.Z shipped with the TCP/IP for VSE/ESA product looks as shown in Figure 44 on page 73:

```

*-----*
*   FOLLOWING ARE THE PPT ENTRIES REQUIRED FOR TCP/IP for VSE/ESA   *
*-----*

DEFINE PROGRAM(TELNET01) GROUP(TCPIP)
LANGUAGE(ASSEMBLER)
DEFINE PROGRAM(FTP01) GROUP(TCPIP)
LANGUAGE(ASSEMBLER)
DEFINE PROGRAM(CLIENT01) GROUP(TCPIP)
LANGUAGE(ASSEMBLER)
*-----*
* FOLLOWING ARE THE PCT ENTRIES REQUIRED FOR TCP/IP for VSE/ESA *
*-----*

DEFINE TRANSACTION(PING) GROUP(TCPIP)
PROGRAM(CLIENT01)
DEFINE TRANSACTION(ping) GROUP(TCPIP)
PROGRAM(CLIENT01)
DEFINE TRANSACTION(TELN) GROUP(TCPIP)
PROGRAM(TELNET01)
DEFINE TRANSACTION(teln) GROUP(TCPIP)
PROGRAM(TELNET01)
DEFINE TRANSACTION(TELC) GROUP(TCPIP)
PROGRAM(TELNET01)
DEFINE TRANSACTION(TELOW) GROUP(TCPIP)
PROGRAM(TELNET01)
DEFINE TRANSACTION(TELR) GROUP(TCPIP)
PROGRAM(TELNET01)
DEFINE TRANSACTION(FTP) GROUP(TCPIP)
PROGRAM(FTP01)
DEFINE TRANSACTION(ftp) GROUP(TCPIP)
PROGRAM(FTP01)
DEFINE TRANSACTION(FTPC) GROUP(TCPIP)
PROGRAM(FTP01)
DEFINE TRANSACTION(FTPW) GROUP(TCPIP)
PROGRAM(FTP01)
DEFINE TRANSACTION(FTPR) GROUP(TCPIP)
PROGRAM(FTP01)
DEFINE TRANSACTION(TCPC) GROUP(TCPIP)
PROGRAM(CLIENT01)
DEFINE TRANSACTION(TCPW) GROUP(TCPIP)
PROGRAM(CLIENT01)
DEFINE TRANSACTION(TCPR) GROUP(TCPIP)
PROGRAM(CLIENT01)
DEFINE TRANSACTION(LPR) GROUP(TCPIP)
PROGRAM(CLIENT01)
DEFINE TRANSACTION(lpr) GROUP(TCPIP)
PROGRAM(CLIENT01)
*-----*
*                               END OF TCP/IP MEMBER                               *
*-----*

```

Figure 44. Member IPNCSD.Z, shipped with TCP/IP for VSE/ESA

With CICS/VSE the CICS CSD entries required for TCP/IP for VSE/ESA can also be created using the job stream shown in Figure 45 on page 74.

```

* $$ JOB JNM=CSDTCPIP,CLASS=0,DISP=D
// JOB CSDTCPIP GENERATE CSD-GROUP TCPIP
// LIBDEF *,SEARCH=(PRD1.BASE)
// EXEC DFHCSDUP
  DELETE ALL GROUP(TEMP)
  MIGRATE TABLE(DFHPPTIP) TOGROUP(TEMP)
  COPY GROUP(TEMP) TO(TCPIP)
  DELETE ALL GROUP(TEMP)
  MIGRATE TABLE(DFHPCTIP) TOGROUP(TEMP)
  COPY GROUP(TEMP) TO(TCPIP)
  DELETE ALL GROUP(TEMP)
  ADD GROUP(TCPIP) LIST(VSELIST)
/*
/&
* $$ EOJ

```

Figure 45. Job to generate DFHCSD entries for TCP/IP for VSE/ESA on CICS/VSE

After generating the DFHCSD entries, the new group must be installed in the active CICS system with the command `CEDA INSTALL GROUP(TCPIP)`. This can also be achieved by recycling CICS (be sure to perform a cold start).

## 2.8.6 VSE/ESA 2.3.1 guest machine (VSE231) definitions

Figure 46 shows the VM/ESA directory entry for VSE231. The only change we made was to add the CTCA entries.

```

USER VSE231  VSE231  256M  256M  ABCDEFG
*   VSE/ESA 2.3.1
  INCLUDE SYSPROF
  ACCOUNT 93 P390
  IPL 140
  DEDICATE 140 140
  DEDICATE 141 141
  DEDICATE 242 242
  DEDICATE 243 243
  CONSOLE 01F 3270
  SPECIAL 200 3270
  SPECIAL 201 3270
  SPECIAL 202 3270
  SPECIAL 203 3270
  SPECIAL 600 CTCA
  SPECIAL 601 CTCA

```

Figure 46. VM/ESA directory entry for the VSE231 guest machine

We added an entry for the CTCA connection to VSE240 in the \$IPLESA.PROC as shown in Figure 47:

```

ADD 600:601,CTCA,EML      CTCA FOR TCP/IP CONNECTION TO VSE240 GUEST

```

Figure 47. VSE231 IPL procedure entries for TCP/IP for VSE

The \$0JCL.PROC member was updated to include the following commands:

```
.
* CP TERM BREAKIN MIN
* CP COUPLE 600 VSE240 403
* CP COUPLE 601 VSE240 402
.
```

Figure 48. VSE231 JCL procedure entries for TCP/IP for VSE

The following definitions were made in TCP/IP for VSE in the VSE/ESA 2.3 (VSE231) system to enable communication with TCP/IP for VSE/ESA in the VSE/ESA 2.4 system (VSE240):

```
.
SET IPADDR    = 192.168.155.124
SET MASK      = 255.255.255.224
.
SET GATEWAY    = OFF
.
DEFINE LINK, ID=VSE240, TYPE=CTCA, DEV=600, MTU=4096
.
DEFINE ROUTE, ID=VSE240, LINKID=VSE240, IPADDR=0.0.0.0
.
DEFINE TELNETD, ID=LU, TERMNAME=TELNLU, TARGET=DBDCCICS, PORT=23, COUNT=5
.
DEFINE FTPD, ID=FTP, PORT=21, COUNT=5
.
DEFINE FILE, PUBLIC='POWER', DLBL=IJQFILE, TYPE=POWER
.
```

Figure 49. TCP/IP for VSE entries for the VSE231 guest machine

---

## 2.9 Install and/or customize TCP/IP applications on other systems

Many remote hosts will need some updating to communicate with TCP/IP for VSE/ESA. The following sections provide descriptions of what we did on each of the platforms.

### 2.9.1 Windows 98

Windows 98 (and Windows 95) includes a line mode Telnet client and a line mode FTP client. We added:

- A TN3270 client, described in 4.1.5, “Configuring a TN3270 client” on page 138
- A graphical FTP client, described in 3.2.1.2, “Graphical FTP clients” on page 88
- A graphical FTP daemon, described in 3.2.1.3, “Graphical FTP daemons” on page 90
- An NFS client, described in 7.4.1, “Configuring the NFS client” on page 191

When using a line mode client it is often easier to use symbolic names rather than IP addresses (for example, “ftp vse240” is generally easier to type and to remember than “ftp 192.168.155.108”). You can define a file named HOSTS (no file type) and include a list of symbolic names and the equivalent IP addresses there; we created the following file in the C:\WINDOWS directory:

```
# project hosts file
192.168.155.108 VSE240
192.168.155.125 VMESA
192.168.155.123 VSE231
192.168.155.99 P500OS2
192.168.155.101 NTSERVER
192.168.155.112 P330VSE
192.168.155.126 PARTF7
```

Figure 50. HOSTS file used on Windows 98

If your network already includes a Domain Name System (DNS), then the HOSTS file is not necessary; both the workstations and TCP/IP for VSE/ESA will contact the DNS to resolve the address. DNS support is available in TCP/IP for VSE/ESA with the usage of the SET DNS1 parameter. This is described in detail in *TCP/IP for VSE Commands, Release 1.4*.

## 2.9.2 OS/2

OS/2 Warp Server 4 includes a line mode Telnet client, a TN3270 client, a line mode FTP client and a graphical FTP client (FTP-PM), and a line mode LPR client. We used only the line mode clients for this project, and no customization was required.

---

## 2.10 Verifying system connectivity

Once you have created your configuration, startup job stream, and all of the other required definitions, and you bring up TCP/IP for VSE/ESA for the first time, it is very likely that some or all of the TCP/IP hosts will refuse to communicate with each other. TCP/IP for VSE/ESA, along with all other TCP/IP hosts, contains the PING command. This is an extremely valuable tool to help determine why TCP/IP systems will not communicate. Some systems, such as Windows 98, also contain the TRACERT (Traceroute) command, which can be used to determine the route a message takes from one host to another (TCP/IP for VSE/ESA does *not* contain a TRACERT command).

### 2.10.1 The PING command

The PING command is very simple:

```
PING IP-address|symbolic_name
```

You can issue this command from all TCP/IP hosts, and it will tell you whether you can communicate with the target host. For example, to ping from Windows 98 to our VSE240 system:

```

Microsoft(R) Windows 98
(C)Copyright Microsoft Corp 1981-1998.

C:\>ping vse240

Pinging VSE240 [192.168.155.108] with 32 bytes of data:

Reply from 192.168.155.108: bytes=32 time=58ms TTL=255
Reply from 192.168.155.108: bytes=32 time=56ms TTL=255
Reply from 192.168.155.108: bytes=32 time=56ms TTL=255
Reply from 192.168.155.108: bytes=32 time=56ms TTL=255

Ping statistics for 192.168.155.108:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 56ms, Maximum = 58ms, Average = 56ms

C:\>

```

Figure 51. Issuing the PING command in Windows 98

You can issue PING from the VSE console:

```

SYSTEM:  VSE/ESA                      VSE/ESA 2.4.0    TURBO (01)      USER:  SYSB
                                           TIME:  16:46:15

87 ping pc2
Z1 0085 TCP910I Client manager connection Established.
Z1 0085 TCP910I PING
Z1 0085 TCP910I PING Ready:
Z1 0085 TCP910I SET HOST= pc2
Z1 0085 TCP910I 192.168.155.121
Z1 0085 TCP910I PING Ready:
Z1 0085 TCP910I PING
Z1 0085 TCP910I PING 1 was successful, milliseconds:000100
Z1 0085 TCP910I PING 2 was successful, milliseconds:000160
Z1 0085 TCP910I PING 3 was successful, milliseconds:000170
Z1 0085 TCP910I PING 4 was successful, milliseconds:000570
Z1 0085 TCP910I PING 5 was successful, milliseconds:000190
Z1 0085 TCP910I PING Ready:
Z1 0085 TCP910I QUIT
Z1 0087 IPN300I Enter TCP/IP Command
Z1-0087
==>

1=HLP 2=CPY 3=END                      6=CNCL 7=BWD 8=FWD 9=EXPL 10=INP      12=INFO

FILTER:  ALL                          BWD                                MODE:  REDISPLAY

```

Figure 52. Issuing the PING command from the VSE console

There is also a CICS PING client:

```

PING PC2
TCP200I Client -- Startup --
TCP207I Copyright (c) 1995-1999 Connectivity Systems Incorporated
TCP202I Attempting to Establish Connection
TCP204I Connection has been Established
Client manager connection Established.
PING Service Client Selected.
192.168.155.121
PING 1 was successful, milliseconds:000090
PING 2 was successful, milliseconds:000090
PING 3 was successful, milliseconds:000100
PING 4 was successful, milliseconds:000080
PING 5 was successful, milliseconds:000080
PING Service Client Completed.
TCP201I Client -- Shutdown --
TCP205I Connection Complete -- Already Closed

```

Figure 53. CICS PING transaction

There is also a batch PING client you can use. How to use the batch PING client is described in the manual *TCP/IP for VSE User's Guide, Release 1.4*.

Just because you can ping in one direction does not mean that it will work in the other. You should very carefully use PING on each system to verify communication with every other system you need to communicate. This is especially true if you are using any gateways, routers, or bridges.

## 2.10.2 The TRACERT (Traceroute) command in Windows 98

The `TRACERT` command reports which systems a message goes through on its way from the originator to the target. This can be very useful to figure out why you cannot communicate between systems, or to investigate network performance problems.

TCP/IP for VSE/ESA does not have this command, but Windows 98, for example, does. Here is an example:

```

C:\>tracert vse240

Tracing route to VSE240 [192.168.155.108]
over a maximum of 30 hops:

  1    57 ms    77 ms    23 ms  VSE240 [192.168.155.108]

Trace complete.

C:\>

```

Figure 54. Using the TRACERT command in Windows 98

This shows that there were no intermediate hops between the PC and the VSE system in our particular setup, but if there were, they would show up here.



---

## 2.11 Operating TCP/IP for VSE/ESA

In normal operation, TCP/IP for VSE/ESA should require relatively little operator intervention. When something does go wrong, or configuration changes need to be made, there are several methods that can be used to enter TCP/IP for VSE/ESA commands. There are also some unique considerations if TCP/IP for VSE/ESA should terminate abnormally.

### 2.11.1 Command entry through the VSE console

The VSE `MSG` command activates the operator communication interface of TCP/IP for VSE/ESA. The `REPLID` will always be a subtask number, never the main task number. Once operator communication has been established, it will remain active until the operator replies with no data; there is no performance impact leaving operator communication active, although it does occupy one line on the VSE console that will not roll off automatically unless `NOHOLD` is set.

Nearly all TCP/IP for VSE/ESA commands can be entered using the operator communication interface.

You can create a `.L` member containing the commands you want executed, and then issue the `EXECUTE` command through the VSE console to process those commands. For example, suppose you wish to create new FTP daemons while TCP/IP for VSE/ESA is active; you could catalog the following `.L` member in a VSE library accessible to TCP/IP for VSE/ESA:

```
* $$ JOB JNM=CMDFILE,DISP=D,CLASS=0
// JOB CMDFILE
// EXEC LIBR
A S=TCPIP.CONFIG
CATALOG CMDFILE.L      R=Y
DEFINE FTPD,  ID=TEMP1, PORT=8021, IPADDR=192.168.155.114, -
        UNIX=YES, TIMEOUT=900
DEFINE FTPD,  ID=TEMP2, PORT=8099, IPADDR=192.168.155.115, -
        UNIX=YES, TIMEOUT=900
/+
/*
/&
* $$ EOJ
```

Figure 55. Cataloging a command file to be processed by the `EXECUTE` command

You can then process this file by issuing the `EXECUTE` command:

```

SYSTEM:  VSE/ESA                      VSE/ESA 2.4.0    TURBO (01)    USER:  SYSE
                                           TIME:  10:53:35

msg z1
AR 0015 1I40I  READY
Z1-0087 IPN300I Enter TCP/IP Command
87 execute cmdfile
Z1 0085 IPN397I Loading command deck CMDFILE
Z1 0085 IPN398I Command deck CMDFILE  has been completely loaded
Z1 0085 0052: FTP900I Daemon Startup FTP Id:TEMP1 Port:8021
Z1-0087 IPN300I Enter TCP/IP Command
Z1 0085 0053: FTP900I Daemon Startup FTP Id:TEMP2 Port:8099

==>

1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD      12=RTRV

ACT_MSG: HOLDRUN          PAUSE: 00  SCROLL: 1          MODE:  CONSOLE

```

Figure 56. Using the EXECUTE command

This technique is particularly useful for making preplanned, regular changes to the TCP/IP environment that involve a large number of key strokes.

### 2.11.2 Command entry using the batch utility program IPNETCMD

A batch utility program, IPNETCMD, is supplied to pass commands to TCP/IP for VSE/ESA. Here is an example of using IPNETCMD to perform the same function as shown earlier with the EXECUTE command:

```

* $$ JOB JNM=TCPCMD,DISP=D,CLASS=5
// JOB TCPCMD      PASS COMMANDS TO TCP/IP FROM BATCH
// LIBDEF *,SEARCH=(TCPIP.FIXES,PRD1.BASE)
// EXEC IPNETCMD,PARM='ID=00'
DEFINE FTPD,  ID=TEMP1, PORT=8021, IPADDR=192.168.155.114, -
          UNIX=YES, TIMEOUT=900
DEFINE FTPD,  ID=TEMP2, PORT=8099, IPADDR=192.168.155.115, -
          UNIX=YES, TIMEOUT=900
/*
/&
* $$ EOJ

```

Figure 57. Job stream for IPNETCMD

Running this job results in the following on the VSE console:

```

SYSTEM:  VSE/ESA                      VSE/ESA 2.4.0    TURBO (01)    USER:  SYSB
                                           TIME:  11:03:46

Z1-0087 IPN300I Enter TCP/IP Command
F5 0005 // JOB TCPCMD      PASS COMMANDS TO TCP/IP FROM BATCH
        DATE 01/19/2000, CLOCK 11/03/38
F5 0101 IPN700I TCP/IP Command Processor -- Version 01.04.00( ), 12/23/99
Z1 0085 0059: FTP900I Daemon Startup FTP Id:TEMP1 Port:8021
Z1 0085 005D: FTP900I Daemon Startup FTP Id:TEMP2 Port:8099
F5 0005 EQJ TCPCMD      MAX.RETURN CODE=0000
        DATE 01/19/2000, CLOCK 11/03/46, DURATION  00/00/07
F5 0001 1Q34I    F5 WAITING FOR WORK

==>

1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD      12=RTRV

ACT_MSG: HOLDRUN          PAUSE: 00  SCROLL: 1          MODE:  CONSOLE

```

Figure 58. Using the IPNETCMD utility program

Again, this technique is useful for preplanned changes to the TCP/IP environment.

### 2.11.3 Shutting down TCP/IP for VSE/ESA

To perform a normal shutdown of TCP/IP for VSE/ESA you enter the command `SHUTDOWN`. You will normally then be asked to confirm that you really do want to shut TCP/IP for VSE/ESA down; if you do, then reply YES. You can suppress this confirmation message by including `SET DOWNCHECK = OFF` in your TCP/IP for VSE/ESA initialization member `IPINITxx.L`.

If you run in a VM/ESA environment, be *very* careful when you issue the `SHUTDOWN` command; that same character string will, when issued to VM/ESA CP, shut down the VM system.

If TCP/IP for VSE/ESA does not shut down when requested, you may need to issue the VSE `CANCEL` command, and even include the `FORCE` operand occasionally.

### 2.11.4 Restarting TCP/IP for VSE/ESA following an abnormal termination

If TCP/IP for VSE/ESA is terminated abnormally, the next startup will detect the abnormal termination and do a quick startup and an immediate, controlled shutdown. This cleans up some internal interfaces that may be left behind during the abnormal termination. Simply restart TCP/IP for VSE/ESA one more time, and it should come up normally.

One word of caution: the internal interfaces are partition dependent, so the restart following abnormal termination *must* be in the same partition (for a dynamic partition, it is the same partition, not the same dynamic class, which may be a

little difficult to manage if you have multiple partitions defined for that dynamic class).

#### **2.11.5 Limiting TCP/IP for VSE/ESA messages on the console**

TCP/IP for VSE/ESA generates a large number of messages during startup, normal operation, and shutdown. Some of the messages contain important information, particularly if you are experiencing problems, but the volume sometimes makes it hard to find the important ones. You can control the quantity and relative importance of the messages displayed on the VSE console and SYSLST using the `SET MESSAGE` command. You can also suppress individual messages using the `MESSAGE MSGID=` command.

We recommend you allow TCP/IP for VSE/ESA to display all messages on both the console and SYSLST until you are comfortable the product is working correctly. You may then want to display the routine messages only on SYSLST, and reserve the console for more important messages.

See *TCP/IP for VSE Installation Guide, Release 1.4* for guidance on message management.

---

## Chapter 3. Using FTP

File Transfer Protocol (FTP) is used to send files between TCP/IP systems.

File transfers using TCP/IP for VSE/ESA can be initiated either from VSE or from a client communicating with VSE. When you initiate a transfer from your client (such as from your PC), you are using both an FTP client and the TCP/IP for VSE/ESA FTP server. When you initiate a transfer from VSE using the FTP command in CICS, automated FTP, or a batch job, you are using the TCP/IP for VSE/ESA FTP client, and must have an FTP server installed on the remote host (the PC).

We will discuss the required definitions and provide examples for using the TCP/IP for VSE/ESA FTP server from various clients, as well as using the VSE FTP clients with a server on a remote host.

---

### 3.1 Definitions for FTP

You must complete the following steps before you can use FTP with TCP/IP for VSE/ESA:

1. Include at least one `DEFINE FTPD` command in the IPINIT member.
2. Include security commands in the IPINIT member if you plan to secure your VSE files.
3. Include `DEFINE FILESYS` and/or `DEFINE FILE` commands in the IPINIT member to build the TCP/IP for VSE/ESA file system, unless you plan to use *only* autonomous FTP batch jobs for file transfer.
4. Include at least one `DEFINE EVENT` command in the IPINIT member if you plan to use automatic FTP to automate the transfer of VSE/POWER LST and PUN queue members to remote hosts.
5. Define the FTP client transaction to CICS if you plan to use that client.

#### 3.1.1 Define FTP daemons

The `DEFINE FTPD` command creates a File Transfer Protocol daemon (server), which provides access to VSE data from all kinds of FTP clients (for example, Windows, DOS, OS/2, OS/390, VSE, UNIX). You must provide one daemon for each concurrent FTP session.

The complete syntax of the `DEFINE FTPD` command can be found in the manual *TCP/IP for VSE Commands, Release 1.4*.

Figure 59 shows the FTPD definition used in this project:

```
DEFINE FTPD, ID=FTP, PORT=21, COUNT=5, UNIX=YES
```

Figure 59. FTP daemon definition

This definition generated five daemons, FTP01 through FTP05, so we could have up to five file transfers occurring at the same time.

We specified UNIX on the daemon because we planned to use a graphical FTP client during the project, and our experience told us that most graphical FTP clients are written to work with UNIX-like daemons (further testing with the specific FTP client we chose to use, BlueZone FTP, indicates that it works satisfactorily with the FTP daemons whether or not UNIX=YES is specified).

### 3.1.2 Define security for FTP

FTP users from all other computer platforms can, in principle, read or write any file on your system. You probably will not permit your entire system to be open to everyone. TCP/IP for VSE/ESA provides several security measures that you can use to control access to your data:

1. You may restrict the access to FTP by user ID by issuing the `SET SECURITY=ON` command and providing TCP/IP for VSE/ESA a set of user IDs and passwords by means of `DEFINE USER` commands. Only users who can provide a valid user ID/password combination are permitted to initiate a file transfer.
2. Files on the VSE system can be made read-only to remote clients by specifying `READONLY=YES` in the `DEFINE FILE` commands. Be careful: `READONLY=NO` is the default, allowing read/write access.
3. You can create a security exit, which must be defined in a `DEFINE SECURITY` command. If you have defined this exit, then the definitions of user ID and password in the `DEFINE USER` statement can be overruled by the exit, as well as other criteria used to allow or deny access.

Figure 60 shows the security definitions used for this project:

```
SET SECURITY          = ON
*
DEFINE USER, ID=TCPA, -
+   PASSWORD=TWINKIE
DEFINE USER, ID=JUST, -
+   PASSWORD=TWINKIE
DEFINE USER, ID=JOHN, -
+   PASSWORD=TWINKIE
DEFINE USER, ID=SYSA, -
+   PASSWORD=TWINKIE
DEFINE USER, ID=SYSB, -
+   PASSWORD=TWINKIE
DEFINE USER, ID=SYSC, -
+   PASSWORD=TWINKIE
DEFINE USER, ID=SYSD, -
+   PASSWORD=TWINKIE
```

Figure 60. FTP security parameters

We set security to ON to validate the user ID and password. We defined several users; they are the only users who can access files through FTP after we have set `SECURITY=ON`. We did not use a security exit, so no `DEFINE SECURITY` command was used.

### 3.1.3 Defining the VSE file system

TCP/IP for VSE/ESA must construct a file system before you can use an FTP client (other than the "Autonomous FTP" functions of the batch FTP clients) to access VSE data. You use the `DEFINE FILESYS` and/or `DEFINE FILE` commands in your IPINIT file to create the file system.

The complete syntax of the `DEFINE FILESYS` and `DEFINE FILE` commands can be found in the manual *TCP/IP for VSE Commands, Release 1.4*.

Figure 61 shows the file definitions we used for this project:

|                                      |               |                 |
|--------------------------------------|---------------|-----------------|
| DEFINE FILE, PUBLIC='TCPIP',         | DLBL=TCPIP,   | TYPE=LIBRARY    |
| DEFINE FILE, PUBLIC='PRD2',          | DLBL=PRD2,    | TYPE=LIBRARY, - |
| READONLY=YES                         |               |                 |
| DEFINE FILE, PUBLIC='POWER',         | DLBL=IJQFILE, | TYPE=POWER      |
| DEFINE FILE, PUBLIC='ICCF',          |               | TYPE=ICCF       |
| DEFINE FILE, PUBLIC='KSDSTCP.PRINT', | DLBL=KSDSPRT, | TYPE=KSDS       |
| DEFINE FILE, PUBLIC='ESDSTCP.PRINT', | DLBL=ESDSPRT, | TYPE=ESDS       |
| DEFINE FILE, PUBLIC='KSDSTCP.FILE',  | DLBL=KSDSFIL, | TYPE=ESDS       |
| DEFINE FILE, PUBLIC='ESDSTCP.FILE',  | DLBL=ESDSFIL, | TYPE=ESDS       |
| DEFINE FILE, PUBLIC='SAMTCP.FILE',   | DLBL=SAMFILE, | TYPE=ESDS       |
| DEFINE FILE, PUBLIC='IJSYSCT',       | DLBL=IJSYSCT, | TYPE=VSAMCAT, - |
| READONLY=YES                         |               |                 |
| DEFINE FILE, PUBLIC='VSESPUC',       | DLBL=VSESPUC, | TYPE=VSAMCAT, - |
| READONLY=YES                         |               |                 |
| DEFINE FILE, PUBLIC='TCPCAT',        | DLBL=TCPCAT,  | TYPE=VSAMCAT, - |
| READONLY=YES                         |               |                 |
| DEFINE FILE, PUBLIC='UCATF01',       | DLBL=UCATF01, | TYPE=VSAMCAT, - |
| READONLY=YES                         |               |                 |

Figure 61. FTP file system definitions

We included definitions for each of the different types of files supported, so that we could demonstrate their use. We also made a number of the entries read-only, to ensure critical entries such as VSAM catalogs did not get destroyed during the project.

#### Note

TCP/IP for VSE/ESA includes a quick and easy way to define a large number of files by using the `DEFINE FILESYS` command. However, we feel this is a very dangerous method to use, because it makes all of the files in your label area available as part of the file system. This typically includes all of your VSE libraries (IJSYSRS, PRD1, PRD2), all of your VSAM catalogs (master catalog and user catalogs), all of the VSE/POWER queues, and all of your master files (VSAM and sequential). We highly recommend you never use the `DEFINE FILESYS` command in a production TCP/IP for VSE/ESA partition, but instead use `DEFINE FILE` commands to define only the files you really want accessible. And we recommend you include `READONLY=YES` on all files except those you definitely want to allow to be updated.

The label information (DLBLs) for each of the files defined in the file system is shown in Figure 62 on page 86.

```

// DLBL SAMFILE, 'VSAM.SAM.FTP.FILE1', 99/365, VSAM, CAT=VSESPUC, X
      RECORDS= (100,100) , RECSIZE=80
// DLBL KSDSFIL, 'VSAM.KSDS.FTP.FILE1' , , VSAM, CAT=VSESPUC
// DLBL ESDSFIL, 'VSAM.ESDS.FTP.FILE1' , , VSAM, CAT=VSESPUC
// DLBL KSDSPRT, 'VSAM.KSDS.PRINT.FILE1' , , VSAM, CAT=VSESPUC
// DLBL ESDSPRT, 'VSAM.ESDS.PRINT.FILE1' , , VSAM, CAT=VSESPUC
// DLBL IJQFILE, 'VSE.POWER.QUEUE.FILE', 99/366, DA
// EXTENT SYS001, DOSRES, 1, 0, 945, 7
// DLBL DTSFILE, 'ICCF.LIBRARY', 99/366, DA
// EXTENT SYS010, SYSWK1, 1, 0, 4530, 1800
// DLBL IJSYSCT, 'VSAM.MASTER.CATALOG', 99/366, VSAM, CAT=IJSYSCT
// DLBL PRD2, 'VSE.PRD2.LIBRARY' , , VSAM, X
      CAT=IJSYSCT, X
      DISP= (OLD, KEEP)
// DLBL VSESPUC, 'VSESP.USER.CATALOG', 0, VSAM, CAT=VSESPUC
// DLBL TCPCAT, 'TCPIP.CATALOG' , , VSAM, CAT=TCPCAT
// DLBL TCPIP, 'VSE.TCPIP.LIBRARY' , , VSAM, CAT=TCPCAT, DISP= (OLD, KEEP)
// DLBL UCATF01, 'UCAT.F01' , , VSAM, CAT=UCATF01

```

Figure 62. DLBLs for each file defined in the file system

Some of the labels in this list came from the TCP/IP startup job stream, and some from the VSE standard labels. Note that the DLBLs for VSAM catalogs have a CAT=value pointing to themselves; this is required by TCP/IP for VSE/ESA when accessing the catalogs using FTP.

### 3.1.4 Define automatic FTP support

TCP/IP for VSE/ESA includes the ability to automatically send VSE/POWER LST and PUN queue entries to other systems, based on the class of the output. If you want to use this support, you must define at least one event.

The complete syntax of the `DEFINE EVENT` command can be found in the publication *TCP/IP for VSE Commands, Release 1.4*.

Figure 63 shows the EVENT definition used in this project:

```

DEFINE EVENT, ID=LST_LISTEN, TYPE=POWER, CLASS=N, QUEUE=LST, ACTION=FTP

```

Figure 63. FTP DEFINE EVENT specification

Details on using this support are in 3.7.2.4, “Automatic FTP” on page 127.

### 3.1.5 Define CICS transaction FTP

TCP/IP for VSE/ESA provides a CICS FTP client, which allows a CICS terminal user to initiate the file transfer between VSE and a remote FTP server. To enable this capability, you must define the FTP transaction to CICS.

TCP/IP for VSE/ESA provides definitions for the CICS transactions and programs. These should already be in your CSD, but you should verify that they are and create them if they are not; see 2.8.5.6, “CICS definitions for TCP/IP for VSE/ESA” on page 71.



---

## 3.2 Configuring non-VSE remote hosts

Whenever you want to transfer files between a VSE system and one or more non-VSE systems, you must have client and/or daemon software on the remote systems. This software may need configuration options set to work properly with the VSE system.

### 3.2.1 Configuring Windows FTP clients and daemons

Windows 95 and Windows 98 include a command-line FTP client, but not a graphical FTP client and an FTP daemon.

#### 3.2.1.1 Command-line FTP clients

A command-line FTP client is included with Windows 95 and Windows 98.

No configuration is generally necessary. You should determine what commands the particular client you are using supports. You can usually invoke the supplied FTP client and issue the `HELP` (or `?`) command to get a list of all commands available, as shown in Figure 64 from a Windows 98 system:

```
C:\>ftp
ftp> help
Commands may be abbreviated.  Commands are:

!            delete          literal          prompt          send
?            debug           ls               put              status
append      dir                  mdelete         pwd              trace
ascii       disconnect        mdir            quit             type
bell        get                mget            quote            user
binary      glob                mkdir           recv             verbose
bye         hash                mls             remotehelp
cd          help                mput            rename
close      lcd                  open            rmdir
ftp>
```

Figure 64. List of Windows 98 FTP commands

The Windows command-line FTP client can be used through `.BAT` files to make repetitive file transfers easier. For example, you can create a batch file on the workstation like this:

```
REM batch file to ftp get a file from VSE240
FTP -s:ftpget.scr VSE240
```

Figure 65. Workstation batch file for FTP

The batch file references a script, in this case, named `FTPGET.SCR`:

```

tcpa
twinkie
quote site cc off
cd power.lst.a
get cicsiccf c:\junk\cicsiccf.000
bye

```

Figure 66. Workstation script for an FTP batch file

You then run the batch file to do the file transfer.

### 3.2.1.2 Graphical FTP clients

No graphical FTP client is included with Windows 95 or Windows 98.

The client we chose to install on our Windows 98 workstations for this project was BlueZone FTP. This client offers several configuration options, but most are not really needed to get started.

Below are a few windows from configuring the BlueZone FTP client to give you an idea of what types of configuration options may be available on the client of your choice.

Figure 67 shows the initial window when BlueZone FTP is started:

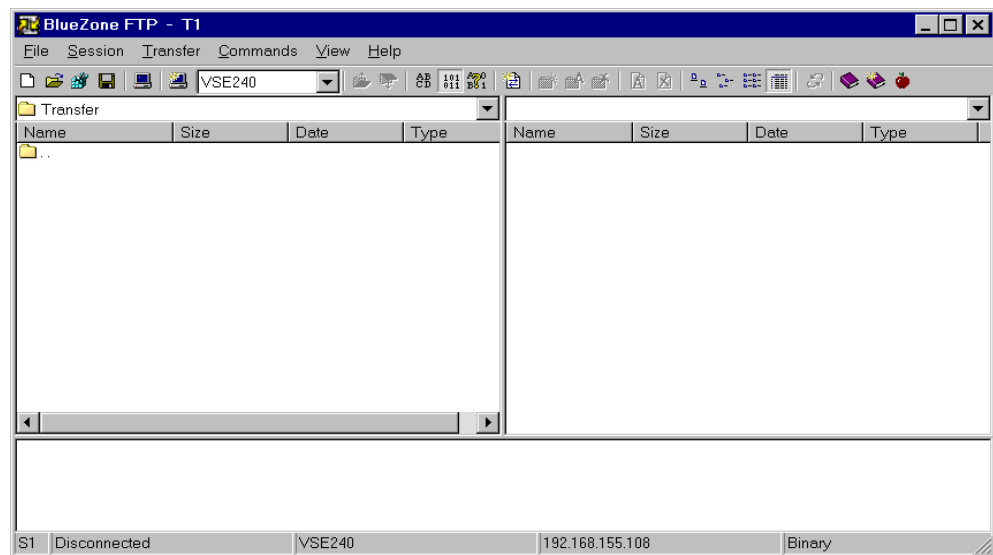


Figure 67. BlueZone FTP initial window

Under the Session pull-down list is a Configure option. Selecting this option displays a list of sessions that have already been configured, allowing you to change existing configurations or add new ones (see Figure 68 on page 89).

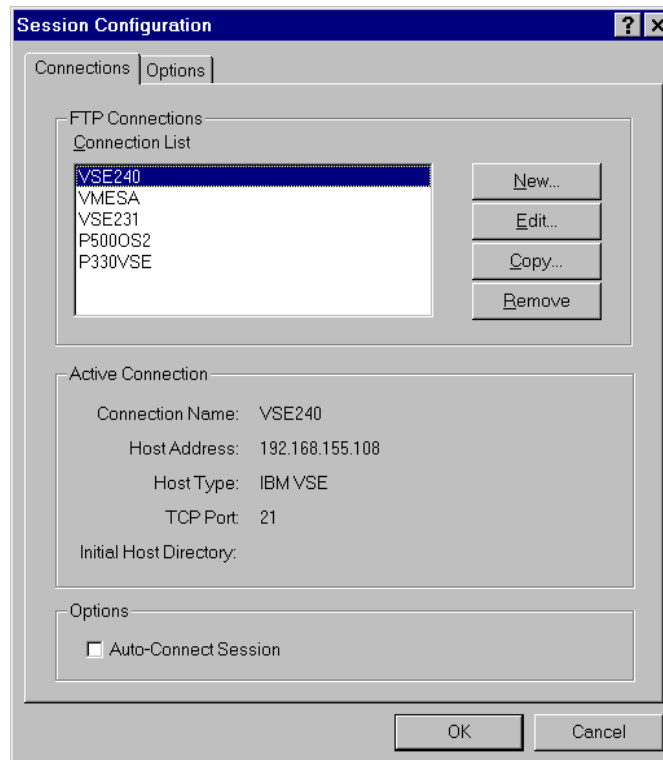


Figure 68. Adding or updating an FTP configuration in BlueZone FTP

We selected our VSE240 configuration to update, and the following window was displayed:

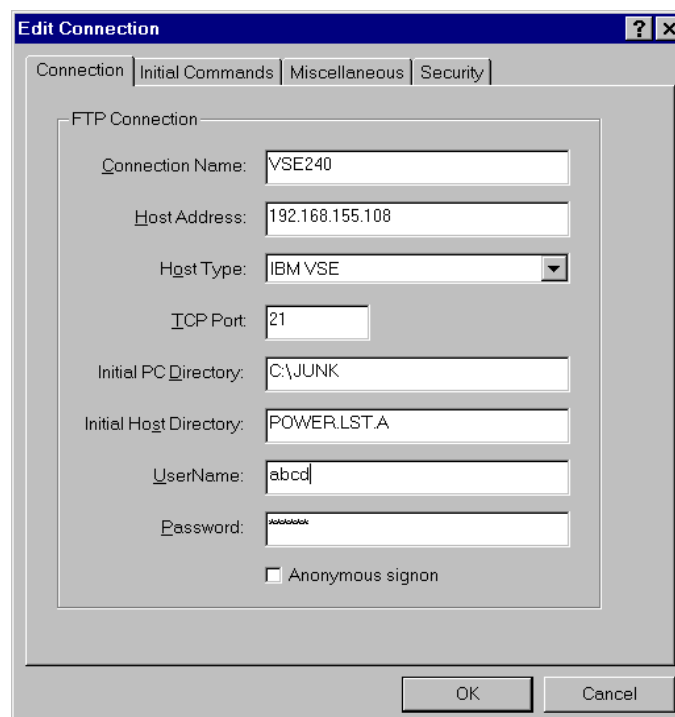


Figure 69. VSE240 configuration in BlueZone FTP

Here we can update the configuration as needed, including setting a user ID and password to automate connection to an FTP daemon.

See Figure 70 for an example of how this client looks when connected to a VSE system:

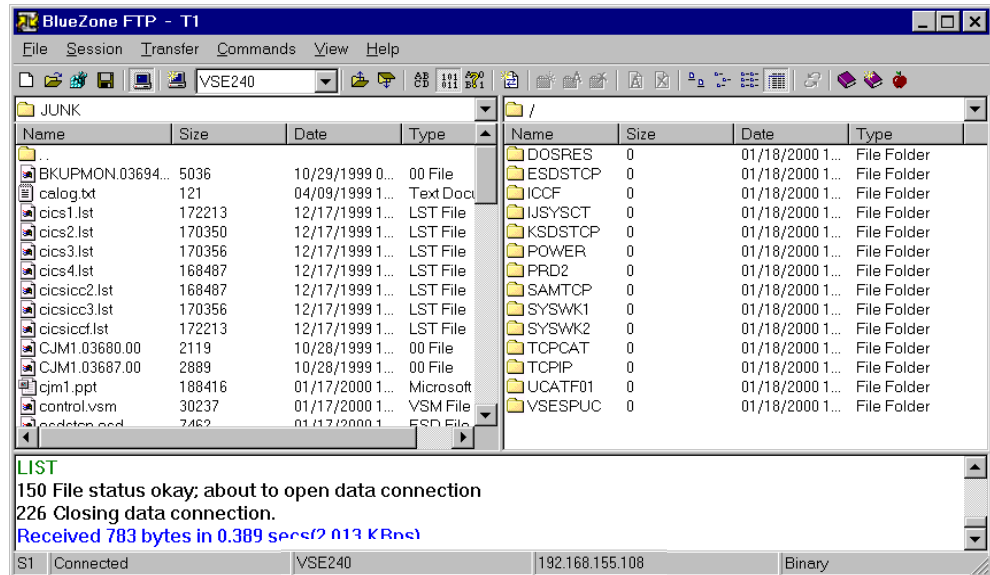


Figure 70. Using the BlueZone FTP graphical client

### 3.2.1.3 Graphical FTP daemons

No FTP daemon is included with Windows 95 or Windows 98.

The daemon we chose to use on our Windows 98 workstations was SmartFTP-D. It is a simple and straightforward daemon with few configuration options. Following are a few windows from the daemon to give you an idea of its use.

Figure 71 on page 91 shows the initial window when SmartFTP-D is started. Once started, it simply monitors a user-specified port (default 21) for FTP requests from clients, and responds to them.

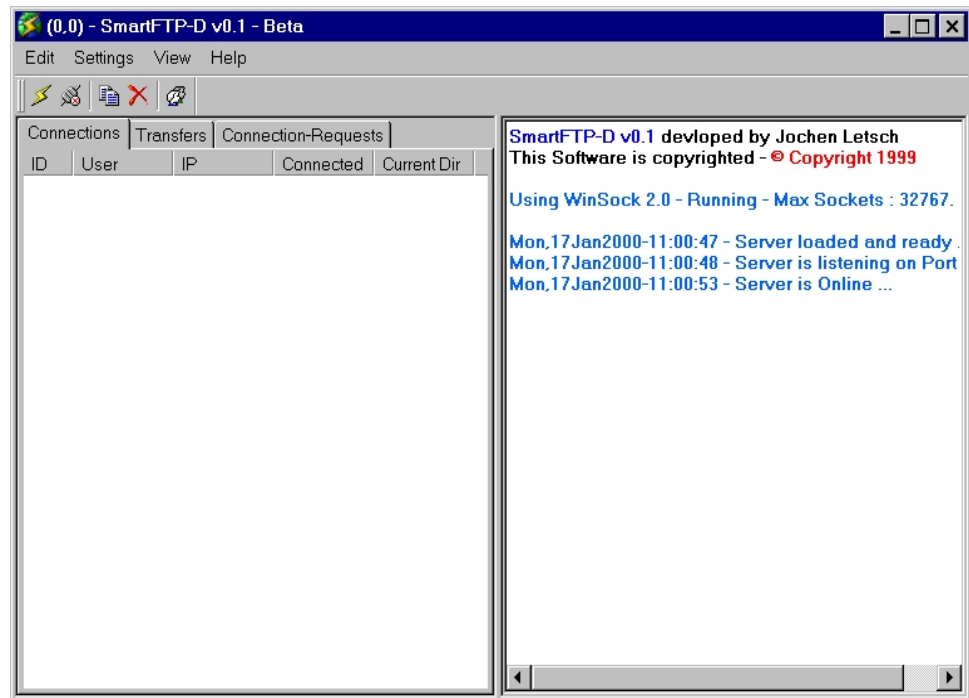


Figure 71. SmartFTP-D initial window

Under the Settings pull-down is a User Management option. Here you can define users and groups of users:

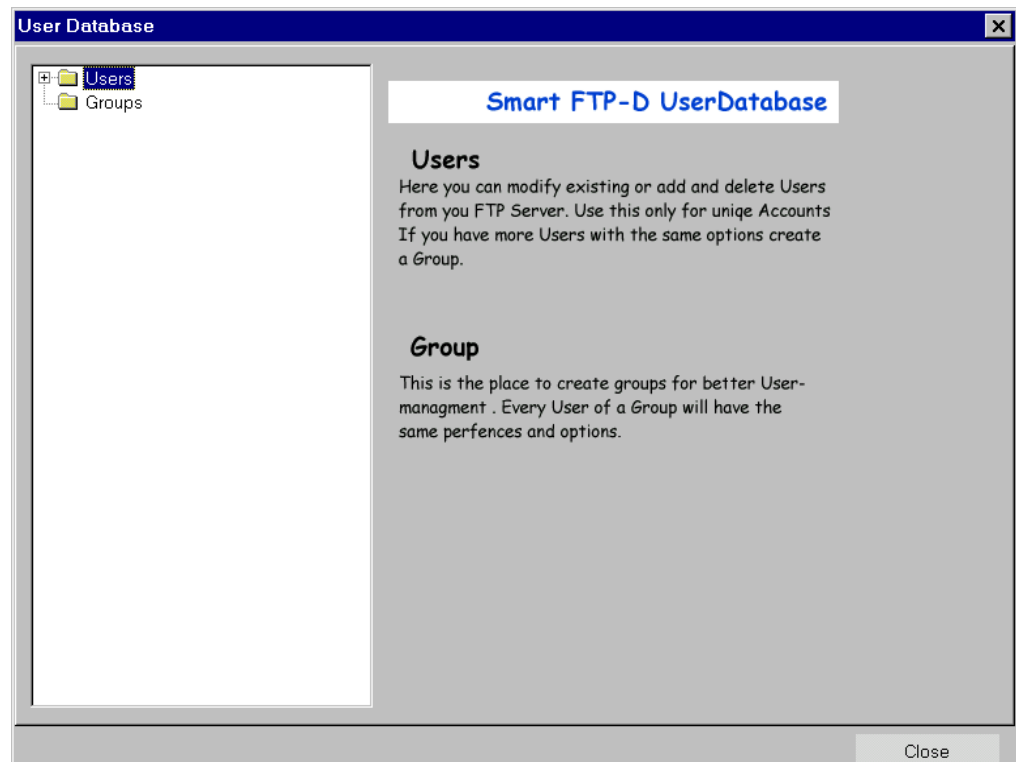


Figure 72. SmartFTP-D user definition

Under the Settings pull-down menu is also an FTP Server Settings option, which allows you to define the port number, maximum number of concurrent sessions, and identify a welcome message if desired.

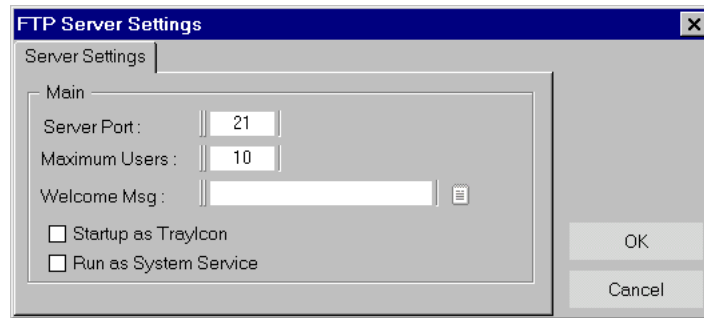


Figure 73. SmartFTP-D settings

### 3.2.2 Configuring OS/2 FTP clients and daemons

TCP/IP for OS/2, contained in OS/2 Warp Server 4, contains a command-line FTP client, a graphical FTP client, and an FTP daemon.

#### 3.2.2.1 Command-line FTP client

The command line FTP client is invoked by opening an OS/2 window and entering `FTP`. No configuration is necessary, although similar to the FTP client in Windows, it does support the use of a `HOSTS` file. It also has a `HELP` command, which will display all of the commands allowed, as shown in Figure 74:

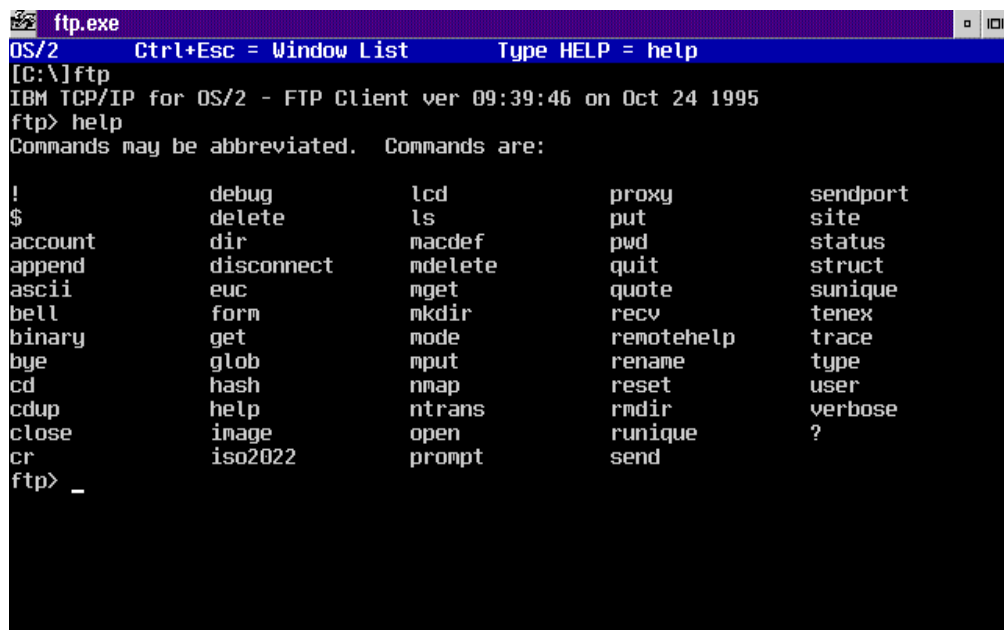


Figure 74. List of OS/2 FTP commands

#### 3.2.2.2 Graphical FTP client

The graphical FTP client included with TCP/IP for OS/2 is named `FTP-PM`. Double-clicking the icon results in the display of an `Open Remote Host` panel, where you enter the name of the remote host and a user ID and password:

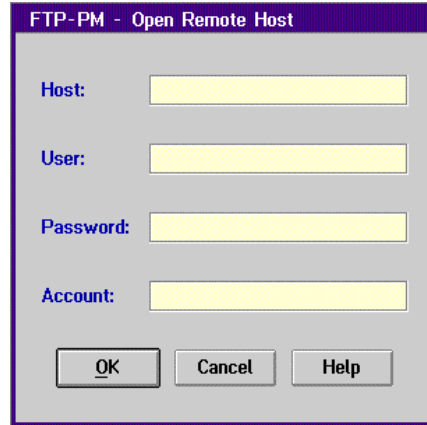


Figure 75. Log on to remote host using OS/2 FTP-PM

The main window for FTP-PM looks like this:

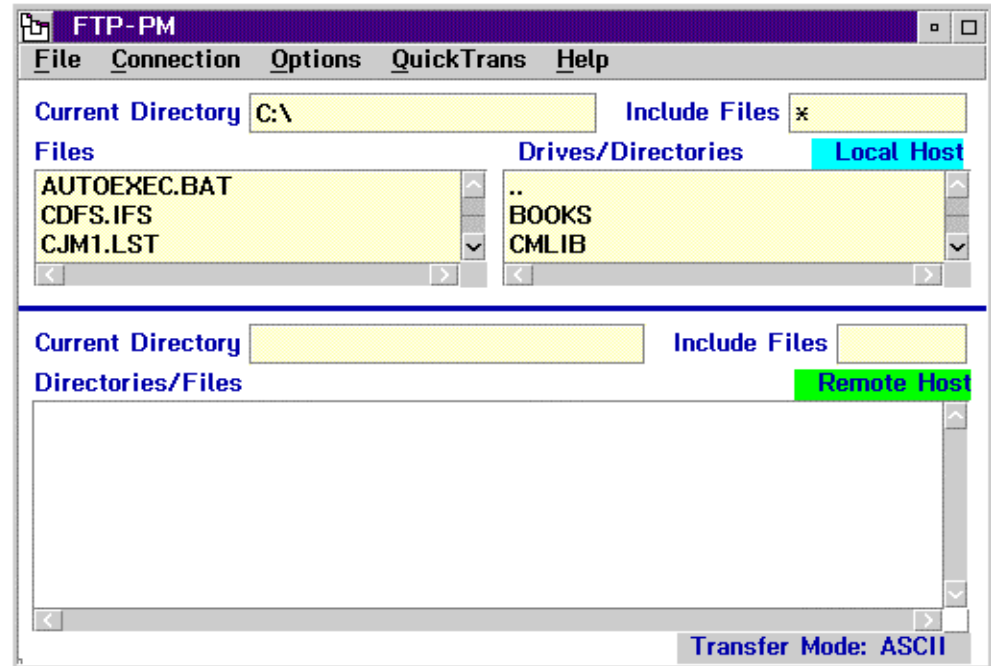


Figure 76. OS/2 FTP-PM main window

Once connected, you can change directories and select the files to be transferred.

### 3.2.2.3 FTP daemon

An FTP daemon is included with TCP/IP for OS/2. It can be automatically started during TCP/IP startup if desired. It appears to have no external interfaces for customizing, it just waits for FTP requests from clients and responds to them. Authorized users can be defined through the TCP/IP for OS/2 customization menus.

### 3.3 Referencing files in the VSE file system

The VSE file system is created by the `DEFINE FILE` and/or `DEFINE FILESYS` commands. Once it is created, you can navigate the file system using standard Change Directory (`CD`) commands. For example, in our project environment, the following file system was created based on our set of `DEFINE FILE` commands (see Figure 61 on page 85):

```
ESDSTCP
  FILE
  PRINT
ICCF
IJSYSCT
  All VSAM files in the catalog
KSDSTCP
  FILE
  PRINT
POWER
  LST
    A - Z, ALL, BIN, 0 - 9
  RDR
    A - Z, ALL, BIN, 0 - 9
  PUN
    A - Z, ALL, BIN, 0 - 9
PRD2
  all sublibraries in the library
SAMTCP
  FILE
TCPCAT
  all VSAM files in the catalog
TCPIP
  all sublibraries in the library
UCATF01
  all VSAM files in the catalog
VSESPUC
  all VSAM files in the catalog
```

Figure 77. Project file system

When we enter the file system, we are normally at the top or ROOT directory; that is, not pointing at any entry. We can issue a `CD` command to point to any entry we wish; for example, `CD TCPIP`. If we then issue a `DIR` command, we would see all of the sublibraries we have defined in that library:

```
CONFIG
FIXES
HTML
PRINT
TEMP
XFER
```

Figure 78. File system under the TCPIP entry



And if we then did a `CD CONFIG` followed by a `DIR`, we would see the members in sublibrary `TCPIP.CONFIG`:

```
AUTOFTP1 L
AUTOFTP2 L
EXTTYPES L
GPSDEF L
GPSDEF01 L
GPSDEF02 L
IPINIT04 L
IPINIT07 L
IPINIT08 L
LPRCFG01 L
LPRSCR01 L
NETWORK L
NFSCFG01 L
VTAM1 L
CUSTDEF PHASE
INSLAND PHASE
LPRISRT1 PHASE
PRODKEYS PHASE
```

Figure 79. File system under the `TCPIP.CONFIG` entry

Of course, we can change to any level of the hierarchy directly, `CD TCPIP.CONFIG`, for example. Note the use of the periods (".") in the file references. When referring to the VSE file system, you should generally use periods, because they will be accepted by the FTP daemon no matter in which mode (UNIX or non-UNIX) it is operating. A forward slash ("/") will be accepted only if the daemon is in UNIX mode, and a backslash ("\") will only be accepted if the daemon is not in UNIX mode.

You can always determine your current position in the file system by issuing the Print Working Directory (`PWD`) command.

You must be careful when issuing FTP commands that you issue them to the correct system. For example, if you issue the `CD` and `GET` commands:

- From the FTP command line on the workstation, they apply to the remote file system (VSE).
- From the CICS FTP client using 3270 emulation on that same workstation, they apply to the workstation's file system, not VSE's.

The VSE FTP clients include `LCD`, `LPWD`, and `LDIR` commands, which reference the "local" (VSE) file system, while `CD`, `PWD`, and `DIR` refer to the "remote" (PC) file system.

---

### 3.4 Data translation during file transfer

All S/390 hosts are EBCDIC systems, while PCs and UNIX systems are ASCII systems. Most of the time you will want the data translated as it is transferred, although there are exceptions, such as transferring VSE object code. TCP/IP for VSE/ESA attempts to handle the conversion automatically, and usually does a

very good job, but sometimes you may need to override what TCP/IP for VSE/ESA does.

TCP/IP for VSE/ESA comes with a member named EXTTYPES.L, which lists various file extensions and how they should be translated (or not translated); see Figure 80 on page 97. The FTP daemons use this table to determine how to translate the files as they are transferred; they compare the file extension specified on the GET or PUT for the VSE file against the table, and perform the translation based on any match found. This search and match overrides any specification you may make to the daemon, so it is important that the table entries are correct.

| CATALOG                                      |        | EXTTYPES.L                 | REPLACE=YES |
|--|--------|----------------------------|-------------|
| * ----- *                                    |        |                            |             |
| * Ext.                                       | Type   | Header                     | *           |
| * =====                                      |        |                            |             |
| AIF  | BIN    | audio/x-aiff               |             |
| AIFC   | BIN    | audio/x-aiff               |             |
| AIFF   | BIN    | audio/x-aiff               |             |
| AU   | BIN    | audio/basic                |             |
| AVI  | BIN    | video/x-msvideo            |             |
| BJB  | BIN80  |                            |             |
| CAB  | BIN    | application/octet-stream   |             |
| CLA  | BIN    | application/java           |             |
| CLASS  | BIN    | application/java           |             |
| FLR  | TEXT   | x-world/x-vrml             |             |
| GIF  | BIN    | image/gif                  |             |
| HTM  | TEXT   | text/html                  |             |
| HTML   | TEXT   | text/html                  |             |
| * HTML                                       | STRING | text/html                  |             |
| ICA  | TEXT   | application/x-ica          |             |
| JAR  | BIN    | application/octet-stream   |             |
| JAVA   | BIN    | text/plain                 |             |
| JPEG   | BIN    | image/jpeg                 |             |
| JPE  | BIN    | image/jpeg                 |             |
| JPG  | BIN    | image/jpeg                 |             |
| MPEG   | BIN    | video/mpeg                 |             |
| MPE  | BIN    | video/mpeg                 |             |
| MPG  | BIN    | video/mpeg                 |             |
| MOV  | BIN    | video/quicktime            |             |
| PDF  | BIN    | application/octet-stream   |             |
| PY   | TEXT   | application/x-pythonwin_py |             |
| QT   | BIN    | video/quicktime            |             |
| SND  | BIN    | audio/basic                |             |
| TXT  | TEXT   | text/plain                 |             |
| TEXT   | TEXT   | text/plain                 |             |
| TIFF   | BIN    | image/tiff                 |             |
| TIF  | BIN    | image/tiff                 |             |
| WAV  | BIN    | audio/x-wav                |             |
| WRL  | TEXT   | x-world/x-vrml             |             |
| WRZ  | TEXT   | x-world/x-vrml             |             |
| *  |        |                            |             |
| * The following entries are used by FTP only |        |                            |             |
| *  |        |                            |             |
| BINJOB                                       | BIN80  |                            |             |
| BJB  | BIN80  |                            |             |

Figure 80. Default entries in the EXTTYPES.L table

We added the following entries to EXTTYPES.L for transferring phases and object modules into and out of VSE libraries:

- PHASE            STRING
- OBJ             BIN80

---

## 3.5 VSE as an FTP server

The FTP daemon on the VSE system lets other TCP/IP systems access VSE data such as:

- VSE libraries and sublibraries (for example, TCPIP.XFER)
- POWER queues (RDR, LST, PUN)
- VSE/ICCF library members (read-only)
- VSE sequential data sets
- VSE/VSAM ESDS data sets
- VSE/VSAM KSDS data sets (in key sequence)
- VSE/VSAM catalogs

Normally the only files available through FTP are those defined using the `DEFINE FILESYS` and/or `DEFINE FILE` commands. However, the VSE batch FTP clients support access to files and libraries that are not defined to the file system; they can be defined directly, through `DLBL` or `TLBL` statements, right in the batch client job stream, using what is called Autonomous FTP.

The most common actions that users will perform from a remote host with FTP are `GET` and `PUT`. The `GET` command will transfer a file from the VSE FTP server to a remote host while the `PUT` command will transfer a file from a remote host to the VSE FTP server.

In the following examples, the input commands are highlighted so that you can easily follow the steps. We will be using clients on a Windows 98 workstation, but similar results would occur using OS/2, UNIX, or S/390 clients.

### 3.5.1 FTP with VSE libraries

Any VSE library member can be transferred using FTP. If the member is type `.OBJ` or `.PHASE`, the member must be transferred in binary (no translation) to prevent loss of data. Otherwise, you would generally want to have the member translated when it is transferred.

#### 3.5.1.1 Get a VSE library member to a workstation

You use the `GET` command to transfer a VSE library member to a remote workstation. See Figure 81 on page 99 for an example of transferring VSE library member `IPINIT04.L` from the `TCPIP.CONFIG` sublibrary to `C:\JUNK\INIT4.TXT` on the workstation.

```

C:\>ftp vse240
Connected to VSE240.
220-TCP/IP for VSE -- Version 01.04.00 -- FTP Daemon
      Copyright (c) 1995,1999 Connectivity Systems Incorporated
220 Service ready for new user.
User (192.168.155.108:(none)): tcpa
331 User name okay, need password.
Password:
230 User logged in, proceed.
ftp> pwd
257 "/"
ftp> cd tcpip/config
250 Requested file action okay, completed.
ftp> pwd
257 "/TCPIP/CONFIG"
ftp> lcd junk
Local directory now C:\junk.
ftp> get ipinit04.1 init4.txt
200 Command okay.
150-File: TCPIP/CONFIG/IPINIT04.L
      Type: ASCII  Recfm: FB Lrecl: 80 Blksize: 80
      CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
      Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent: 15,088
      Records sent: 184
      Transfer Seconds: 1.69 ( 14K/Sec)
      File I/O Seconds: .05 ( 0K/Sec)
226 Closing data connection.
ftp: 15088 bytes received in 2.30Seconds 6.56Kbytes/sec.
ftp> bye
221 Service closing control connection.

C:\>dir c:\junk\init*

Volume in drive C has no label
Volume Serial Number is 3214-0F08
Directory of C:\junk

INIT4      TXT           15,088  12-17-99 11:21a init4.txt
          1 file(s)             15,088 bytes
          0 dir(s)             4,578.12 MB free

C:\>

```

Figure 81. Transfer a VSE library member to a remote workstation

In this example we chose to set our position in both the VSE and workstation file systems by using `PWD`, `CD`, and `LCD` commands before issuing the `GET` command. We could have more easily accomplished the file transfer in one command by specifying the file position explicitly, as shown in Figure 82 on page 100:

```

ftp> get tcpip.config.ipinit04.1 c:\junk\newipinit.txt
200 Command okay.
150-File: TCPIP.CONFIG.IPINIT04.L
    Type: ASCII Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON UNIX=ON RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:          15,088
    Records sent:        184
    Transfer Seconds:    1.69 (    14K/Sec)
    File I/O Seconds:    .05 (    0K/Sec)
226 Closing data connection.
ftp: 15088 bytes received in 2.31Seconds 6.53Kbytes/sec.
ftp>

```

Figure 82. FTP explicitly specifying the file position

If you want to transfer object code into or out of a VSE library, either .OBJ members or .PHASE members, the transfer must be done in binary. Figure 83 is an example of transferring a phase from VSE library TCPIP.CONFIG to the workstation and then back to the VSE system to library TCPIP.XFER under another name.

```

ftp> get tcpip.config.prodkeys.phase c:\junk\prodkeys
200 Command okay.
150-File: TCPIP.CONFIG.PRODKEYS.PHASE
    Type: Binary Recfm: SV Lrecl:  4096
    CC=ON UNIX=ON RECLF=OFF TRCC=OFF CRLF=ON
150 File status okay; about to open data connection
226-Bytes sent:          158
    Records sent:        2
    Transfer Seconds:    1.59 (    0K/Sec)
    File I/O Seconds:    .06 (    0K/Sec)
226 Closing data connection.
ftp: 158 bytes received in 2.15Seconds 0.07Kbytes/sec.
ftp> put c:\junk\prodkeys tcpip.xfer.savekeys.phase
200 Command okay.
150-File: TCPIP.XFER.SAVEKEYS.PHASE
    Type: Binary Recfm: SV Lrecl:  4096
    CC=ON UNIX=ON RECLF=OFF TRCC=OFF CRLF=ON
150 File status okay; about to open data connection
226-Bytes sent:          158
    Records sent:        1
    Transfer Seconds:    1.79 (    0K/Sec)
    File I/O Seconds:    .44 (    0K/Sec)
226 Closing data connection.
ftp: 158 bytes sent in 0.00Seconds 0.00Kbytes/sec.
ftp>

```

Figure 83. Transferring a .PHASE member with FTP

Notice that the system automatically recognized, based on our entry in EXTYPES.L, that we were transferring a phase, and chose binary and RECFM of SV. Transferring a .OBJ member would be similar except the RECFM chosen would be FB.

Figure 84 on page 101 shows the results on the VSE console:

```

Z1 0085 01D9: FTP936I FTP Daemon Retrieving File, Count: 2 Userid: TCPA
Z1 0085 File: TCPIP.CONFIG.PRODKEYS.PHASE
Z1 0085 01D9: FTP936I FTP Daemon Storing File, Count: 1 Userid: TCPA
Z1 0085 File: TCPIP.XFER.SAVEKEYS.PHASE

```

Figure 84. VSE console output during transfer of the .PHASE member

You should *always* check the displayed summary information to ensure the file transfer took place as you expected: translation on or off, fixed or variable length records, carriage control on or off, and so forth.

### 3.5.1.2 Put a workstation file into a VSE library

You use the `PUT` command to transfer your remote workstation file into a VSE library. We took one of the files previously downloaded (C:\JUNK\NEWIPINIT.TXT) and transferred it to VSE sublibrary TCPIP.XFER as member NEWCFG.A:

```

ftp> put c:\junk\newipinit.txt tcpip.xfer.newcfg.a
200 Command okay.
150-File: TCPIP.XFER.NEWCFG.A
    Type: Ascii  Recfm: FB Lrecl:   80 Blksize:   80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US ENG 03
150 File status okay; about to open data connection
226-Bytes sent:      15,088
    Records sent:      184
    Transfer Seconds:    2.29 (    7K/Sec)
    File I/O Seconds:    .59 (    0K/Sec)
226 Closing data connection.
ftp: 15088 bytes sent in 0.11Seconds 137.16Kbytes/sec.
ftp>

```

Figure 85. Transfer a remote workstation file into a VSE library

## 3.5.2 FTP with POWER queues

With FTP you can access the POWER LST, RDR, and PUN queues.

### 3.5.2.1 Get a POWER queue entry to a workstation

We used the following command to get the POWER list queue entry CICSICCF (POWER job number 666, class A) from our VSE system to our workstation as C:\JUNK\CICSICCF.LST):

```

ftp> get power.lst.a.cicsiccf.666 c:\junk\cicsiccf.lst
200 Command okay.
150-File: POWER.LST.A.CICSICCF.666
    Type: ASCII  Recfm: V  Lrecl:    80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:          172,213
    Records sent:        1,863
    Transfer Seconds:      3.10 (    56K/Sec)
    File I/O Seconds:     .60 (    0K/Sec)
226 Closing data connection.
ftp: 172213 bytes received in 3.68Seconds 46.80Kbytes/sec.
ftp>

```

Figure 86. Transfer a VSE/POWER LST queue entry to our workstation

This transferred the LST queue entry to the workstation with the carriage control character in the first byte. You have several options when transferring VSE/POWER files to your workstation:

- Carriage control character transferred unchanged (the default):  
 QUOTE SITE CC ON  
 QUOTE SITE TRCC OFF  
 GET ...
- Carriage control character removed during transfer:  
 QUOTE SITE CC OFF  
 QUOTE SITE TRCC OFF  
 GET ...
- Carriage control character replaced with ANSI control characters for PC printers:  
 QUOTE SITE CC OFF  
 QUOTE SITE TRCC ON

### 3.5.2.2 Put a workstation file into a POWER queue

To put the file back into the VSE/POWER LST queue from our workstation, we used the `PUT` command. However, we must first tell FTP the logical record length; we do that using the `QUOTE SITE LRECL 133` command.



```

ftp> quote site lrecl 133
200 Command okay.
ftp> put c:\junk\cicsiccf.lst power.lst.m.cicsiccf
200 Command okay.
150-File: POWER.LST.M.CICSICCF
    Type: Ascii  Recfm: FB Lrecl: 133 Blksize: 80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent: 172,213
    Records sent: 1,863
    Transfer Seconds: 13.89 ( 12K/Sec)
    File I/O Seconds: 2.35 ( 84K/Sec)
226 Closing data connection.
ftp: 172213 bytes sent in 3.35Seconds 51.41Kbytes/sec.
ftp>

```

Figure 87. Transfer a workstation file into the POWER LST queue

We transferred to the VSE/POWER LST queue in class M the file we had just previously transferred down using the `GET` command.

To transfer the workstation file `C:\JUNK\JOB1.JCL` to the VSE/POWER RDR queue for execution, we used the following commands:

```

ftp> CD POWER.RDR.0
250 Requested file action okay, completed.
ftp> PUT C:\JUNK\JOB1.JCL
200 Command okay.
150-File: POWER/RDR/0/JOB1.JCL
    Type: Ascii  Recfm: FB Lrecl: 80 Blksize: 80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent: 136
    Records sent: 8
    Transfer Seconds: 1.49 ( 0K/Sec)
    File I/O Seconds: .04 ( 0K/Sec)
226 Closing data connection.
ftp: 136 bytes sent in 0.00Seconds 0.00Kbytes/sec.
ftp>

```

Figure 88. Transfer a workstation file to the VSE/POWER RDR queue for execution

Notice that we specified a VSE/POWER RDR class of 0 in the change directory command; however, as shown in Figure 89 on page 104, the file transferred included a `POWER * $$ JOB` card that specified `CLASS=8`, so the job actually ran in F8. TCP/IP for VSE/ESA honors the RDR class in the `FTP` command only if there is no `* $$ JOB` statement, or the `* $$ JOB` statement does not contain a `CLASS=` parameter.

```

* $$ JOB JNM=JOB1,DISP=D,CLASS=8
// JOB JOB1
// EXEC LSERV
// ASSGN SYS000,F01
// ASSGN SYS005,SYSLST
// EXEC LVTOC
/&
* $$ EOJ

```

Figure 89. Job submitted to the VSE/POWER RDR queue

### 3.5.3 FTP from the ICCF library

You can transfer a member from the VSE/ICCF library to a workstation. You cannot, however, transfer a member into the VSE/ICCF library. The user ID used to log on through FTP must be known to VSE/ICCF.

#### 3.5.3.1 Get an ICCF library member to a workstation

We transferred ICCF library member JCLPROCS from library 10 to our workstation (C:\JUNK\JCLPROCS) using the following example:

```

ftp> get iccf.10.jclprocs c:\junk\jclprocs
200 Command okay.
150-File: ICCF.10.JCLPROCS
    Type: ASCII  Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:          46,166
    Records sent:          563
    Transfer Seconds:      2.69 (    22K/Sec)
    File I/O Seconds:     1.68 (    45K/Sec)
226 Closing data connection.
ftp: 46166 bytes received in 3.18Seconds 14.52Kbytes/sec.
ftp>

```

Figure 90. Get VSE/ICCF library member to the workstation

Note that the source is specified as ICCF\_public\_name.ICCF\_library\_number.member\_name (in our case, ICCF.10.JCLPROCS).

Note that to get an ICCF library member you have to use a valid CICS user ID during FTP login.

The ICCF library file is unique in that you cannot issue the change directory (CD) command to the ICCF library itself, or to any of its libraries, such as 10, because the file system simulated by TCP/IP for VSE/ESA does not map it to FTP that way.

### 3.5.4 FTP with VSAM files

We can transfer VSAM ESDS and KSDS files in either direction.

### 3.5.4.1 Get a VSAM KSDS file to a workstation

We moved a VSAM KSDS file to our workstation, as shown in the following example. All of the records in the KSDS are transferred to the workstation. The public name for the file is KSDSTCP.FILE.

```
ftp> get ksdstcp.file c:\junk\ksdstcp.vsm
200 Command okay.
150-File: KSDSTCP.FILE
  Type: ASCII  Recfm: FB Lrecl:   80 Blksize:   80
  CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
  Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:          7,462
  Records sent:          91
  Transfer Seconds:      1.69 (    7K/Sec)
  File I/O Seconds:      .46 (    0K/Sec)
226 Closing data connection.
ftp: 7462 bytes received in 2.25Seconds 3.32Kbytes/sec.
ftp>
```

Figure 91. Transfer a VSAM KSDS file to the workstation

The file was translated from EBCDIC to ASCII during the transfer. If the file to be transferred contains binary or packed decimal fields, those fields will be destroyed during the transfer. Of course, the file could be transferred using binary (no translation), but then the file would be unreadable on the workstation.

### 3.5.4.2 Put a workstation file into a VSAM KSDS file

We attempted to transfer the file we just downloaded to the workstation back up to VSE. However, writes to VSAM KSDS files by TCP/IP for VSE/ESA are *always* treated as inserts, and because the file being uploaded contained records with the same keys as already existed in the file, all inserts were rejected (although from the PC end it appeared the file transfer completed successfully).

```
ftp> put c:\junk\ksdstcp.vsm ksdstcp.file
200 Command okay.
150-File: KSDSTCP.FILE
  Type: Ascii  Recfm: FB Lrecl:   80 Blksize:   80
  CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
  Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:          7,462
  Records sent:          91
  Transfer Seconds:      1.78 (    7K/Sec)
  File I/O Seconds:      .50 (    0K/Sec)
226 Closing data connection.
ftp: 7462 bytes sent in 0.06Seconds 124.37Kbytes/sec.
ftp>
```

Figure 92. Transferring a workstation file into a VSAM file

A look at the VSE console shows the error.

```

Z1 0085 IPF300I I/O Handler Startup VSAM KSDS
Z1 0085 0024: FTP936I FTP Daemon Storing File, Count: 91 Userid: TCPA
Z1 0085 File: KSDSTCP.FILE
Z1 0085 IPF306W VSAM KSDS File duplicate record, update ignored.
Z1 0085 DLBL:KSDSFIL

```

Figure 93. VSE console output from failed transfer of KSDS

For this file transfer to work correctly, all of the records in the VSAM file with duplicate keys must be deleted, or the file deleted and redefined. Once we deleted and redefined the VSAM KSDS, the transfer completed successfully:

```

Z1 0085 0022: FTP936I FTP Daemon Storing File, Count: 91 Userid: TCPA
Z1 0085 File: KSDSTCP.FILE

```

Figure 94. VSE console output from successful transfer of KSDS

The results at the PC end looked exactly the same for the failed and for the successful transfer.

### 3.5.4.3 Get a VSAM ESDS file to a workstation

This example is almost identical to the KSDS example, except this time we transferred an ESDS file. All of the records in the ESDS are transferred to the workstation.

```

ftp> get esdstcp.file c:\junk\esdstcp.esd
200 Command okay.
150-File: ESDSTCP.FILE
    Type: ASCII  Recfm: FB Lrecl:   80 Blksize:   80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:          7,462
    Records sent:         91
    Transfer Seconds:      1.58 (    7K/Sec)
    File I/O Seconds:      .63 (    0K/Sec)
226 Closing data connection.
ftp: 7462 bytes received in 2.09Seconds 3.57Kbytes/sec.
ftp>

```

Figure 95. Transfer an ESDS VSAM file to the workstation

### 3.5.4.4 Put a workstation file into a VSAM ESDS file

Putting records into an ESDS file with TCP/IP for VSE/ESA is different from using a KSDS file. Here you have two choices:

- Append the records to the existing file. This is done using the `APPEND` command of FTP, and results in the records being added *after* any existing records in the file.
- Replace the existing file. This is done by defining the file with the `REUSE` attribute. Because this is not an attribute that can be altered, we had to delete and redefine the ESDS file with the `REUSE` attribute. Only the new records transferred into the ESDS will be present after the file transfer.

We used both techniques. First, we transferred the file to VSE using the `APPEND` command, as shown in Figure 96 on page 107:

```
ftp> append
Local file c:\junk\esdstcp.esd
Remote file esdstcp.file
200 Command okay.
150-File: ESDSTCP.FILE
  Type: Ascii  Recfm: FB Lrecl:   80 Blksize:   80
  CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
  Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:          7,462
  Records sent:          91
  Transfer Seconds:      1.58 (    7K/Sec)
  File I/O Seconds:      .29 (    0K/Sec)
226 Closing data connection.
ftp: 7462 bytes sent in 2.09Seconds 3.57Kbytes/sec.
ftp>
```

Figure 96. Transfer workstation file to VSAM ESDS using `APPEND`

The `APPEND` command requires the user to enter the name of the Local file, which is the file on the workstation, and the name of the Remote file, which is the file on the VSE system. The result was twice as many records in the file as were there originally.

The VSE console displayed the following:

```
Z1 0085 00BA: FTP936I FTP Daemon Storing File, Count: 91 Userid: TCPA
Z1 0085 File: ESDSTCP.FILE
```

Figure 97. VSE console after successful ESDS file transfer using `APPEND`

We then deleted and redefined the ESDS, this time including the `REUSE` attribute, and the file transfer completed successfully:

```
ftp> put c:\junk\esdstcp.esd esdstcp.file
200 Command okay.
150-File: ESDSTCP.FILE
  Type: Ascii  Recfm: FB Lrecl:   80 Blksize:   80
  CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
  Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:          7,462
  Records sent:          91
  Transfer Seconds:      1.68 (    7K/Sec)
  File I/O Seconds:      .41 (    0K/Sec)
226 Closing data connection.
ftp: 7462 bytes sent in 0.06Seconds 124.37Kbytes/sec.
ftp>
```

Figure 98. Transfer workstation file to VSAM ESDS using `PUT`

### 3.5.5 FTP using the VSAM catalog

TCP/IP allows you to define a VSAM catalog as a file and then be able to upload or download any file to/from that VSAM catalog.

Following is an FTP session for a file of this type.

```
ftp> cd vsespuc
250 Requested file action okay, completed.
dir
.
.
-rw-rw-rw- 1 vse direct      7280 AUG 99 12:50 VSAM.ESDS.FTP.FILE1
-rw-rw-rw- 1 vse direct         0 AUG 99 12:50 VSAM.ESDS.PRINT.FILE1
-rw-rw-rw- 1 vse direct    12:00 JAN 17 17:07 VSAM.KSDS.FTP.FILE1
-rw-rw-rw- 1 vse direct         0 DEC 00 16:45 VSAM.KSDS.PRINT.FILE1
-rw-rw-rw- 1 vse direct    12:00 JAN 17 17:07 VSAM.SAM.FTP.FILE1
-rw-rw-rw- 1 vse direct    12:00 JAN 17 17:07 VSE.CONTROL.FILE
-rw-rw-rw- 1 vse direct    12:00 JAN 17 17:07 VSE.MESSAGE.ROUTING.FILE
-rw-rw-rw- 1 vse direct  825978 OCT 00 11:11 VSE.ONLINE.PROB.DET.FILE
-rw-rw-rw- 1 vse direct    12:00 JAN 17 17:07 VSE.PRIMARY.LIBRARY
-rw-rw-rw- 1 vse direct    12:00 JAN 17 17:07 VSE.TEXT.REPSTORY.FILE
-rw-rw-rw- 1 vse direct    12:00 JAN 17 17:07 VSESP.USER.CATALOG
226 Closing data connection.
ftp: 1876 bytes received in 5.05Seconds 0.37Kbytes/sec.
ftp> get vse.control.file c:\junk\control.vsm
200 Command okay.
150-File: VSESPUC/VSE.CONTROL.FILE
      Type: ASCII  Recfm: FB Lrecl:   80 Blksize:   80
      CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
      Translate with US_ENG_03
150 File status okay; about to open data connection
226-Bytes sent:      30,237
      Records sent:      206
      Transfer Seconds:      1.99 (      29K/Sec)
      File I/O Seconds:      1.04 (      29K/Sec)
226 Closing data connection.
ftp: 30237 bytes received in 2.58Seconds 11.72Kbytes/sec.
ftp>
```

Figure 99. Transfer a file from a VSAM catalog to the workstation

This example shows a sample FTP session with a file defined with TYPE=VSAMCAT. First we do a CD to VSESPUC. Then we list the entries in this VSAM catalog by issuing the DIR command. All files defined in the VSAM catalog are listed. Next we do a GET of the file VSE.CONTROL.FILE. Because this particular VSAM file contains packed decimal and binary fields, some of the data was lost due to translation from EBCDIC to ASCII.

---

## 3.6 Using a graphical client on the workstation

Using a graphical FTP client is very different from line mode (where you explicitly control the environment by issuing commands):

- It generates many of the FTP commands under the covers; you need to be very careful that it is doing what you want it to do (that is, CC on or off, translation or binary transfer, and so forth).
- You can see at a glance where you are in both file systems (VSE and client).
- You can typically drag and drop files in either direction.

- You can sometimes double-click to view the file in Notepad without transferring the file to the hard drive.

#### Note

Do not forget that most graphical clients require the VSE FTP daemon to be in UNIX mode to display the file system entries correctly.

Here is just one example of using a graphical client for FTP. We started the client at the top of the VSE file system, so the public names of all of the files defined in the file system are displayed; see Figure 100:

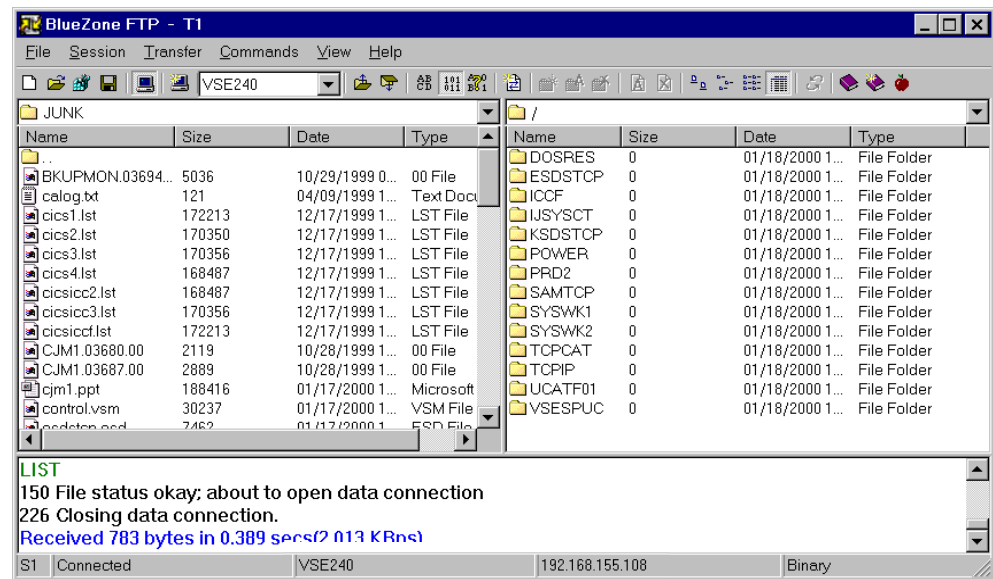


Figure 100. Initial display from a graphical FTP client

We then double-clicked the TCPIP entry on the right side (which is a VSE library) to get a list of subdirectories:

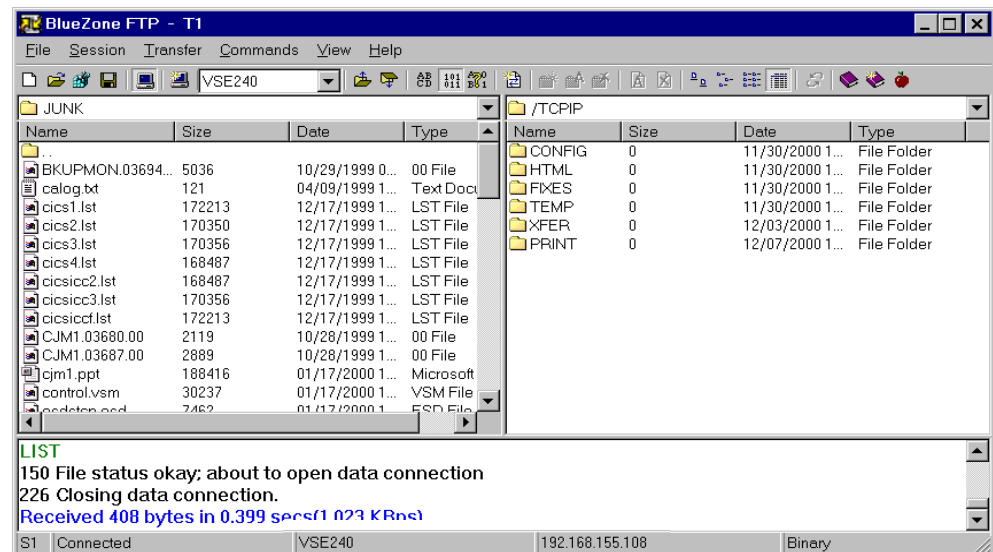


Figure 101. Graphical client listing of sublibraries in the VSE library TCPIP

Double-clicking **CONFIG** gave us the following list of members in sublibrary TCP/IP.CONFIG:

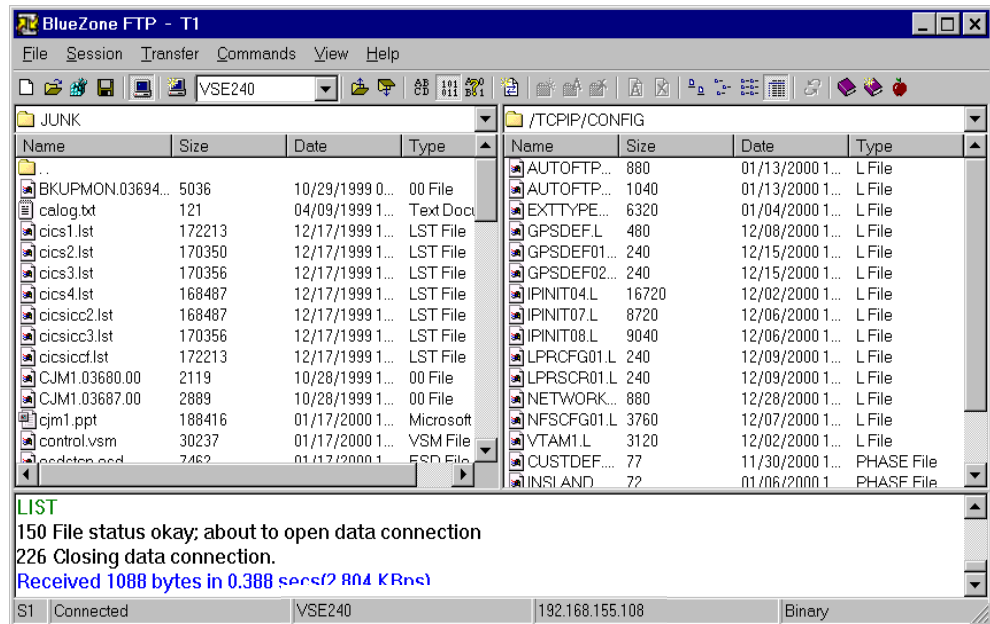


Figure 102. Graphical client listing of members in VSE sublibrary TCP/IP.CONFIG

Clicking **LPRCFG01.L** allowed us to drag and drop the entry onto the workstation hard drive, in the C:\JUNK directory, as shown in Figure 103:

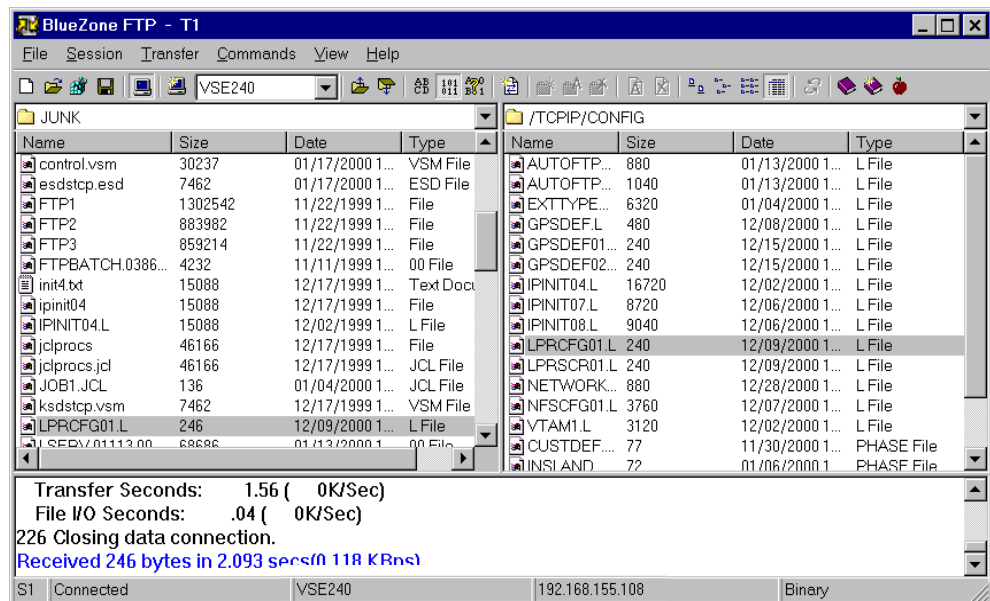


Figure 103. Following FTP of member from VSE library using graphical client

And finally we double-clicked **LPRCFG01.L** on the workstation side to view the member in Windows Notepad. We could have viewed the member also without performing the file transfer by double-clicking **LPRCFG01.L** on the VSE side of the window.



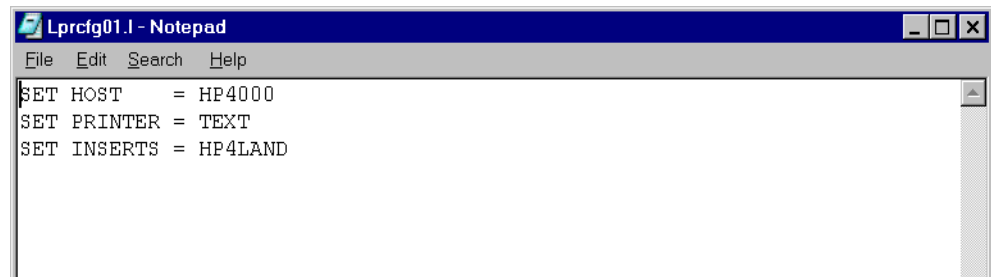


Figure 104. Displaying the transferred member using Notepad

---

### 3.7 VSE as an FTP client

TCP/IP for VSE/ESA provides FTP clients on VSE that can be initiated as a CICS transaction, from batch, or through various programming APIs. To start from a CICS session, the FTP transactions must have been previously defined to CICS.

There is a series of `FTP` commands that you can use with the VSE FTP clients. The commands and their description can be found in *TCP/IP for VSE User's Guide, Release 1.4*. A summary of the commands is available from the VSE CICS FTP client by entering `HELP`:

```

help
----- Help System Display (FTPC      ) -----
The following is a list of FTP client commands:
  ACCT      - Send account information to the remote session
  ASCII     - Change to ASCII type
  BINARY    - Change to binary (image) type
  BYE       - Quit
  CD        - Change the remote working directory
  CDUP      - Change the remote working directory up one level
  CLOSE     - Close the remote session
  DELETE    - Erase a file at the remote host
  DIR       - List the remote working directory contents
  EBCDIC    - Change to EBCDIC type
  ERASE     - Erase a file at the remote host
  EXECUTE   - Fetch a member and execute the commands
  GET       - Transfer a member from remote system
  GETX      - Transfer a member from remote system w/delimiters
  HELP      - Display help information
  IMAGE     - Change to image (binary) type
  LACCT     - Send account information to the local session
  LCD       - Change the local working directory
  LCDUP     - Change the local working directory up one level
  LCLOSE    - Close the local session
  LERASE    - Erase a file at the local host
  LDELETE   - Erase a file at the local host
  LDIR      - List the local working directory contents
  LMKDIR    - Create a directory on the local host
  LNLIST    - Retrieve qualified list of names from local host
  LNOP      - Issue no-op to the local host
  LOPEN     - Open the local session
  LPASS     - Send a password to the local session
  LPWD      - List the local directory name
  LQUOTE    - Pass string as command to local host
  LRENAME   - Rename a file at the local host
  LRENAMEX  - Rename a file at the local host w/delimiters
  LRMDIR    - Remove a directory at the local host
  LSITE     - Issue site command to local host
  LSTATUS   - Issue a status command to the local host
  LSYSTEM   - Determine the local system type
  LUSER     - Send a user-id to the local session
  MGET      - Fetch multiple members from remote system
  MGETX     - Fetch multiple members from remote system w/delimiters
  MODE      - Set transmission mode
  MPUT      - Send multiple members from remote system
  MPUTX     - Send multiple members from remote system w/delimiters
  NLIST     - Retrieve qualified list of names from remote host
  NOP       - Issue no-op to the remote host
  OPEN      - Open the remote session
  PASS      - Send a password to the remote session
  PUT       - Transfer a member to remote system
  PUTX      - Transfer a member to remote system w/delimiters
  PWD       - List the remote directory name
  QUIT      - Terminate the client application
  QUOTE     - Pass string as command to remote host
  RENAME    - Rename a file at the remote host
  RENAMEX   - Rename a file at the remote host w/delimiters

```

Figure 105. VSE FTP client HELP output (part 1 of 2)

```
RMDIR    - Remove a directory at the remote host
SETVAR    - Set the value of a variable
SITE      - Issue site command to remote host
STATUS    - Issue a status command to the remote host
SYSTEM    - Determine the remote system type
STRU      - Set transmission structure
TERSE     - Limit messages displayed to user
TYPE      - Set transmission type
USER      - Send a user-id to the remote session
VERBOSE   - Do not limit message displayed to user

Ready:
```

Figure 106. VSE FTP client HELP output (part 2 of 2)

Using the VSE FTP clients, you can access any other TCP/IP system in the network that runs an FTP daemon.

On the local VSE system, the FTP clients can access data in any of the VSE file system entries.

In some of the following examples, we are using the FTP clients on our VSE system (VSE240) to access our second VSE system (VSE231) where an FTP daemon is running.

#### Note

The VSE clients support file transfer with all types of remote hosts. However, the remote host must be running an FTP daemon. Most Windows-based PCs, for example, will not have an FTP daemon running, unless you have added one, since there is no FTP daemon supplied with Windows. We installed SmartFTP-D on a Windows 98 system to demonstrate using the VSE FTP clients with a Windows workstation.

### 3.7.1 CICS FTP client transaction

Here is an example of starting the CICS client and logging on to a remote host (in this case, another VSE system):

```

FTP VSE231
FTP211I Connecting to Port: 000021 at IP: 192.168.155.124 Id:00
FTP209I Establishing connection to TCP/IP partition
FTP212I Connection has been established
F: 220-TCP/IP for VSE -- Version 01.03.00 -- FTP Daemon
    Copyright (c) 1995,1997 Connectivity Systems Incorporated
220 Service ready for new user.
Enter Foreign User ID or "LOGOFF":
abcd
F: USER abcd
F: 331 User name okay, need password.
Enter Foreign Password or "LOGOFF"
abcd
F: PASS XXXX
F: 230 User logged in, proceed.
Foreign host connection established.
L: 220-TCP/IP for VSE -- Version 01.04.00 -- FTP Daemon
    Copyright (c) 1995,1999 Connectivity Systems Incorporated
220 Service ready for new user.
Enter Local User ID, null, or "LOGOFF":
tcpa
L: USER tcpa
L: 331 User name okay, need password.
Enter Local Password, null, or "LOGOFF"
twinkie
L: PASS XXXXXX
L: 230 User logged in, proceed.
Local host connection established.
Ready:
More...

```

Figure 107. Logging on to a remote host using the VSE FTP client

Note that the VSE FTP client prefaces all nonprompting messages from the foreign system with the tag F:. This contrasts with messages from the local host, which are prefaced with the tag L:.

One other important point about the screen when using the CICS client is the More... indicator at the bottom of the screen. This indicates that there are more messages to be displayed and that you need to press Enter or Clear to see them. It is displayed highlighted even though it was not entered by the user. When no more messages are waiting to be displayed, the CICS client will prompt you for the next command.

### 3.7.1.1 GET a file from a remote Windows workstation

Here is an example of transferring a file from a remote workstation to the VSE system.

```

get autoexec.bat tcpip.xfer.autoexec.bat
L: PASV
L: 227 Entering Passive Mode (204,155,155,108,016,016) .
F: PORT 204,155,155,108,016,016
F: 200 PORT command succeed.
L: STOR tcpip.xfer.autoexec.bat
L: 150-File: TCPIP.XFER.AUTOEXEC.BAT
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
F: RETR autoexec.bat
F: 150 Binary data connection for transferring file autoexec.bat.
L: 226-Bytes sent:          152
    Records sent:          5
    Transfer Seconds:      1.18 (      K/Sec)
    File I/O Seconds:     .14 (      0K/Sec)
226 Closing data connection.
F: 226 Transfer complete.
Ready:

```

Figure 108. Transfer a file from a workstation using the CICS FTP client

### 3.7.1.2 PUT a file to a remote Windows workstation

Here is an example of transferring the same file back down to the workstation from the VSE system:

```

put tcpip.xfer.autoexec.bat abcdef.123
L: PASV
L: 227 Entering Passive Mode (204,155,155,108,016,023) .
F: PORT 204,155,155,108,016,023
F: 200 PORT command succeed.
F: STOR abcdef.123
F: 150 Binary data connection for file abcdef.123.
L: RETR tcpip.xfer.autoexec.bat
L: 150-File: TCPIP.XFER.AUTOEXEC.BAT
    Type: ASCII  Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
F: 226 Transfer complete.
L: 226-Bytes sent:          410
    Records sent:          5
    Transfer Seconds:      1.17 (      0K/Sec)
    File I/O Seconds:     .07 (      0K/Sec)
226 Closing data connection.
Ready:

```

Figure 109. Transfer a file to a remote workstation using the CICS FTP client

### 3.7.1.3 GET a VSE library member from a remote VSE system

We established a connection to our remote VSE system (VSE231) and logged on to both systems. We then were able to transfer a VSE library member (VTMNSNA.B) from the VSE231 system to the VSE240 system, as shown in Figure 110 on page 116:

```

Ready:
pwd
F: PWD
F: 257 ""
Ready:
lpwd
L: PWD
L: 257 "/"
Ready:
cd prd2.config
F: CWD prd2.config
F: 250 Requested file action okay, completed.
F: PWD
F: 257 "PRD2.CONFIG"
Ready:
lcd tcpip.xfer
L: CWD tcpip.xfer
L: 250 Requested file action okay, completed.
L: PWD
L: 257 "/TCPIP/XFER"
Ready:
get vtmsna.b vtmsna.b
L: PASV
L: 227 Entering Passive Mode (204,155,155,108,016,084) .
F: PORT 204,155,155,108,016,084
F: 200 Command okay.
L: STOR vtmsna.b
L: 150-File: TCPIP/XFER/VTMSNA.B
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US ENG 03
150 File status okay; about to open data connection
F: RETR vtmsna.b
F: 150-File: PRD2.CONFIG.VTMSNA.B
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
    Translate with DEFAULT
150 File status okay; about to open data connection
L: 226-Bytes sent:      4,674
    Records sent:      57
    Transfer Seconds:    1.45 (    4K/Sec)
    File I/O Seconds:    .22 (    0K/Sec)
226 Closing data connection.
F: 226-Bytes sent:      4,674
    Records sent:      57
    Transfer Seconds:    .16 (    0K/Sec)
    File I/O Seconds:    .04 (    0K/Sec)
226 Closing data connection.
Ready:

```

Figure 110. Get VSE library member from a remote VSE system to a local VSE

We used the `CD` and `LCD` commands first to position ourselves correctly in each system's file system before issuing the `GET`, but we could have included the positioning information in the `GET` command.

#### 3.7.1.4 PUT a VSE library member to a remote VSE system

In this example we transfer a library member from our VSE240 system to the VSE231 system.

Using the FTP PUT command, we transfer the member ATCCON00.B in sublibrary PRD2.CONFIG on our local VSE system into the sublibrary PRD2.SAVE of the foreign VSE system as member VTMCFG.SAVE.

```
put prd2.config.atccon00.b prd2.save.vtmcfg.save
L: PASV
L: 227 Entering Passive Mode (204,155,155,108,016,091) .
F: PORT 204,155,155,108,016,091
F: 200 Command okay.
F: STOR prd2.save.vtmcfg.save
F: 150-File: PRD2.SAVE.VTMCFG.SAVE
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
    Translate with DEFAULT
150 File status okay; about to open data connection
L: RETR prd2.config.atccon00.b
L: 150-File: PRD2.CONFIG.ATCCON00.B
    Type: ASCII  Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
F: 226-Bytes sent:          984
    Records sent:           12
    Transfer Seconds:       4.12 (      K/Sec)
    File I/O Seconds:       .17 (      0K/Sec)
226 Closing data connection.
L: 226-Bytes sent:          984
    Records sent:           12
    Transfer Seconds:       1.14 (      0K/Sec)
    File I/O Seconds:       .13 (      0K/Sec)
226 Closing data connection.
Ready:
```

Figure 111. Transfer a local VSE library member to a remote VSE system

### 3.7.1.5 Get a POWER queue entry from a remote VSE system

In this example, we will transfer a POWER list queue entry from the VSE231 system into the POWER list queue of our VSE240 system.

```

cd power.lst
F: CWD power.lst
F: 250 Requested file action okay, completed.
F: PWD
F: 257 "POWER.LST"
Ready:
cd s
F: CWD s
F: 250 Requested file action okay, completed.
F: PWD
F: 257 "POWER.LST.S"
Ready:
dir
F: PORT 204,155,155,108,016,028
F: 200 Command okay.
F: LIST
F: 150 File status okay; about to open data connection
PERF7.00390.00      124      13      101  3 D SYSA
PERF5.00388.00      251      19      216  3 D SYSA
PERF6.00389.00      107      10       90  3 D SYSA
PERF0.00387.00     10105     237     9,864  3 D SYSA
F: 226 Closing data connection.
Ready:
site lrecl 133
F: SITE lrecl 133
F: 200 Command okay.
Ready:
lsite lrecl 133
L: SITE lrecl 133
L: 200 Command okay.
Ready:
get perf0 power.lst.a.perf0
L: PASV
L: 227 Entering Passive Mode (204,155,155,108,016,030) .
F: PORT 204,155,155,108,016,030
F: 200 Command okay.
L: STOR power.lst.a.perf0
L: 150-File: POWER.LST.A.PERF0
    Type: Ascii  Recfm: FB Lrecl:  133 Blksize:      80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
F: RETR perf0
F: 150-File: POWER.LST.S.PERF0
    Type: Ascii  Recfm: V Lrecl:  133
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
    Translate with DEFAULT
150 File status okay; about to open data connection
L: 226-Bytes sent:      1,124,753
    Records sent:        9,866
    Transfer Seconds:    23.67 (    47K/Sec)
    File I/O Seconds:    21.56 (    52K/Sec)
226 Closing data connection.
F: 226-Bytes sent:      1,124,753
    Records sent:        9,866
    Transfer Seconds:    19.91 (    57K/Sec)
    File I/O Seconds:    6.50 (   183K/Sec)
226 Closing data connection.
Ready:

```

Figure 112. Get a remote POWER LST queue entry into a local POWER LST queue



We first changed to the POWER.LST directory on the remote VSE system, then changed to class S and issued a `DIR` command to see what files existed in that class. Since the default record length for file transfer is 80, and VSE/POWER LST queue entries are larger than that, we issued `SITE LRECL 133` and `LSITE LRECL 133` commands to set the record lengths at both ends. We then selected **PERF0** as the file we wanted to transfer, and issued the `GET` command.

### 3.7.1.6 Put a local POWER queue entry into a remote VSE system

As in the previous example, for transferring entries between POWER list queues we had to set the logical record length on both VSE systems to 133 with the `SITE LRECL 133` and `LSITE LRECL 133` commands.

To transfer the DTRPTF03 entry from the LST queue on VSE240 to the LST queue on VSE231, we issued the `PUT` command, fully qualifying the reference to each file. The result was file PTFLST in the VSE/POWER LST queue on VSEE231 in class N.

```

site lrecl 133
F: SITE lrecl 133
F: 200 Command okay.
Ready:
lsite lrecl 133
L: SITE lrecl 133
L: 200 Command okay.
Ready:
put power.lst.a.dtrptf03.1183 power.lst.n.ptflst
L: PASV
L: 227 Entering Passive Mode (204,155,155,108,016,031) .
F: PORT 204,155,155,108,016,031
F: 200 Command okay.
F: STOR power.lst.n.ptflst
F: 150-File: POWER.LST.N.PTFLST
    Type: Ascii Recfm: FB Lrecl: 133 Blksize: 80
    CC=ON UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
    Translate with DEFAULT
150 File status okay; about to open data connection
L: RETR power.lst.a.dtrptf03.1183
L: 150-File: POWER.LST.A.DTRPTF03.1183
    Type: ASCII Recfm: V Lrecl: 133
    CC=ON UNIX=ON RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
F: 226-Bytes sent: 174,810
    Records sent: 2,814
    Transfer Seconds: 7.89 ( 24K/Sec)
    File I/O Seconds: 2.99 ( 85K/Sec)
226 Closing data connection.
L: 226-Bytes sent: 174,810
    Records sent: 2,814
    Transfer Seconds: 4.54 ( 42K/Sec)
    File I/O Seconds: 1.44 ( 170K/Sec)
226 Closing data connection.
Ready:

```

Figure 113. Put a POWER LST queue entry into a remote VSE POWER LST queue

### 3.7.2 The VSE batch FTP clients

In some cases it is much more convenient to use one of the batch FTP clients. For example, let's say that you regularly execute a batch job and *always* want to FTP the resulting sequential file to another system at the job's conclusion. In this instance, you might want to automate your interactions with the FTP client.

TCP/IP for VSE/ESA provides two easy-to-use batch clients:

- Internal batch FTP client (program name FTP)
- External batch FTP client (program name FTPBATCH)

Both programs perform the same FTP functions, but in very different ways, and with different effects on the VSE system.

With the internal batch client (FTP), all of the processing is performed in the TCP/IP for VSE/ESA partition. That is, the file open, reading or writing of the file, transmission of the file, and the file close are all done in that partition, even if Autonomous FTP is used (see 3.7.2.1, "Autonomous FTP" on page 120). Because FTP can consume a large amount of CPU cycles and may perform many I/O operations, this can have a negative effect on the performance of other partitions running with a lower VSE priority, such as CICS.

With the external batch client (FTPBATCH), the file open, reading or writing of the file, and the file close are performed in the batch partition, not the TCP/IP partition. Only the transmission of the file is done in the TCP/IP partition. This allows most of the CPU and I/O overhead to be moved to a lower priority partition. Other benefits include:

- Exploitation of the VSE/ESA Turbo Dispatcher with multiple processors, because VSE can dispatch the two partitions on different processors.
- Reduced LTA waiting by the TCP/IP partition during open/close, because the files are opened and closed in the FTPBATCH partition.
- FTP for tape files, without concern for tape open/close processing affecting the TCP/IP for VSE/ESA partition.
- Better job accounting because the CPU cycles and I/O of the FTP daemon will also occur in the FTPBATCH partition.

Despite the significant benefits of using FTPBATCH, you may want to use FTP some or all of the time, as the total CPU consumption will be less with FTP, and generally file transfers will complete more quickly with FTP. Which client to use is really a trade-off, resource consumption and control versus performance.

#### 3.7.2.1 Autonomous FTP

Both FTP and FTPBATCH support Autonomous FTP, which allows you to transfer files that are not defined in the TCP/IP for VSE/ESA file system. To reference VSE files not defined in the file system, you must:

- Specify %dtfname on PUT and GET requests.
- Ensure the file label information (DLBL or TLBL) is available in the batch client partition, either included directly in the job or in one of the accessible standard label areas.

The formats of the `GET` and `PUT` commands are:

GET remote\_file %dlbl,type,recfm,lrecl,blksize

PUT %dlbl,type,recfm,lrecl,blksize remote\_file

All parameters are required (except blksize if the file is unblocked):

dlbl      The name of the VSE DLBL (or TLBL) statement defining the file.

type      The file's type. Supported values are SAM, ESDS, TAPE, and KSDS.

recfm     The file's record format. Allowable values are F, FB, V, and VB.

lrecl      The file's logical record length.

blksize   The file's block size.

A detailed description of Autonomous FTP can be found in the manual *TCP/IP for VSE User's Guide, Release 1.4*.

### 3.7.2.2 The internal batch FTP client (FTP)

A typical job stream for the internal batch client might look like this:

```
* $$ JOB JNM=FTP2,DISP=D,CLASS=0
// JOB FTP2
// LIBDEF *,SEARCH=(TCPIP.FIXES,PRD1.BASE)
// EXEC FTP,PARM='IP=PC2,PORT=21'
tcpa
twinkie
TCPA
TWINKIE
ftp command 1
ftp command 1
etc...
/*
/&
* $$ EOJ
```

Figure 114. A typical FTP client batch job

The `IP=` parameter specifies the IP address of the remote server; it can be either a real IP address or a specified NAME. `PORT` defaults to 21 and `ID` defaults to 00 if not specified. There are additional parameters that can be specified, primarily to control the number of connection attempts, wait time, and debugging options; see *TCP/IP for VSE User's Guide, Release 1.4* for more information.

The `FTP` commands are placed directly in the stream. The first four interactions (that is, the first four commands) will probably be two pairs of user IDs and passwords, the first for the remote daemon and the second for the local (VSE) daemon.

Your last interaction with the FTP client can be the `QUIT` (or `BYE`) command, although the `/*` serves the same purpose.

Another way to identify the remote host and user IDs and passwords is using the `OPEN`, `USER`, and `PASS` commands:

```

* $$ JOB JNM=FTP1,DISP=D,CLASS=0
// JOB FTP1
// LIBDEF *,SEARCH=(TCP/IP.FIXES,PRD1.BASE)
// EXEC FTP
OPEN PC2 21
USER tcpa
PASS twinkie
LOPEN
LUSER TCPA
LPASS TWINKIE
ftp command 1
ftp command 2
etc...
/*
/&
* $$ EOJ

```

Figure 115. A typical FTP batch job with USER and PASS commands

Here all parameters are specified in the FTP commands.

You can establish FTP sessions with multiple remote hosts in one job stream. The following is a sample batch job stream, which will perform a file transfer from a PC to a VSE library on VSE240, and from the VSE library to the VSE/POWER PUN queue on the VSE231 system.

```

* $$ JOB JNM=FTP3,DISP=D,CLASS=0
// JOB FTP3
// LIBDEF *,SEARCH=(TCP/IP.FIXES,PRD1.BASE)
// EXEC FTP
OPEN PC2
USER tcpa
PASS twinkie
LOPEN
LUSER TCPA
LPASS TWINKIE
CD junk
GET jclprocs.jcl TCP/IP.XFER.JCLPROCS.JCL
CLOSE
OPEN VSE231
USER ABCD
PASS ABCD
EBCDIC
PUT TCP/IP.XFER.JCLPROCS.JCL POWER.PUN.M.JCLPROCS
/*
/&
* $$ EOJ

```

Figure 116. An FTP batch job transferring files among multiple systems

This job illustrates several capabilities of batch FTP:

- We can OPEN FTP sessions with multiple servers in one job step, although only one can be open at a time. Here we have opened a session with PC2, then with the VSE server using the LOPEN command. Later, we closed the PC2 session and opened one with VSE231.

- Case is important. Notice that some of the user IDs, passwords, and the directory name and file name on the PC, are in lowercase. This was required in this case because the daemon being used on the PC (SmartFTP-D) is a UNIX-style daemon, and is case sensitive. Since the directory name (JUNK) was originally created in lowercase, all reference to it must be in lowercase.
- The `EBCDIC` command can be used to reduce overhead when transferring between EBCDIC systems, such as two VSE systems or VSE and VM. Using this command prevents the EBCDIC to ASCII translation on the sending side and the ASCII to EBCDIC translation on the receiving side.

The list output from this job is shown in Figure 117 on page 124.

```

// JOB FTP3
// LIBDEF *,SEARCH=(TCPIP.FIXES,PRD1.BASE)
// EXEC FTP
NOUPRMPT
Ready:
Ready:
OPEN PC2
F: 220-Smart-FTP-D v0.1 --*** Beta ***--
220-Get your FREE Version at http://daemon.smartftp.com
220-
220 Server ready...
Foreign host connection established.
Ready:
USER tcpa
F: USER tcpa
F: 331 Need password.
Ready:
PASS twinkie
F: PASS XXXXXX
F: 230 User "tcpa" logged in.
Ready:
LOPEN
L: 220-TCP/IP for VSE -- Version 01.04.00 -- FTP Daemon
    Copyright (c) 1995,1999 Connectivity Systems Incorporated
220 Service ready for new user.
Local host connection established.
Ready:
LUSER TCPA
L: USER TCPA
L: 331 User name okay, need password.
Ready:
LPASS TWINKIE
L: PASS XXXXXX
L: 230 User logged in, proceed.
Ready:
CD junk
F: CWD junk
F: 250 CWD command successfull.
F: PWD
F: 257 "/C:/junk/" is current directory.
Ready:
GET jclprocs.jcl TCPIP.XFER.JCLPROCS.JCL
L: PASV
L: 227 Entering Passive Mode (204,155,155,108,016,068) .
F: PORT 204,155,155,108,016,068
F: 200 PORT command succeed.
L: STOR TCPIP.XFER.JCLPROCS.JCL
L: 150-File: TCPIP.XFER.JCLPROCS.JCL
    Type: Ascii  Recfm: FB Lrecl:    80 Blksize:    80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
    Translate with US_ENG_03
150 File status okay; about to open data connection
F: RETR jclprocs.jcl
F: 150 Binary data connection for transferring file jclprocs.jcl.
L: 226-Bytes sent:          46,166
    Records sent:          563
    Transfer Seconds:      1.68 (    45K/Sec)
    File I/O Seconds:      .21 (    0K/Sec)
226 Closing data connection.
F: 226 Transfer complete.

```

Figure 117. List output of our FTP client batch job FTP3 (part 1 of 2)

```

Ready:
CLOSE
Foreign host connection closed.
Ready:
OPEN VSE231
F: 220-TCP/IP for VSE -- Version 01.03.00 -- FTP Daemon
    Copyright (c) 1995,1997 Connectivity Systems Incorporated
220 Service ready for new user.
Foreign host connection established.
Ready:
USER ABCD
F: USER ABCD
F: 331 User name okay, need password.
Ready:
PASS ABCD
F: PASS XXXX
F: 230 User logged in, proceed.
Ready:
EBCDIC
F: TYPE E N
F: 200 Command okay.
L: TYPE E N
L: 200 Command okay.
Ready:
PUT TCPIP.XFER.JCLPROCS.JCL POWER.PUN.M.JCLPROCS
L: PASV
L: 227 Entering Passive Mode (204,155,155,108,016,070) .
F: PORT 204,155,155,108,016,070
F: 200 Command okay.
F: STOR POWER.PUN.M.JCLPROCS
F: 150-File: POWER.PUN.M.JCLPROCS
    Type: EbcDic Recfm: FB Lrecl:      80 Blksize:      80
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
150 File status okay; about to open data connection
L: RETR TCPIP.XFER.JCLPROCS.JCL
L: 150-File: TCPIP.XFER.JCLPROCS.JCL
    Type: EBCDIC Recfm: FB Lrecl:      80 Blksize:      80
    CC=ON  UNIX=ON  RECLF=OFF TRCC=OFF CRLF=ON
150 File status okay; about to open data connection
F: 226-Bytes sent:      45,603
    Records sent:      563
    Transfer Seconds:      4.83 (      11K/Sec)
    File I/O Seconds:      .07 (      0K/Sec)
226 Closing data connection.
L: 226-Bytes sent:      45,603
    Records sent:      563
    Transfer Seconds:      1.29 (      44K/Sec)
    File I/O Seconds:      .08 (      0K/Sec)
226 Closing data connection.
Ready:
QUI
F: QUIT
F: 221 Service closing control connection.
L: QUIT
L: 221 Service closing control connection.
1S55I  LAST RETURN CODE WAS 0000
EOJ FTP3      MAX.RETURN CODE=0000

```

Figure 118. List output of our FTP client batch job FTP3 (part 2 of 2)

### 3.7.2.3 External batch FTP client (FTPBatch)

The following job illustrates using FTPBatch to transfer a VSE sequential file (a VSAM-managed SAM file) to a workstation using Autonomous FTP:

```
* $$ JOB JNM=FTP4,DISP=D,CLASS=0
// JOB FTP4
// DLBL SAMFILE,'VSAM.SAM.FTP.FILE1',99/365,VSAM,CAT=VSESPUC,          X
      RECORDS=(100,100),RECSIZE=80
// EXEC FTPBATCH
LOPEN
LUSER TCPA
LPASS TWINKIE
OPEN PC2
USER tcpa
PASS twinkie
CD \junk
DEL TESTSEQ
PUT %SAMFILE,SAM,FB,80,2000 TESTSEQ
/*
/&
* $$ EOJ
```

Figure 119. FTPBatch job transferring a VSE sequential file to a workstation

The FTPBatch program requires `LOPEN` to be the first command.

The following job illustrates using FTPBatch to transfer a workstation file into a dynamically defined VSAM-managed SAM file named LISTFL using Autonomous FTP:

```
* $$ JOB JNM=FTP5,DISP=D,CLASS=0
// JOB FTP5
// DLBL LISTFL,'LIST.FILE',99/365,VSAM,CAT=VSESPUC,          X
      RECORDS=(1000,100),RECSIZE=133
// EXEC FTPBATCH
LOPEN
LUSER TCPA
LPASS TWINKIE
OPEN PC2
USER tcpa
PASS twinkie
CD \junk
GET cicsiccf.lst %LISTFL,SAM,F,133,133
/*
/&
* $$ EOJ
```

Figure 120. FTPBatch job transferring a workstation file to a VSAM-managed SAM file

Transfer of tape files is supported with FTPBatch. Operands are the same as those used for transferring a sequential disk (SAM) file, except the type is TAPE. Here is an example of transferring a workstation file to a tape using Autonomous FTP:



```

* $$ JOB JNM=FTP6,DISP=D,CLASS=0
// JOB FTP6
// LIBDEF *,SEARCH=(TCPIP.FIXES,PRD1.BASE)
// ASSGN SYS007,570
// MTC REW,SYS007
// TLBL LISTFL,'LIST.FILE'
// EXEC FTPBATCH,SIZE=FTPBATCH
LOPEN
LUSER TCPA
LPASS TWINKIE
OPEN PC2
USER tcpa
PASS twinkie
CD \junk
GET cicsiccf.lst %LISTFL,TAPE,FB,133,1330
/*
/&
* $$ EOJ

```

Figure 121. FTPBATCH job transferring a workstation file to a VSE magnetic tape

### 3.7.2.4 Automatic FTP

Automatic FTP allows VSE/POWER PUN and LST queue entries to be automatically sent to any other system using FTP. To use this support you must:

1. Add a `DEFINE EVENT` command in your IPINIT member. The `DEFINE EVENT` used for this project is shown in Figure 63 on page 86.
2. Catalog an FTP script file containing the `FTP` commands to be issued; for example:

```

* $$ JOB JNM=AUTOFTP1,DISP=D,CLASS=0
// JOB AUTOFTP1
// EXEC LIBR
A S=TCPIP.CONFIG
CATALOG AUTOFTP1.L      R=Y
LOPEN
LUSER TCPA
LPASS TWINKIE
OPEN PC2
USER tcpa
PASS twinkie
CD junk
LCD POWER.LST.N
LSITE CC OFF
LSITE TRCC ON
PUT LSERV LSERV.LST
/+
/*
/&
* $$ EOJ

```

Figure 122. Basic script for Automatic FTP

3. Run a job referencing this script in the DEST= parameter; for example:

```
* $$ JOB JNM=LSESV,DISP=D,CLASS=0
* $$ LST CLASS=N,DISP=D,DEST=(*,AUTOFTP1)
// JOB LSESV
// EXEC LSESV
/*
/&
* $$ EOJ
```

Figure 123. Job using Automatic FTP script

The console output from running this job looks like this:

```
BG 0000 // JOB LSESV
      DATE 01/13/2000, CLOCK 15/23/02
BG 0000 EOJ LSESV
      DATE 01/13/2000, CLOCK 15/23/04, DURATION 00/00/02
BG 0001 1Q34I BG WAITING FOR WORK
Z1 0085 0027: TCP911I Processing Event:FTP_N Type:POWER Action:FTP
Z1 0085 0027: TCP912I Processing Class:N Job:LSESV 01111-000
Z1 0085 0027: TCP910I Commands will be taken from AUTOFTP1
Z1 0085
Z1 0085 0021: FTP934I FTP Session Established with:TCPA from
Z1 0085 Ipaddr:192.168.155.108 Id:FTP05 Port:21
Z1 0085 0021: FTP936I FTP Daemon Retrieving File, Count: 1241 Userid:
Z1 0085 TCPA File: POWER/LST/N/LSESV
Z1 0085 0027: TCP909I Processing Output:LSESV ,01111-000 has completed
Z1 0085 0021: FTP900I Daemon Startup FTP Id:FTP05 Port:21
```

Figure 124. Console output from processing an Automatic FTP job

The simple script used in this example contains all the parameters needed to establish an FTP session with the remote FTP server and transfer the print file. However, it is not very usable in the real world, because it has the source and target file names hardcoded in the script. You probably want to use unique file names, based on the job name and job number in the VSE/POWER LST queue, and perhaps other information. TCP/IP for VSE/ESA provides a series of variables to help you do that, and you can create your own variables in a script.

See *TCP/IP for VSE User's Guide, Release 1.4* for details on creating and using variables.

A more practical script might be:

```

* $$ JOB JNM=AUTOFTP2,DISP=D,CLASS=0
// JOB AUTOFTP2
// EXEC LIBR
A S=TCPIP.CONFIG
CATALOG AUTOFTP2.L      R=Y
LOPEN
LUSER TCPA
LPASS TWINKIE
OPEN PC2
USER tcpa
PASS twinkie
CD junk
LSITE CC OFF
LSITE TRCC ON
SETVAR &LPATH = "POWER" + "." + &PWRQUE + "." + &PWRCLAS
LCD &LPATH
SETVAR &LFN = &PWRNAME + "." + &PWRNUMB + "." + SUBSTR(&PWRSUFF,2,2)
PUT &LFN
/+
/*
/&
* $$ EOJ

```

Figure 125. Advanced script with variables for Automatic FTP

This script uses the `SETVAR` command to create two variables:

- `LPATH` contains the path to the local file in the VSE/POWER queue:  
`POWER.LST.N.`
- `LFN` contains the complete file name in the VSE/POWER LST queue:  
`job_name.job_number.segment_number` (in this case, `LSERV.01121.00`).

The variables are then used in `FTP` commands to position within the local (VSE) file system, and to identify the file being transmitted.

The only change needed in the application job stream to use the new script is to change the `DEST=` parameter in the `* $$ LST` statement to `DEST=(*,AUTOFTP2)`.

The console output for processing the script looks like this:

```

BG 0000 // JOB LSERV
        DATE 01/13/2000, CLOCK 16/20/51
BG 0000 EOJ LSERV
        DATE 01/13/2000, CLOCK 16/20/53, DURATION 00/00/02
BG 0001 1Q34I BG WAITING FOR WORK
Z1 0085 0027: TCP911I Processing Event:FTP_N Type:POWER Action:FTP
Z1 0085 0027: TCP912I Processing Class:N Job:LSERV 01121-000
Z1 0085 0027: TCP910I Commands will be taken from AUTOFTP2
Z1 0085
Z1 0085 0021: FTP934I FTP Session Established with:TCPA from
Z1 0085 Ipaddr:192.168.155.108 Id:FTP05 Port:21
Z1 0085 0021: FTP936I FTP Daemon Retrieving File, Count: 1241 Userid:
Z1 0085 TCPA File: POWER/LST/N/LSERV.01121.00
Z1 0085 0027: TCP909I Processing Output:LSERV ,01121-000 has completed
Z1 0085 0021: FTP900I Daemon Startup FTP Id:FTP05 Port:21

```

Figure 126. Console output from Automatic FTP script using variables

If the Automatic FTP fails, you can use the command `SET DIAGNOSE=AFTP` to obtain more detailed information to determine why the FTP failed.

### 3.7.3 Managing FTP daemons

At any time you can `QUERY` or `DELETE` the FTP daemons, as well as define new ones using `DEFINE`. This would normally be done using the VSE console by replying to the prompt from the TCP/IP for VSE/ESA partition. If you do not have any outstanding prompt, you can get one by issuing the `MSG part_id` command.

#### 3.7.3.1 QUERY FTPDS command

The command format is:

```
Query FTPds [,ID=name}{[,SYSLST]
```

The `QUERY FTPDS` command without the `ID=` operand displays all defined FTP daemons with the corresponding status (active or inactive):

```

87 query ftpds
Z1 0085 IPN434I (( TCP/IP FTP Daemons ))
Z1 0085 IPN435I ID: FTP01 Port: 21 Driver: FTPD
Z1 0085 IPN350I Current Status: Inactive
Z1 0085 IPN435I ID: FTP02 Port: 21 Driver: FTPD
Z1 0085 IPN350I Current Status: Inactive
Z1 0085 IPN435I ID: FTP03 Port: 21 Driver: FTPD
Z1 0085 IPN350I Current Status: Inactive
Z1 0085 IPN435I ID: FTP04 Port: 21 Driver: FTPD
Z1 0085 IPN350I Current Status: Inactive
Z1 0085 IPN435I ID: FTP05 Port: 21 Driver: FTPD
Z1 0085 IPN350I Current Status: Inactive

```

Figure 127. Display status of all FTP daemons

To query the status of an individual FTP daemon, use the following format of the `QUERY` command:

```
87 query ftpds,id=ftp03
Z1 0085 IPN434I (( TCP/IP FTP Daemons ))
Z1 0085 IPN435I ID: FTP03 Port: 21 Driver: FTPD
Z1 0085 IPN350I Current Status: Inactive
```

Figure 128. Display status of individual FTP daemon

### 3.7.3.2 DELETE FTPD command

The `DELETE FTPD` command allows you to dynamically remove a single FTP daemon from your configuration:

```
DELEte FTPd,ID=name [,FORCE]
```

For example:

```
87 delete ftpd,id=temp2
Z1 0085 005D: FTP908I Daemon Shutdown FTP Id:TEMP2 Port:8099
```

Figure 129. Delete an FTP daemon

To delete multiple daemons, you must enter the `DELETE` command once for each daemon.



---

## Chapter 4. Using Telnet

Telnet is the protocol that allows users to log on from one TCP/IP system to another.

TCP/IP for VSE/ESA includes a Telnet daemon that supports the TN3270 protocol. This allows you to log on to VTAM applications directly from any TCP/IP-connected TN3270 clients. No line mode Telnet daemon is provided.

TCP/IP for VSE/ESA also provides two VSE Telnet clients, a CICS transaction and a batch program.

- The CICS transaction supports TN3270 sessions with daemons supporting that protocol (generally OS/390, VM/ESA, and VSE/ESA systems), and line mode sessions with other daemons (for example, UNIX and Windows systems).
- The batch program supports line mode sessions only.

---

### 4.1 Telnet implementation

To use the Telnet client and server applications in TCP/IP for VSE/ESA, you must define the following:

1. One Telnet daemon in your IPINITxx.L for each TN3270 client to be supported concurrently
2. VTAM APPL definitions for the TN3270 sessions
3. Optionally, one or more menus for VTAM application selection
4. CICS definitions for CICS Telnet client transactions

#### 4.1.1 Telnet daemon definitions

For each concurrent Telnet (TN3270) session you want to offer on your VSE system, you must define a Telnet daemon in your IPINIT initialization member. If you want to connect to a specific VTAM application on your VSE system, you can specify the VTAM application as the TARGET= parameter. To distinguish between different VTAM applications when connecting to VSE, you can either use different port numbers (for example, 23 and 5023), or you can use a menu.

Figure 130 on page 134 shows the Telnet-related statements in our initialization member IPINIT04.L.

```

SET TELNETD_BUFFERS = 20
.
SET SECURITY          = ON
.
*-----*
*               *
*   Define Telnet Daemons   *
*           and Menu         *
*               *
*-----*
DEFINE TELNETD, ID=TRMA, TERMNAME=TELNA, PORT=5023, COUNT=5, TARGET=DBDCCICS
DEFINE TELNETD, ID=TRMB, TERMNAME=TELNB, PORT=23, COUNT=5, MENU=VTAM1
DEFINE MENU, ID=VTAM1, MEMBER=VTAM1

```

Figure 130. TELNETD definitions in the IPINIT04.L member

Each `DEFINE TELNETD` command creates five daemons, so the maximum number of concurrent TN3270 users is 10; five can be connected directly to DBDCCICS (CICSICCF), and five are presented the VTAM1 menu when they connect.

The `SET TELNETD_BUFFERS` command and the `SET SECURITY` command are included here to remind you that the number of buffers allocated has an effect on Telnet performance, and the security option determines whether the user ID and password must be included on any menu you define.

#### 4.1.2 VTAM APPL definitions

To access your VSE VTAM applications using TN3270 clients, you must define one or more VTAM APPLs to be used as virtual terminal LU names. One ID (LUname) will be required for each concurrent Telnet session. Each APPL name will be referenced by a corresponding TCP/IP for VSE/ESA `DEFINE TELNETD` command.

Figure 131 is an example of the VTAM APPL definitions for our Telnet daemons. Note that each APPL name matches a generated TERMNAME value in the `DEFINE TELNETD` command.

```

.
TELNA01  APPL  AUTH= (ACQ)
TELNA02  APPL  AUTH= (ACQ)
TELNA03  APPL  AUTH= (ACQ)
TELNA04  APPL  AUTH= (ACQ)
TELNA05  APPL  AUTH= (ACQ)
TELNB01  APPL  AUTH= (ACQ)
TELNB02  APPL  AUTH= (ACQ)
TELNB03  APPL  AUTH= (ACQ)
TELNB04  APPL  AUTH= (ACQ)
TELNB05  APPL  AUTH= (ACQ)
.

```

Figure 131. Define VTAM APPLs for the Telnet daemons

The APPLs are used as follows:



- TELNA01 through TELNA05 are five APPLs that will be used for Telnet daemons for direct connection to application DBDCCICS (job name CICSICCF) when the TN3270 client connects to port 5023.
- TELNB01 through TELNB05 are five APPLs for use with menu VTAM1 when the client connects using port 23 (the default port).

### 4.1.3 Telnet menu definition

Many VSE users use a VTAM Unformatted System Services Table (USSTAB) to provide the end users with a method of selecting the VTAM application to which they want to connect. This technique will not work with TCP/IP for VSE/ESA, because VTAM is not controlling the terminal, TCP/IP is. So a menu facility is included with TCP/IP for VSE/ESA; it is similar to, but not completely compatible with, the USSTAB facility in VTAM.

TCP/IP for VSE/ESA comes with two sample menus, MENU1 and MENU2. While perhaps a good starting point for creating your own menu, neither comes close to directly replacing the IBM-supplied VTAM USSTAB. We created a menu, VTAM1, that comes close.

Figure 132 on page 136 shows the source for VTAM1.L:

```

* $$ JOB JNM=VTAM1,DISP=D,CLASS=0
// JOB VTAM1      CATALOG TCP/IP VTAM-LIKE USSTAB
// EXEC LIBR
A S=TCPIP.CONFIG
CATALOG VTAM1.L                      REPLACE=YES
HI=#
LO=$
VAR=@
CMDLINE=?
INPUT=+
PF3=EXIT
CLEAR=REFRESH

CHAR=A=LOGON APPL (DBDCCICS)
CHAR=B=LOGON APPL (PRODCICS)

MSGLINE=23
TRIES=3

IMAGE
$TCPUSSTB          #TCP/IP Application Selection Menu$

LUNAME: @LUNAME                      $IP ADDR: @IPADDR

$Enter a character to select the desired application followed by your
  USER ID and PASSWORD, then press the Enter key:

          #A$ DBDCCICS          Application Development CICS

          #B$ PRODCICS          Production CICS

#=====>      ?                      $

#USER ID:$  +USERID      $

#PASSWORD:$  +PASSWORD  $

/+
/*
/&
* $$ EOJ

```

Figure 132. Source for the VTAM1.L menu

The part of the member preceding the `IMAGE` statement defines the special characters used to identify high intensity and normal intensity fields, PF key commands, character-equated commands, and so forth.

The part following the `IMAGE` statement is a 24x80 image of how the screen will appear. It is very important that exactly 24 lines appear in the member between the line containing `IMAGE` and the line with the `/+`.

*TCP/IP for VSE Installation Guide, Release 1.4* contains more detail on creating a menu.

One limitation on using the TCP/IP for VSE/ESA menu versus a VTAM USSTAB is that the method of selecting an application by character is limited to a single character in TCP/IP for VSE/ESA while VTAM supports multiple characters. This may affect the migration of existing USSTABs.

Connecting from a TN3270 client to TCP/IP for VSE/ESA using a port associated with a daemon pointing to this menu (for example, `ID=TRMB`) will result in the following screen being displayed:

```
TCPUSSTB          TCP/IP Application Selection Menu

LUNAME: TELNB02          IP ADDR: 192.168.155.118

Enter a character to select the desired application followed by your
USER ID and PASSWORD, then press the Enter key:

          A  DBDCCICS          Application Development CICS
          B  PRODCICS          Production CICS

=====>

USER ID:

PASSWORD:
```

*Figure 133. Our Telnet menu*

This looks very much like the IBM-supplied VTAM USSTAB, except that it includes fields for entry of a user ID and password. These fields are necessary only if security has been turned on in TCP/IP for VSE/ESA (we set `SECURITY=ON` for this project).

#### 4.1.4 CICS Telnet client definition

TCP/IP for VSE/ESA also provides a CICS Telnet client. This allows CICS users to establish a Telnet session to access other TCP/IP systems that are running a Telnet daemon.

TCP/IP for VSE/ESA provides definitions for the CICS transactions and programs. These should already be in your CSD, but you should verify that they are and create them if they are not; see 2.8.5.6, “CICS definitions for TCP/IP for VSE/ESA” on page 71.

#### 4.1.5 Configuring a TN3270 client

There are a large number of TN3270 clients available for various platforms. Most will require some type of configuration before they can be used.

##### 4.1.5.1 Configuring TN3270 Plus for Windows 98

The client we chose to use on our Windows 98 workstations was TN3270 Plus. This client offers a fairly large number of configuration options, but most are not really needed to get started; for example, it offers a keyboard configuration dialog, but the client can be used without doing any reconfiguration (on the other hand, using the default key assignments may be difficult, depending on what you are used to).

Figure 134 shows the initial window when TN3270 Plus is started:



Figure 134. Initial window from TN3270 Plus

The Setup pull-down menu offers a Session configuration option, shown in Figure 135 on page 139.

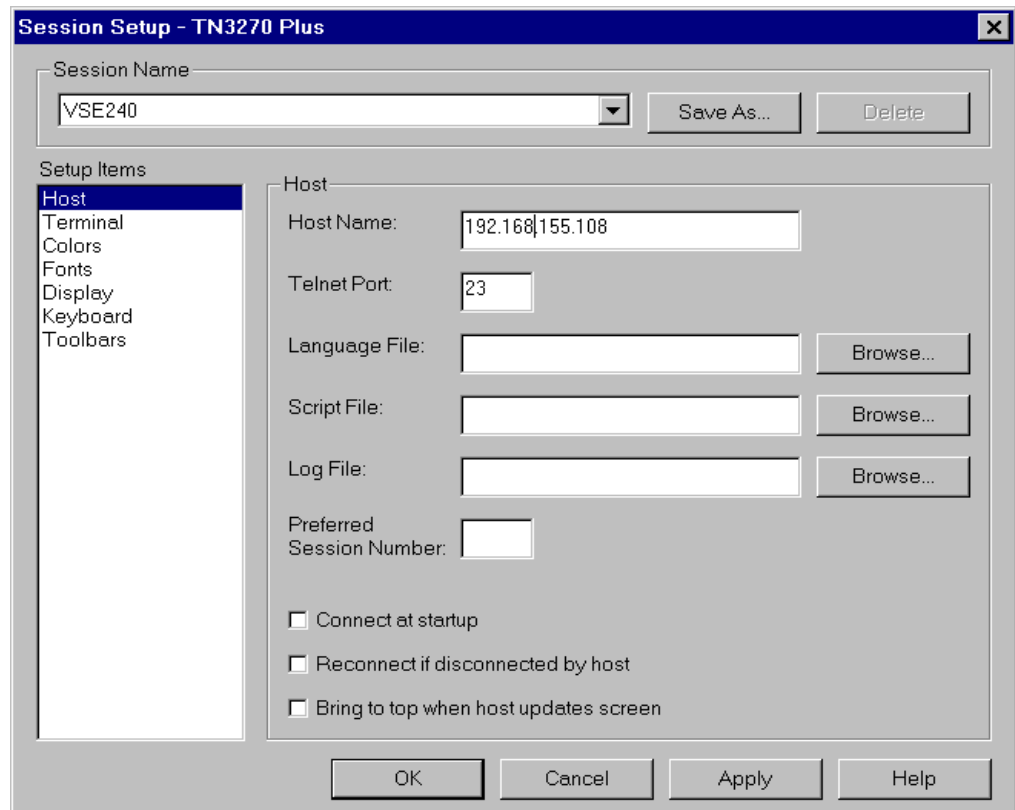


Figure 135. Configuring TN3270 Plus

Here you can create a profile for each IP address/port number combination you wish to access. You can also specify a script file to automate logging on.

## 4.2 Using a TN3270 client

The particular TN3270 client we used provides a selection menu at startup to select the previously configured session you wish to use. In our environment (see Figure 21 on page 45) we have multiple sessions defined:

- VSE240 (port 23) that connects to guest VSE240 and displays the TCP/IP menu VTAM1
- VSE240-CICSICCF (port 5023) that connects to guest VSE240 and takes us directly to VTAM application DBDCCICS (CICSICCF) without the menu
- VSE240F7 (port 23) that connects with TCP/IP in partition F7 of guest VSE240
- VSE231 (port 23) that connects with TCP/IP in guest VSE231
- VMESA (port 23) that connects with TCP/IP in our VM/ESA system

Figure 136 on page 140 shows the session selection menu from TN3270 Plus with the VSE240 session already selected.

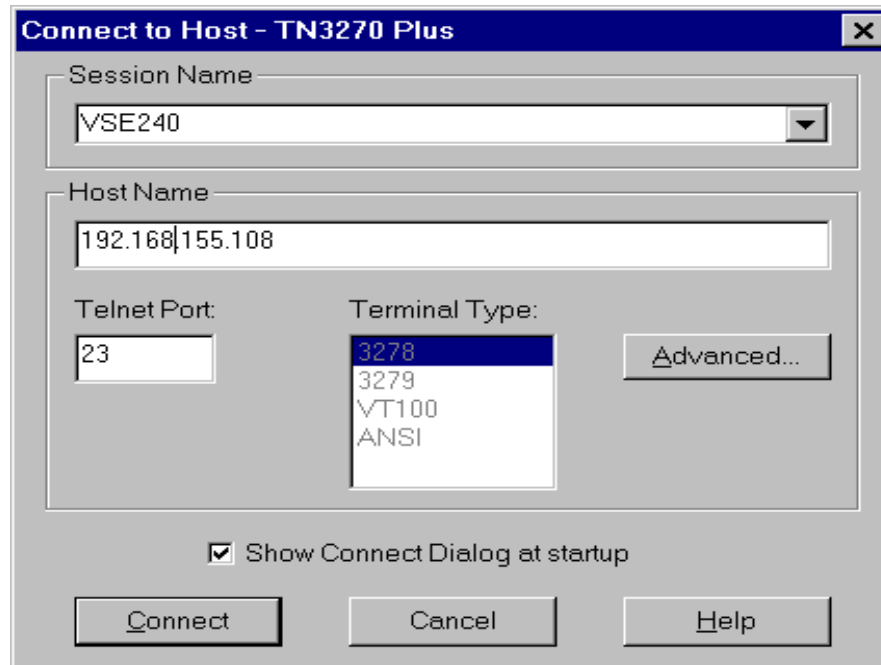


Figure 136. TN3270 client application selection menu

Selecting **VSE240**, which is configured to use the default Telnet port 23, causes one of the TELNBxx Telnet daemons to be driven, and because they specify VTAM1 as the menu, it appears as shown in Figure 133 on page 137.

If, however, you selected session **VSE240-CICSICCF**, which has the same IP address but uses port 5023, the normal VSE/ESA logo would appear as shown Figure 137, because a different daemon (TELNAxx), which does not use a menu but instead connects directly to VTAM APPL DBDCCICS, would be used:

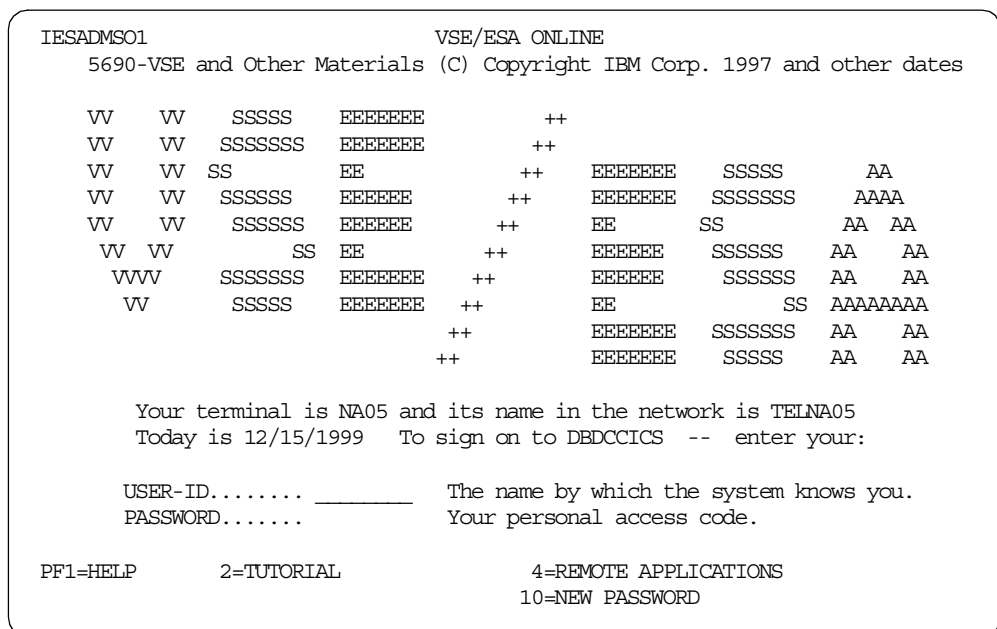


Figure 137. Connecting to CICSICCF without a menu

## 4.3 Using the Telnet CICS client

The Telnet CICS client can be started as a CICS transaction to establish a Telnet session to any other TCP/IP system that runs a Telnet daemon.

The command syntax to access another TCP/IP host is:

```
TELNET ip_addr[,port] [,ID=nn] [,LINEMODE]
```

Remember that you can use a symbolic name anywhere you would normally enter an IP address if you have defined it in the IPINIT member using the `DEFINE NAME` command. This frequently makes it much easier to initiate clients because you do not have to remember the IP address.

Using the VSE Telnet client CICS transaction is described in *TCP/IP for VSE User's Guide, Release 1.4*.

The CICS Telnet client supports both 3270 and line mode operation. If you do not specify line mode when you invoke the client, it will first attempt to connect in 3270 mode, and if the daemon on the remote host rejects that, it will try line mode.

### Note

Many remote hosts, such as UNIX and OS/2, are sensitive to text case, so you should ensure the CICS terminal is set to mixed case before invoking the Telnet client. If you use the Interactive Interface and escape to CICS using PF6, the terminal will be set for uppercase, and if the remote host expects a lowercase password, logon will fail. But escaping with PF9 will set the terminal to mixed case, and the logon will succeed.

### 4.3.1 Connecting in 3270 mode

To access our VM/ESA system from a CICS terminal using the Telnet transaction, we did the following:

```
TELNET VMESA
TEL200I Telnet client -- Startup --
TEL220I Copyright (c) 1995-1999 Connectivity Systems Incorporated
TEL211I Connecting to port: 000023 at IP: 192.168.155.125 Id:00
TEL209I Attempting to establish connection
TEL219I The PA3 Key functions as a close or escape
TEL212I Connection has been established
```

Figure 138. Starting the Telnet CICS client to VM/ESA

Once the connection was made through TCP/IP, we received the VM/ESA logo:

```

VM/ESA ONLINE
      PPPPPPPPPP      33333333      99999999      00000000
      PPPPPPPPPPPP      3333333333      999      999      0000000000
      PPP      PPP      33      333      999      999      00      00
      PPP      PP      333      999      999      00      00
      PPP      PPP      333      999      999      00      00
      PPPPPPPPPPPP      33333333      999      999      00      00
      PPPPPPPPPP      33333333      99999999      00      00
      PPP      333      999      00      00
      PPP      333      999      00      00
      PPP      33      333      999      00      00
      PPP      3333333333      999      0000000000
      PPP      333333333      999      00000000

      Virtual Machine/Enterprise Systems Architecture
      Version 2 Release 3.0 9801

Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)
USERID   ===>
PASSWORD ===>

COMMAND  ===>

RUNNING  SYS1

```

Figure 139. Connecting to VM/ESA in 3270 mode

Using the CICS client, you can connect in 3270 mode to any VSE/ESA, VM/ESA, and OS/390 (MVS) systems running TCP/IP.

### 4.3.2 Connecting in line mode to OS/2

Because we did not have access to any other systems, such as Windows 98 or Windows 95, Windows NT, or UNIX, with a Telnet daemon active, we could not demonstrate those interactions during this project. To connect to other systems using line mode would be very much like the OS/2 example described below.

To access our remote OS/2 system (P500OS2), we started the CICS transaction TELNET in a CICS screen on our local VSE system and specified the symbolic name of the remote OS/2 system as a parameter. We logged on to OS/2, changed the directory to SPOOL\HP4000, displayed the directory, renamed a member, and issued the DIR command to verify the rename worked.

Commands that can be used on the remote host depend on which ones the Telnet daemon on that host supports, but they would generally include making, deleting, and changing directories, deleting, renaming, and copying files, executing files, and so forth. You can even initiate FTP through the Telnet connection, but be careful; when initiated that way, the remote host you have "Telneted" to is the FTP client, and whatever host was specified in the FTP command is the FTP server (this can get really confusing).

The PA3 key can be used to terminate the line mode Telnet client at any time. Some daemons may also support the EXIT command.

The complete interaction is shown in Figure 140 on page 143. All of our commands are highlighted. We terminated the session using the PA3 key.



```

TELNET P500S2
TEL200I Telnet client -- Startup --
TEL220I Copyright (c) 1995-1999 Connectivity Systems Incorporated
TEL211I Connecting to port: 000023 at IP: 192.168.155.099 Id:00
TEL209I Attempting to establish connection
TEL219I The PA3 Key functions as a close or escape
TEL212I Connection has been established

OS/2 Version 2.3 (isip390)

Enter your password:
abcd
....

[<isip390>-C:\]
cd spool\hp4000
c
d spool\hp4000

[<isip390>-C:\spool\hp4000]
dir
d
ir

The volume label in drive C is OS2.
The Volume Serial Number is A7B3:C014.
Directory of C:\spool\hp4000

12-16-99  12:33p    <DIR>          0  .
1-08-99  11:43a    <DIR>          0  ..
9-03-99   1:05p     9239          0  00002.SPL
9-03-99   1:05p     555          0  abcd.xyz
      4 file(s)          9794 bytes used
                        91969024 bytes free

[<isip390>-C:\spool\hp4000]
rename 00002.spl xxxxx.abc
r
ename 00002.spl xxxxx.abc

[<isip390>-C:\spool\hp4000]
dir
d
ir

The volume label in drive C is OS2.
The Volume Serial Number is A7B3:C014.
Directory of C:\spool\hp4000

12-16-99  12:37p    <DIR>          0  .
1-08-99  11:43a    <DIR>          0  ..
9-03-99   1:05p     555          0  abcd.xyz
9-03-99   1:05p     9239          0  xxxxx.abc
      4 file(s)          9794 bytes used
                        91969024 bytes free

[<isip390>-C:\spool\hp4000]

TEL201I Telnet client -- Shutdown --
TEL214I Connection complete

```

Figure 140. Telnet CICS client connection to OS/2

---

## 4.4 Using the Telnet batch client

Sometimes it is very desirable to automate a procedure if it is to be done on a regular basis. It is also useful in reducing errors to create the dialog once instead of creating it each time you need to run it. The Telnet batch client allows you to set up a specific dialog and run it as often as you need.

The command syntax for executing the Telnet batch client is:

```
// EXEC TELNET,PARM='IP=ip_addr,PORT=port,ID=nn'
```

The commands to log on to the remote host and access and process files and directories are the same as for the CICS client.

Here is an example of a batch job to connect to OS/2 and perform the same actions as in the CICS Telnet example earlier:

```
* $$ JOB JNM=BATCHTLN,DISP=D,CLASS=0
// JOB BATCHTLN      BATCH TELNET CLIENT JOB
// EXEC TELNET,PARM='IP=P500OS2,PORT=23,ID=00'
abcd
cd spool\hp4000
dir
rename 00002.spl xxxxx.abc
dir
/*
/&
* $$ EOJ
```

*Figure 141. Batch Telnet client job*

The SYSLST output from the batch job is:

```

// JOB BATCHTLN      BATCH TELNET CLIENT JOB
// EXEC TELNET,PARM='IP=P5000S2,PORT=23,ID=00'
TEL100I TCP/IP -- Telnet Batch Client -- Version 01.04.00( ), 12/23/99
TEL101I Copyright 1995,1999 (C) Connectivity Systems Inc.
TEL105I Processing the execution parm override
TEL109I IP Address is set to 192.168.155.099
TEL110I PORT Number is set to 000023
TEL111I System ID is set to 00
TEL106I Processing of parm override complete
TEL115I Connecting with TCP/IP for VSE partition
TEL117I Now connected with partition

OS/2 Version 2.3 (isip390)

Enter your password:
abcd
.
...

[<isip390>-C:\]
cd spool\hp4000
c
d spool\hp4000

[<isip390>-C:\spool\hp4000]
dir
d
ir

```

Figure 142. Telnet batch job output (part 1 of 2)

```

The volume label in drive C is OS2.
The Volume Serial Number is A7B3:C014.
Directory of C:\spool\hp4000

12-16-99  12:43p    <DIR>            0  .
  1-08-99  11:43a    <DIR>            0  ..
  9-03-99   1:05p     9239            0  00002.SPL
  9-03-99   1:05p     555            0  abcd.xyz
          4 file(s)          9794 bytes used
                                91969024 bytes free

[<isip390>-C:\spool\hp4000]
rename 00002.spl xxxxx.abc
r
ename 00002.spl xxxxx.abc

[<isip390>-C:\spool\hp4000]
dir
d
ir

The volume label in drive C is OS2.
The Volume Serial Number is A7B3:C014.
Directory of C:\spool\hp4000

12-16-99  12:45p    <DIR>            0  .
  1-08-99  11:43a    <DIR>            0  ..
  9-03-99   1:05p     555            0  abcd.xyz
  9-03-99   1:05p     9239            0  xxxxx.abc
          4 file(s)          9794 bytes used
                                91969024 bytes free

[<isip390>-C:\spool\hp4000]
TEL102I TCP/IP -- Telnet Batch Client -- Complete
TEL118I Terminating connection
TEL120I Connection has closed, successfully
1S55I  LAST RETURN CODE WAS 0000
EOJ BATCHTLN  MAX.RETURN CODE=0000

```

Figure 143. Telnet batch job output (part 2 of 2)

## 4.5 Managing Telnet daemons

At any time you can QUERY or DELETE the Telnet daemons, as well as define new ones using DEFINE. This would normally be done using the VSE console by replying to the prompt from the TCP/IP for VSE/ESA partition. If you do not have any outstanding prompt, you can get one by issuing the `MSG part_id` command.

### 4.5.1 QUERY TELNETDS command

The command format is:

```
Query TELnetds [,ID=name}{,SYSLST]
```

The `QUERY TELNETDS` command without the `ID=` operand displays all defined Telnet daemons with the corresponding status (active or inactive):

```

87 query telnetds
Zl 0085 IPN469I (( TCP/IP TELNET Daemons ))
Zl 0085 IPN470I ID: TRMB01
Zl 0085 IPN471I Terminal: TELNB01 Target:
Zl 0085 IPN472I Port: 23 Driver: TELNETD
Zl 0085 IPN473I Menu: VTAM1
Zl 0085 IPN549I Logmode2: S3270 Logmode3: D4B32783
Zl 0085 IPN356I Logmode4: D4B32784 Logmode5: D4B32785
Zl 0085 IPN350I Current Status: Inactive
Zl 0085 IPN470I ID: TRMB02
Zl 0085 IPN471I Terminal: TELNB02 Target:
Zl 0085 IPN472I Port: 23 Driver: TELNETD
Zl 0085 IPN473I Menu: VTAM1
Zl 0085 IPN549I Logmode2: S3270 Logmode3: D4B32783
Zl 0085 IPN356I Logmode4: D4B32784 Logmode5: D4B32785
Zl 0085 IPN350I Current Status: Inactive
Zl 0085 IPN470I ID: TRMB03
Zl 0085 IPN471I Terminal: TELNB03 Target:
Zl 0085 IPN472I Port: 23 Driver: TELNETD
Zl 0085 IPN473I Menu: VTAM1
Zl 0085 IPN549I Logmode2: S3270 Logmode3: D4B32783
Zl 0085 IPN356I Logmode4: D4B32784 Logmode5: D4B32785
Zl 0085 IPN350I Current Status: Active
Zl 0085 IPN351I Current IPaddr: 192.168.155.118
Zl 0085 IPN352I Current Applid: DBDCCICS
Zl 0085 IPN354I Current Logmod: S3270
Zl 0085 IPN355I Current Userid:
.
.
.

```

Figure 144. Display the status of all Telnet daemons

To query the status of an individual Telnet daemon, use the following format of the `QUERY` command:

```

87 query telnetds,id=trmb05
Zl 0085 IPN470I ID: TRMB05
Zl 0085 IPN471I Terminal: TELNB05 Target:
Zl 0085 IPN472I Port: 23 Driver: TELNETD
Zl 0085 IPN473I Menu: VTAM1
Zl 0085 IPN549I Logmode2: S3270 Logmode3: D4B32783
Zl 0085 IPN356I Logmode4: D4B32784 Logmode5: D4B32785
Zl 0085 IPN350I Current Status: Active
Zl 0085 IPN351I Current IPaddr: 192.168.155.120
Zl 0085 IPN352I Current Applid: DBDCCICS
Zl 0085 IPN354I Current Logmod: S3270
Zl 0085 IPN355I Current Userid:
Zl 0087 IPN300I Enter TCP/IP Command
Zl-0087

```

Figure 145. Display the status of individual Telnet daemon

## 4.5.2 DELETE TELNETD command

The `DELETE TELNETD` command allows you to dynamically remove a Telnet daemon from your configuration:

```
DELEte TELnetd, ID=name [, FORCE]
```

For example:

```
87 del tel, id=trma03  
Z1 0085 0019: TEL918I Daemon Shutdown Telnet Termname:TELNA03
```

*Figure 146. Delete a Telnet daemon*

---

## Chapter 5. Network printing using LPD/LPR

TCP/IP for VSE/ESA provides support for your VSE system to participate in network printing using the TCP/IP Line Printer Daemon (LPD) and Line Printer Requester (LPR) applications. Your VSE system can be an LPD print server, an LPR client, or both.

The TCP/IP Line Printer Daemon (LPD) allows you to utilize VSE host-based printer resources for any TCP/IP client in your network that supports the line printer protocol. LPD provides this support by interfacing with the POWER LST queue, so you can essentially do file transfers directly into the print queues and let POWER take it from there. LPD supports some other capabilities as well, for example, the ability to print directly to VSE disk-based files such as a sublibrary member or a VSAM file.

The TCP/IP clients use the Line Printer Requester (LPR) application to send print data to the LPD running on VSE. On workstations, the LPR client can easily be started via a window or full screen command-line interface.

TCP/IP for VSE/ESA also provides a Line Printer Requester (LPR) client on VSE. Using the LPR client, you can send print data from your VSE system to any other TCP/IP host that runs an LPD.

---

### 5.1 Defining the Line Printer Daemon on VSE

Before any clients can make use of the LPDs running on VSE, each LPD has to be defined in the TCP/IP for VSE initialization file IPINITxx.L.

During TCP/IP for VSE startup, each `DEFINE LPD` command in the TCP/IP initialization file IPINITxx starts a Line Printer Daemon (server). A single Line Printer Daemon can handle any number of simultaneous requests.

The syntax of the `DEFINE LPD` statements can be found in *TCP/IP for VSE Commands, Release 1.4*.

Figure 147 on page 150 shows the part of our TCP/IP initialization file IPINIT04.L where we defined five different Line Printer Daemons.

```

      . . .
      . . .
*-----*
*
*          LINE PRINTER DAEMONS
*
*-----*
DEFINE LPD,PRINTER=FAST,QUEUE='POWER.LST.A'
DEFINE LPD,PRINTER=FASTLIB,QUEUE='TCPIP.PRINT'
DEFINE LPD,PRINTER=LOCAL,QUEUE='POWER.LST.A',LIB=TCPIP,SUBLIB=TEMP
DEFINE LPD,PRINTER=KSDS,QUEUE='KSDSTCP.PRINT',LIB=TCPIP,SUBLIB=TEMP
DEFINE LPD,PRINTER=ESDS,QUEUE='ESDSTCP.PRINT',LIB=TCPIP,SUBLIB=TEMP
*

```

Figure 147. Line Printer Daemon definitions in our IPINIT04 file

### General parameter explanation

**PRINTER=** This parameter sets the unique name of the LPD printer. The LPR clients have to specify this name as the printer when they want to send data to this LPD.

**QUEUE=** This parameter specifies the destination for the print data. You can specify POWER queues, VSE libraries, or VSAM KSDS or ESDS files if they are defined in the TCP/IP file system as public file names.

Defining these public file names to TCP/IP is done with the `DEFINE FILE` command (see 3.1.3, “Defining the VSE file system” on page 85).

**LIB= ,SUBLIB=** You can specify a VSE sublibrary where the print data is temporarily stored until completely transferred to its destination. If no LIB/SUBLIB is specified, the print data will be temporarily stored in the TCP/IP partition’s GETVIS storage instead. This increases the print performance but puts a limitation on large print files.

### Explanation of our LPD definitions

```

DEFINE LPD,PRINTER=FAST,QUEUE='POWER.LST.A'
DEFINE LPD,PRINTER=LOCAL,QUEUE='POWER.LST.A',LIB=TCPIP,SUBLIB=TEMP

```

With these commands, we defined two LPD printers, FAST and LOCAL, that put the data into the POWER list queue in class A. POWER is defined as a public name in the TCP/IP file system (see 3.1.3, “Defining the VSE file system” on page 85).

When a client LPR prints to this daemon, it may pass a file name. If this file name meets the syntactic requirements of a POWER queue entry, the file name will be used. Otherwise, the queue entry name will be generically built as LPDFAnnn, where *nnn* is the three-digit number transmitted by the LPR client.



Because the first LPD definition does not have the LIB/SUBLIB defined all print data for this LPD is stored in the GETVIS area of the TCP/IP partition. Therefore, this LPD should mainly be used for small print output only.

```
DEFINE LPD, PRINTER=FASTLIB, QUEUE='TCPIP.PRINT'
```

We used this LPD to send data from an LPR client into the VSE sublibrary TCPIP.PRINT. When a client LPR establishes a link to this daemon, it may pass a file name. If this job name meets the syntactic requirements of a member name, it will be stored using this name. Otherwise, the member will have the generic name LPDFA $nnn$ , where  $nnn$  is the three-digit number transmitted by the LPR client. The member type will be derived from the transmitted *origin name*. If no origin name is available, or if it is unsuitable for use as a member type, the member type will be LISTING. If a duplicate member already exists, it will be overwritten.

```
DEFINE LPD, PRINTER=KSDS, QUEUE='KSDSTCP.PRINT', LIB=TCPIP, SUBLIB=TEMP  
DEFINE LPD, PRINTER=ESDS, QUEUE='ESDSTCP.PRINT', LIB=TCPIP, SUBLIB=TEMP
```

These two Line Printer Daemons can be used to write data to VSAM files from an LPR client. Writing to an ESDS file appends the records sent by the LPR client to the end of the data set whereas writing to a KSDS file performs a VSAM INSERT operation of the records.

### 5.1.1 LPD operation commands

On the VSE system, Line Printer Daemons can be defined or deleted dynamically with TCP/IP commands that are issued on the VSE console to the partition where TCP/IP for VSE/ESA is running. A detailed description of the LPD commands can be found in *TCP/IP for VSE Commands, Release 1.4*.

#### 5.1.1.1 The QUERY LPDS command

The QUERY LPDS command permits you to display the status of currently defined Line Printer Daemons. Figure 148 on page 152 shows an example of what we entered on our VSE console in order to verify the LPDs that we had defined previously.

```

msg z1
AR 0015 1I40I  READY
Z1 0087 IPN300I Enter TCP/IP Command
Z1-0087
87 query lpds
Z1 0085 IPN443I (( TCP/IP Line Printer Daemons ))
Z1 0085 IPN444I Print Name: ESDS
Z1 0085 IPN445I Queue: ESDSTCP.PRINT.
Z1 0085 IPN446I Library: TCPIP .TEMP
Z1 0085 IPN444I Print Name: KSDS
Z1 0085 IPN445I Queue: KSDSTCP.PRINT.
Z1 0085 IPN446I Library: TCPIP .TEMP
Z1 0085 IPN444I Print Name: LOCAL
Z1 0085 IPN445I Queue: POWER.LST.A.
Z1 0085 IPN446I Library: TCPIP .TEMP
Z1 0085 IPN444I Print Name: FASTLIB
Z1 0085 IPN445I Queue: TCPIP.PRINT.
Z1 0085 IPN446I Library: MEMORY .
Z1 0085 IPN444I Print Name: FAST
Z1 0085 IPN445I Queue: POWER.LST.A.
Z1 0085 IPN446I Library: MEMORY .
Z1 0087 IPN300I Enter TCP/IP Command
Z1-0087

```

Figure 148. Example of the QUERY LPDS command

## 5.2 Access the VSE Line Printer Daemon from a workstation

Your interface to the VSE Line Printer Daemon will primarily be through an LPR client residing on a TCP/IP workstation, UNIX-based processor or other TCP/IP platforms. LPR capability varies widely between the different applications, so you will want to select TCP/IP application software that mirrors your requirements.

OS/2, Windows NT and UNIX platforms provide an LPR client as part of the standard TCP/IP support. The LPR client allows you to enter LPR commands via the command-line interface on these platforms. For some platforms such as Windows 95 and Windows 98 you have to acquire LPR client software separately.

The two LPR command parameters that are required by the VSE LPD are the printer name and the server (or host) name. Additionally, you could specify other options such as a job name or the job disposition. The printer names for the VSE LPDs must have been defined to TCP/IP for VSE/ESA with the DEFINE LPD command.

To check for the printers that are defined on the VSE system, you can use the TCP/IP for VSE/ESA command QUERY LPDS on the VSE console (see Figure 148).

Here we will show you an example of how to send a file from our OS/2 workstation to an LPD (printer) on our VSE system. We used the printer FAST, which is defined to transfer the data sent from the LPR client into the POWER list queue with class A. The IP address of our VSE system is 192.168.155.108.

To print the workstation file CONFIG.SYS from drive C:, we issue the following LPR command from an OS/2 command line window:

```
lpr -p fast -s 192.168.155.108 config.sys
```

Having entered the `LPR` command, the following was displayed in the OS/2 window:

```

OS/2 Window
OS/2      Ctrl+Esc = Window List      Type HELP = help
[C:\]lpr -p fast -s 192.168.155.108 config.sys

Printing C:\CONFIG.SYS:
Trying LPD print server 192.168.155.108(192.168.155.108), devi
Sent 8647 bytes.
The entire document was sent.

[C:\]

```

Figure 149. After issuing the `LPR` command

Now, we checked the contents of the `POWER` list queue using the `VSE` Interactive Interface.

|                                   |                 |              |     |             |          |          |          |          |          |      |      |                  |
|-----------------------------------|-----------------|--------------|-----|-------------|----------|----------|----------|----------|----------|------|------|------------------|
| IESBQUL                           |                 | LIST QUEUE   |     | Page 1 of 1 |          |          |          |          |          |      |      |                  |
|                                   |                 |              |     | Class: A    |          |          |          |          |          |      |      |                  |
| OPTIONS: 1 = DISPLAY              |                 | 2 = CHANGE   |     | 3 = PRINT   |          |          |          |          |          |      |      |                  |
|                                   |                 |              |     | 5 = DELETE  |          |          |          |          |          |      |      |                  |
| OPT                               | JOBNAME         | NUMBER       | SFX | S           | PR       | DIS      | CL       | PAGES    | CC       | FORM | TO   | FROM             |
| —                                 | CICSICCF        | 00666        |     |             | 4        | D        | A        | 35       | 1        |      | SYSB | .SYSB            |
| —                                 | VIAMSTR         | 00664        |     |             | 4        | D        | A        | 4        | 1        |      | SYSB | .SYSB            |
| —                                 | TCPAPPL         | 00624        |     |             | 3        | D        | A        | 3        | 1        |      | TCPA | .TCPA            |
| —                                 | IPINIT07        | 00627        |     |             | 3        | D        | A        | 3        | 1        |      | TCPA | .TCPA            |
| —                                 | PAUSEF4         | 00633        |     |             | 3        | D        | A        | 2        | 1        |      | SYSA | .SYSA            |
| —                                 | TPRINT          | 00680        |     |             | 3        | D        | A        | 5        | 1        |      | SYSG | .SYSG            |
| —                                 | JCLPROCS        | 00691        |     |             | 3        | D        | A        | 4        | 1        |      | SYSH | .SYSH            |
| —                                 | INSERTS         | 00713        |     |             | 3        | D        | A        | 14       | 1        |      | SYSB | .SYSB            |
| —                                 | IPINIT04        | 00725        |     |             | 3        | D        | A        | 3        | 1        |      | SYSD | .SYSD            |
| —                                 | <b>LPDFA714</b> | <b>00735</b> |     |             | <b>3</b> | <b>D</b> | <b>A</b> | <b>1</b> | <b>1</b> |      |      | <b>.SYSTCPIP</b> |
| PF1=HELP 2=REFRESH 3=END 4=RETURN |                 |              |     |             |          |          |          |          |          |      |      |                  |
| LOCATE JOBNAME ==> _____          |                 |              |     |             |          |          |          |          |          |      |      |                  |

Figure 150. Display the `POWER` list queue after `LPR`

Our PC file was transferred, but because we did not specify a job name, TCP/IP for VSE/ESA generated the new POWER list queue entry as LPDFA714. The LPD also sets the FROM user ID of this entry automatically to SYSTCPIP.

---

## 5.3 Using the Line Printer Requester on VSE

TCP/IP for VSE/ESA also provides a Line Printer Requester (LPR) client on VSE. Using this LPR client, you can send files from your VSE system to any other TCP/IP host that runs an LPD.

The following four methods can be used to invoke the LPR client on the VSE system:

- Interactively, through the CICS transaction LPR
- Automatically, through the automatic LPR client (AUTOLPR)
- Via a batch job, through the LPR batch client
- From an application program, using the REXX, Assembler, COBOL, or PL/I Sockets Interface

We describe the first three methods in this chapter. Refer to *TCP/IP for VSE Programmer's Reference, Release 1.4* for guidance on using Sockets to invoke the LPR client.

### 5.3.1 Using the CICS transaction LPR

On the VSE system, the LPR can be started as a CICS transaction from a CICS command mode screen by just entering LPR. This transaction will then prompt you for input, allowing you to specify various `LPR` commands in the CICS terminal session.

All `LPR` commands for the VSE LPR Client are described in *TCP/IP for VSE Commands, Release 1.4*.

#### Note

Before you can start the LPR as a CICS transaction, CICS must have been customized accordingly. We described the required setup in 2.8.5.6, "CICS definitions for TCP/IP for VSE/ESA" on page 71.

We used the VSE LPR client to print a POWER list queue entry on:

- Our second VSE system
- Our network printer

As a *first step*, we started the LPD server on the target system, that is, we started TCP/IP for VSE/ESA on our second VSE system P330VSE (IP address 192.168.155.112) where we had defined an LPD with `PRINTER=FAST`. The LPD on our network printer is built into the HP JetDirect card and automatically starts when the printer is powered on.

In the *second step*, we used a terminal emulator session to sign on to CICS. On the CICS command screen, we entered LPR to start the CICS transaction LPR on

VSE. In the following CICS screens, the commands that we entered are highlighted in bold.

```
LPR
TCP200I Client -- Startup --
TCP207I Copyright (c) 1995-1999 Connectivity Systems Incorporated
TCP202I Attempting to Establish Connection
TCP204I Connection has been Established
Client manager connection Established.
LPR Ready:
```

Figure 151. Invoking the CICS transaction LPR

Now, we set the target host system to P330VSE (our second VSE system at IP address 192.168.155.112) and the printer we want to use, which is FAST with the following commands:

```
LPR Ready:
SET HOST=P330VSE
192.168.155.112
LPR Ready:
SET PRINTER=FAST
```

Figure 152. SET HOST and PRINTER

Note that TCP/IP resolves the name to the IP address specified with the `DEFINE NAME` commands in our configuration and the LPR client displays the IP address.

We want to print the list output from the last run of the TCP/IP for VSE/ESA (POWER list queue entry TCPSZ1). We use `CD` commands to navigate to the VSE POWER LST queue class A entries (POWER.LST.A) and use the `PRINT` command to actually send this entry to the target LPD. The `Print Working Directory (PWD)` command is used to verify that we are positioned in the correct directory. The `DIR` command displays the contents of the POWER.LST.A directory. This is how our CICS screen looked:

```

LPR Ready:
CD POWER.LST.A
Change has completed.
LPR Ready:
PWD
"POWER.LST.A"
LPR Ready:
DIR
CICSICCF.00666.00      1893      35      1,856  4 D SYSB
VTAMSTRT.00664.00      26        4        22  4 D SYSB
TCPAPPL.00624.00       12        3        8  3 D TCPA
IPINIT07.00627.00      12        3        8  3 D TCPA
PAUSEF4.00633.00        7        2        5  3 D SYSA
TPRINT.00680.00       115        5       109  3 D SYSG
JCLPROCS.00691.00       61        4        56  3 D SYSH
INSERTS.00713.00       191       14       175  3 D SYSB
IPINIT04.00725.00       12        3        8  3 D SYSD
LPDFA714.00735.00      227        1       227  3 D SYSTCPIP
TCPSZ1.00696.00      426        7       413  3 L SYSB
TCPSF7.00679.00       367        7       354  3 L TCPA
LPR Ready:

```

Figure 153. Navigate to the POWER list queue

As we found the queue entry TCPSZ1, which is highlighted above, we now print this entry via the FAST printer of the second VSE system by using the `PRINT` command.

```

PRINT TCPSZ1.00696.00
Opening the file
Establishing connection with LP Daemon
Remote port:  515  Local Port:  721
Request the queue for output
Checking LP daemon
Transferring the data file
Data file transfered.
Transferring the control file
Control file transfered.
Closing the connection with LP Daemon
Connection complete

```

Figure 154. Print to the remote LPD on VSE system P330VSE

Note that the command `PRINT TCPSZ1` would be valid also, that is, the spool ID of the POWER queue entry can be omitted. If multiple files exist with the same file name, TCP/IP will print the queue entry with the lowest spool ID.

The transfer has been successfully completed. Using the same session, we now also want to print the TCP/IP for VSE/ESA initialization file IPINIT04.L in sublibrary TCPIP.CONFIG to the same printer.

We used the following highlighted commands to do that:

```

LPR Ready:
CD ..
Change has completed.
LPR Ready:
PWD
"POWER.LST"
LPR Ready:
CD ..
Change has completed.
LPR Ready:
CD ..
Change has completed.
LPR Ready:
PWD
""
LPR Ready:
CD TCPIP.CONFIG
Change has completed.
LPR Ready:
DIR IPINIT*.L
IPINIT04.L          14720          7  99/12/02  13:34  99/12/08  16:37  FN
IPINIT07.L          9200          5  99/12/06  11:35  99/12/07  16:04  FN
IPINIT08.L          9040          5  99/12/06  11:28             FN
LPR Ready:

```

Figure 155. Get to the TCP/IP for VSE/ESA initialization file

The `PWD` command is used again to verify that we are positioned to the root directory ("" ) before we issue the `CD` command to position to a new directory.

Because the member being printed does not have standard carriage control characters, we first turned off the CC option. To check all the options in effect, we used the `QUERY OPT` command with the following result:

```

SET CC=NO
LPR Ready:
QUERY OPT
HOST           :192.168.155.112
PRINTER        :FAST
FCB             :
INSERTS         :
CC              :NO
CRLF            :YES
NOEJECT         :NO
DELREQ         :YES
DISP            :KEEP
USER            :TCPIP
PASSWORD        :
CLASS           :
COPIES          :01
JOBNAME         :
MAIL            :
TITLE           :
TRANSLATE       :US_ENG_03
LPR Ready:
More...

```

Figure 156. SET CC off, and QUERY OPT commands

If the output display exceeds the 3270 screen the character string More... is displayed. Pressing the Clear key displays a new screen with the remainder of the output.

Now we can print the file using the `PRINT` command.

```
PRINT IPINIT04.L
Opening the file
Establishing connection with LP Daemon
Remote port:  515  Local Port:  721
Request the queue for output
Checking LP daemon
Transferring the data file
Data file transfered.
Transferring the control file
Control file transfered.
Closing the connection with LP Daemon
Connection complete
LPR Ready:
```

Figure 157. Print the IPINIT04.L sublibrary member

Additionally, we wanted to print our IPINIT04.L member on our network printer. The name defined for our network printer where the LPD was running is HP4000. For HP printers you can choose between the two queues TEXT or RAW, where the TEXT queue can be used for unformatted text. For details of the differences refer to the description of the `SET PRINTER` command in *TCP/IP for VSE User's Guide, Release 1.4*. Still using our CICS LPR session, we only had to set the HOST and the PRINTER parameters to the values for the PC printer as shown in Figure 158:

```
SET HOST=HP4000
192.168.155.103
LPR Ready:
SET PRINTER=TEXT
LPR Ready:
PRINT IPINIT04.L
Opening the file
Establishing connection with LP Daemon
Remote port:  515  Local Port:  721
Request the queue for output
Checking LP daemon
Transferring the data file
Data file transfered.
Transferring the control file
Control file transfered.
Closing the connection with LP Daemon
Connection complete
LPR Ready:
```

Figure 158. Print a file on our network printer

#### 5.3.1.1 SHORT and LONG commands from the VSE LPR client

The LPR client on VSE also supports query commands. To obtain printer queue information, such as how many jobs are sent to a printer, job name, current job printing status and so on from LPD servers, you can use the VSE LPR commands `LONG` or `SHORT`. These commands display this information for LPDs that manage



print queues, however, they do not display any information for LPDs such as the HP JetDirect printers that do not spool their print request.

To show the difference between the two commands, we first issued the `LONG` command in our CICS LPR client session to an LPD running on one of our PCs.

```
SET HOST=PC2
204.155.155.121
LPR Ready:
SET PRINTER=PRINTQ01
LPR Ready:
LONG
Establishing connection with LP Daemon
Remote port: 515 Local Port: 721
Requesting the queue information
Receiving LP daemon response
Queue 'PRINTQ01' is Enabled, Processing, Holding
Queue type: Text, printer Default PostScript Printer, winspool, LPT1:
Queue status: Printer 'Default PostScript Printer' is defined
Seq # UserID Files Size Date/Time
11 AUTOLPR GPSRPT01.PRINT 28 12/21/99 12:48:22 PM
14 VSEUSER IPINIT04.L 9109 12/23/99 06:59:31 PM
Closing the connection with LP Daemon
Connection complete - Already closed
LPR Ready:
```

Figure 159. The VSE LPR command `LONG`

Because the `HOST` is set to `PC2` (the name of our PC workstation) and the `PRINTER` to `PRINTQ01` (our workstation LPD printer queue), the `LONG` command displayed the `PRINTQ01` printer queue on our workstation where the `IPINIT04.L` and other files were queued to be printed.

When using the `SHORT` command, we received the following data on our CICS screen:

```
LPR Ready:
SHORT
Establishing connection with LP Daemon
Remote port: 515 Local Port: 721
Requesting the queue information
Receiving LP daemon response
Seq # UserID Files Size
11 AUTOLPR GPSRPT01.PRINT 28
14 VSEUSER IPINIT04.L 9109
Closing the connection with LP Daemon
Connection complete - Already closed
LPR Ready:
```

Figure 160. Report of the `SHORT` command from the LPR client

After successfully printing on our network printer, we finally terminated the CICS LPR transaction with the `QUIT` command. This is shown in Figure 161:

```

QUIT
TCP201I Client -- Shutdown --
TCP205I Connection Complete -- Already Closed

```

Figure 161. Leave the CICS LPR client transaction

## 5.4 The automatic LPR client

You can use the automatic LPR client (AUTOLPR) to automatically send the output from a VSE job to a given LPD server that is identified by the HOST= and PRINTER= parameter values. To enable this facility, you need to:

- Define one or more POWER LST queue classes that are to be used for automatic printing. This is done with the TCP/IP for VSE/ESA command `DEFINE EVENT`.
- Change parameters in the POWER LST statement in the VSE batch jobs to specify the destination HOST and PRINTER.
- Code and assemble an INSERTS phase if you want to control printer formatting.

### 5.4.1 Defining events

For the automatic LPR client, we have to define one or more POWER LST queue classes to be monitored by TCP/IP for VSE/ESA. This is done with the `DEFINE EVENT` command, which permits you to establish actions to be performed when the specified events occur on your VSE system. The general syntax of this command is documented in *TCP/IP for VSE Commands, Release 1.4*.

On our VSE system, we defined such an event in the TCP/IP for VSE/ESA initialization file. Figure 162 shows the part of our file IPINIT04.L where we defined the event LST\_LISTEN:

```

. . .
*-----*
*                                     *
*      Automated Line Printer Client  *
*                                     *
*-----*
DEFINE EVENT,ID=LST_LISTEN,TYPE=POWER,CLASS=M,QUEUE=LST,ACTION=LPR
*-----*
*                                     *
*      Define LPR Scripts             *
*                                     *
*-----*
DEFINE NAME,NAME=LPRSCR01,SCRIPT=LPRSCR01
. . .

```

Figure 162. `DEFINE EVENT` and `SCRIPT` commands in initialization file IPINIT04.L

This `DEFINE EVENT` causes TCP/IP for VSE/ESA to monitor class M of the POWER LST queue and to start the LPR client automatically with queue entries that are put into this class.

When TCP/IP for VSE/ESA is up and running, you can use the `QUERY EVENTS` command in a VSE console session to check for the active events that are defined to TCP/IP for VSE/ESA.

On our VSE console, we first entered `MSG Z1` (TCP/IP for VSE/ESA is running in partition Z1) followed by the `QUERY EVENTS`. The following figure shows how our VSE console looked:

```
msg z1
AR 0015 1I40I  READY
Z1-0087 IPN300I Enter TCP/IP Command
87 query events
Z1 0085 IPN425I (( TCP/IP Events ))
Z1 0085 IPN587I Event Pause Interval: 9000 (30 Seconds)
Z1 0085 IPN426I Event ID: LST_LISTEN
Z1 0085 IPN427I Type: POWER Driver:
Z1 0085 IPN428I Class: M Queue: LST Action: LPR
Z1 0085 IPN429I Host field: USER User ID: SYSTCPIP Single: N
Z1 0085 IPN430I Action: LPR Retries: 1 Time: 13500
Z1-0087 IPN300I Enter TCP/IP Command
```

Figure 163. The `QUERY EVENTS` command

## 5.4.2 Using the automatic LPR client

Once a given LST queue is being monitored, TCP/IP looks at the `USER` and `DEST` parameters of each output file that is queued for the class. The contents of the `USER` parameter are assumed to be the host where the output should be sent. TCP/IP for VSE/ESA then looks at the destination parameter (`DEST=`) to determine the printer name to use. The syntax of the `POWER LST` statement would look like this:

```
* $$ LST CLASS=class,DISP=D,UINF=name,DEST=(*,printer)
```

Figure 164. Syntax of the `POWER LST` statement

**class** Identifies the `POWER LST` queue that TCP/IP for VSE/ESA is monitoring. This queue is specified to TCP/IP for VSE/ESA by way of the `DEFINE EVENT` command.

**name** This field specifies one of three possible values:

1. The IP address of the host owning the printer. For example, "UINF=192.168.155.103".
2. Symbolic host name of the IP address, as set with the TCP/IP for VSE/ESA `DEFINE NAME` command as defined in our configuration in Figure 37 on page 59. For example, UINF=HP4000.
3. Name of a script that contains `LPR` commands, as set with the TCP/IP for VSE/ESA `DEFINE NAME` command shown in Figure 162 on page 160. For example, UINF=LPRSCR01. A script example is described in 5.4.3, "Using scripts for the automatic LPR client" on page 162. You can learn more about using scripts in *TCP/IP for VSE User's Guide, Release 1.4*.

**Note:** The `POWER UINF` parameter was first available with VSE/ESA 2.3; this is the recommended usage. It is also possible to use the

USER parameter of the \$\$ LST statement to provide this information to TCP/IP.

**printer** Names the specific printer where you want your output to appear. If you are using a script file containing a SET PRINTER= command, then this DEST= parameter is not required.

Because we had already defined the required EVENT and NAME in our TCP/IP for VSE/ESA initialization file (see Figure 162 on page 160), we only have to add the appropriate POWER LST statement in a VSE job in order to have the job output automatically printed via the LPR client on the printer of the target system.

Figure 165 shows which POWER LST statement we used to print the VSE job output on our network printer.

```
* $$ JOB JNM=AUTOLPR,CLASS=0,DISP=D
* $$ LST CLASS=M,DISP=K,DEST=(*,TEXT),UINF=HP4000
// JOB AUTOLPR          TCPIP AUTOMATIC LPR EXAMPLE
// EXEC LIBR
ACC S=TCPIP.CONFIG
L IPINIT04.L
/*
/&
* $$ EOJ
```

Figure 165. Job using the automatic LPR client

After submitting the job, its output was directly sent to our network printer. Figure 166 shows the VSE console output from the TCP/IP partition that processes the LPR event.

```
Z1 0085 0027: TCP911I Processing Event:LST_LIST Type:POWER Action:LPR
Z1 0085 0027: TCP907I Processing Output:AUTOLPR ,00758-000 for automated
Z1 0085 printing at:HP4000          on:TEXT
Z1 0085 0027: TCP909I Processing Output:AUTOLPR ,00758-000 has completed
```

Figure 166. VSE console output for the automatic LPR event

### 5.4.3 Using scripts for the automatic LPR client

Due to the minimal amount of information that can be passed on the \* \$\$ LST statement, TCP/IP for VSE/ESA implements a scripting facility. In a script, you may specify a variety of LPR commands that will control the print operation.

To use a script with the automatic LPR client, we did the following:

1. We prepared a script by coding LPR commands and cataloging them in a member LPRSCR01.L in a sublibrary of the TCP/IP partition search chain.
2. We defined the LPR script with the symbolic name LPRSCR01 in our TCP/IP configuration file (see Figure 162 on page 160).
3. We specified this symbolic name LPRSCR01 in the UINF parameter of the \* \$\$ LST statement for our VSE job.

A script can also be executed with the `EXEC` command. The `EXEC` command can be issued during an LPR session invoked by the CICS transaction LPR or by the batch LPR client (see “Using the INSERTS phase to control printers” on page 163).

The job stream to catalog our LPR script sublibrary member LPRSCR01.L, referenced by LPRSCR01 in the `DEFINE NAME` statement is shown in the following figure:

```
* $$ JOB JNM=LPRSCRIPT,CLASS=0,DISP=D
// JOB LPRSCRIPT CATALOG LPR SCRIPT FILE
// EXEC LIBR
ACC S=TCPIP.CONFIG
CATALOG LPRSCR01.L                      REPLACE=YES
SET HOST      = HP4000
SET PRINTER = TEXT
SET CC        = YES
SET INSERTS = HP4LAND
/+
/*
/&
* $$ EOJ
```

Figure 167. Job stream to catalog our script file LPRSCR01.L

To use this script in connection with the automatic LPR client, we only had to specify the symbolic name LPRSCR01 in the `UINF=` parameter of the `* $$ LST` statement in our test job as shown in Figure 168:

```
* $$ JOB JNM=AUTOLPR,CLASS=0,DISP=D
* $$ LST CLASS=M,DISP=K,UINF=LPRSCR01
// JOB AUTOLPR      TCPIP AUTOMATIC LPR WITH SCRIPT EXAMPLE
// EXEC LIBR
ACC S=TCPIP.CONFIG
L IPINIT04.L
/*
/&
* $$ EOJ
```

Figure 168. Job using the automatic LPR client with a script file

## 5.5 Using the INSERTS phase to control printers

Desktop printers generally have more features than the mainframe printers that they replace. In most cases, you need to set modes of operation before you print your reports. For example, you may need to set landscape mode or a monospaced font.

The LPR support in TCP/IP for VSE/ESA has a facility that permits you to include control data that is automatically merged with the print stream. Control data may precede the report, precede each page of the report, and follow the last page of the report. To accomplish this, you generate INSERTS phases containing printer control codes used by the network printers to which your VSE LPR clients will be sending print requests.

### 5.5.1 Creating the INSERTS phase

The INSERTS macro is coded once to generate each INSERTS phase. The macro is coded as follows:

```
phase      INSERTS DEFINE [,HEADER=head]          *
              [,TRAILER=trail]                    *
              [,PAGE=page]
```

Figure 169. INSERTS macro format

The parameters in lowercase are explained below:

- phase**     The name to be assigned to the INSERTS phase.
- head**     A string of hexadecimal characters (0 - 9, A - F) that will be transmitted in advance of the print file.
- trail**     A string of hexadecimal characters (0 - 9, A - F) that will be transmitted following the print file.
- page**     A string of hexadecimal characters (0 - 9, A - F) that will be transmitted at the start of each page, following the form feed character. The page data will be suppressed for any file that has an associated FCB.

We coded the following job stream to assemble and link the INSERTS phase HP4LAND. As coded, this phase will cause our HP printers to shift into landscape mode and print semibold typeface at 13 characters per inch and 8 lines per inch. After the last page has been printed, the printer is reset to default status.

```
* $$ JOB JNM=INSERTS,CLASS=0
* $$ LST CLASS=Q,DISP=D
// JOB INSERTS
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF PHASE,CATALOG=TCPIP.CONFIG
// OPTION CATAL
// EXEC ASMA90,PARM='EXIT(LIBEXIT(EDECKXIT)),SIZE(MAX-200K,ABOVE)'
* PAGE ORIENTATION (LANDSCAPE)
* 1B266C314F
* LINE SPACING (8 LINES/INCH)
* 1B266C3844
* CHARACTER PITCH (13), SEMI-BOLD TYPEFACE
* 1B28733133483142
HP4LAND INSERTS DEFINE,
              HEADER=1B451B266C314F1B266C38441B28733133483142,
              TRAILER=1B45
              END
/*
// EXEC LNKEDT,SIZE=256K
/*
/&
* $$ EOJ
```

Figure 170. Example of INSERTS phase HP4LAND

Figure 171 and Figure 172 on page 165 show two more examples of INSERTS phases we used in our LPR test environment. The INSERTS phase GPS1LEGL

in Figure 171 will cause printing on legal size paper pulled from the lower tray of the printer at 8 lines per inch.

```
* $$ JOB JNM=GPS1LEGL,CLASS=0
* $$ LST CLASS=Q,DISP=D
// JOB GPS1LEGL
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF PHASE,CATALOG=TCPIP.CONFIG
// OPTION CATAL
// EXEC ASMA90,PARM='EXIT(LIBEXIT(EDECKXIT))',SIZE(MAX-200K,ABOVE) '
* PAGE SIZE (LEGAL)
* 1B266C3341
* PAPER SOURCE (LOWER TRAY)
* 1B266C3448
* LINE SPACING (8 LINES/INCH)
* 1B266C3844
GPS1LEGL INSERTS DEFINE,
                                HEADER=1B451B266C33411B266C34481B266C3844,
                                TRAILER=1B45
                                END
/*
// EXEC LNKEDT,SIZE=256K
/*
/&
* $$ EOJ
```

Figure 171. Job stream to create INSERTS phase GPS1LEGL

The second INSERTS phase GPS1LETR in Figure 172 will print on letter size paper, pulled from the upper tray of the printer at 6 lines per inch.

```
* $$ JOB JNM=GPS1LETR,CLASS=0
* $$ LST CLASS=Q,DISP=D
// JOB GPS1LETR
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF PHASE,CATALOG=TCPIP.CONFIG
// OPTION CATAL
// EXEC ASMA90,PARM='EXIT(LIBEXIT(EDECKXIT))',SIZE(MAX-200K,ABOVE) '
* PAGE SIZE (LETTER)
* 1B266C3241
* PAPER SOURCE (UPPER TRAY)
* 1B266C3148
* LINE SPACING (6 LINES/INCH)
* 1B266C3644
GPS1LETR INSERTS DEFINE,
                                HEADER=1B451B266C32411B266C31481B266C3644,
                                TRAILER=1B45
                                END
/*
// EXEC LNKEDT,SIZE=256K
/*
/&
* $$ EOJ
```

Figure 172. Job stream to create INSERTS phase GPS1LETR

### 5.5.2 Invoking the INSERTS phase

You tell the LPR clients to use an INSERTS phase by coding the `SET INSERTS` command in a script file or by entering it as an input command for the batch LPR client or CICS LPR client. You can also use the UCS parameter on the POWER LST card to tell the automatic LPR client to use an INSERTS phase.

Figure 167 on page 163 shows our script file with the `SET INSERTS` command to invoke our HP4LAND INSERTS phase.

---

## 5.6 Using the batch LPR client

The batch mode of the TCP/IP for VSE/ESA LPR client allows you to start the LPR client and issue `LPR` commands from within a VSE batch job.

We used the following batch job to print the member TCPAPPL.B in sublibrary PRD2.CONFIG on our network printer HP4500 with IP address 192.168.155.105, using the printer queue TEXT. For HP printers you can choose between the two queues TEXT or RAW, where the TEXT queue can be used for unformatted text. For details of the differences, refer to the description of the `SET PRINTER` command in *TCP/IP for VSE User's Guide, Release 1.4*.

```
* $$ JOB JNM=LPRBATCH,CLASS=0,DISP=D
* $$ LST CLASS=Q
// JOB LPRBATCH TCPIP BATCH LPR EXAMPLE
// LIBDEF *,SEARCH=PRD1.BASE
// EXEC CLIENT,PARM='ID=00,APPL=LPR'
SET HOST=HP4500
SET PRINTER=TEXT
SET CC=NO
PRINT PRD2.CONFIG.TCPAPPL.B
QUIT
/*
/&
* $$ EOJ
```

Figure 173. Our VSE batch job to invoke the LPR

### Notes

// EXEC CLIENT,PARM='ID=00,APPL=LPR' starts the batch LPR. Thereafter, you can code any number of LPR commands. The ID=00 tells the batch LPR client to use TCP/IP partition with SYSID=00 to process the request.

You should provide at least the two commands `SET HOST=` and `SET PRINTER=` to fully qualify the target printer. You could also create a script file with these commands and code just the `EXEC` script command in the batch job.

Your last command should always be a `QUIT` to properly close the LPR batch session.



After the job execution, we looked at the POWER list queue entry to check that everything was correct.

```
// JOB LPRBATCH TCPIP BATCH LPR EXAMPLE
// LIBDEF *,SEARCH=PRD1.BASE
// EXEC CLIENT,PARM='ID=00,APPL=LPR'
TCP100I TCP/IP -- Misc Batch Client -- Version 01.04.00( ), 12/23/99
TCP101I Copyright 1995,1999 (C) Connectivity Systems Inc.
TCP105I Processing the execution parm override
TCP111I System ID is set to 00
TCP106I Processing of parm override complete
TCP123I Selected application: LPR
TCP115I Establishing connection
TCP117I Connection has been Established
Client manager connection Established.
LPR
LPR Ready:
SET HOST=HP4500
192.168.155.105
LPR Ready:
SET PRINTER=TEXT
LPR Ready:
SET CC=NO
LPR Ready:
PRINT PRD2.CONFIG.TCPAPPL.B
Opening the file
Establishing connection with LP Daemon
Remote port: 515 Local Port: 721
Request the queue for output
Checking LP daemon
Transferring the data file
Data file transferred.
Transferring the control file
Control file transferred.
Closing the connection with LP Daemon
Connection complete
LPR Ready:
QUIT
TCP124W Flushing the rest of system input
TCP102I TCP/IP -- Misc Batch Client -- Complete
TCP118I Terminating connection
TCP119I Connection is already closed
1S55I LAST RETURN CODE WAS 0000
EOJ LPRBATCH MAX.RETURN CODE=0000
```

Figure 174. Check the batch job entry in the POWER list queue

### 5.6.1 Return codes of batch LPR

For further analysis, the batch LPR provides return codes defined for the LPR batch client as follows:

- 0** Success. Your LPR commands completed normally.
- 4** A syntax error occurred in one of the LPR commands in the job stream. You will need to examine the job output to see what the problem was. Common problems include the inability to change to the specified directory, file not found conditions, printer not found conditions, and so forth.

- 8 Sometime after the session was established with the foreign host, either TCP/IP for VSE/ESA or the foreign host, terminated the connection. You will need to examine the availability of both TCP/IP for VSE/ESA and the host with which you were connected in order to resolve the problem. In addition, several LPR commands may have completed prior to the error, so you will need to examine the output of the job to see if a rerun is required.
- 12 One of the parameters was incorrectly specified.
- 16 TCP/IP for VSE/ESA is unable to establish a session with the foreign host.

---

## 5.7 Diagnosing LPR problems

An LPR request can fail for any number of reasons:

- Invalid LPD server IP address, host name, or script name
- Incorrect printer or queue name
- LPD print server not operational or accessible

Figure 175 shows the messages that are displayed on the VSE console if an automatic LPR request fails.

```
Z1 0085 0027: TCP911I Processing Event:LST_LIST Type:POWER Action:LPR
Z1 0085 0027: TCP907I Processing Output:AUTOLPR ,00760-000 for automated
Z1 0085 printing at:LPRSCR01          on:SYSB
Z1 0085 0027: TCP908I Processing Output:AUTOLPR ,00760-000 has failed
```

*Figure 175. VSE console message for unsuccessful LPR*

You can use the `DIAGNOSE LPR` command to obtain more detailed information that may help you determine why the LPR request failed. In Figure 176 on page 169 you can see the VSE console output during processing of an LPR client request when the LPR diagnostic function has been activated. You can enter this command in either the IPINITxx deck or from the VSE console after TCP/IP has finished initialization.

The `DIAGNOSE LPR` function only displays diagnostic output for LPR clients invoked by automatic LPR and GPS. It does not display LPR processing invoked by the batch job and CICS LPR client. The output includes parameter settings and interactions with the remote LPD during the LPR process. These are all useful in debugging LPR failures.

```

Z1 0087 IPN300I Enter TCP/IP Command
87 diagnose lpr
Z1 0085 IPN524I Diagnose status set to LPR
Z1 0085 0027: TCP911I Processing Event:LST_LIST Type:POWER Action:LPR
Z1 0085 0027: TCP907I Processing Output:AUTOLPR ,00760-000 for automated
Z1 0085 printing at:LPRSCR01          on:SYSB
Z1 0085 0027: TCP910I Client manager connection Established.
Z1 0085 0027: TCP910I LPR
Z1 0085 0027: TCP910I LPR Ready:
Z1 0085 0027: TCP910I SET USER=AUTOLPR
Z1 0085 0027: TCP910I LPR Ready:
Z1 0085 0027: TCP910I SET COPIES= 01
Z1 0085 0027: TCP910I LPR Ready:
Z1 0085 0027: TCP910I SET PRINTER= SYSB
Z1 0085 0027: TCP910I LPR Ready:
Z1 0085 0027: TCP910I SET JOBNAME= AUTOLPR
Z1 0085 0027: TCP910I LPR Ready:
Z1 0085 0027: TCP910I SET DISP=DELETE
Z1 0085 0027: TCP910I LPR Ready:
Z1 0085 0027: TCP910I SET CC=YES
Z1 0085 0027: TCP910I LPR Ready:
Z1 0085 0027: TCP910I SET HOST= LPRSCR01
Z1 0085 0027: TCP910I Error: Unable to locate the execute.
Z1 0085 0027: TCP908I Processing Output:AUTOLPR ,00760-000 has failed

```

*Figure 176. DIAGNOSE LPR command with console output*

In this example, TCP/IP could not locate the script file LPRSCR01 because it was either not defined in the IPINITxx deck or not cataloged in a VSE library accessible by the TCP/IP partition.



---

## Chapter 6. 3270 printing using GPS

The General Print Server (GPS) enables your existing VTAM-based 3270 printer applications to send printed output to TCP/IP-based printers without modifying the applications. The applications are typically CICS-based applications, however, they may also be any stand-alone VTAM application.

A GPS daemon is defined in the TCP/IP partition. It communicates to the application through VTAM as a logical unit with the characteristics of a locally attached 3287 printer. Either GPS or the application initiates a VTAM BIND to establish communication. Once connected, the GPS daemon receives data from the application through VTAM, just like a physical printer. The data is reformatted with ASA carriage control and is written to a staging member in a library or it may use 31-bit partition GETVIS for the staging area.

When the application terminates, the standard TCP/IP LPR client is invoked to transfer the accumulated data to a remote Line Printer Daemon (LPD). The LPR client can also be invoked to transfer the accumulated data in segments based on the amount of data received from the application or after a predefined idle time.

---

### 6.1 Installing GPS

GPS is a priced feature of the TCP/IP for VSE/ESA product and already part of TCP/IP for VSE/ESA 1.4.

Since GPS is a separately priced feature, you must obtain a separate product key for the GPS feature and add it to the TCP/IP product key file. The procedure described in 2.4, "Obtain TCP/IP product keys from IBM and define to TCP/IP for VSE/ESA" on page 42 shows how to accomplish this.

Detailed documentation for GPS can be found in *TCP/IP for VSE Optional Products, Release 1.4*.

---

### 6.2 GPS definitions

Before you can use GPS, you have to define the following:

1. VTAM application logical units for GPS 3270 printer sessions
2. CICS terminal entries for GPS printers
3. GPS daemons in your TCP/IP initialization

#### 6.2.1 VTAM definitions for GPS

Each GPS daemon requires a corresponding VTAM application (APPL) logical unit (LU) to be defined. The VTAM APPL LU provides the connection between the printing application, typically CICS, and the GPS daemon in your TCP/IP partition.

Our VTAM APPL LU definitions to support two GPS daemons are shown in Figure 177.

```

* $$ JOB JNM=GPSAPPL,DISP=D,CLASS=0
// JOB GPSAPPL      CATALOG TCP/IP VTAM GPS DEFINITIONS
// EXEC LIBR
A S=PRD2.CONFIG
CATALOG GPSAPPL.B                      REPLACE=YES
GPSAPPL  VBUILD TYPE=APPL
GPSPRT01 APPL  AUTH= (ACQ) ,DLOGMOD=DSC2K
GPSPRT02 APPL  AUTH= (ACQ) ,DLOGMOD=DSC2K
/+
/*
/&
* $$ EOJ

```

Figure 177. VTAM APPL definitions for GPS

We will use the VTAM APPL LU net names GPSPRT01 and GPSPRT02 in our CICS 3270 printer definitions and GPS daemon definitions. The LOGMODE DSC2K, referenced in Figure 177, is supplied by IBM in the default VTAM LOGMODE table ISTINCLM. It is a non-SNA 3270 printer log mode and is suitable for BIND negotiation.

### 6.2.2 Defining GPS 3270 printers to CICS

Although GPS works with any VTAM application, the majority of printing will probably come from CICS. To use a GPS printer under CICS, you must define it to CICS as a locally attached non-SNA 3270 printer device. We used the CICS resource definition online (RDO) transaction CEDA to do this.

There are several parameters that you can specify in the TERMINAL and TYPETERM definitions. We copied the TYPETERM VSEDSCP from the IBM supplied group VSETYPE and changed the PRINTERTYPE parameter from 3284 to 3287.

Figure 178 on page 173 and Figure 179 on page 174 show the values we used in the RDO TERMINAL and TYPETERM definitions for one of our GPS printers.

The GPS printer will be referenced by print applications in our CICS partition using CICS terminal ID GPS1. The CICS partition will communicate to the GPS daemon definition in the TCP/IP partition using the VTAM APPL LU net name GPSPRT01.

|                            |                        |   |
|----------------------------|------------------------|---|
| OBJECT CHARACTERISTICS     |                        | CICS RELEASE = 0410                                 |
| CEDA View Terminal( GPS1 ) |                        |   |
| Terminal                   | : GPS1                 |   |
| Group                      | : GPSPRT               |   |
| Description                | : GPS TERMINAL EXAMPLE |   |
| AUTINSTModel               | : No                   | No   Yes   Only                                     |
| AUTINSTName                | :                      |   |
| TERMINAL IDENTIFIERS       |                        |   |
| Typeterm                   | : GPSPRT               |   |
| Netname                    | : GPSPRT01             |   |
| Consname                   | :                      |   |
| REMOTESYSem                | :                      |   |
| REMOTENAME                 | :                      |   |
| REMOTESYSNet               | :                      |   |
| Modename                   | :                      |   |
| ASSOCIATED PRINTERS        |                        |   |
| PRINTER                    | :                      |   |
| PRINTERCopy                | : No                   | No   Yes  |
| ALTPRINTER                 | :                      |   |
| ALTPRINTCopy               | : No                   | No   Yes  |
| SPOOLTo                    | :                      |   |
| PIPELINE PROPERTIES        |                        |   |
| Pool                       | :                      |   |
| TASKlimit                  | : No                   | No   1-32767  |
| OPERATOR DEFAULTS          |                        |   |
| OPERId                     | :                      |   |
| OPERPriority               | : 000                  | 0-255   |
| OPERRsl                    | : 0                    | 0-24,...  |
| OPERSecurity               | : 1                    | 1-64,...  |
| PRESET SECURITY            |                        |   |
| USERId                     | :                      |   |
| NAtlang                    | :                      |   |
| TERMINAL USAGES            |                        |   |
| TRansaction                | :                      |   |
| TEmpriority                | : 000                  | 0-255   |
| Inservice                  | : Yes                  | Yes   No  |
| PRINTER DATA               |                        |   |
| SPOOLDest                  | :                      |   |
| SPOOLPRTRsl                | : 00                   | 0-24   Public                                       |
| SPOOLPRTTo                 | : 00                   | 0-59  |
| SPOOLFcb                   | :                      |   |
| PRINTEDmsg                 | : No                   | No   Yes  |
| PRINTImmed                 | : No                   | No   Yes  |
| SESSION SECURITY           |                        |   |
| SEcurityname               | :                      |   |
| ATTachsec                  | : Local                | Local   Identify   Verify   Persistent<br>  Mixidpe |
| BINDPassword               | :                      | PASSWORD NOT SPECIFIED                              |
| BINDSecurity               | : No                   | No   Yes  |
| USEDfltuser                | : No                   | No   Yes  |

Figure 178. CICS RDO TERMINAL definition for the GPS printer

|                              |                 |                     |
|------------------------------|-----------------|---------------------|
| OBJECT CHARACTERISTICS       |                 | CICS RELEASE = 0410 |
| CEDA View TYPeterm( GPSPRT ) |                 |                     |
| TYPeterm                     | : GPSPRT        |                     |
| Group                        | : GPSPRT        |                     |
| DEscription                  | :               |                     |
| RESOURCE TYPE                |                 |                     |
| DEVice                       | : 3270P         |                     |
| TERmodel                     | : 2             |                     |
| SESSiontype                  | :               |                     |
| PRINTERtype                  | : 3287          |                     |
| LDclist                      | :               |                     |
| SHippable                    | : No            | No   Yes            |
| MAPPING PROPERTIES           |                 |                     |
| PAGesize                     | : 024 , 080     | 0-999               |
| ALTPage                      | : 024 , 080     | 0-999               |
| ALTSuffix                    | :               |                     |
| FMhparm                      | : No            | No   Yes            |
| OBOperid                     | : No            | No   Yes            |
| PAGING PROPERTIES            |                 |                     |
| AUTOPage                     | : Yes           | No   Yes            |
| DEVICE PROPERTIES            |                 |                     |
| DEFscreen                    | : 024 , 080     | 0-999               |
| ALTScreen                    | : 024 , 080     | 0-999               |
| APLKybd                      | : No            | No   Yes            |
| APLText                      | : No            | No   Yes            |
| AUDiblealarm                 | : No            | No   Yes            |
| COLor                        | : No            | No   Yes            |
| COPy                         | : No            | No   Yes            |
| DUAlcasekybd                 | : No            | No   Yes            |
| EXtendedds                   | : No            | No   Yes            |
| HILight                      | : No            | No   Yes            |
| Katakana                     | : No            | No   Yes            |
| LIGHtpen                     | : No            | No   Yes            |
| MSrcontrol                   | : No            | No   Yes            |
| OBFormat                     | : No            | No   Yes            |
| PARTitions                   | : No            | No   Yes            |
| PRINTAdapter                 | : No            | No   Yes            |
| PROgsymbols                  | : No            | No   Yes            |
| VALidation                   | : No            | No   Yes            |
| FORMfeed                     | : Yes           | No   Yes            |
| HORizform                    | : No            | No   Yes            |
| VERTicalform                 | : No            | No   Yes            |
| TEXTKybd                     | : No            | No   Yes            |
| TEXTPrint                    | : No            | No   Yes            |
| Query                        | : No            | No   Cold   All     |
| OUTline                      | : No            | No   Yes            |
| SOSi                         | : No            | No   Yes            |
| BACKtrans                    | : No            | No   Yes            |
| CGcsgid                      | : 00000 , 00000 | 0-65535             |

Figure 179. CICS RDO TYPETERM definition for the GPS printer (part 1 of 2)



|                              |                 |  |
|------------------------------|-----------------|--|
| SESSION PROPERTIES           |                 |  |
| AScii                        | : No            | No   7   8   |
| SENDsize                     | : 00000         | 0-30720  |
| RECEivesize                  | : 00256         | 0-30720  |
| BRacket                      | : Yes           | Yes   No   |
| LOGMODE                      | :               |  |
| LOGMODECom                   | : No            | No   Yes   |
| DIAGNOSTIC DISPLAY           |                 |  |
| ERRLastline                  | : No            | No   Yes   |
| ERRIntensify                 | : No            | No   Yes   |
| ERRColor                     | : NO            | NO   Blue   Red   Pink   Green<br>  Turquoise   Yellow   NEutral |
| ERRHilight                   | : No            | No   Blink   Reverse   Underline                                 |
| OPERATIONAL PROPERTIES       |                 |  |
| AUTOConnect                  | : No            | No   Yes   All   |
| ATi                          | : Yes           | No   Yes   |
| TTi                          | : Yes           | Yes   No   |
| CRatesess                    | : Yes           | No   Yes   |
| RELreq                       | : Yes           | No   Yes   |
| DIScreq                      | : Yes           | Yes   No   |
| Nepclass                     | : 000           | 0-255  |
| SIGNoff                      | : Yes           | Yes   No   Logoff  |
| Xrfsignoff                   | : Noforce       | Noforce   Force  |
| MESSAGE RECEIVING PROPERTIES |                 |  |
| ROUTedmsgs                   | : All           | All   None   Specific  |
| LOGOnmsg                     | : No            | No   Yes   |
| APPLICATION FEATURES         |                 |  |
| BUILDchain                   | : No            | No   Yes   |
| USERarealen                  | : 255           | 0-255  |
| IOarealen                    | : 00256 , 00000 | 0-32767  |
| UCTran                       | : No            | No   Yes   Tranid  |
| RECOVERY                     |                 |  |
| RECOVOption                  | : Sysdefault    | Sysdefault   Clearconv   Releasesess<br>  Uncondrel   None       |
| RECOVNotify                  | : None          | None   Message   Transaction                                     |

Figure 180. CICS RDO TYPETERM definition for the GPS printer (part 2 of 2)

### 6.2.3 Defining the GPS daemon on VSE

The `DEFINE GPSD` command is used to define each GPS daemon (server) in the TCP/IP partition. You must define one daemon for each 3270 printer logical unit that you want to emulate.

Figure 181 on page 176 shows the part of our TCP/IP initialization file `IPINIT04.L` where we defined two of our GPS daemons. A complete description of the `DEFINE GPSD` command and its parameters can be found in *TCP/IP for VSE Optional Products, Release 1.4*.

```

*-----*
*
*      Define GPS Daemon
*
*-----*
DEFINE GPSD, ID=GPS1, PRINTER=TEXT, IPADDR=HP4000, INSERTS=HP4LAND, -
      TERMNAME=GPSVRT01, TARGET=DBCDCICS, INSESS=YES, LOGMODE=DSC2K, -
      QUEUING=MEMORY, STORAGE='TCPIP.TEMP', RETRY_COUNT=3
DEFINE GPSD, ID=GPS2, PRINTER=TEXT, IPADDR=HP4500, INSERTS=HP4LAND, -
      TERMNAME=GPSVRT02, TARGET=DBCDCICS, INSESS=YES, LOGMODE=DSC2K, -
      QUEUING=MEMORY, STORAGE='TCPIP.TEMP', RETRY_COUNT=3

```

Figure 181. TCP/IP GPS daemon definitions

### ***Explanation of our GPS daemon definitions***

In the above example we defined two GPS daemons with the IDs of GPS1 and GPS2, the names by which the GPS daemons are referenced in TCP/IP commands.

With the first set of parameters we provide the information TCP/IP needs to route the LPR request to the remote LPD:

|                        |  |
|------------------------|--|
| <b>PRINTER=TEXT</b>    | The name of our print queue defined on the remote LPD  |
| <b>IPADDR=HP4000</b>   | The name (or could also be IP address) of our remote LPD print server                              |
| <b>INSERTS=HP4LAND</b> | The name of our INSERTS phase that contains printer control data to be transmitted with the report |

Next we specify the parameters TCP/IP needs to connect to our 3270 print application, in our case, CICS:

|                          |  |
|--------------------------|--|
| <b>TERMNAME=GPSVRT01</b> | The VTAM LU name specified in our VTAM APPL definitions and our CICS 3270 printer definitions. You must have a unique VTAM APPL defined for each GPS daemon.   |
| <b>INSESS=YES</b>        | This parameter tells TCP/IP to bind the session to the application specified in the TARGET parameter when the GPS daemon is started. If you code NO in this parameter then TCP/IP waits for CICS to acquire the session. |
| <b>TARGET=DBCDCICS</b>   | The VTAM application (APPLID) to which this GPS daemon will connect. The APPLID of our CICS partition containing the 3270 GPS printer definitions is DBCDCICS.   |
| <b>LOGMODE=DSC2K</b>     | This is the VTAM log mode that will be used when the session is bound. It can be coded here or in the VTAM APPL definition.  |

Lastly, we provide information TCP/IP uses during the processing of the LPR request:

|                             |  |
|-----------------------------|--|
| <b>QUEUING=MEMORY</b>       | This parameter tells TCP/IP where to stage LPR requests for the GPS daemon. We specified all intermediate staging will be done in 31-bit partition GETVIS (MEMORY) instead of in a library member (DISK). Using memory-based queuing can reduce the 24-bit storage requirements for each daemon by approximately 100K.   |
| <b>STORAGE='TCPIP.TEMP'</b> | This parameter identifies the VSE library and sublibrary that TCP/IP uses to stage LPR print requests. This library would be used for staging if we specified DISK rather than MEMORY in the QUEUING parameter. The library is also used to log LPR activity if LOG=YES is specified in the DEFINE GPSD and to record debugging information if DEBUG=YES is specified. |
| <b>RETRY_COUNT=3</b>        | This is the number of times GPS will attempt to perform the LPR request before it shuts down and deletes the GPS daemon. You will have to redefine the GPS daemon with the <code>DEFINE GPSD</code> command if this occurs.  |

## 6.3 Operating GPS

The GPS daemon can be defined dynamically while TCP/IP is running. You can enter the `DEFINE GPSD` command from the VSE console, however, the command may require multiple lines of input. As long as you supply the hyphen (-) continuation character TCP/IP will respond with: *IPN300I Continue TCP/IP Command*, and you can enter the rest of the commands of a multiline define.

A better method is to use the `EXECUTE` command to execute a VSE library member containing the `DEFINE GPSD` commands. You can catalog a member in a library that is accessible to the TCP/IP partition and enter the command `EXECUTE memname` (where memname is the name of the member cataloged in a VSE library that contains the `DEFINE GPSD` statements).

Figure 182 shows the job stream we used to catalog a member containing one of our GPS daemon definitions.

```
* $$ JOB JNM=GPSDEF,DISP=D,CLASS=0
// JOB GPSDEF    CATALOG TCP/IP GPSDEF01.L
// EXEC LIBR
A S=TCPIP.CONFIG
CATALOG GPSDEF01.L                      REPLACE=YES
DEFINE GPSD, ID=GPS1, PRINTER=TEXT, IPADDR=HP4000, INSERTS=HP4LAND, -
        TERMNAME=GPSPT01, TARGET=DBDCCICS, INSESS=YES, LOGMODE=DSC2K, -
        QUEUING=MEMORY, STORAGE='TCPIP.TEMP', RETRY_COUNT=3

/+
/*
/&
* $$ EOJ
```

Figure 182. *DEFINE GPS command file for the EXECUTE command*

Figure 183 shows the `EXECUTE` command we entered to define our GPS daemon and the VSE console output from the command.

```
87 exec gpsdef01
Z1 0085 IPN397I Loading command deck GPSDEF01
Z1 0085 IPN398I Command deck GPSDEF01 has been completely loaded
Z1 0085 00FF: GPS900I GPS1 GPS Daemon starting
Z1 0085 00FF: GPS927I GPS1 GPS Version dated 12/23/99, 1.4 Prod
Z1 0085 00FF: GPS917I GPS1 Waiting for BIND
Z1 0085 00FF: GPS910I GPS1 GPS Bind to application complete
Z1-0087 IPN300I Enter TCP/IP Command
```

Figure 183. Use of the `EXECUTE` command to define the GPS daemon

### 6.3.1 The `DELETE GPSD` command

Because there is no `MODIFY GPSD`, if changes are required you must first delete the GPS daemon then redefine it. The following is an example of the command we entered to delete the GPS daemon with an ID of `GPS1` and the resultant VSE console messages:

```
Z1 0087 IPN300I Enter TCP/IP Command
Z1-0087
87 delete gpsd,id=gps1
Z1 0085 0029: GPS922I GPS1 GPS Shutting down
Z1 0085 0029: GPS924I GPS1 GPS Shutdown complete
```

Figure 184. `DELETE GPS daemon` command

Note that TCP/IP will automatically shut down and delete the GPS daemon if the LPR request fails and the number of retries have been attempted in the `RETRY_COUNT` parameter.

### 6.3.2 The `QUERY GPSD` command

You can `QUERY` the GPS daemon to determine the status of a specific GPS daemon and the parameters defined for it. Figure 185 on page 179 is an example of a `QUERY` command and the resulting output from TCP/IP.

```

87 query gpsd,id=gps1
Z1 0085 IPN253I (( TCP/IP GPS Daemons ))
Z1 0085 GPS930I GPS1 Host: HP4000          Printer: TEXT
Z1 0085 GPS931I GPS1 Termname: GPSPRT01 Logmode: DSC2K
Z1 0085 GPS932I GPS1 User:  Translate:  Script:
Z1 0085 GPS933I GPS1 Log=NO  Debug=NO
Z1 0085 GPS934I GPS1 Insess=DBDCCICS Target:
Z1 0085 GPS935I GPS1 Storage=TCPIP.TEMP
Z1 0085 GPS936I GPS1 Retries: 10  Delay: 18000
Z1 0085 GPS937I GPS1 Maxpages: 100 Now queued: 0
Z1 0085 GPS938I GPS1 Maxlines: 10000 Now queued: 0
Z1 0085 GPS939I GPS1 Maxchars: 1000000 Now queued: 0
Z1 0085 GPS940I GPS1 Maxidle: 3000 (10 seconds)
Z1 0087 IPN300I Enter TCP/IP Command
Z1-0087

```

Figure 185. QUERY GPSD command

### 6.3.3 Testing GPS setup

You can use the CICS-supplied CMSG transaction to test your CICS, VTAM, and GPS definitions to support printing to network printers. The syntax for the CMSG transaction is described for example in *CICS Transaction Server for VSE/ESA CICS-Supplied Transactions*, SC33-1655. We entered the following from a blank CICS screen to test our GPS setup:

```
cmsg msg='this is a test of our gps setup',route=gps1,send
```

MRS OK MESSAGE HAS BEEN ROUTED

In this example, the text in the MSG parameter is sent to our CICS GPS printer, GPS1, identified with the ROUTE parameter.

The GPS daemon uses the TCP/IP LPR client to send print files to the remote LPD. If the LPR request fails, GPS will display messages on the VSE console indicating the reason for the failure and retry the LPR operation after a time interval specified in the DEFINE GPSD. The GPS daemon is shut down if the LPR request is not successful after the specified number of retries.

Figure 186 on page 180 shows the GPS console messages from one of our tests where the printer queue name was not defined on the remote LPD.

```
Z1 0085 00B8: GPS900I GPS2 GPS Daemon starting
Z1 0085 00B8: GPS927I GPS2 GPS Version dated 12/23/99, 1.4 Prod
Z1 0085 00B8: IPN694I Unsent data: 50
Z1 0085 00B8: GPS905W GPS2 GPS processing previously queued data
Z1 0085 00B8: GPS916E GPS2 Error: Printer name has been rejected.
Z1 0085 00B8: GPS928I GPS2 LPR retry 1 of 3 in 60 seconds
Z1 0085 00B8: GPS916E GPS2 Error: Printer name has been rejected.
Z1 0085 00B8: GPS928I GPS2 LPR retry 2 of 3 in 60 seconds
Z1 0085 00B8: GPS916E GPS2 Error: Printer name has been rejected.
Z1 0085 00B8: GPS928I GPS2 LPR retry 3 of 3 in 60 seconds
Z1 0085 00B8: GPS916E GPS2 Error: Printer name has been rejected.
Z1 0085 00B8: GPS919E GPS2 LPR operation has failed
Z1 0085 00B8: GPS922I GPS2 GPS Shutting down
Z1 0085 00B8: GPS924I GPS2 GPS Shutdown complete
```

*Figure 186. VSE console messages for a GPS retry of an LPR request*

You have to restart the GPS daemon with the `DEFINE GPSD` command after correcting the GPSD daemon definition or the remote LPD setup. Print files are queued if the LPR request fails and are resent when the GPS daemon is restarted.

You can also use the `DIAGNOSE LPR` command described in 5.7, “Diagnosing LPR problems” on page 168 to obtain additional details on the settings GPS used for the LPR request.

You can find more information for troubleshooting common GPS problems in the manual *TCP/IP for VSE Optional Products, Release 1.4*.

---

## Chapter 7. Accessing VSE files using NFS

Network File System (NFS) is a feature of UNIX TCP/IP systems that was developed to allow files on remote systems to be transparently accessed as though they were on the local UNIX system. The TCP/IP for VSE/ESA NFS feature allows another system running an NFS client to access VSE files directly. This could be a UNIX system, or a PC with NFS client support.

You can access VSE files through NFS the same as you do for files that are local to your workstation. For example, on a Windows 98 system you could use Notepad to open and modify a book in a VSE source library. You could also do a COPYFILE directly from the network drive to the local system, which is quite a bit simpler than using FTP. Finally, you could write a workstation program that retrieves and updates a VSE file just as though it were a local file.

There are some limitations for VSE NFS servers. Because NFS was primarily designed to link UNIX systems, it is dependent on having a UNIX file system available on the server. Therefore, the NFS feature of TCP/IP for VSE/ESA cannot process files that do not have an emulated directory structure. These include ICCF, sequential (flat) files, and VSAM files defined individually in the file system.

VSAM ESDS and KSDS file types are supported for retrieval when accessed using a VSAM catalog entry in the file system. ESDS is the only VSAM file type that NFS supports for output on the VSE system. You can also access VSE libraries and the POWER RDR, PUN, and LST queues.

NFS V2 (which is supported by TCP/IP for VSE/ESA) uses the unreliable UDP protocol. Unless the network is quite robust and reliable, this can result in hangs or incomplete transmissions. The NFS feature of TCP/IP for VSE/ESA may encounter problems with dial-up connections or networks with transmission problems.

TCP/IP for VSE/ESA offers NFS server support only. It does not offer an NFS client on VSE. Therefore, you cannot directly access files on another system, even if the other system is running an NFS daemon.

NFS clients are widely available for a number of PC systems. Configuration information for clients that have been tested with the NFS feature of TCP/IP for VSE/ESA is provided in the *TCP/IP for VSE Optional Products, Release 1.4* documentation. The list does not include UNIX systems; for support you must consult with UNIX vendors or the UNIX community.

---

### 7.1 Installing NFS

NFS is a priced feature of the TCP/IP for VSE/ESA product and already part of TCP/IP for VSE/ESA 1.4.

Since NFS is a separately priced feature, you must obtain a separate product key for the NFS feature and add it to the TCP/IP for VSE/ESA product key file. The procedure described in 2.4, "Obtain TCP/IP product keys from IBM and define to TCP/IP for VSE/ESA" on page 42 shows how to accomplish this.

Detailed documentation for NFS can be found in *TCP/IP for VSE Optional Products, Release 1.4*.

---

## 7.2 Configuring NFS

Before you can use NFS, you must do the following:

- Define NFS mount points.
- Optionally, customize NFS configuration files and set your system clock.

### 7.2.1 Define NFS mount points

File structures accessed by NFS are called mount points. TCP/IP for VSE/ESA uses the same file system structures that you create with the `DEFINE FILE` statements and use with FTP. NFS clients can mount and access file system entries that have a valid directory structure. These file system entries include VSAM catalogs containing VSAM KSDS and ESDS files, VSE libraries, and the POWER RDR, PUN, and LST queues.

You should avoid using the `DEFINE FILESYS` entry as provided in the supplied sample initialization member `IPINIT00.L`. This causes TCP/IP for VSE/ESA to define every file in the system standard label area. While it is good for getting acquainted with TCP/IP for VSE/ESA, it represents a security exposure because it allows access to all files defined in the standard label area. Moreover, it can have an impact on NFS performance when doing directory displays. Using `DIR` through NFS can be rather slow, and the impact of doing a `DIR` on a large library is significant. You should only add `DEFINE FILE` statements for those resources that will be accessed by NFS or FTP.

Another way to accomplish the same reduction in files is to specify a mount point that is restricted in scope at open time to a specific directory, for example, `//VSE/PRD2.CONFIG`, rather than all entries in the file system. (Note the forward slashes. NFS clients conform to UNIX file syntax.)

Figure 187 on page 183 shows the part of our `IPINIT04.L` configuration file containing the `DEFINE FILE` entries and the `DEFINE NFSD` command used to start NFS.

The `DEFINE NFSD` command must follow the `DEFINE FILE` entries that you want to be available as NFS mount points. Any `DEFINE FILE` entries that follow the `DEFINE NFSD` command will not be accessible by NFS clients. NFS will ignore the `DEFINE FILE` entries that it does not support.



```

*-----*
*
*      Setup the File System
*
*-----*
DEFINE FILE, PUBLIC='TCPIP',          DLBL=TCPIP,   TYPE=LIBRARY
DEFINE FILE, PUBLIC='PRD2',          DLBL=PRD2,    TYPE=LIBRARY, -
      READONLY=YES
DEFINE FILE, PUBLIC='POWER',        DLBL=IJQFILE,  TYPE=POWER
DEFINE FILE, PUBLIC='ICCF',          DLBL=ICCF,    TYPE=ICCF
DEFINE FILE, PUBLIC='KSDSTCP.PRINT', DLBL=KSDSPRT, TYPE=KSDS
DEFINE FILE, PUBLIC='ESDSTCP.PRINT', DLBL=ESDSPRT, TYPE=ESDS
DEFINE FILE, PUBLIC='KSDSTCP.FILE',  DLBL=KSDSFIL, TYPE=ESDS
DEFINE FILE, PUBLIC='ESDSTCP.FILE',  DLBL=ESDSFIL, TYPE=ESDS
DEFINE FILE, PUBLIC='SAMTCP.FILE',    DLBL=SAMFILE, TYPE=SAM
DEFINE FILE, PUBLIC='IJSYSCT',       DLBL=IJSYSCT, TYPE=VSAMCAT, -
      READONLY=YES
DEFINE FILE, PUBLIC='VSESPUC',       DLBL=VSESPUC, TYPE=VSAMCAT, -
      READONLY=YES
DEFINE FILE, PUBLIC='TCPCAT',        DLBL=TCPCAT,  TYPE=VSAMCAT, -
      READONLY=YES
DEFINE FILE, PUBLIC='UCATF01',       DLBL=UCATF01, TYPE=VSAMCAT, -
      READONLY=YES
*-----*
*
*      Define NFS Daemon
*
*-----*
DEFINE NFSD, ID=NFSD, CONFIG=NFSCFG01

```

Figure 187. DEFINE FILE and DEFINE NFSD entries

## 7.2.2 NFS configuration files

Several NFS configuration files are supplied with the NFS feature in the PRD1.BASE library. The supplied defaults are adequate for the initial testing of NFS. You can copy these into your source editor library, modify them as appropriate and recatalog them into your configuration library.

### 7.2.2.1 NFSCFG.L configuration file

NFS configuration options are kept in a library member that is by default named NFSCFG.L. You can alter the configuration options to tune NFS performance, to control the presentation of information to the NFS client, and to collect debugging information.

The following figure shows the options that we used in our NFSCFG01.L configuration file for our NFS startup:

```

CATALOG NFSCFG01.L                                REPLACE=YES
*
* Sample NFS for VSE configuration file
*
*
* The following are options that the typical site may use for tuning
*
DirCacheSize=512K      /* Here goes 40 to 84 byte entries (RECFM=V) */
DirGroupSize=10        /* The number of entries to send during I/O */
DirlistExit=No         /* DIRLIST exit name, or YES for "NFSEX01" */
DirlistNullFiles=Yes   /* Yes=Include zero-length files in DirList */
LimitVSAMToESDS=Yes    /* Yes=Display only ESDS dataset via VSAM */
LoadPowerClasses=No    /* Yes=Define all classes at startup */
LoadPowerQueues=No     /* Yes=Define all queue names at startup */
MaxAllocErrs=3         /* Maximum consecutive ALLOC FRBLOK errors */
MaxRequests=1000       /* Maximum requests that can be serviced */
OS2Support=Yes         /* Perform additional routines for OS/2 */
PrintIDCAMS=Yes        /* Yes=Output IDCAMS results to SYSLST */
ReadCacheSize=128k     /* The total cache size for any READ */
ReadCacheTime=30       /* Max seconds a recently-read entry remains */
Security=No            /* Yes=Use the same user exit that FTP does */
ShowActive=No          /* Yes=Show Power queue entries with disp=* */
ShowNonExported=No     /* Yes=List truncated UDP EXPORT lists */
ShowPhases=No          /* No=Suppress .PHASE info in DIRLIST */
ShowRPCError=No        /* Yes=Show info when GETPORT fails via RPC */
ShowStaleInfo=No       /* Yes=Show info when request goes stale */
TimeOffset=+360        /* Number of minutes to use for accurate TOD */
Truncate=Yes           /* No=Don't truncate WRITE records to VSE */
UnixTextMode=Off       /* ON=Omit CR character from end-of-record */
VSAMTableSize=64k      /* Maximum size for the VSAM attribute table */
WakeUpTime=5           /* # of seconds to check cache usage */
WriteCacheTime=30      /* Max seconds a written entry remains */
XDRExit=No             /* Indicate XDR support and possible name */
*
* The following are options are for internal use and should not be
* used unless requested to do so by Technical Support.
*
DatagramTrace=No       /* Yes=produce SYSLST output of I/O traffic */
Debug=No               /* Yes=Output a lot of debugging info */
ListExports=No         /* Yes=List all mount points that we know of */
MaxExportSize=4K       /* Maximum outgoing EXPORT datagram (MOUNTD) */
Version3=No            /* To allow version 3 protocol */
WatchDirCache=No       /* Yes=Show DIR cache real time statistics */
WatchReadCache=No      /* Yes=Show READ cache real time statistics */
WatchReads=No          /* Yes=Show READ real time activity */
WatchWrites=No         /* Yes=Show WRITE real time activity */
WatchWriteCache=No     /* Yes=Show WRITE cache real time statistics */
/+

```

Figure 188. NFSCFG01.L configuration file

We increased the DIRCACHESIZE parameter from 80K to 512K to improve the performance of library directory displays by providing more caching for directory entries.

NFS is sensitive to correct time settings. Ideally your VSE machine should be set to GMT (Greenwich Mean Time) with an appropriate SET ZONE in the VSE IPL procedure to provide local time correction. If this is not feasible, the setting TimeOffset must be initialized in the configuration file NFSCFG.L. Note that this value is in minutes rather than hours. Our VSE system was IPLed using wall clock time and did not specify a relative offset from GMT so we set the TimeOffset parameter to +360 minutes in our configuration to reflect six hours from GMT.

The various tuning options may be experimented with as desired once the system is up and running. The configuration options are described in *TCP/IP for VSE Optional Products, Release 1.4*.

The user should be aware that by default SECURITY=NO provides no logon control whatsoever. If SECURITY=YES is specified, the same logon security as FTP is used, which may be more satisfactory.

### 7.2.2.2 NFSTYPES.L configuration file

NFS does not possess the override capabilities of FTP (for example, commands to specify BINARY or ASCII transfer), therefore, it may be necessary to provide defaults for translation when copying files to VSE. Translation is based on the file extension on the NFS client side. These defaults are defined in the NFSTYPES.L configuration file.

Figure 189 and Figure 190 on page 186 show the two sections of the NFSTYPES.L file supplied with TCP/IP.

```

CATALOG NFSTYPES.L                                REPLACE=YES
# -----#
#               N F S T Y P E S . L               #
#               #                                  #
# The following is a list of file types that will allow data being #
# written *TO* the VSE operating system to be accessible to the  #
# mainframe-based programs. Any file type not listed will still be #
# usable by the NFS client user, but VSE-based programs will not  #
# be able to use the data if the created file does not have an    #
# extension that is in this list.                                #
# -----#
#
# The BINARY entries will not be converted to EBCDIC, nor will
# we worry about eliminating the CR/LF. The difference between BINARY
# and BIN80 has to do with the method that is used to store the data
# in a VSE library. BINARY will use "STRING" mode, while BIN80 will
# write in 80-byte records instead of the continuous byte-string that
# is used by BINARY.
#
BIN      BINARY      <---- Store it in string mode
BJB      BIN80
DUMP     BINARY      <---- Stored in string mode
E        BIN80       <---- Edited macros (Non-HLASM) end in ".E"
JOB      BIN80
OBJ      BIN80
W        BIN80       <---- PRD2.PWS binary entries end in ".W"
#

```

Figure 189. NFSTYPES.L configuration file (part 1 of 2)

```

# The ASCII entries will be converted to EBCDIC and will have their
# CR/LF sequences removed.
#
&BLANK  ASCII
A        ASCII
ASM      ASCII
B        ASCII
C        ASCII
CATALOG  ASCII
D        ASCII
F        ASCII
FILE     ASCII
G        ASCII
H        ASCII
HTM      ASCII
HTML     ASCII
I        ASCII
J        ASCII
JCL      ASCII
K        ASCII
L        ASCII
LST      ASCII
M        ASCII
N        ASCII
O        ASCII
OUT      ASCII
OUTPUT   ASCII
P        ASCII
PROC     ASCII
PUN      ASCII
PRT      ASCII
Q        ASCII
R        ASCII
RDR      ASCII
S        ASCII
SLI      ASCII
T        ASCII
TEXT     ASCII
TXT      ASCII
U        ASCII
V        ASCII
VRML     ASCII
X        ASCII
Y        ASCII
/+

```

Figure 190. NFSTYPES.L configuration file (part 2 of 2)

The entries supplied are more or less self-explanatory, and any additional file types desired can simply be added to the NFSTYPES.L file. If you use a file type that is not in the NFSTYPES.L, NFS assumes the file is binary and no translation will be performed when the file is written to VSE. This may result in a file with data that is not usable on VSE. Once a file type is established, the option will remain in effect for the duration of the NFS session.

You should standardize your file type names used with NFS to match file types you have defined in NFSTYPES.L rather than continually changing the entries.

**Note**

This controls only the move of files from a workstation to VSE.

### 7.2.2.3 NFSMODEL.L configuration file

When NFS moves a new file from the workstation to VSAM, it must perform an explicit IDCAMS DEFINE to create an entry in the catalog. The parameters for the define are controlled by NFSMODEL.L. The sample provided in PRD1.BASE is as follows:

```
CATALOG NFSMODEL.L                                REPLACE=YES
*
* This is a sample VSAM model. The following variables can be used:
*
*   &NAME: File ID (1-44 bytes long)
*   &CATALOG: (1-44 bytes long) catalog file ID
*   &VOLID: VOLID (6 bytes) where the file will reside
*   &RECLEN: Logical record length
*
* The following two parameter will be used to establish primary and
* secondary allocation values. If the file already exists and is
* about to be overwritten, then the original values will be used. If
* the file name never existed in the catalog, then default values will
* be necessary. These are the default values.
*
&PRI=2000
&SEC=2000
*
* Here is the actual DEFINE CLUSTER that is processed to CREATE a new
* entry in the VSAM catalog.
*
  DEFINE CLUSTER (NAME -
    (&NAME) -
    NONINDEXED RECORDFORMAT(FIXUNB) -
    RECORDSIZE(&RECLEN &RECLEN) -
    RECORDS(&PRI &SEC) -
    VOLUMES(&VOLID)) -
  DATA (NAME -
    (&NAME.D)) -
  CATALOG -
    (&CATALOG)

/+
```

Figure 191. NFSMODEL.L configuration file

The only controllable variables you can change are &PRI and &SEC, the primary and secondary allocations in records. If these are insufficient for the size of the files you are moving to VSE, then this is where to increase them.

---

## 7.3 Operating NFS

There are several commands provided by TCP/IP for VSE/ESA to start and stop NFS and to manage the NFS server.

### 7.3.1 Starting NFS

NFS can be activated by including the `DEFINE NFSD` command in the `IPINITxx.L` configuration file or by entering the command from the VSE console to the TCP/IP for VSE/ESA partition. The format of the command is:

```
DEFINE NFSD, ID=identifier, TRANSLATE=name, CONFIG=name
```

where identifier is an arbitrary name such as `NFSD`.

The `TRANSLATE` name is the name of an ASCII/EBCDIC translate table (the default is `US_ENG_03`) and `CONFIG` is the name of the configuration file (the default is `NFSCFG`).

Figure 187 on page 183 shows the `DEFINE NFSD` command we used in our `IPINIT04.L` initialization file to cause NFS to start automatically when TCP/IP for VSE/ESA starts. This command must come after the `DEFINE FILE` commands; otherwise, NFS will start up with no available mount points.

In the following figure we show the startup of NFS from the console command and the VSE console messages that are issued during NFS initialization:

```
Z1-0087 IPN300I Enter TCP/IP Command
87 define nfsd, id=nfsd, config=nfscfg01
Z1 0085 01BB: IP0668I Prod Prog=01.04.00 Ver=12/23/99 14:32
Z1 0085 01BB: IP0705I NFS will use configuration file NFSCFG01
Z1 0085 01BC: IP0601I NFSCMD Initialization complete for NFSD
Z1 0085 01BD: IP0601I MOUNTD Initialization complete for NFSD
Z1 0085 01BD: IP0715I MOUNTD total mountpoints built: 00014
Z1 0085 01BE: IP0624I NFSUDPD Security is turned off
Z1 0085 01BE: IP0709I NFSUDPD XLATE table name is US_ENG_03
Z1 0085 01BE: IP0722I TimeOffset=X'00000000' GMT Offset=X'00000000'
Z1 0085 01BE: IP0601I NFSUDPD Initialization complete for NFSD
Z1 0085 01BF: IP0601I PCNSFD Initialization complete for NFSD
Z1 0085 01C0: IP0601I RPCUDPD Initialization complete for NFSD
Z1 0085 01C1: IP0601I RPCTCPD Initialization complete for NFSD
Z1 0085 01C3: IP0601I NFSXPCC Initialization complete for NFSD
Z1 0085 01BB: IP0601I NFSD Initialization complete for NFSD
```

Figure 192. NFS startup console messages

The “total mountpoints built” message indicates how many `DEFINE FILE` entries NFS detected as eligible mount points. After NFS has completed initialization, NFS clients can access the TCP/IP for VSE/ESA system and select any of these mount points for mounting.

### 7.3.2 Terminating NFS

The NFS feature of TCP/IP for VSE/ESA terminates automatically when TCP/IP for VSE/ESA is terminated. Should it be necessary to terminate it without

terminating the rest of TCP/IP for VSE/ESA, the following command can be entered from the console to the TCP/IP for VSE/ESA partition:

```
DELETE NFSD,ID=idname
```

where `idname` is the identifier used in the `DEFINE NFSD` command at startup (NFSD in our example).

In either event, the following messages are produced during NFS termination:

```
Z1-0087 IPN300I Enter TCP/IP Command
_87 delete nfsd,id=nfsd
Z1 0085 004F: IP0602I NFSD Shutdown in progress...
Z1 0085 0032: IP0620I RPCTCPD Shutdown complete for NFSD
Z1 0085 002D: IP0620I NFSCMD Shutdown complete for NFSD
Z1 0085 0053: IP0620I PCNFSH Shutdown complete for NFSD
Z1 0085 0052: IP0620I NFSUDD Shutdown complete for NFSD
Z1 0085 0054: IP0620I RPCUDD Shutdown complete for NFSD
Z1 0085 0050: IP0620I NFSCMD Shutdown complete for NFSD
Z1 0085 0051: IP0620I MOUNTD Shutdown complete for NFSD
Z1 0085 0055: IP0620I RPCTCPD Shutdown complete for NFSD
Z1 0085 0057: IP0620I NFSXPCC Shutdown complete for NFSD
Z1 0085 004F: IP0620I NFSD Shutdown complete for NFSD
```

Figure 193. NFS termination console messages

### 7.3.3 Query and control commands

NFS provides several commands that let you obtain current status information and to update NFS dynamically. These can be entered using the batch utility `IPNETCMD` or from the VSE console. The commands are prefaced by `NFS`.

#### 7.3.3.1 NFS QUERY command

The `NFS QUERY` command has several useful options. You can check your configuration settings by entering the command as shown in the following figure:

```
87 nfs query config
Z1 0085 01BC: IP0713I ConfigName=NFSCFG01
Z1 0085 01BC: IP0713I DatagramTrace=No
Z1 0085 01BC: IP0713I DBCS=No
Z1 0085 01BC: IP0713I Debug=No
Z1 0085 01BC: IP0713I DirCacheSize=524288
.
.
.
Z1 0085 01BC: IP0713I WatchReadCache=No
Z1 0085 01BC: IP0713I WatchReads=No
Z1 0085 01BC: IP0713I WatchWriteCache=No
Z1 0085 01BC: IP0713I WatchWrites=No
Z1 0085 01BC: IP0713I WriteCacheTime=30
Z1 0085 01BC: IP0713I XDRExit=<none>
```

Figure 194. NFS QUERY CONFIG command

By entering the command shown in Figure 195 on page 190 you can see all the mount paths known to NFS:

```
87 nfs query paths
Z1 0085 01BC: IP0713I Path=\
Z1 0085 01BC: IP0713I Path=ESDSTCP
Z1 0085 01BC: IP0713I Path=IJSYSCT
Z1 0085 01BC: IP0713I Path=KSDSTCP
Z1 0085 01BC: IP0713I Path=KSDSTCP\FILE
Z1 0085 01BC: IP0713I Path=KSDSTCP\PRINT
Z1 0085 01BC: IP0713I Path=POWER
.
.
.
Z1 0085 01BC: IP0713I Path=TCPIP\PRINT
Z1 0085 01BC: IP0713I Path=UCATF01
Z1 0085 01BC: IP0713I Path=VSESPUC
```

Figure 195. NFS QUERY PATHS command

The following NFS QUERY command shows you the user IDs of NFS clients who are accessing your system and their active mount points:

```
87 nfs query users
Z1 0085 01BC: IP0713I User=jlawson
Z1 0085 01BC: IP0713I Path=\
Z1 0085 01BC: IP0713I Path=VSESPUC
Z1 0085 01BC: IP0713I Path=UCATF01
Z1 0085 01BC: IP0713I Path=TCPIP
Z1 0085 01BC: IP0713I Path=TCPCAT
Z1 0085 01BC: IP0713I Path=SAMTCP
Z1 0085 01BC: IP0713I Path=PRD2
Z1 0085 01BC: IP0713I Path=POWER
Z1 0085 01BC: IP0713I Path=KSDSTCP
Z1 0085 01BC: IP0713I Path=IJSYSCT
Z1 0085 01BC: IP0713I Path=ESDSTCP
```

Figure 196. NFS QUERY USERS command

### 7.3.3.2 NFS FLUSH command

When NFS performs a directory list, the results are retained in cache, but updates other than through NFS are not reflected in the cached copy. Over time, this can cause the cached version to become inaccurate. The NFS FLUSH command causes the cache to be flushed so that fresh I/Os will be done for the next directory list command received.

For example, you would issue the command:

```
NFS FLUSH PRD2.CONFIG
```

to flush the cache for directory PRD2.CONFIG.



You can also add the parameter NFSTIMER to file system entries to cause their cache to be automatically flushed if the cache has not been accessed after a specified time interval.

---

## 7.4 Installing and using an NFS client

An NFS client is required on your workstation to access the TCP/IP for VSE/ESA NFS server. You can find information for configuring five different NFS clients in *TCP/IP for VSE Optional Products, Release 1.4*. We tested our NFS configuration using one of these clients, InterDrive from FTP Software, on a Windows 98 workstation.

Installation of this NFS client is typical of most PC product installs, therefore, we do not cover that in this book. However, we will show the windows we used to configure the InterDrive NFS client and examples of accessing our VSE system using this client.

### 7.4.1 Configuring the NFS client

The InterDrive NFS client is installed in a directory on you workstation and an icon is created in Entire Network under Network Neighborhood. After you install the InterDrive software, you begin by first double-clicking **Network Neighborhood** and then double-clicking **Entire Network**. The InterDrive icon will be displayed.

Next, double-click the **InterDrive** icon, right-click **NFS Servers I Have Configured** and then left-click **Properties** in the pull-down list. Figure 197 on page 192 shows the windows displayed for these steps.

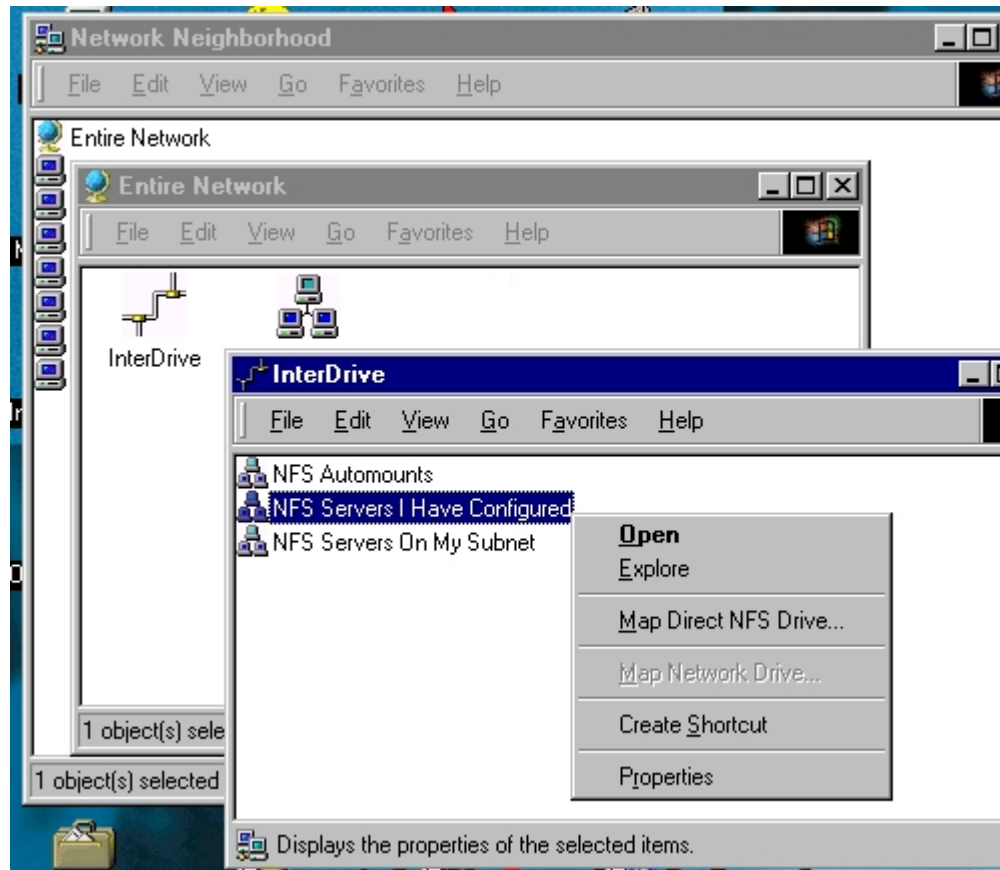


Figure 197. NFS client configuration window (1 of 4)

In the “New server to add” field of the Properties window that is displayed in Figure 198 on page 193, you enter the IP address or name of your TCP/IP for VSE/ESA host system running the NFS server. We entered the name of our VSE host, VSE240.

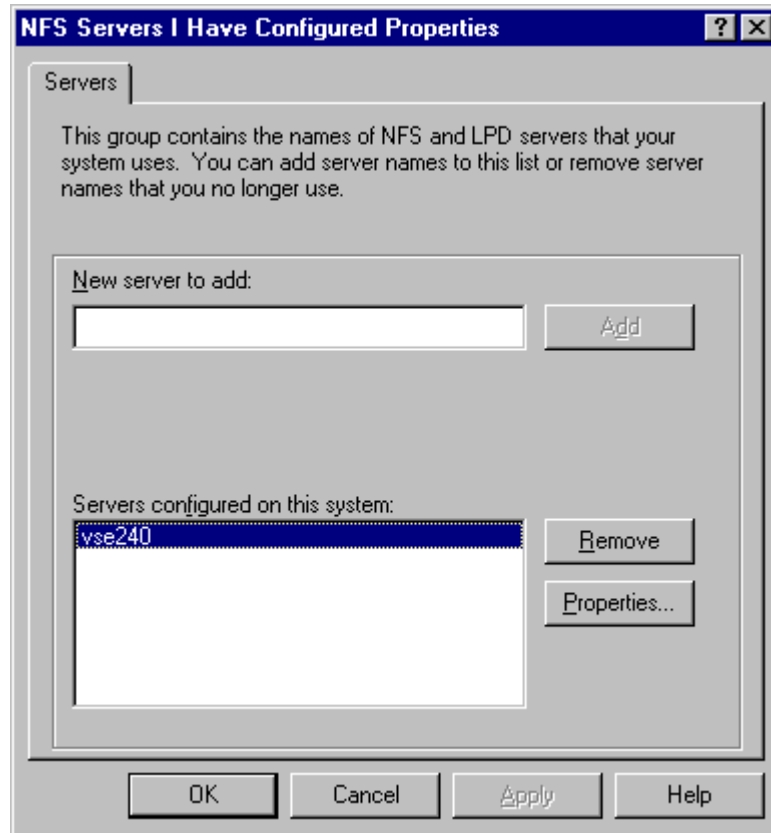


Figure 198. NFS client configuration window (2 of 4)

Next, you have to set the properties of the NFS server that are required by TCP/IP for VSE/ESA. From the list of Servers configured on this system, select your server name or IP address and click **Properties** to define the properties for your VSE NFS server.

Figure 199 on page 194 and Figure 200 on page 195 show two of the sections of properties that we needed to configure. The required settings are listed in the NFS client configuration information in *TCP/IP for VSE Optional Products, Release 1.4*.

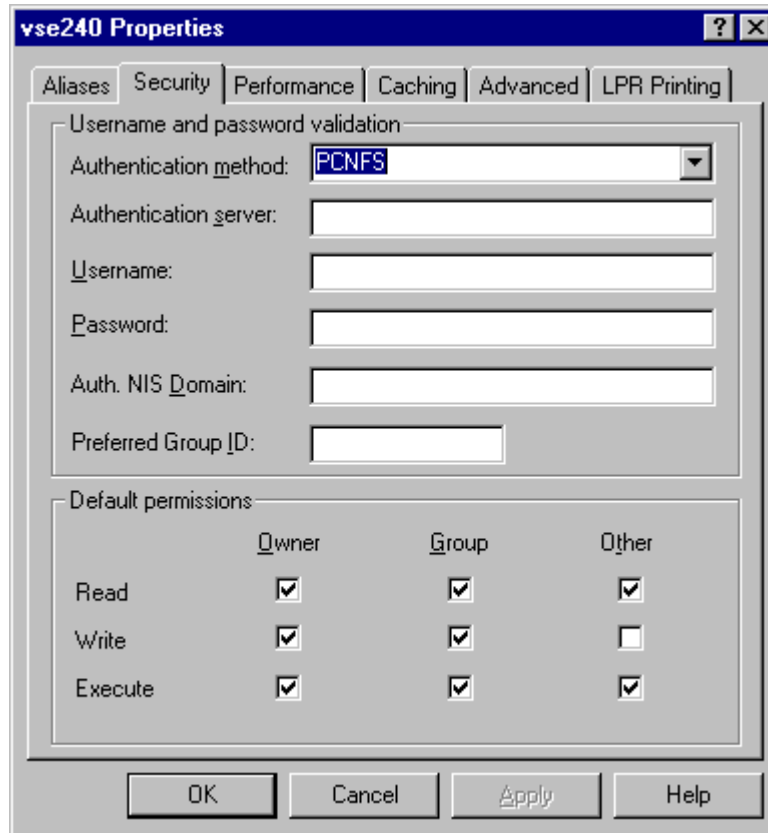


Figure 199. NFS client configuration window (3 of 4)

No tailoring was required in the Aliases section.

TCP/IP for VSE/ESA requires the authentication method to be changed to PCNFS in the Security section.

The changes that we had to make in the Performance section were selecting **2** for the Streaming (requests) and **Use version 2 only** for the NFS protocol. The transport protocol and other options already defaulted to the correct settings as shown in Figure 200 on page 195.

The defaults were taken in the Caching section.

All options were turned off or disabled in the Advanced and LPR Printing sections.

After we completed these configuration changes, we closed the Properties windows and were ready to use the NFS client to access our TCP/IP for VSE/ESA NFS server.

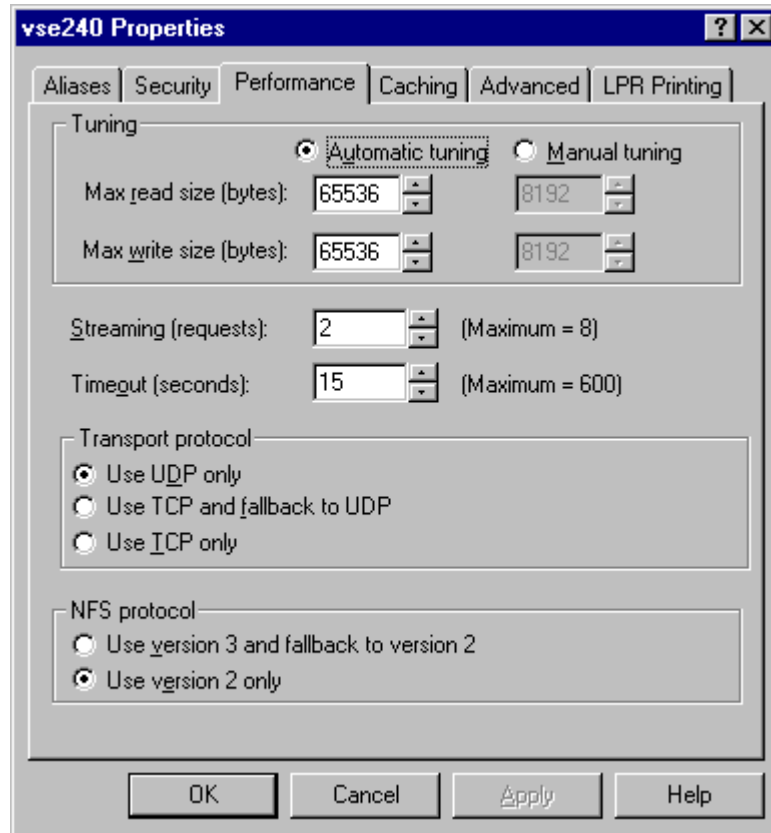


Figure 200. NFS client configuration window (4 of 4)

### 7.4.2 Using the NFS client

After we define our NFS server to the NFS client, we can now access files on VSE. If you double-click the **NFS Servers I Have Configured** selection in Figure 197 on page 192, you will see a window with an icon representing your NFS server. You can double-click this icon and you will see a display of directory folders representing the file system structures that NFS can access. Figure 201 on page 196 illustrates these steps.

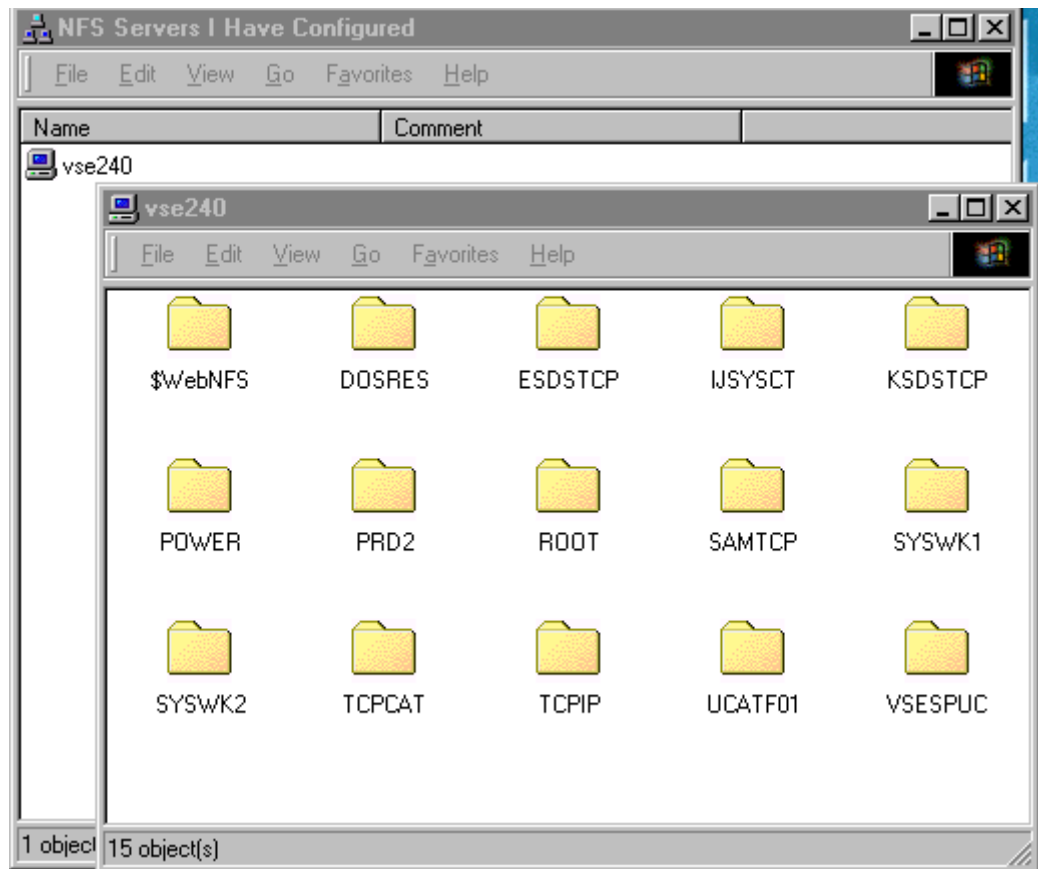


Figure 201. NFS server file system window

You can access these folders and the file entries in them as you do for any entries on your workstation.

You can also select any of these folders as a mount point and mount them as network drives. You can also mount the entire file system as an NFS drive. To do this you right-click the **vse240** icon mount point, select **Map Direct NFS Drive** from the pull-down list and select a drive letter to assign to the mount point. If you click a directory folder as a mount point, then select **Mount Network Drive** from the pull-down list and select a drive letter to assign to the mount point.

Figure 202 on page 197, Figure 203 on page 197, Figure 204 on page 198, and Figure 205 on page 198 illustrate these steps.

In Figure 202 we right-clicked the **NFS server** icon and selected **Map Direct NFS Drive**. The window in Figure 203 was displayed and we selected the letter **V** as our network drive letter used to access our NFS server on VSE.

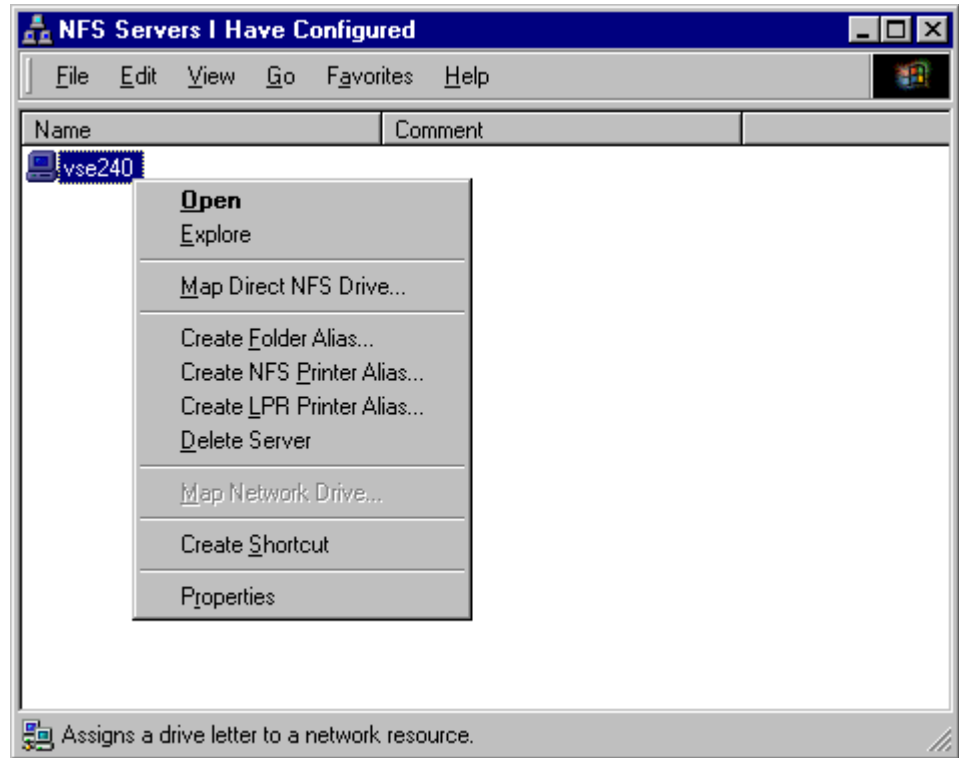


Figure 202. Mapping the NFS server to a network drive (1 of 2)

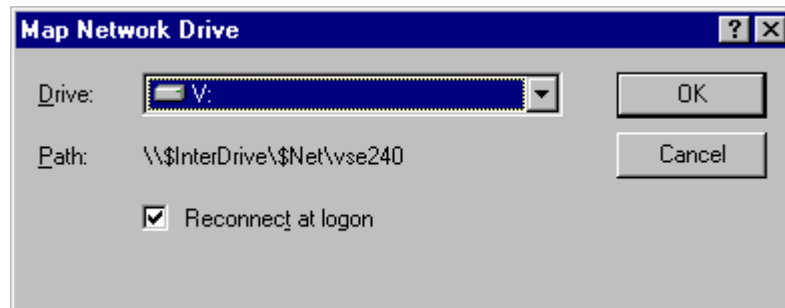


Figure 203. Mapping the NFS server to a network drive (2 of 2)

As an example of mounting an individual directory, we mounted the **POWER** directory as a network drive. To do this we right-clicked the **POWER** folder in Figure 204 on page 198 and clicked **Mount Network Drive** from the pull-down list. Then we selected the letter **P** as our network drive letter as shown in Figure 205 on page 198.

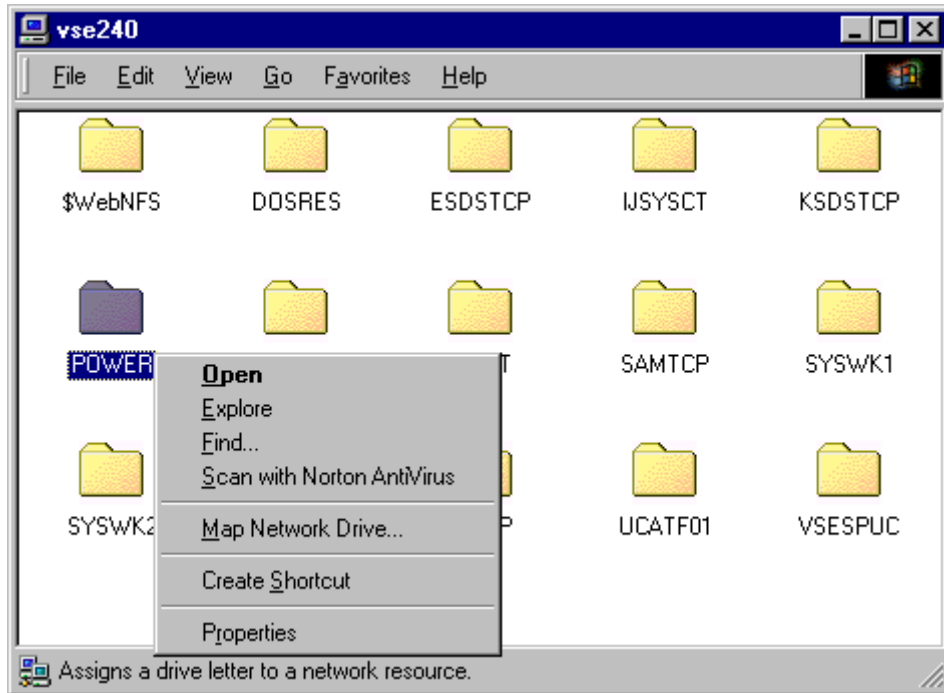


Figure 204. Mapping the POWER queue as a network drive (1 of 2)

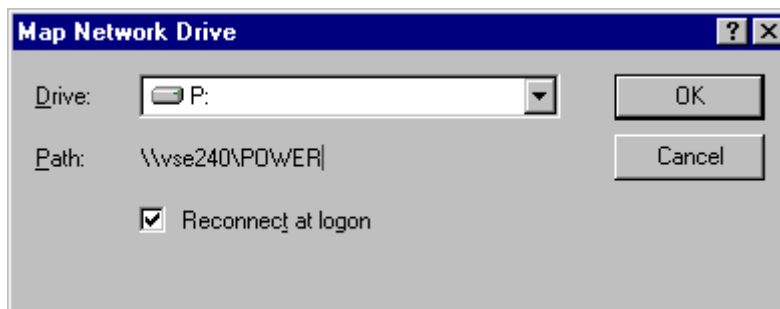


Figure 205. Mapping the POWER queue as a network drive (2 of 2)

We can now access our POWER queue as the P: drive or our VSE NFS server as the V: drive.

#### 7.4.2.1 Accessing your VSE NFS files

After you have mapped your VSE file system to network drives, you can access them through the same workstation facilities that you use to access files on your workstation or on other servers in your network.

For example, in Figure 206 on page 199 you can see the windows produced by double-clicking **My Computer** on our desktop, then double-clicking the **POWER P:** drive icon. Our mapped NFS mount points are displayed as network drives along with other folders and drives defined on the workstation.

By double-clicking the **VSE network drives** you can display the directory folders and file entries accessed by the NFS server on VSE. Figure 206 shows the directory folders representing the POWER RDR, PUN, and LST queues. If we



select one of these folders we would see folders representing the various POWER classes. We could then select a POWER class folder to see the individual entries in that class of the selected POWER queue.

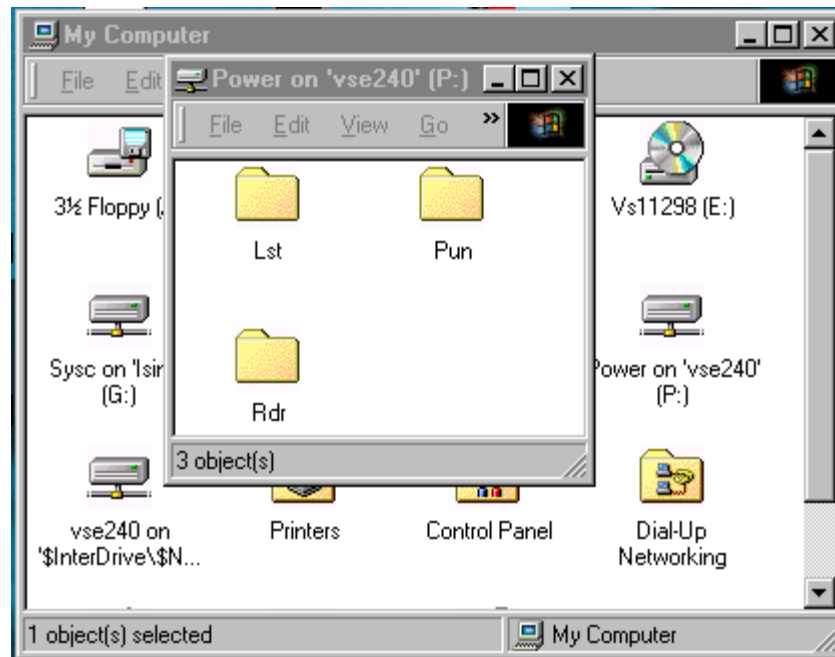


Figure 206. My Computer display of POWER queues

You can also access the VSE network drives from Windows Explorer as illustrated in Figure 207 on page 200. When we bring up Windows Explorer, we see our VSE network drives P: and V: listed with our other network drives and folders.

In this example, we selected the **V:** drive and displayed the members in library TCP/IP.CONFIG. We right-clicked member **Gpsdef** to show that you can perform the same PC operations on VSE NFS entries that you can on any other file on your workstation. You can open the member in a PC editor, copy and paste, drag and drop, or even delete the entry from the VSE system.

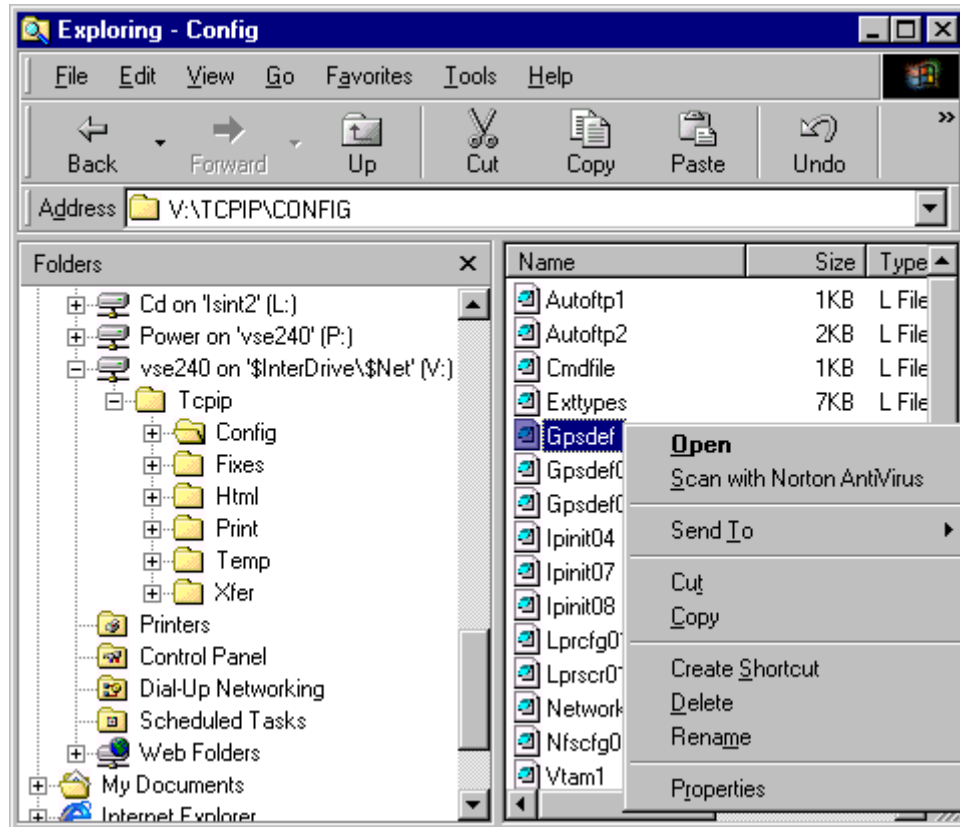
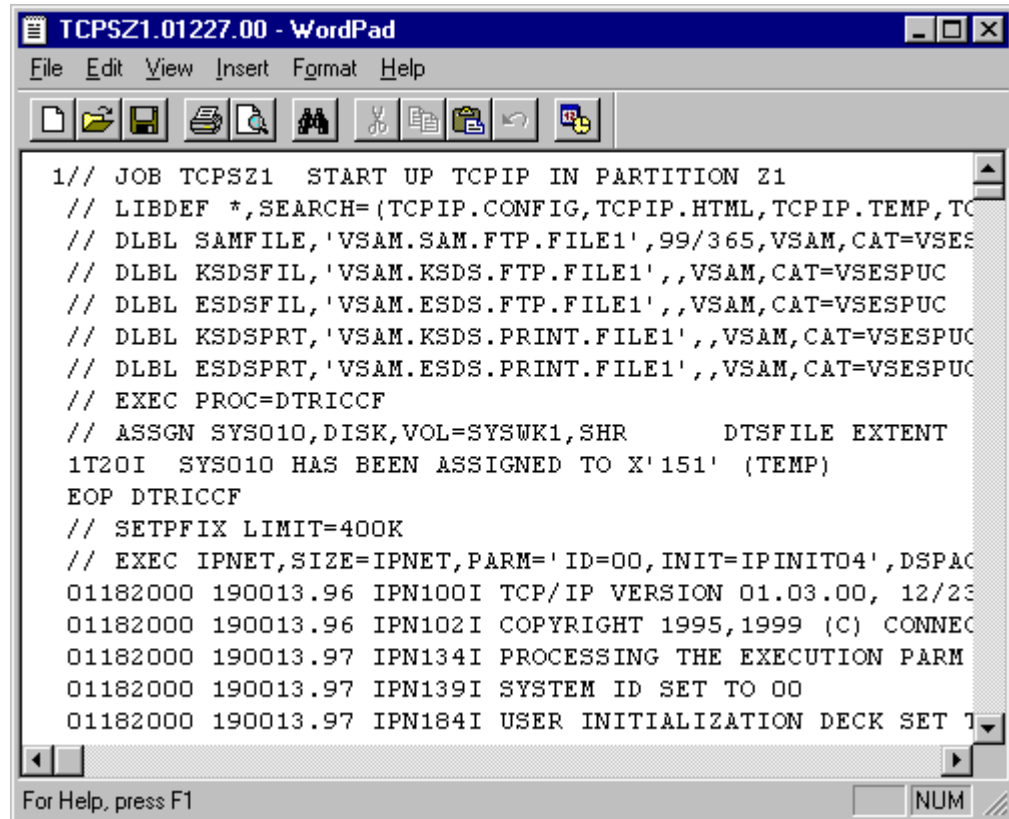


Figure 207. Windows Explorer access to VSE file system entries

In Figure 208 on page 201 you can see the results of selecting one of our TCP/IP partition output listings, TCPSZ1, from the POWER LST queue and opening it. Windows attempted to open it in Notepad, but because it was too large, Windows prompted us to open the file in WordPad.



```
1// JOB TCPSZ1 START UP TCPIP IN PARTITION Z1
// LIBDEF *,SEARCH=(TCPIP.CONFIG,TCPIP.HTML,TCPIP.TEMP,TC
// DLBL SAMFILE,'VSAM.SAM.FTP.FILE1',99/365,VSAM,CAT=VSES
// DLBL KSDSFIL,'VSAM.KSDS.FTP.FILE1',,,VSAM,CAT=VSESPUC
// DLBL ESDSFIL,'VSAM.ESDS.FTP.FILE1',,,VSAM,CAT=VSESPUC
// DLBL KSDSPRT,'VSAM.KSDS.PRINT.FILE1',,,VSAM,CAT=VSESPUC
// DLBL ESDSPRT,'VSAM.ESDS.PRINT.FILE1',,,VSAM,CAT=VSESPUC
// EXEC PROC=DTRICCF
// ASSGN SYS010,DISK,VOL=SYSWK1,SHR      DTSFILE EXTENT
1T20I  SYS010 HAS BEEN ASSIGNED TO X'151' (TEMP)
EOP DTRICCF
// SETPFIX LIMIT=400K
// EXEC IPNET,SIZE=IPNET,PARM='ID=00,INIT=IPINIT04',DSPAC
01182000 190013.96 IPN100I TCP/IP VERSION 01.03.00, 12/23
01182000 190013.96 IPN102I COPYRIGHT 1995,1999 (C) CONNEC
01182000 190013.97 IPN134I PROCESSING THE EXECUTION PARM
01182000 190013.97 IPN139I SYSTEM ID SET TO 00
01182000 190013.97 IPN184I USER INITIALIZATION DECK SET 7
```

Figure 208. Windows WordPad display of POWER LST queue entry

In Figure 209 on page 202 we selected the directory representing the VSAM catalog VSESPUC and displayed its contents. From this display we could select an ESDS or KSDS file, open it with one of the PC's facilities and process it on the workstation. These examples could be performed on any of the supported directories and file entries that NFS displays to our NFS client.

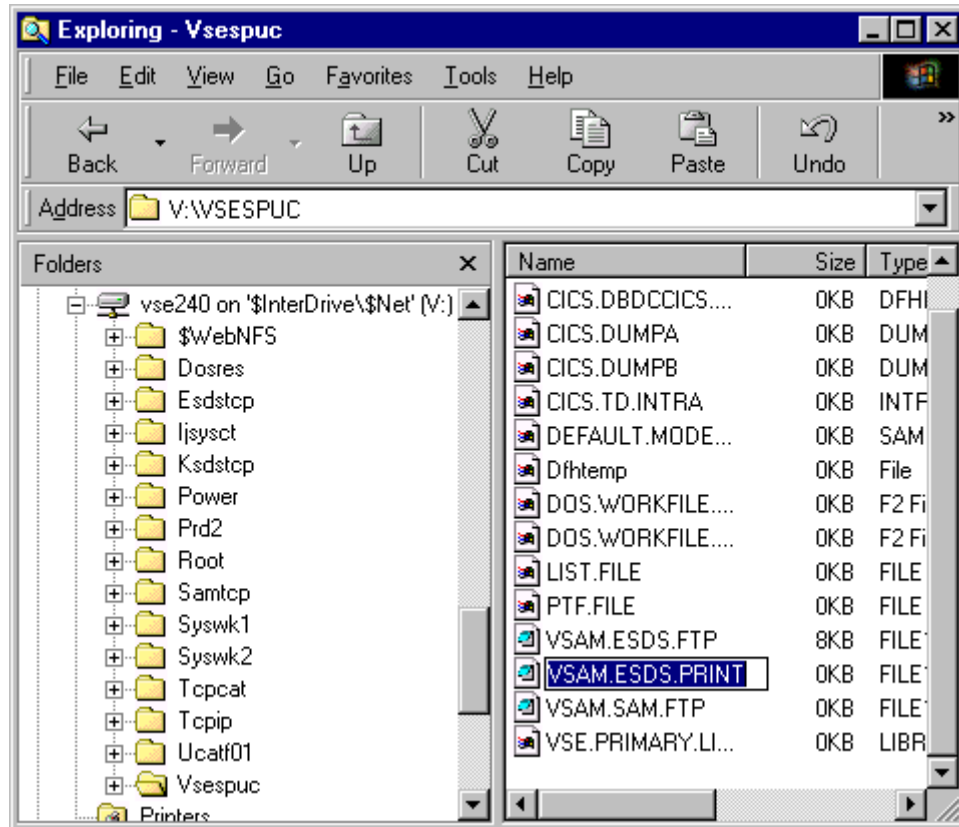


Figure 209. Windows Explorer display of VSAM catalog

### 7.4.3 Other NFS considerations

Although NFS operation is generally transparent to the client, here are some considerations you need to be aware of.

NFS is a file access method rather than a record or field access method. This means that even if you are only updating one record in a large file, the entire file must be read and rewritten to do this. Therefore, a few updates can take a long time.

When you put a file to VSE (explicitly or implicitly) the name must follow VSE rules for the type of file being written. For example, if you are writing to the POWER queue, the name of the entry can be no more than eight characters.

In NFS the file type on the client side controls whether ASCII to EBCDIC translation is done or not when a file is written to VSE. See NFSTYPES.L to pick an appropriate file type. If you guess wrong, the most likely result will be that the file looks like junk on the VSE side.

These and other considerations are documented in *TCP/IP for VSE Optional Products, Release 1.4*.

---

## Chapter 8. Using VSE as a Web server

TCP/IP for VSE/ESA includes a Hypertext Transfer Protocol (HTTP) daemon to support the use of Web browsers such as Netscape Communicator and Microsoft Internet Explorer to access a variety of data residing on the VSE system. The data can consist of:

- Static Web pages that are documents coded using the Hypertext Markup Language (HTML).
- Dynamic Web pages that are HTML documents that are dynamically created by programs that use the Common Gateway Interface (CGI) protocol. CGI programs execute on the VSE Web server system.
- Objects referenced by these Web pages such as graphics, video, and audio.
- Java applets that are executable programs that are sent to and executed on the client system running the Web browser.
- Application data retrieved from files and databases on the VSE system in response to input from the Web browser.

The role of the Web server is to access this data and send it to the Web browser. The Web browser receives the data, interprets it, and formats it for presentation to the Web user.

A Web site is accessed from a Web browser using a Universal Resource Locator (URL). The format of a URL to access a Web server is:

`http://location:port/path/filename`

|                 |   |
|-----------------|---|
| <b>location</b> | The IP address or domain name of the host Web server.   |
| <b>port</b>     | The TCP/IP port number on which the host Web server is listening. The default listen port for HTTP is 80. |
| <b>path</b>     | Path or directories on the host where the data resides.   |
| <b>filename</b> | File name and extension of the file containing the data you want. The default is INDEX.HTML.              |

A Web site is created and maintained in the same manner whether it is accessible via an intranet or the Internet. The steps you need to perform to create a Web site on VSE are:

1. Define the HTTP daemons in your TCP/IP for VSE/ESA configuration.
2. Create a set of HTML documents, possibly with embedded links to other objects (graphics, video, and so forth). These have to be placed into a VSE sublibrary or VSAM data sets that you specify when defining an HTTP daemon.
3. Create and define CGI programs that add intelligence to your Web site by processing input data from Web browsers.

We have provided some basic examples for each of these steps in this chapter.

For a more extensive description and examples of using VSE/ESA as a Web server see the redbook *VSE/ESA as a Web Server*, SG24-2040. Most of the examples in that redbook are still valid for the current release of TCP/IP for VSE/ESA even though they were developed using an older level of the product.

## 8.1 Defining the HTTP daemon

The HTTP daemon is defined and started with the `DEFINE HTTPD` command, entered from the VSE console or as part of the TCP/IP initialization file. A single HTTP daemon can serve many Web browser clients at the same time. However, you can also start several HTTP daemons in parallel with different definitions in order to operate several Web sites concurrently.

Multiple Web sites can be defined by the use of different sublibraries or by defining multiple HTTP daemons, each with its own port number. The HTTP daemon can be defined as secured, requiring a valid logon for access, or nonsecured.

We defined two HTTP daemons in our initialization file `IPINIT04.L` as shown in Figure 210. The figure also contains definitions for CGI programs, which we describe in 8.4.2.1, “Defining CGI programs” on page 214.

```
*-----*
*                                           *
*           Setup the File System          *
*                                           *
*-----*
DEFINE FILE, PUBLIC='TCPIP',           DLBL=TCPIP,   TYPE=LIBRARY
DEFINE FILE, PUBLIC='PRD2',           DLBL=PRD2,     TYPE=LIBRARY, -
READONLY=YES
.
.
.
*-----*
*                                           *
*           Define HTTP Daemons           *
*                                           *
*-----*
DEFINE HTTPD, ID=HTTP1, ROOT='TCPIP.HTML', SECURE=NO, CONFINES=YES
DEFINE HTTPD, ID=HTTP2, ROOT='TCPIP.HTML', PORT=5080, SECURE=YES, CONFINES=NO
*-----*
*                                           *
*           Define CGI Programs           *
*                                           *
*-----*
DEFINE CGI, PUBLIC='ASMC11', TYPE=CGI
DEFINE CGI, PUBLIC='ASMC12', TYPE=CGI-BAL
DEFINE CGI, PUBLIC='REXXCGI1', TYPE=CGI-REXX
```

Figure 210. Define HTTP and DEFINE CGI statements in the `IPINIT04.L` file

Each HTTP daemon definition requires a unique identifier for use internally and in console commands, therefore, we used `HTTP1` and `HTTP2` as the IDs for our two HTTP daemon definitions.

We defined the sublibrary `TCPIP.HTML` as the root for both of our HTTP daemons. This is the sublibrary where these HTTP daemons will look for HTML documents and other objects referenced by these documents. The HTTP daemon will look for the member `INDEX.HTML` as the default document unless the Web user enters a different file name in the URL.

The root library must be a public library name in the TCP/IP for VSE/ESA file system. This sublibrary must be in the LIBDEF search chain for the TCP/IP for VSE/ESA partition. Figure 210 on page 204 shows the required DEFINE FILE entry for library TCPIP in our configuration file.

We let the port number on our HTTP1 daemon definition default to 80 so that Web users would automatically connect to it as the default listen port. Our second daemon, HTTP2, was defined to use port number 5080. This is a way to distinguish HTTP daemons that run concurrently on the same VSE system. To access HTTP daemons with a port number other than 80, the Web browsers have to specify this port number explicitly in the URL.

For example, to access our second HTTP daemon on port 5080 on our VSE system named VSE240, we would enter the following URL:

```
http://vse240:5080
```

The CONFINE=YES parameter in our HTTP1 definition limits the HTTP daemon to searching only the root sublibrary for documents and objects. The CONFINE=NO option on our HTTP2 definition allows that HTTP daemon to search using any fully qualified public name defined in the file system. The name can come from HTML documents or can be entered by the Web user in the URL.

We specified SECURE=YES on our HTTP2 daemon definition to create an example of a secured daemon. We will describe the setup of a secured Web server in 8.3, “Creating a secured Web site” on page 209.

See *TCP/IP for VSE Installation Guide, Release 1.4* for additional details on setting up the HTTP daemon. The complete syntax for the `DEFINE HTTPD` command is described in *TCP/IP for VSE Commands, Release 1.4*.

### 8.1.1 Managing HTTPD definitions

After the HTTP daemons have been defined and started, we use the `QUERY HTTPDS` command from the VSE console to examine the characteristics of the active HTTP daemons. The following figure shows our VSE console log from this command and other HTTPD commands.

```
Z1-0087 IPN300I Enter TCP/IP Command
87 query httpds
Z1 0085 IPN508I (( TCP/IP HTTP Daemons ))
Z1 0085 IPN509I ID: HTTP2 Port: 5080 Driver: HTTPD
Z1 0085 IPN510I Confine: OFF Root: TCPIP.HTML.
Z1 0085 IPN509I ID: HTTP1 Port: 80 Driver: HTTPD
Z1 0085 IPN510I Confine: ON Root: TCPIP.HTML.
Z1 0087 IPN300I Enter TCP/IP Command
Z1-0087
87 delete httpd,id=http1
Z1 0085 002B: HTT903I Daemon Shutdown HTTP
Z1 0087 IPN300I Enter TCP/IP Command
87 define httpd,id=http1,root='tcpip.html',confine=yes
Z1 0085 006C: HTT900I Daemon Startup Hyper Text Transfer Protocol
87 query httpd,id=http1
Z1 0085 IPN508I (( TCP/IP HTTP Daemons ))
Z1 0085 IPN509I ID: HTTP1 Port: 80 Driver: HTTPD
Z1 0085 IPN510I Confine: ON Root: TCPIP.HTML.
```

Figure 211. Check the status of the HTTP daemons

You can use the `DELETE HTTPD` command to stop an HTTP daemon. If the daemon is currently in use, it is flagged for deletion as soon as the current operation finishes. If you need to terminate an HTTP daemon immediately, use the `FORCE` option.

**Note**

There is no `MODIFY HTTPD` command. To change the specifications of a defined HTTP daemon, you have to `DELETE` and re-`DEFINE` it.

A detailed description of the syntax for the `QUERY HTTPDS` and `DELETE HTTPD` commands can be found in *TCP/IP for VSE Commands, Release 1.4*.

---

## 8.2 Creating documents for your Web site

To set up a Web site, you have to have at least one HTML document as a starting point. This is generally referred to as the home page and is the first page a Web user sees when he or she connects to your Web site. If the Web user does not request a specific document in his or her URL, then the `INDEX.HTML` will be the default document retrieved and presented to the Web browser.

We created the HTML document `INDEX.HTML` and cataloged it in the sublibrary `TCPIP.HTML`, which is defined as the root for our daemons. The contents of this member will be presented as the default document to all Web browsers accessing our VSE system.

HTML documents are created as standard text files with embedded HTML tags. HTML tags are recognizable by the less than and greater than symbols, for example, `<tag>`. HTML documents have to be stored in a VSE sublibrary in EBCDIC because they will be translated to ASCII during transmission to a client Web browser. You can create these on VSE with any VSE editor, however, there are no editing tools on VSE to validate the syntax of the HTML document. There are several PC tools that will create and validate HTML documents.

Graphic images (audio, video, and virtual reality files) are not translated during transmission. This is why they have to be created in ASCII format (for example, on a workstation) and then stored as binary files in a VSE sublibrary. The easiest way to upload graphics and other binary files into a VSE library member is to use FTP. From our FTP client on the workstation, we had to set the `TYPE` to `BINARY` and the record format to `S`.

### 8.2.1 HTTP file types

Once all the required files are stored in the VSE library members, the HTTP daemon determines the different types of files by looking at the member type. The daemon then generates a file header record that notifies the Web browsers of the file type.

Table 9 on page 207 lists the various types of files and the acceptable extension names/member types recognized by the HTTP daemons. These are defined in the `EXTTYPES.L` member described in 3.4, "Data translation during file transfer" on page 95.



Both FTP and HTTP use this table for the transmission of these file types.

Table 9. Member types recognized by the HTTP daemon

| Type of file            | Member type               |
|-------------------------|---------------------------|
| HTML                    | HTML HTM                  |
| Plain text              | TEXT TXT JAVA             |
| Graphics                | GIF JPEG JPG JPE TIFF TIF |
| Audio                   | AU SND AIF AIFF AIFC WAV  |
| Video                   | MPEG MPG MPE QT MOV AVI   |
| Virtual reality         | WRL WRZ FLR               |
| Java                    | CLASS CLA                 |
| Other application types | CAB ICA JAR PDF PY        |

## 8.2.2 Documents on our Web site

Figure 212 shows the source code for our INDEX.HTML member and the job stream to catalog it into our root sublibrary TCPIP.HTML:

```
// JOB INDEXPAG    CATALOG TCP/IP WEB INDEX PAGE
// EXEC LIBR
A S=TCPIP.HTML
CATALOG INDEX.HTML                REPLACE=YES
<html>
<!-- @(#)TCP/IP for VSE/ESA Redbook Index Page 12/17/99
* /-->
<head><title>Using TCP/IP Redbook home page</title></head>
<body BGCOLOR="#C0C0C0"><font color="#0000FF">
<center><h1><i>Welcome</i></h1>
<font color="black">
<H2>to the <I>TCP/IP for VSE/ESA Redbook</I> Home Page</H2><BR><HR>
<br>
</img>
<br>
<p><font size=+3>This is our Web Server running on VSE/ESA!!!
<br><br>
<font size=+1>
<a href="http://www.redbooks.ibm.com">
Click here for the IBM Redbooks Home Page </a>
<br>
<a href="http://www.s390.ibm.com/vse">
Click here for the VSE Home Page </a>
<br>
<a href="itsoform.html">Click here to tell us about yourself </a>
</body>
</html>
/+
/*
/&
```

Figure 212. Default page INDEX.HTML in sublibrary TCPIP.HTML

Because our Web page includes an image file referenced by the <img> HTML tag, we had to create the file ITSOLOGO.GIF on our workstation and transfer it with FTP using type BINARY and record format S into the sublibrary TCPIP.HTML.

Our Web page also has links to other Web pages. The HTML <a> tag with the HREF parameter defines a link to another document from this Web page. These links are called hyperlinks. When the Web user clicks the text displayed by the hyperlink definition, the Web browser will send a request for the document defined in the HREF parameter to the Web server. The hyperlink could be a reference to a document on the same Web site or it could be a URL linking to another Web site.

In our Web page we have a hyperlink to retrieve a form document called ITSOFORM.HTML from our Web site. We will describe the use of this form in 8.4.1, "Creating forms" on page 212.

We have also defined two hyperlinks, one to link to the IBM redbook home page and one to link to the IBM VSE home page. Both of these hyperlinks use a fully qualified URL in our Web page.

Finally, to test accessibility to our Web site on the VSE system from a Web browser client, we did the following on our Windows 98 workstation:

1. Start **Microsoft Internet Explorer**.
2. Specify **http://vse240** in the Location field.

Because no port number is specified, we connect to our default HTTP daemon, HTTP1, at the default port 80. The URL does not contain library/sublibrary information nor the member name, therefore, the Web browser on our workstation received the contents of the default member, which is INDEX.HTML in the sublibrary TCPIP.HTML. Figure 213 on page 209 illustrates how Internet Explorer displayed the home page from our VSE Web site.



Figure 213. Web browser display of VSE INDEX.HTML Web page

We can click one of the hyperlinks at the bottom of the Web page and the Web browser will request the document defined in our Web page.

## 8.3 Creating a secured Web site

The HTTP daemon operates primarily in read-only mode when accessing VSE data. The only exception is when CGI programs are invoked and write data to the VSE system, thus security may not be an issue. However, if you are concerned about limiting access to your VSE system, there are several ways to secure your VSE Web server.

- You can rely on security systems outside of your VSE system such as firewalls in your network to limit access from unauthorized users.
- You can implement your own security exit in TCP/IP for VSE/ESA to control who can access the HTTP daemon and its documents.
- You can define the HTTP daemon with the CONFINE=YES parameter to limit access to the root sublibrary only. Our HTTP1 daemon definition:

```
DEFINE HTTPD, ID=HTTP1, ROOT='TCP/IP.HTML', SECURE=NO, CONFINE=YES
```

is an example of an HTTP definition that provides this level of security.

- You can define a secured HTTP daemon. The following section describes how to implement this type of HTTP security.

### 8.3.1 Implementing a secured HTTP daemon

You can force Web users to log on to your Web site with valid user IDs and passwords. To implement this level of security we had to do the following:

- Activate security in our configuration file with the `SET SECURITY=ON` command.
- Define authorized user IDs and their passwords with `DEFINE USER` commands.
- Define an HTTP daemon with `SECURE=YES`.

We defined our second HTTP daemon, HTTP2, with security using the following command:

```
DEFINE HTTPD, ID=HTTP2, ROOT='TCPIP.HTML', PORT=5080, SECURE=YES, CONFINES=NO
```

Security must be activated with the `SET SECURITY=ON` command before this HTTP daemon is defined, otherwise, the `SECURE=YES` option will be ignored.

We also had to catalog three HTML security documents, `PASSWORD.HTML`, `VIOLATED.HTML`, and `BLANKING.HTML`, into the root sublibrary `TCPIP.HTML`, which was defined for this HTTP daemon. An installation job stream containing these documents is supplied in member `HTMLINST.Z` in the `PRD1.BASE` sublibrary.

### 8.3.2 Logging on to the secured Web server

To access our secured Web server HTTP2 we did the following:

1. Start Microsoft Internet Explorer.
2. Specify `http://vse240:5080` in the Location field.

Because our secured Web server HTTP2 is defined with a port number other than the default 80, we had to include the port number 5080 in the URL. Our secured HTTP daemon first retrieves the `PASSWORD.HTML` document for the user to enter a name and password as shown in Figure 214 on page 211. If a valid name and password are entered, our secured HTTP daemon will then send the `INDEX.HTML` document.



Figure 214. *PASSWORD.HTML* document from our secured HTTP daemon

If an invalid user ID or password are entered, the HTTP daemon sends the *VIOLATED.HTML* Web page shown in Figure 215. The user will have to reaccess the Web site and enter a valid logon ID.

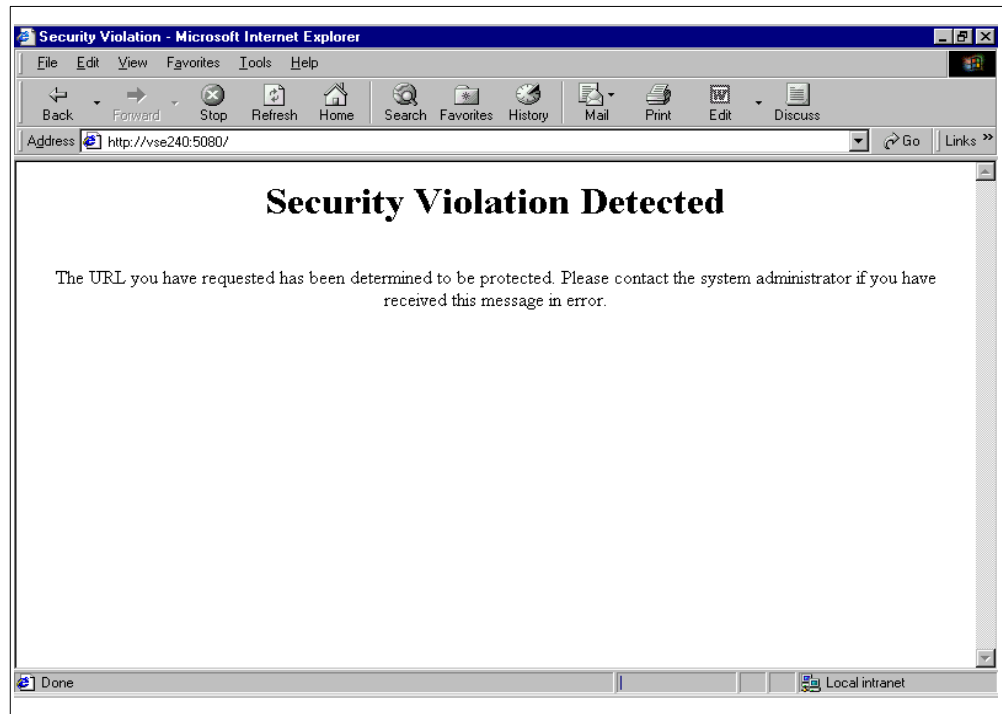


Figure 215. *VIOLATED.HTML* Web page from our secured HTTP daemon

---

## 8.4 Processing data on a VSE Web site

Most Web sites do not only display documents but allow the Web user to enter data that needs to be processed by the host running the Web server. You can create Web pages with forms containing fields, push buttons, selection list, and other variable input that the Web user can enter or select.

The programs that process this data are called CGI programs and execute on the host where the Web server is running.

### 8.4.1 Creating forms

Forms are created using the same HTML structure we have shown in previous examples. Special tags are used to define a form and the various input selections the user can choose on the form.

Figure 212 on page 207 shows the source code for our form page ITSOFORM.HTML and the job stream to catalog it into our root sublibrary TCPIP.HTML.

```

// JOB ITSOFORM    CATALOG TCP/IP WEB FORM PAGE
// EXEC LIBR
A S=TCPIP.HTML
CATALOG ITSOFORM.HTML                      REPLACE=YES
<html>
<!-- This web page contains:
      * a form using METHOD=GET
      * two input text boxes which hold 30 characters
      * pull down list containing 3 items, default first item
      * a submit push button labeled Click here
      * hyperlink to index.html
      * /-->
<HEAD><TITLE>Using TCP/IP Redbook Information Form</TITLE></HEAD>
<body>
<H2><I>TCP/IP for VSE/ESA Redbook</I> Information Form</H2><BR><HR>
<form name=REXXCGI1 action="REXXCGI1",method=get>
<p>Enter your name: <input type="text" size=30
                        name="yourname" value="">
<p>Enter your company: <input type="text" size=30
                        name="company" value="">
<P>Do you have TCP/IP for VSE/ESA installed?
<select name="tcpinst" SIZE=1>
<option selected>yes
<option>no
<option>I don't know
</select>
</p>
<p><input type="submit" value="Click here">
      to send the form.</P>
</form>
<a href="index.html">Go to the home page</a>
<br>
</body>
</html>
/+
/*
/&

```

Figure 216. Form page ITSOFORM.HTML

The HTML `<form>` tag defines the form and the type of fields in it. Our form has two input fields called `yourname` and `company`, and a selection pull-down list with a default value of `yes`. It also includes a hyperlink back to the `INDEX.HTML` page.

In the form we tell the HTTP daemon the action to take when the data is sent by the Web browser. In our example, we are invoking a CGI program written in REXX to process the data sent by the Web browser.

Figure 213 on page 209 illustrates how Internet Explorer displays our form page sent from our VSE Web site.

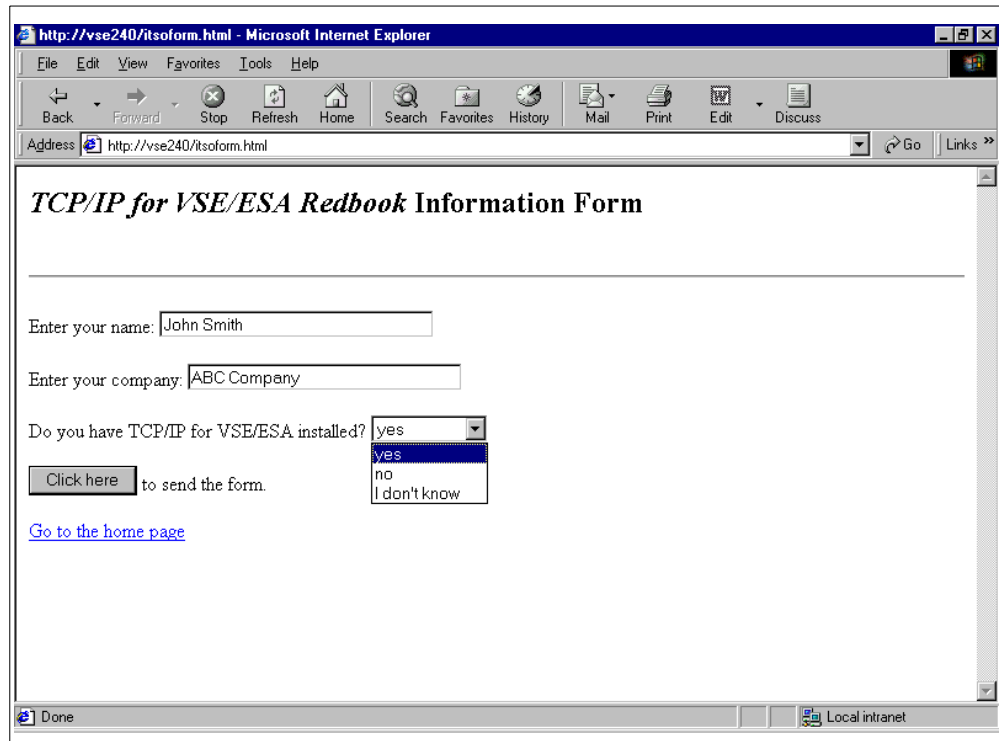


Figure 217. Web browser display of ITSIFORM.HTML Web page

The user fills out the input fields, selects an entry in the pull-down list and clicks the **Click here** button to send the data to the Web server. The Web server will invoke the CGI program we defined in the HTML form document to process the data sent by the Web browser.

## 8.4.2 Using CGI programs

TCP/IP for VSE/ESA supports CGI programs written in Assembler and REXX. The function of the CGI program is to process data sent by the Web browser and to dynamically build a document to be sent back to the Web browser.

Data is passed to the CGI program as a string of variable fields. TCP/IP for VSE/ESA passes this data along with the user ID and password of the logged on user to the CGI program. If security is not activated for the VSE Web site, then the user ID is set to ANONYMOUS and the password is blank.

To use CGI programs you have to:

- Define the programs in the TCP/IP for VSE/ESA configuration.
- Code the CGI programs and catalog them into a VSE sublibrary.
- Create an HTML document that invokes the CGI program such as our ITSIFORM.HTML document in Figure 216 on page 213.

### 8.4.2.1 Defining CGI programs

The `DEFINE CGI` command is used to define CGI programs that will be used by our VSE Web server. Figure 210 on page 204 lists the CGI programs that we defined in our configuration file. We defined two assembler language CGI programs and one REXX CGI program.



TCP/IP for VSE/ESA supports two types of assembler language CGI programs called format 1 and format 2 assembler CGIs.

The format 1 CGI uses an older macro interface supplied by the TCP/IP for VSE/ESA product with the FILEIOHD macro. Programs written to this interface are still supported, however, they can only be written as 24-bit applications. The following command is used to define a format 1 assembler CGI:

```
DEFINE CGI,PUBLIC='ASMCGI1',TYPE=CGI
```

The examples in the redbook *VSE/ESA as a Web Server*, SG24-2040 are format 1 assembler CGIs.

The format 2 assembler CGI uses a parameter list interface mapped with a CGIDATA macro. This interface is easier to use than the older format 1 macro interface and is recommended for new assembler CGI programs. Programs written to this interface can be 31-bit applications. The following command is used to define a format 2 assembler CGI:

```
DEFINE CGI,PUBLIC='ASMCGI2',TYPE=CGI-BAL
```

For assembler CGIs the public name is the phase name under which the CGI program was cataloged into a VSE sublibrary. For example, ASMCGI1.PHASE and ASMCGI2.PHASE are the member names of the phases in our TCPIP.HTML sublibrary. The sublibrary must be in the LIBDEF search string of the TCP/IP for VSE/ESA partition.

Examples for both types of assembler CGIs are in *TCP/IP for VSE Programmer's Reference, Release 1.4*.

We implemented a REXX CGI program to process the data from our form document. We used the following command to define our REXX CGI program:

```
DEFINE CGI,PUBLIC='REXXCGI1',TYPE=CGI-REXX
```

For REXX CGIs the public name is the procedure name under which the REXX procedure was cataloged in a VSE sublibrary. The REXX procedure must be cataloged in a sublibrary that is defined in the LIBDEF search chain for the TCP/IP for VSE/ESA partition.

For example, REXXCGI1.PROC is the name of our member in sublibrary TCPIP.HTML.

#### Note

The format of the `DEFINE` command to define CGI programs has changed with the current level 1.4 of TCP/IP for VSE/ESA. The old format used a `DEFINE FILE,TYPE=CGI` format. CGI programs are now defined separately from `DEFINE FILE` entries using the `DEFINE CGI` command. The old format is still accepted, however, you should use the new format when defining CGI programs to TCP/IP for VSE/ESA.

#### 8.4.2.2 Managing CGI programs

After we have defined our CGI programs in the TCP/IP for VSE/ESA configuration, we can use the `QUERY CGIS` command from the VSE console to

check our definitions. The following figure shows our VSE console log from this command:

```
Z1 0087 IPN300I Enter TCP/IP Command
87 query cgis
Z1 0085 IPN431I (( TCP/IP CGIs ))
Z1 0085 IPN432I Public Name: ASMCGI1
Z1 0085 IPN433I Type: CGI DLBL: Driver: IPNFCGIX
Z1 0085 IPN432I Public Name: ASMCGI2
Z1 0085 IPN433I Type: CGI-BAL DLBL: Driver: IPNFCGIB
Z1 0085 IPN432I Public Name: REXXCGI1
Z1 0085 IPN433I Type: CGI DLBL: CGI-REX Driver: IPNFCGI
Z1 0087 IPN300I Enter TCP/IP Command
```

Figure 218. Console output from the QUERY CGIS command

There are no commands to modify or remove CGI definitions from the running TCP/IP for VSE/ESA configuration. The `DELETE CGI` command is available to allow you to remove a CGI program from memory and reload an updated copy. To load a new copy of our REXX CGI program we would enter the following command:

```
DELETE CGI,PUBLIC='REXXCGI1'
```

#### Note

An older method of deleting the CGI used in the redbook *VSE/ESA as a Web Server*, SG24-2040 and earlier levels of the TCP/IP for VSE/ESA documentation is to delete the REXX driver module IPNFCGI used by the REXX CGI programs. The following is the command used to do this:

```
DELETE FILEIO,ID=IPNFCGI
```

The REXX CGI definition is not deleted as a result of this `DELETE` command, however, the REXX CGI program is reloaded the next time it is requested.

### 8.4.3 Our sample REXX CGI program

Figure 219 on page 217 shows the source code for our sample REXX CGI program and the job stream to catalog it into our TCPIP.HTML library. TCP/IP for VSE/ESA passes to our program three parameters consisting of the user ID, password and data sent by the Web browser.

The data passed by the Web browser to the VSE Web server is formatted into a string of variables. Each variable starts with an ampersand (&) followed by the name defined in the form, an equal sign (=), and then the data value. For example, the data parameter passed to our REXX CGI program from our ITSOFORM.HTML would look like the following:

```
&yourname=John Smith&company=ABC Company&tcpinst=yes
```

The REXX program would use the REXX parse commands to select each of the fields for processing.

```

// JOB REXXCGI      CATALOG TCP/IP REXX CGI
// EXEC LIBR
A S=TCPIP.HTML
CATALOG REXXCGI1.PROC                      REPLACE=YES
/* REXX */
userid=arg(1)
password=arg(2)
data=arg(3)
Say "userid="userid
Say "password="password
Say "data="data
/* construct html response */
x=html('<html><head><title>')
x=html('VSE REXX CGI Program')
x=html('</title></head>')
x=html('<body text=black bgcolor="#C0C0C0">')
x=html('<center><h1>TCP/IP Redbook REXX CGI Example</h1>')
x=html('<h2><i>')
x=html('The CGI request was processed by a REXX CGI program.')
x=html('</i></h2></center>')
x=html('<br><br>The input data received at' date() time()':<br>')
x=html(data)
x=html('</body></html>')
EXIT 0
/* end of rexx cgi html response */
/+
/*
/&

```

Figure 219. Source code for our sample REXX CGI program

After the data is processed, the CGI program constructs a document that will be sent to the Web browser. The REXX CGI program uses the REXX HTML function to pass HTML source to the HTTP daemon, which in turn, sends the dynamically constructed document to the Web browser.

In our example we constructed a Web page displaying the date and time and the string of data that was received from the Web browser. In a real environment the REXX CGI program may have stored the data received from the Web browser in a database or file and sent some acknowledgment confirming the input.

Figure 220 on page 218 shows the Web page displayed in our Web browser.

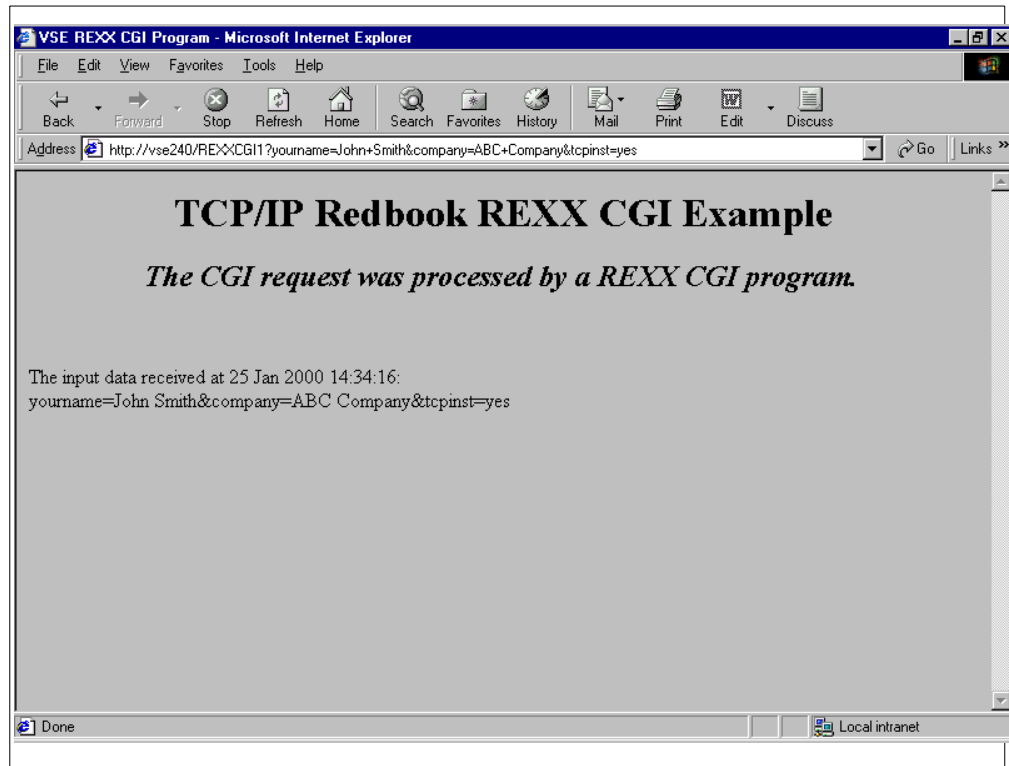


Figure 220. Web page created by our REXX CGI program

See *TCP/IP for VSE Programmer's Reference, Release 1.4* for more examples of writing CGI programs. You can also obtain several CGI program examples from Connectivity System's Web site at:

<http://www.tcpip4vse.com/progsamps.html>

---

## Appendix A. Special notices

This publication is intended to help customers and IBM technical personnel in the installation and usage of IBM TCP/IP for VSE/ESA Release 1.4. The information in this publication is not intended as the specification of any programming interfaces that are provided by TCP/IP for VSE/ESA. See the PUBLICATIONS section of the IBM Programming Announcement for TCP/IP for VSE/ESA for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

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

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

|                    |                    |
|--------------------|--------------------|
| eNetwork           | ESCON              |
| IBM Global Network | IBM ®              |
| MQ                 | MQSeries           |
| Netfinity          | Operating System/2 |
| OS/2               | OS/390             |
| RISC System/6000   | RS/6000            |
| S/390              | SP                 |
| System/390         | VM/ESA             |
| VSE/ESA            | VTAM               |
| XT                 | 400                |

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

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

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

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

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

UNIX is a registered trademark in the United States and other countries licensed exclusively through the Open Group.

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

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

---

## Appendix B. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

---

### B.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 223.

- *VSE/ESA as a Web Server*, SG24-2040
- *MQSeries for VSE/ESA*, SG24-5647
- *The Native TCP/IP Solution for VSE*, SG24-2041
- *Network Products Reference*, GX28-8002

---

### B.2 IBM Redbooks collection

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at <http://www.redbooks.ibm.com/> for information about all the CD-ROMs offered, updates and formats.

| CD-ROM Title   | Collection Kit Number |
|--|-----------------------|
| System/390 Redbooks Collection                                 | SK2T-2177             |
| Networking and Systems Management Redbooks Collection          | SK2T-6022             |
| Transaction Processing and Data Management Redbooks Collection | SK2T-8038             |
| Lotus Redbooks Collection                                      | SK2T-8039             |
| Tivoli Redbooks Collection                                     | SK2T-8044             |
| AS/400 Redbooks Collection                                     | SK2T-2849             |
| Netfinity Hardware and Software Redbooks Collection            | SK2T-8046             |
| RS/6000 Redbooks Collection (BkMgr Format)                     | SK2T-8040             |
| RS/6000 Redbooks Collection (PDF Format)                       | SK2T-8043             |
| Application Development Redbooks Collection                    | SK2T-8037             |
| IBM Enterprise Storage and Systems Management Solutions        | SK3T-3694             |

---

### B.3 Other resources

These publications are also relevant as further information sources:

#### IBM publications

- *TCP/IP for VSE/ESA - IBM Program Setup and Supplementary Information*, SC33-6601
- *CICS Transaction Server for VSE/ESA CICS-Supplied Transactions*, SC33-1655

#### Connectivity Systems, Inc. publications

- *TCP/IP for VSE Installation Guide, Release 1.4*
- *TCP/IP for VSE Commands, Release 1.4*
- *TCP/IP for VSE User's Guide, Release 1.4*
- *TCP/IP for VSE Programmer's Reference, Release 1.4*

- *TCP/IP for VSE Messages and Codes, Release 1.4*
- *TCP/IP for VSE Optional Products, Release 1.4*

#### **TCP/IP and Internet publications**

- *Internetworking with TCP/IP, Volume I, Principles, Protocols and Architecture*, Third Edition, Prentice-Hall, Inc., 1995, by Douglas E. Comer; ISBN 0-13-216987-8.
- *TCP/IP Tutorial and Technical Overview*, Sixth Edition, IBM Corp., 1998, GG24-3376-05, and Prentice-Hall, Inc., 1998, by Martin W. Murhammer, Orcun Atakan, Stefan Bretz, Larry R. Pugh, Kazunari Suzuki, David H. Wood; ISBN 0-13-020130-8.
- *IPng and the TCP/IP Protocols*, John Wiley & Sons, Inc., 1996, by Stephen A. Thomas; ISBN 0-471-13088-5.
- *IP Multicasting*, McGraw-Hill, 1999, by Marcus Goncalves and Kitty Niles; ISBN 0-07-913791-1.

---

## **B.4 Referenced Web sites**

These Web sites are also relevant as further information sources:

<http://www.isoc.org/internet-history/>  
<http://www.ietf.org/rfc.html>  
<http://rs.internic.net/>  
<http://www.ibm.com/javainfo/>  
<http://www.java.sun.com/>  
<http://www.socks.nec.com>  
<http://www.tcpip4vse.com/progsamps.html>  
<http://www.adobe.com>



---

## How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** <http://www.redbooks.ibm.com/>

Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

Send orders by e-mail including information from the IBM Redbooks fax order form to:

|                       |   |
|-----------------------|---|
|                       | <b>e-mail address</b>   |
| In United States      | usib6fpl@ibmmail.com  |
| Outside North America | Contact information is in the "How to Order" section at this site:<br><a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a> |

- **Telephone Orders**

|                           |  |
|---------------------------|--|
| United States (toll free) | 1-800-879-2755   |
| Canada (toll free)        | 1-800-IBM-4YOU   |
| Outside North America     | Country coordinator phone number is in the "How to Order" section at this site:<br><a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a> |

- **Fax Orders**

|                           |  |
|---------------------------|--|
| United States (toll free) | 1-800-445-9269   |
| Canada                    | 1-403-267-4455   |
| Outside North America     | Fax phone number is in the "How to Order" section at this site:<br><a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a> |

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

### IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at <http://w3.itso.ibm.com/> and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at <http://w3.ibm.com/> for redbook, residency, and workshop announcements.

---

## IBM Redbooks fax order form

Please send me the following:

| Title | Order Number | Quantity |
|-------|--------------|----------|
|       |              |          |
|       |              |          |
|       |              |          |
|       |              |          |
|       |              |          |
|       |              |          |
|       |              |          |
|       |              |          |

---

|            |           |
|------------|-----------|
| First name | Last name |
|------------|-----------|

---

|         |
|---------|
| Company |
|---------|

---

|         |
|---------|
| Address |
|---------|

---

|      |             |         |
|------|-------------|---------|
| City | Postal code | Country |
|------|-------------|---------|

---

|                  |                |            |
|------------------|----------------|------------|
| Telephone number | Telefax number | VAT number |
|------------------|----------------|------------|

---

|   |  |
|---|--|
| <input type="checkbox"/> Invoice to customer number |  |
|---|--|

---

|   |  |
|---|--|
| <input type="checkbox"/> Credit card number |  |
|---|--|

---

|                             |                |           |
|-----------------------------|----------------|-----------|
| Credit card expiration date | Card issued to | Signature |
|-----------------------------|----------------|-----------|

**We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.**

## Abbreviations and acronyms

|                 |  |               |   |
|-----------------|--|---------------|---|
| <b>ACF/VTAM</b> | Advanced Communications Facility/Virtual Telecommunications Access Method  | <b>DASD</b>   | direct access storage device                        |
| <b>AH</b>       | Authentication Header  | <b>DHCP</b>   | Dynamic Host Configuration Protocol                 |
| <b>APAR</b>     | authorized program analysis report   | <b>DLBL</b>   | Disk Label  |
| <b>API</b>      | application programming interface  | <b>DNS</b>    | Domain Name System                                  |
| <b>APPN</b>     | Advanced Peer-to-Peer Networking   | <b>DOD</b>    | U.S. Department of Defense                          |
| <b>ARP</b>      | Address Resolution Protocol  | <b>DOS</b>    | Disk Operating System                               |
| <b>ARPA</b>     | Advanced Research Projects Agency  | <b>DVMRP</b>  | Distance Vector Multicast Routing Protocol          |
| <b>ASCII</b>    | American Standard Code for Information Interchange                         | <b>EBCDIC</b> | Extended Binary Communication Data Interchange Code |
| <b>ASI</b>      | Automatic System Initialization  | <b>ESCON</b>  | Enterprise Systems Connection                       |
| <b>ATM</b>      | asynchronous transfer mode   | <b>ESDS</b>   | Entry Sequenced Data Set                            |
| <b>BIND</b>     | Berkeley Internet Name Domain  | <b>ESP</b>    | Encapsulating Security Payload                      |
| <b>BOOTP</b>    | bootstrap protocol   | <b>FDDI</b>   | Fiber Distributed Data Interface                    |
| <b>BSD</b>      | Berkeley Software Distribution   | <b>FR</b>     | frame relay   |
| <b>CCITT</b>    | Comite Consultatif International Telegraphique et Telephonique (now ITU-T) | <b>FSU</b>    | Fast Service Upgrade                                |
| <b>CD</b>       | compact disc   | <b>FTP</b>    | File Transfer Protocol                              |
| <b>CERN</b>     | Conseil Européen pour la Recherche Nucléaire                               | <b>GMT</b>    | Greenwich Mean Time                                 |
| <b>CGI</b>      | Common Gateway Interface   | <b>GPS</b>    | General Print Server                                |
| <b>CICS</b>     | Customer Information Control System  | <b>GUI</b>    | graphical user interface                            |
| <b>CLAW</b>     | Common Link Access to Workstation  | <b>HTML</b>   | Hypertext Markup Language                           |
| <b>COBOL</b>    | Common Business Oriented Language  | <b>HTTP</b>   | Hypertext Transfer Protocol                         |
| <b>CPU</b>      | central processing unit  | <b>I/O</b>    | input/output  |
| <b>CSD</b>      | CICS System Definition   | <b>IAB</b>    | Internet Activities Board                           |
| <b>CSI</b>      | Connectivity Systems, Inc.   | <b>IANA</b>   | Internet Assigned Number Authority                  |
| <b>CSMA/CD</b>  | carrier sense multiple access with collision detection                     | <b>IBM</b>    | International Business Machines Corporation         |
| <b>CTC</b>      | Channel-to-Channel Unit  | <b>ICCF</b>   | Interactive Computing and Control Facility          |
| <b>CTCA</b>     | Channel-to-Channel Adapter   | <b>ICMP</b>   | Internet Control Message Protocol                   |
| <b>DARPA</b>    | Defense Advanced Research Projects Agency                                  | <b>IEEE</b>   | Institute of Electrical and Electronics Engineers   |
|                 |  | <b>IETF</b>   | Internet Engineering Task Force                     |
|                 |  | <b>IGMP</b>   | Internet Group Management Protocol                  |
|                 |  | <b>IGN</b>    | IBM Global Network                                  |

|               |   |                |  |
|---------------|---|----------------|--|
| <b>IMAP</b>   | Internet Message Access Protocol                          | <b>NNTP</b>    | Network News Transfer Protocol   |
| <b>IP</b>     | Internet Protocol   | <b>NSF</b>     | National Science Foundation  |
| <b>IPL</b>    | Initial Program Load                                      | <b>OEM</b>     | original equipment manufacturer  |
| <b>IPSec</b>  | IP Security Architecture                                  | <b>OS/2</b>    | Operating System/2   |
| <b>IPv4</b>   | Internet Protocol Version 4                               | <b>OSA</b>     | Open Systems Adapter   |
| <b>IPX</b>    | Internetwork Packet Exchange                              | <b>OSI</b>     | Open Systems Interconnect  |
| <b>IRFT</b>   | Internet Research Task Force                              | <b>OSPF</b>    | Open Shortest Path First   |
| <b>ISAKMP</b> | Internet Security Association and Key Management Protocol | <b>OS/2</b>    | Operating System/2   |
|               |   | <b>OS/390</b>  | Operating System for the System/390 platform                             |
| <b>ISDN</b>   | integrated services digital network                       | <b>PC</b>      | personal computer  |
| <b>ISO</b>    | International Organization for Standardization            | <b>PDA</b>     | personal digital assistant   |
| <b>ITSO</b>   | International Technical Support Organization              | <b>PIM</b>     | Protocol Independent Multicast   |
| <b>ISOC</b>   | Internet Society  | <b>PING</b>    | packet internet groper   |
| <b>JCL</b>    | job control language                                      | <b>POP</b>     | Post Office Protocol   |
| <b>JVM</b>    | Java Virtual Machine                                      | <b>PPP</b>     | Point-to-Point Protocol  |
| <b>KSDS</b>   | Key Sequenced Data Set                                    | <b>PPTP</b>    | Point-to-Point Tunneling Protocol  |
| <b>LAN</b>    | local area network  | <b>PTF</b>     | program temporary fix  |
| <b>LDAP</b>   | Lightweight Directory Access Protocol                     | <b>RARP</b>    | Reverse Address Resolution Protocol                                      |
| <b>LLC</b>    | logical link control                                      | <b>RDO</b>     | Resource Definition Online   |
| <b>LPD</b>    | Line Printer Daemon                                       | <b>REXEC</b>   | Remote Execution Command Protocol  |
| <b>LPR</b>    | Line Printer Requester                                    | <b>REXX</b>    | Restructured Extended Executor   |
| <b>LU</b>     | logical unit  | <b>RFC</b>     | Request for Comments   |
| <b>L2F</b>    | Layer 2 Forwarding  | <b>RIP</b>     | Routing Information Protocol   |
| <b>L2TP</b>   | Layer 2 Tunneling Protocol                                | <b>RPC</b>     | remote procedure call  |
| <b>MAC</b>    | media access control                                      | <b>RSH</b>     | Remote Shell   |
| <b>MB</b>     | Megabyte  | <b>RS/6000</b> | IBM RISC System/6000   |
| <b>MIB</b>    | Management Information Base                               | <b>SAM</b>     | Sequential Access Method   |
| <b>MIME</b>   | Multipurpose Internet Mail Extensions                     | <b>SLIP</b>    | Serial Line Internet Protocol  |
|               |   | <b>SMI</b>     | Structure of Management Information                                      |
| <b>MOSPF</b>  | Multicast Open Shortest Path First                        | <b>SMTP</b>    | Simple Mail Transfer Protocol  |
| <b>MTU</b>    | maximum transmission unit                                 | <b>SNA</b>     | Systems Network Architecture   |
| <b>MVS</b>    | Multiple Virtual Storage Operating System                 | <b>SNMP</b>    | Simple Network Management Protocol                                       |
| <b>NAT</b>    | network address translation                               | <b>SOCKS</b>   | SOCK-et-S (An internal NEC development name that remained after release) |
| <b>NCSA</b>   | National Computer Security Association                    | <b>SSL</b>     | Secure Sockets Layer   |
| <b>NFS</b>    | Network File System                                       |                |  |

|                  |   |
|------------------|---|
| <b>TCP</b>       | Transmission Control Protocol   |
| <b>TCP/IP</b>    | Transmission Control Protocol<br>/Internet Protocol   |
| <b>TFTP</b>      | Trivial File Transfer Protocol  |
| <b>TLBL</b>      | Tape Label  |
| <b>TTL</b>       | time to live  |
| <b>UDP</b>       | User Datagram Protocol  |
| <b>URL</b>       | Uniform Resource Locator  |
| <b>USSTAB</b>    | Unformatted System Services<br>Table  |
| <b>VM</b>        | virtual machine   |
| <b>VM/ESA</b>    | Virtual Machine/Enterprise<br>Systems Architecture  |
| <b>VPN</b>       | virtual private network   |
| <b>VRID</b>      | virtual route identifier  |
| <b>VRML</b>      | Virtual Reality Modeling<br>Language  |
| <b>RRP</b>       | Virtual Router Redundancy<br>Protocol   |
| <b>VSAM</b>      | Virtual Storage Access<br>Method  |
| <b>VSE/ESA</b>   | Virtual Storage<br>Extended/Enterprise Systems<br>Architecture                                    |
| <b>VSE/POWER</b> | Virtual Storage<br>Extended/Priority Output<br>Writers, Execution processor,<br>and input Readers |
| <b>VSE/VSAM</b>  | Virtual Storage<br>Extended/Virtual Storage<br>Access Method                                      |
| <b>VTAM</b>      | Virtual Telecommunications<br>Access Method   |
| <b>WAN</b>       | wide area network   |
| <b>WARP</b>      | Workstation Asset Reduction<br>Program  |
| <b>WWW</b>       | World Wide Web  |
| <b>XDR</b>       | External Data Representation  |
| <b>X11</b>       | X Window System Version 11  |
| <b>X.25</b>      | CCITT Packet Switching<br>Standard  |



---

# Index

## Numerics

3172 Interconnect Controller  
    DEFINE ADAPTER command 63  
    DEFINE LINK command 63

## A

Address Resolution Protocol (ARP) 7  
AppleTalk 17, 18  
ARPANET 1  
ATM 20  
ATM switch 20  
AUTOLPR 160  
automatic LPR 160

## B

batch FTP client 120  
batch LPR 166  
BOOTP 7  
bridges 18  
bridging 14

## C

carriage control characters 102, 157  
CC option 157  
CGI 203, 212  
    commands 216  
    definition 214  
    format 1 215  
    format 2 215  
    program invocation 213  
    program use 214  
channel-to-channel adapter  
    DEFINE LINK command 63  
Channel-to-Channel support (CTC) 6  
CICS  
    customizing 71  
    definitions for GPS 172  
    DFHCSD entries 74  
    FTP client 111  
    FTP transaction 86  
    IPNCSD.Z 72  
    table entries 72  
    Telnet client 137, 141  
    transaction LPR 154  
circuit switching 17  
class A addresses 9  
class B addresses 9  
class C addresses 9  
class D addresses 9  
CLAW protocol 6  
command-line interface 152  
Common Link Access to Workstation 6  
creating forms 212

## D

daemons  
    File Transfer Protocol 83  
    FTP 65, 83  
    GPS 68  
    HTTP 68  
    Hypertext Transfer Protocol 203  
    LPD 149  
    NFS 68, 182  
    Telnet 64, 134  
DECnet 18  
DEFINE commands  
    CGI 68, 214  
    CGI program 204  
    DEFINE EVENT 127  
    EVENT 160  
    FILE 67, 85, 98  
    FILESYS 85  
    FTPD 65, 83  
    FYLESYS 98  
    GPSD 68, 175  
    HTTPD 68, 204  
    LINK 63  
    LPD 65, 149, 152  
    NAME 66, 161  
    NFSD 68, 182, 188  
    ROUTE 64  
    TELNETD 64, 71, 134  
    USER 67  
DELETE CGI command 216  
DELETE commands  
    HTTPD 206  
DELETE FTPD command 131  
DELETE GPSD command 178  
DELETE NFSD command 188  
DELETE TELNETD command 147  
DFHCSD entries 74  
DHCP 7  
DIAGNOSE LPR command 168  
DNS 28  
    Domain Name Server (DNS) 4  
Domain Name System 4  
DSPACE parameter 52

## E

ENTR 5  
ETHERNET/IEEE 15

## F

Fast Ethernet 5  
FDDI 5  
File Transfer Protocol (FTP) 26, 28  
Firewalls 36  
    proxy servers 36  
    screening filters 36  
    SOCKS servers 37

## FTP

- automatic 127
- Autonomous 98, 120, 126
- batch client 120
- CICS transaction 86, 111
- definitions 83
- DELETE command 131
- external batch client 120, 126
- file types 207
- FTPBatch 120, 126
- internal batch client 120, 121
- POWER queues 101
- QUERY FTPDS 130
- security 84
- VSAM files 105
- VSE as server 98

FTPBatch 120, 126

## G

GET command 98, 101, 105, 120

Gopher 31

## GPS

- CICS definitions 172
- command file for EXECUTE 177
- daemon definition parameter 176
- definition 175
- DELETE command 178
- DIAGNOSE LPR command 180
- installation 171
- overview 171
- product key 171
- QUERY command 178
- test definitions 179
- VTAM definitions 171

## H

Hardware Address Resolution 7

HTML document 206

- creation 206
- example 207
- security 210
- tags 206

HTML tag 206

HTTP 32

- HTTP daemon 204
  - commands 205
  - default port 208
  - definition 204
  - file type 206
  - security 209

hub 17

hyperlink 208

## I

IAB 3

IBM 3088 6

IBM 9221 Integrated Communication Processor 5

IBM Open Systems Adapter 2 5

ICMP 14

INSERTS macro 164

INSERTS phase 160, 163, 176

- example 164

Internet Activities Board (IAB) 2

Internet Protocol (IP) 8

- address 61, 66
- direct routing 13
- gateway 13
- indirect routing 13
- router 13
- routing 13
- routing table 13

Internet security 34

Internetwork layer 4

IP

- datagram 12

IPINIT file 53, 85, 133, 149, 175, 182, 188, 204

IPNETCMD 80, 189

IPX 2, 17

ISOC 2

## J

Java 33

- applets 33, 203
- byte-code 33
- compiler 33
- HotJava 34
- Java Beans 34
- JavaOS 34
- programming language 33
- servlets 34
- signed applets 34
- Virtual Machine (JVM) 33

JavaScript 34

## L

LAN 15, 19

- bridges 16
- extenders 16
- media-access methods 15
- switch 20
- switches 16
- topology 16

Logical Link Control (LLC) 19

logical record length 119

LONG command 159

loopback 10

LPD 28, 149

- access from workstation 152
- daemon definition parameter 150
- definition 149
- example 152
- operation commands 151
- QUERY LPDS command 151

LPR 28, 72, 149, 154

- automatic 160

- batch 166

- CICS transaction 154



- commands 155
- DEST parameter 160
- DIAGNOSE command 168, 180
- events 160
- HOST parameter 160
- INSERTS phase 164
- LONG command 159
- POWER LST statement 161, 162
- PRINT command 158
- PRINTER parameter 160
- QUIT command 159
- return codes 167
- scripting facility 162
- SHORT command 159
- USER parameter 161
- luname 71

## M

- mapping 29
- maximum transmission unit (MTU) 63
- Media Access Control (MAC) 19
- Multiaccess broadcast networks 7

## N

- NetBIOS 18
- network
  - interface layer 4
  - protocols 24
- Network Computer 34
- Network File System (NFS) 31
- NFS
  - access NFS server on VSE 196
  - client configuration 191
  - configuration 182, 183
  - file type considerations 202
  - FLUSH command 190
  - installation 181
  - mount point 182
  - overview 181
  - product key 181
  - QUERY command 189
  - start 188
  - termination 188
- NFSCFG.L configuration file 183
- NFSMODEL.L configuration file 187
- NFSNET 1
- NFSTYPES.L configuration file 185

## O

- OPEN command 121
- OSA 6
  - DEFINE ADAPTER command 63
  - DEFINE LINK command 63
- OSA-2 5

## P

- packet switching 17
- partition size 50

- PASS command 121
- path determination 22
- PCT entries 72
- ping 14, 76
- point-to-point
  - link 17
  - networks 7
- port 24
- port number 133
- POWER
  - LST queue classes 160
  - LST statement 161, 162
  - queues 119
- PPT entries 72
- PRINT command 156, 158
- print on workstation 158
- product key 42, 171, 181
- protocols
  - File Transfer 83
  - HTT 203
  - TN3270 133
- public name 65, 105, 150
- PUT command 101, 105, 116, 119, 120

## Q

- QUERY commands
  - CGIS 215
  - EVENTS 160
  - FTPDS 130
  - GPSD 178
  - HTTPDS 205
  - LPDS 151
  - NFS 189
  - OPT 157
  - TELNETDS 130, 146
- QUIT command 159

## R

- RARP 7
- remote bridges 19
- remote login 27
- Request for Comments (RFC) 2
- return codes for LPR 167
- REXX CGI program 215, 216
- RFC 3
- ring topology 16
- routing 14, 21, 23
  - bandwidth 24
  - delay 23
  - dynamic 23
  - load 24
  - metrics 23
  - multipath 23
  - path length 23
  - reliability 23
  - single-path 23
  - static 23
- routing table 13

## S

- scripting facility 162
- Secure Sockets Layer (SSL) 35
- secured Web site 209
- security exit 63, 67
- SET commands
  - CC 157
  - GATEWAY 62
  - HOST 155, 158
  - INSERTS 166
  - MASK 61
  - PRINTER 155, 158
  - SECURITY 63, 84
- SETVAR command 129
- SHORT command 159
- SNA/APPN 6
- sockets 24
  - interface 25
- SSL
  - authentication 35
  - integrity 35
  - privacy 35
- star topology 16
- switches 18
- switching 14
- symbolic name 66, 161

## T

- TCP/IP
  - documentation 39
  - installation for VSE 40
  - IPINIT file 53
  - partition size 50
  - product key 42
  - protocol 27
- TCP/IP operation 79
  - batch utility 80
  - console input 79
  - console messages 82
  - restart 81
  - shutdown 81
- Telnet
  - batch client 144
  - CICS client 141
  - CICS definitions 137
  - CICS transaction 141, 142
  - command 141
  - DEFINE TELNETD command 134
  - definition 133
  - DELETE command 147
  - line mode connection 142
  - menu definition 135
  - QUERY command 146
  - remote login 27
  - terminal emulation 27
  - TN3270 Plus 138
  - VTAM APPL definitions 134
- terminal emulator session 154
- The Internet Architecture Board (IAB) 2

- The Internet Assigned Number Authority (IANA) 3
- The Internet Engineering Task Force (IETF) 3
- The Internet Research Task Force (IRTF) 3
- TN3270 28
  - protocol 133
- TN3270 Plus 138
  - usage 139
- Traceroute 14, 78
- Transmission Control Protocol (TCP) 26
- transport layer 4
- tree topology 16

## U

- USER command 121
- User Datagram Protocol (UDP) 25

## V

- VCTC 6
- VSAM
  - ESDS file transfer 106
  - file transfer 105
  - files 105, 151
  - KSDS file transfer 105
- VSE
  - DEFINE LINK command 63
  - FTP batch 120
  - FTP daemon 98
  - GET to workstation 98
  - local POWER queues 119
  - LPR commands 155
  - partition size for TCP 50
  - PUT workstation file 101
  - startup job 50
- VTAM
  - customizing 71
  - DEFINE TELNETD command 71
  - definitions for GPS 171
  - DSPACE parameter 52
  - luname 71
  - Telnet definitions 134

## W

- Web
  - browser 203
  - CGI programs 212
  - dynamic page 203
  - HTTP daemon definition 204
  - hyperlink 208
  - Java applets 203
  - port number 205
  - securing server 209
  - server 203
  - site creation 203
  - source code for Web site 207
  - static page 203
- Web browser 34
- wide area network (WAN) 17
- Winsock 25

**X**  
XNS 18



---

## IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at <http://www.redbooks.ibm.com/>
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to [redbook@us.ibm.com](mailto:redbook@us.ibm.com)

|   |  |  |   |  |  |  |  |
|---|--|--|---|--|--|--|--|
| <b>Document Number</b>  | SG24-5626-00   |  |   |  |  |  |  |
| <b>Redbook Title</b>  | Getting Started with TCP/IP for VSE/ESA 1.4  |  |   |  |  |  |  |
| <b>Review</b>   | <table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr></table>  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
| <b>What other subjects would you like to see IBM Redbooks address?</b>  | <table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr></table>  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
|   |  |  |   |  |  |  |  |
| <b>Please rate your overall satisfaction:</b>   | <input type="radio"/> Very Good <input type="radio"/> Good <input type="radio"/> Average <input type="radio"/> Poor  |  |   |  |  |  |  |
| <b>Please identify yourself as belonging to one of the following groups:</b>  | <input type="radio"/> Customer<br><input type="radio"/> Business Partner<br><input type="radio"/> Solution Developer<br><input type="radio"/> IBM, Lotus or Tivoli Employee<br><input type="radio"/> None of the above               |  |   |  |  |  |  |
| <b>Your email address:</b><br>The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities. | <table><tr><td></td></tr><tr><td><input type="radio"/> Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.</td></tr></table> |  | <input type="radio"/> Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction. |  |  |  |  |
|   |  |  |   |  |  |  |  |
| <input type="radio"/> Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction. |  |  |   |  |  |  |  |
| <b>Questions about IBM's privacy policy?</b>  | The following link explains how we protect your personal information.<br><a href="http://www.ibm.com/privacy/yourprivacy/">http://www.ibm.com/privacy/yourprivacy/</a>   |  |   |  |  |  |  |

SG24-5626-00

Printed in the U.S.A.

