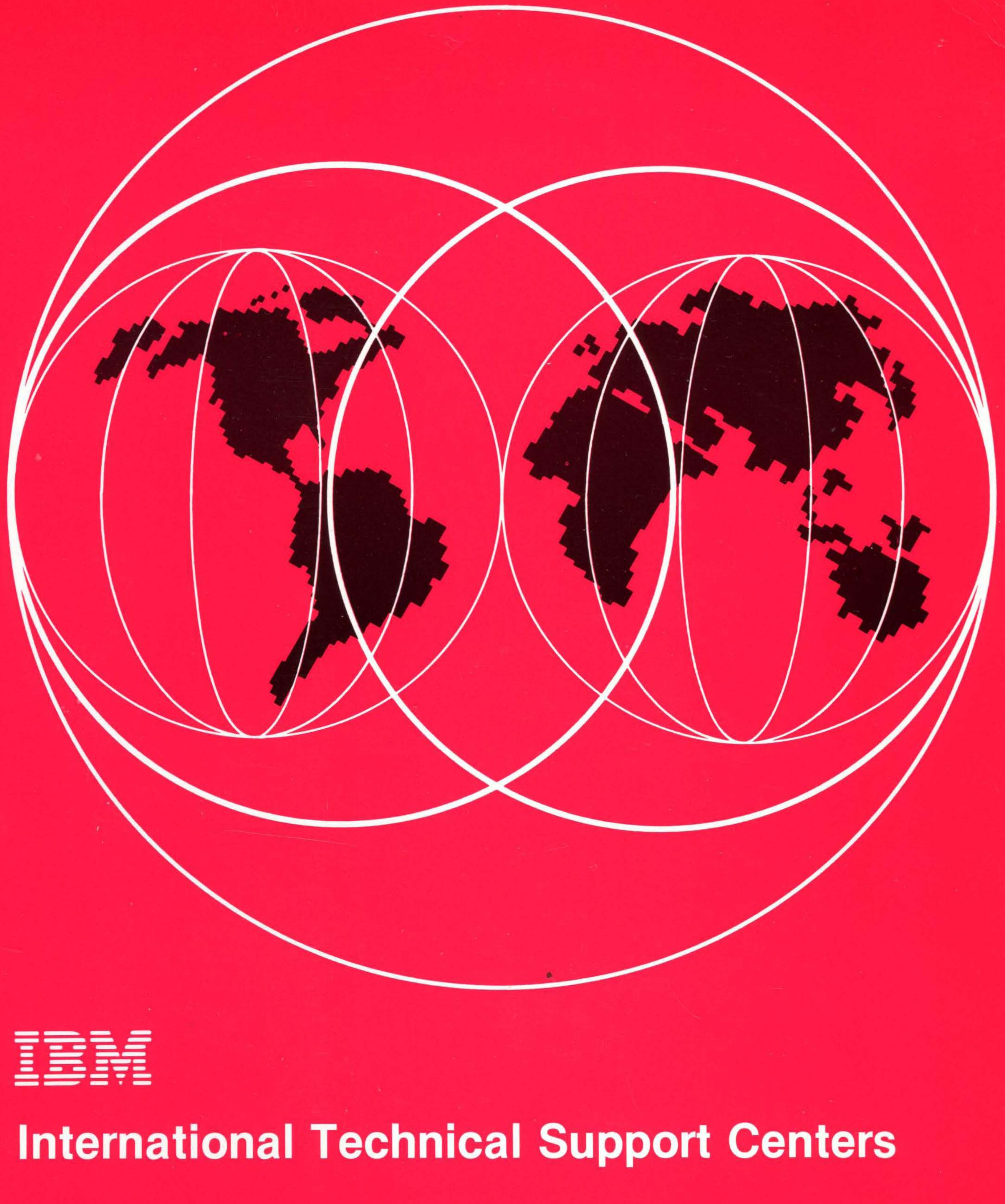


IBM

International Technical Support Centers

AIX COMMUNICATIONS HANDBOOK

GG24-3381-00

AIX Communications Handbook

Document Number GG24-3381-0

June 16, 1989

International Technical Support Center
Austin, Texas

First Edition (June 1989)

This edition applies to the Advanced Interactive Executive/RT Operating System (AIX/RT) Version 2.2.1 and the Advanced Interactive Executive PS/2 (AIX PS/2) Operating System Version 1.1.

Abstract

This publication is intended for IBM systems engineers, authorized dealers of AIX-based systems and customers as an aid to using and customizing most of the AIX communications products. Products covered in this publication are:

1. AIX Base Operating System
2. Asynchronous Terminal Emulation (ATE)
3. Basic Networking Utilities (BNU)
4. SNA Services
5. Distributed Services (DS)
6. 3278/79 Emulation Program
7. Workstation Host Interface Program (WHIP)
8. Network 3270-PLUS
9. TCP/IP
10. Network File System (NFS)
11. Simple Mail and Mail Handler (MH)
12. DOS Server
13. AIX Access for DOS Users (AADU)
14. X-Windows
15. X-Windows for DOS
16. IBM 6150 X.25 Communications Support

This publication renders obsolete the following publications:

- *IBM RT PC Communications Cookbook Volume 1*, GG24-3124
- *IBM RT PC Communications Cookbook Volume 2*, GG24-1542
- *IBM RT PC Communications Cookbook Volume 3*, GG24-3244.

AIX (630 pages)

Acknowledgements

We gratefully acknowledge the contributions of many IBMers from all over the world. We'd like to thank the following for direct assistance and for their permission to include excerpts from various documents.

AS/400 Communications:
Thanks to Marcela Adan (ITSC Rochester) and Stella Currie (ITSC Poughkeepsie) for assistance during tests with AS/400 and contributions to this publication.

3270 Host Communications:
Thanks to Bernard Pusch (IBM UK) for permission to use parts of his documents about Network 3270-Plus.

DS and the SLIP for TCP/IP:
Thanks to Neil Young and David Williams (IBM UK) for permission to use their documents and for useful hints.

X.25 Communications:
Thanks to Martin Haardt (IBM Germany) whose paper on X.25 communications was the basis for the chapter about X.25.

TCP/IP:

Thanks to Alan Reinhold (IBM U.S.A., Raleigh) for permission to use excerpts from his *Communications Systems Bulletin* about TCP/IP (GG22-9125).

Network 3270-PLUS translate tables:

Thanks to Einar Johannesson (IBM Iceland) for permission to use his notes about changing the ASCII/EBCDIC translation tables of Network 3270-PLUS.

Network 3270-PLUS sample programs:

Thanks to Phillippe Crepin (Currently IBM Germany) for permission to include the sample programs for Network 3270-PLUS file transfer adapted by him.

Last, but certainly not least, we'd like to express our appreciation of the help we have received from many IBMers of the IBM Austin Laboratory. They have jumped in with help when everything looked hopeless, and they have helped us by reading through this document and correcting our numerous errors and misunderstandings.

Jane Britton
Carylin Greene
Daryl Greene
Liz Hughes
Greg Joyce
Kent Malave
Hye-Young McCreary
Lance Russell
Wayne Vicknair

Authors

The authors of this publication are:

Dieter Hepper, IBM Germany

Jan Norbäck, IBM Sweden

Johnny Lauridsen, IBM Denmark

Niels Christiansen, ITSC Austin (project advisor)

Trademarks

The following trademarks apply to this publication.

- IBM, Personal Computer AT and AT, RT, RT PC, and RT Personal Computer, Personal System/2 and PS/2 are registered trademarks of the International Business Machines Corporation.
- AIX, Personal Computer XT and XT, Micro Channel, MVS/ESA, MVS/XA and PROFS are trademarks of the International Business Machines Corporation.
- ARPANET is developed by the United States Department of Defense.
- DEC VT100, DEC VT220 and DEC VT320, VAX, VMS and ULTRIX are trademarks of Digital Equipment Corporation.
- Ethernet and Xerox are trademarks of Xerox, Inc.
- Etherlink is a trademark of the 3COM Corporation.
- HYPERchannel is a trademark of Network Systems Corporation.
- INed, INmail and INnet are trademarks of INTERACTIVE Systems Corporation.
- Intel is a trademark of Intel Corporation.
- Network 3270-PLUS and Network RJE-PLUS are trademarks of Rabbit Software Corporation.
- NIC and NICps/2 are trademarks of Ungermann-Bass.
- ProNET is a trademark of Proteon, Inc.
- SIM3278/TCPIP is a trademark of Simware Inc.
- Smartcom II and Smartmodem are trademarks of Hayes Microcomputer Products, Inc.
- Sun Microsystems, NFS and SunOS are trademarks of SUN Microsystems, Inc.
- UNIX is a trademark of AT&T Bell Laboratories.
- X-Window System is a trademark of the Massachusetts Institute of Technology.

Preface

This publication summarizes the results of an IBM residency held at the International Technical Support Center in Austin during March and April 1989. Our intention with the publication is to provide you with a comprehensive guide to the communications facilities of the currently available versions of the AIX Operating System, whether AIX/RT or AIX PS/2. Where information about communications facilities of AIX/370 has been available it's included, but only very limited testing has been done.

Throughout the publication we refer to the IBM system units 6150 and 6151 with the generic name ***IBM RT***. Unless otherwise specifically stated, the term "IBM RT" does not imply any specific model.

Some machine numbers used in this publication will reflect an entire family of machines. One example is 3x74, which means that an IBM 3174 or IBM 3274 may be used without any difference in function.

Whenever the IBM RT X.25 support is mentioned in this publication we refer to the program *IBM 6150 X.25 Communications Support* (07F3233), which is not available in all countries.

Audience

This publication is intended for IBM system engineers and for the communications specialists of authorized dealers and customers. It is assumed that the reader has a good understanding of the AIX Operating System and knows how to enter commands and install products from distribution diskettes. A working knowledge of the IBM RT and IBM Personal System/2 hardware is also required.

Structure

The publication is divided into the following major sections:

1. General Introduction
2. AIX Base Operating System
3. Asynchronous Communications
4. Basic Networking Utilities (BNU)
5. IBM RT SNA Services
6. Distributed Services (DS)
7. APPC/LU 6.2 Communication
8. 3270 Emulation and Remote Job Entry
9. 3278/79 Terminal Emulation Program
10. Workstation Host Interface Program
11. Network 3270-PLUS
12. Transmission Control Protocol/Internet Protocol (TCP/IP)
13. Network File System (NFS)
14. AIX Mail
15. AIX Access for DOS Users and AIX DOS Server
16. X-Windows in a Network Environment
17. X.25 Communications

The publication also has the following appendixes:

Contents

Figures

Tables

General Introduction

This publication is intended for IBM Systems Engineers and communications specialists of authorized dealers and customers as an aid in installation, customizing and use of selected AIX communication products.

It is ***not*** the intention that this publication replace the publications related to specific products where detailed information is given for each product but rather to give the customizing aspect of the products and thereby suggest ways in which they might be integrated in a user's environment. Where applicable, this publication will provide the communications specialist with hints for choosing between alternative options.

The tables on the following pages illustrate the richness in connectivity function currently available in the AIX communication area. Not all of these communication options are described in this publication.

Notation

Throughout this publication we attempt to use a consistent selection of typefaces:

ndtable	names of executable programs
Network File System	names of products
Using the AIX Operating System	names of publications
Automatic Call Unit	first time a term is used
This is important	emphasized text
Unknown server	error message
`/etc/qconfig`	path- and file names; program listings
`snatrace -b -l KDEFAULT`	what you type at the command line
getenv	system calls, standard C language structures

Softcopy of Programs

In the appendixes of this publication you'll find several sample programs. A softcopy of these programs is available for IBM employees from the IBM internal use repository ***RTTOOLS***. See Appendix A, "Obtaining a Software Softcopy" on page 325 for details.

Using the Connectivity Tables

The first three of the tables on the following pages describe the communication options of AIX as seen from each of the three AIX platforms: *AIX/RT*, *AIX PS/2* and *AIX/370*. The last table describes the functional capability of each communications product across all AIX platforms.

You use the first three tables to determine what physical communications method is available for communication to other systems. Pick the table that describes the connectivity options for your ***local*** system when you want to see its communication capabilities. The first column to the left shows possible target systems; pick the one you have in mind, then select the software product you want to use.

For example, suppose you want to see what means an IBM RT has for TCP/IP connection to VM:

- Pick the connectivity table for AIX/RT.
- Go to column "Target System". Search for "/370 VM".
- Go to column "AIX/RT SW PRODUCT". Search for "TCP/IP".
- Go to the communication path columns and search for "X".

You will find Token-Ring, Ethernet and X.25 as possible communication paths.

Table 1 (Page 1 of 2). Connectivity of AIX/RT. This table is intended to give you an overview of the connectivity of AIX/RT.

Target System	**AIX/RT SW Product**	**Async**	**BSC**	**SDLC**	**Token Ring**	**Ether net**	**CUT/ DFT**	**X.25**
IBM RT AIX/RT	LU6.2			X	X	X		X
	TCP/IP	X			X	X		X
	DS			X	X	X		X
	NFS				X	X		
	ATE	X						
	BNU	X			X	X		X
	X-Windows				X	X		X
PS/2 AIX PS/2	TCP/IP				X	X		X 1)
	DS 1)				X	X		
	NFS 1)				X	X		
	ATE	X						
	BNU	X			X	X		
	X-Windows				X	X		
AIX/370	TCP/IP 1)				X	X		
	NFS 1)				X	X		
PC, PS/2 PC DOS	LU6.2			X 2)	X 2)			
	TCP/IP				X	X		
	NFS					X 3)		
	DOS-Server/ AIX Access	X			X	X		
	X-Windows for DOS				X	X		
PC, PS/2 OS/2	LU6.2			X	X			X 5)
	ATE	X 4)						
AS/400	LU6.2			X	X			X
	3278/79-Emulation						CUT	
	3270-PLUS(SNA)			X	X			X
	ATE	X 6)						
/370 VM	LU6.2			X	X			X
	TCP/IP				X	X		X 9)
	WHIP						DFT 7)	
	3278/79-Emulation						CUT	
	3270-PLUS(SNA)			X	X			X
	3270-PLUS(BSC)		X				DFT 8)	
	RJE-PLUS(SNA)			X 12)	X 12)			X 12)
	RJE-PLUS(BSC)		X					
	NFS 10)				X	X		X 9)
	ATE	X						
	X-Windows 11)				X	X		X 9)

Table 1 (Page 2 of 2). Connectivity of AIX/RT. This table is intended to give you an overview of the connectivity of AIX/RT.

Target System	**AIX/RT SW Product**	**Async**	**BSC**	**SDLC**	**Token Ring**	**Ether net**	**CUT/ DFT**	**X.25**
/370 MVS	LU6.2			X	X			X
	TCP/IP 1)				X	X		X
	WHIP						DFT 7)	
	3278/79-Emulation						CUT	
	3270-PLUS(SNA)			X	X			X
	3270-PLUS(BSC)		X				DFT 8)	
	RJE-PLUS(SNA)			X	X			X
	RJE-PLUS(BSC)		X					
	NFS 1) 13)				X	X		X
	ATE	X						
	X-Windows 1) 14)				X	X		X
OEM	TCP/IP					X		
	NFS				X	X		
	ATE	X						
	BNU	X				X		
	X-Windows					X		

Note:

1. Announced but not available at date of the residency.
2. APPC/PC
3. PC-NFS from SUN Microsystems.
4. From OS/2 you can log into an AIX/RT as an ASCII terminal. OS/2 does not support login from other systems.
5. Only supported with OS/2 Extended Edition Communications Manager Version 1.2 with the X.25 Interface Coprocessor/2 Adapter.
6. ATE to the IBM 5208 ASCII-5250 Link Protocol Converter or to IBM AS/400 ASCII Workstation Controller.
7. With Advanced 3278/79 Emulation Adapter only through Non-SNA or BSC controllers. With the IBM RT S/370 Host Interface Adapter through 5088 controller.
8. Only through Non-SNA or BSC controller.
9. Support for 9370 X.25 Communication Subsystem with TCP/IP for VM Release 1.2.
10. The NFS Feature of TCP/IP for VM Release 1.2 provides file server functions for systems that have the NFS 3.2 client function installed. The VM NFS Feature does not include the NFS client function.
11. TCP/IP for VM Release 1.2 provides X-Windows client function for X-Windows System Version X.11.
12. Only supported with RSCS Version 2 Release 3.
13. The NFS Feature of MVS provides file server functions for client systems that have the NFS 3.2 client function installed. The MVS NFS Feature does not include the NFS client function.
14. TCP/IP for MVS provides X-Windows client function for X-Windows System Version X.11.

Table 2 (Page 1 of 2). Connectivity of AIX PS/2. This table is intended to give you an overview of the connectivity of AIX PS/2.

Target System	AIX PS/2 SW Product	Async	Token Ring	Ether net	CUT/ DFT	X.25 1)
PS/2 AIX PS/2	TCP/IP		X	X		
	DS 1)		X	X		
	TCF 1)		X	X		
	NFS 1)		X	X		
	ATE	X				
	BNU	X	X	X		
	X-Windows		X	X		
IBM RT AIX/RT	TCP/IP		X	X		
	DS 2)		X	X		
	NFS		X	X		
	ATE	X				
	BNU	X	X	X		
	X-Windows		X	X		
AIX/370	TCP/IP 1)		X	X		
	TCF 1)		X	X		
	NFS 1)		X	X		
PC, PS/2 PC DOS	TCP/IP		X	X		
	NFS			X 3)		
	DOS-Server/ AIX Access	X	X	X		
	X-Windows for DOS		X	X		
PC, PS/2 OS/2	ATE	X 3)				
AS/400	ATE	X 5)				
	DOS Merge with 6) 3270 Emulation				CUT	
/370 VM	TCP/IP		X	X		
	WHIP				DFT 7)	
	NFS 8)		X	X		
	ATE	X				
	X-Windows 9)		X	X		
	DOS Merge with 6) 3270 Emulation				CUT	
/370 MVS	TCP/IP 1)		X	X		
	WHIP				DFT 7)	
	NFS 1) 10)		X	X		
	ATE	X				
	X-Windows 1) 11)		X	X		
	DOS Merge with 6) 3270 Emulation				CUT	

Table 2 (Page 2 of 2). Connectivity of AIX PS/2. This table is intended to give you an overview of the connectivity of AIX PS/2.

Target System	**AIX PS/2 SW Product**	**Async**	**Token Ring**	**Ether net**	**CUT/ DFT**	**X.25 1)**
OEM	TCP/IP			X		
	NFS		X	X		
	ATE	X				
	BNU	X		X		
	X-Windows			X		

Note:

1. Announced but not available at date of the residency.
2. Requires DS version 1.3 on the IBM RT.
3. PC-NFS from SUN Microsystems.
4. From OS/2 you can log into the AIX PS/2 as an ASCII terminal. OS/2 does not support a login from other systems.
5. ATE to the IBM 5208 ASCII-5250 Link Protocol Converter.
6. DOS Merge with the IBM PC 3270 Emulation Program, Entry Level installed. It simulates an IBM 3278 Model 2 Display Station or an IBM 3279 Model 2A or S2A Color Display Station. File transfer is supported for VM/CMS and MVS/TSO. NLS is supported.
7. With 3270 Connection Adapter only through Non-SNA or BSC controllers.
8. The NFS Feature of TCP/IP for VM Release 1.2 provides file server functions for client systems that have the NFS 3.2 client function installed. The VM NFS Feature does not include the NFS client function.
9. TCP/IP for VM Release 1.2 provides X-Windows client function for X-Windows System Version X.11.
10. The NFS Feature of MVS can provides file server functions for client systems that have the NFS 3.2 client function installed. The MVS NFS Feature does not include the NFS client function.
11. The TCP/IP for MVS provides X-Windows client function for X-Windows System Version X.11.

Table 3. Connectivity of AIX/370. This table is intended to give you an overview of the connectivity of AIX/370.

Target System	AIX/370 SW	CTC	VCTC	RSCS	Token Ring	Ether net
AIX/370	TCP/IP 1)	X	X		X	X
	TCF 1)	X	X		X	X
	NFS 1)				X	X
	UVCP/VUCP 1)			X 2)		
	SENDMAIL			X 2)		
PS/2	TCP/IP				X	X
AIX PS/2	TCF 1)				X	X
	NFS 1)				X	X
IBM RT	TCP/IP				X	X
AIX/RT	NFS				X	X
PC, PS/2	TCP/IP				X	X
PC DOS	NFS					X 3)
/370	TCP/IP				X	X
VM	NFS 4)				X	X
	UVCP/VUCP			X 2)		
	SENDMAIL			X 2)		
/370	TCP/IP 1)				X	X
MVS	NFS 1) 5)				X	X
	UVCP/VUCP			X 2)		
OEM	TCP/IP					X
	NFS					X

Note:

1. Announced but not available at date of the residency.
2. Files are transferred using the Network Job Entry facility of the Remote Spooling Communication Subsystem (RSCS). The AIX/370 user may send files to and receive files from a VM/CMS user, an MVS/TSO user or another AIX/370 user in a network. Files from AIX/370 users can be transferred, using the command *uvcp*, directly to the local CMS users without the need for RSCS. Local/remote file transfer is implemented using the NETDATA protocol supported by both VM/CMS and MVS/TSO. UVCP means UNIX to VM copy and VUCP means VM to UNIX copy.
3. PC-NFS from SUN Microsystems.
4. The NFS Feature of TCP/IP for VM Release 1.2 provides file server functions for client systems that have the NFS 3.2 client function installed. The VM NFS Feature does not include the NFS client function.
5. The NFS Feature of MVS provides file server functions for client systems that have the NFS 3.2 client function installed. The MVS NFS Feature does not include the NFS client function.

Table 4. Functions supported by the AIX Systems

AIX SW Product	**Distributed Computing**	**File Server**	**File Transfer**	**Mail**	**Multitasking Windowing**	**Print Server**	**Programming Interface**	**Remote Execution**	**Remote Login**	**Remote Presentation**	**RJE Workstation Emulation**	**Single System Image**	**Transaction Processing**	**Terminal Emulation**
ATE			X						X					X
BNU			X	X				X	X					
DOS Merge with 3270 Emulation			X						X					X
DOS-Server /AIX Access		X				X			X					X
DS		X				X						X		
JSB Multiview					X									
LU6.2 1)			X				X						X	
NFS		X					X	X				X		
RJE-PLUS(BSC)			X								X			
RJE-PLUS(SNA)			X								X			
TCF	X	X				X		X				X		
TCP/IP			X	X		X	X	X	X					X
WHIP			X				X		X					X
X-Windows							X			X				
3270-PLUS(BSC)			X				X		X					X
3270-PLUS(SNA)			X				X		X					X
3278/79-Emulation			X						X					X

Note:

1. LU 6.2 provides no applications for file transfer, but you can use the LU 6.2 programming interface and make transaction programs that provide this function. Appendix B, "LU 6.2 Sample Programs" on page 327 lists sample programs for file transfer between the IBM RT and several other systems.

AIX Base Operating System

Although the topic of this publication is AIX communications, we have found it appropriate to include this initial chapter about the AIX Base Operating System. We will introduce you to what's new in AIX/RT Version 2.2.1 and we'll give you a few hints about using AIX that may be helpful for you as a communications specialist or systems administrator.

News in AIX/RT Version 2.2.1

AIX/RT Version 2.2.1 is a maintenance release of AIX and should be installed by all users of AIX/RT. Despite being a maintenance release it includes several new or considerably enhanced capabilities, some of which are separately priced products. The most important changes are:

- ***AIX Network File System*** is new in AIX/RT Version 2.2.1.
- ***AIX DOS Server*** is new in AIX/RT Version 2.2.1.
- ***C2 Security*** is new in AIX/RT Version 2.2.1.
- ***TCP/IP*** is completely changed in AIX/RT Version 2.2.1.
- ***AIX X-Windows*** is completely rewritten for AIX/RT Version 2.2.1.

In addition to the major changes listed above, there are several functional enhancements, the most significant of which we shall cover below.

The file /README supplied with AIX/RT Version 2.2.1 gives an overview of the changes and also provides hints about the use of some of the new or changed functions.

AIX/RT Network File System

AIX Network File System (NFS) is an implementation of SUN Microsystem's NFS Version 3.2. NFS permits a user on a client machine to access data from server machine(s) transparently, as if the remote files were stored at the local machine. Mount commands can be issued at system start-up time or directly by users and associate a local directory with a directory or file system on a remote host.

Based upon TCP/IP, Network File System complements Distributed Services by allowing file sharing in multi-vendor environments where all systems support TCP/IP and NFS. In networks of IBM RTs where more than one network is used, Network File System allows file sharing across network boundaries.

The initial version of Network File System on AIX/RT does not support file- or record locking.

Network File System is covered in detail in the chapter "Network File System (NFS)" on page 247.

AIX/RT DOS Server

Earlier available as a Program RPQ, *AIX DOS Server* is now included in AIX/RT Version 2.2.1 Base Operating System as an installable option.

AIX DOS Server provides IBM Personal System/2 and PC systems using IBM Disk Operating System (PC-DOS) and the *AIX Access for DOS Users* product access to the AIX system running *DOS Server*. The user of the PC-DOS system can:

- Use the file system of one or more AIX hosts running *DOS Server* as a logical extension of the local PC-DOS file system.
- Use the printers connected to AIX hosts running *DOS Server*.
- Connect into an AIX system running *DOS Server* with the PC-DOS system appearing to AIX as a DEC VT100 terminal.

Refer to "AIX Access for DOS Users and AIX DOS Server" on page 291 for a description of *AIX DOS Server* and *AIX Access for DOS Users*.

C2 Security

New and enhanced security features are added in AIX/RT 2.2.1 for compliance with the US Department of Defense requirements. The new features allow the system administrator to configure the AIX Base Operating System and TCP/IP for C2 Security.

The security features have induced several changes to AIX, the most apparent being the new directory `/etc/security` holding several files used to enforce C2 security. One of the files in the new directory is `/etc/security/passwd` where user passwords are now kept.

C2 security can be configured separately for the Base Operating System and TCP/IP. See the publications *Managing the AIX Operating System* and *Using the AIX Operating System* for a description of the new security functions of AIX/RT Version 2.2.1.

TCP/IP

TCP/IP of AIX/RT Version 2.2.1 is entirely rewritten from earlier versions and is now based upon the BSD implementation using the socket interface. For the most part, the changed implementation will not impact end-users; programmers of TCP/IP based applications, the communications specialist and the system administrator must adapt to the new environment.

Some of the most significant changes are:

- TCP/IP no longer supports the ***hostname*** (flat) name server protocol. Existing nameservers must be converted to ***domain*** nameservers and hosts using nameservers must be changed accordingly.

- Additional gateway support for the ***EGP***, ***HELLO*** and ***RIP*** protocols through the ***gated*** daemon. The ***GGP*** protocol is no longer supported.

- Only a subset of the earlier system call API is now supported and the libraries `/usr/lib/lininternet.a`, `/usr/lib/libI.a` and `/usr/lib/libS.a` are no longer available. Conversion to the ***socket*** interface API is strongly recommended.

- C2 security is provided as an option when using TCP/IP.

- Support for X.25 is implemented. See "Using X.25 for TCP/IP" on page 209 and "Running TCP/IP via X.25" on page 318.
- New server commands (daemons) are added and some no longer exist. The new *inetd* (often referred to as the "super daemon") invokes other daemons as required, eliminating the need to have many daemons active at all times. This affects the customization of `/etc/rc.tcpip` where you need only start a few daemons.
- The *tnamed* and *tnd* daemons are no longer supported.
- Significantly changed commands include *netstat, ping, rsh, telnet, tftp* and *utftp*.
- New commands include *arp, hostid, ifconfig, trpt* and *securetcpip*.
- Several commands are renamed, but most of the new commands have a link to the former command name. Examples are:
  - *ftp* replaces *xftp*.
  - *rsh* replaces *remsh*.
  - *telnet* replaces *tn*.

Please refer to "Transmission Control Protocol/Internet Protocol (TCP/IP)" on page 165 and the publication *IBM RT Interface Program for use with TCP/IP* for further information.

AIX X-Windows Version 2.1

Version 2.1 of the IBM AIX X-Windows System is new with AIX/RT Version 2.2.1. The initial shipment of X-Windows with this version of AIX/RT was an implementation of X-Windows 11 at release level 2. Updates are now available to upgrade to X-Windows 11 Release 3 at no extra cost.

Only the networking aspects of X-Windows are covered in this publication (see "AIX X-Windows in a Network Environment" on page 301). Please refer to the publications *IBM AIX X-Windows Programming Guide*, *IBM AIX X-Windows User's Guide* and *IBM AIX X-Windows Programmer's Reference* for detailed information about AIX X-Windows.

Additional Printer Support

AIX/RT 2.2.1 adds support for the following printer types:

- IBM 4224-301, 302 and 3C2 wire matrix printers.
- IBM 5204 Quickwriter printer in IBM 5202 emulation mode.
- IBM 4216 Personal Pageprinter. See the `/README` file in AIX/RT 2.2.1 for information on how to define this printer to AIX/RT.

Applications

In order to utilize the new functions and improved performance of AIX/RT Version 2.2.1, changes are made to various AIX applications, some of which are:

- Distributed Services is upgraded to Version 1.2.1 for use with AIX/RT 2.2.1.
- Workstation Host Interface Program (WHIP) 1.1 is required for AIX/RT 2.2.1 and is also enhanced to provide support for the 3278/79 Emulation Adapter. See "Workstation Host Interface Program" on page 111.

- Professional CADAM 2.0.2 is required on AIX/RT 2.2.1. You may be able to run Version 2.0.1 on AIX/RT 2.2.1 if it was installed on AIX/RT 2.2.0 and AIX/RT 2.2.1 was merged into AIX/RT 2.2.0, but it is recommended that the new version of Professional CADAM is installed.

For further information on these applications and others, see the /README file of AIX/RT 2.2.1.

Installation

If you have an AIX/RT 2.2.0 system, you can install AIX/RT 2.2.1 using the ***merge*** option (after installing VRM 2.2.1). It is, however, recommended that you back up all user data and use the ***reinstall*** option. If you do want to merge AIX/RT 2.2.1 with AIX/RT 2.2.0, please refer to the file /README of AIX/RT 2.2.1. It is located on the base operating system diskettes, so you should restore this single file to your AIX/RT 2.2.0 system with the *restore* command and read it before proceeding.

System calls

The system call interface for the **vmount**, **uvmount** and **mntctl** system calls is changed. Refer to *AIX/RT Operating System Technical Reference* for information on the new interface.

Basic Network Utilities (BNU)

Support is added for running BNU using the TCP/IP protocol over a local area network. The security features of BNU are enhanced as well. See "Basic Networking Utilities (BNU)" on page 29 for more information about BNU.

Useful Hints

Some questions are asked over and over by users of AIX. The intention of this section is to answer some of those questions. In addition to the information contained herein, have a look at Appendix H, "IBM RT Hardware Installation" on page 531 where you'll find important information about hardware. Actually, you should ***not*** install communications adapters in an IBM RT or IBM Personal System/2 without consulting this appendix.

This section will not contain hints about customizing AIX communications products or features. Such hints are given in the chapter concerned with each product or feature. The topic of performance tuning is covered in Appendix I, "AIX/RT Performance Tuning" on page 535.

Automating Installation and Updating

For the system administrator responsible for applying updates to many IBM RT systems, the *updatep* and *installp* processes where diskettes must be taken to every machine on the network can be pretty tedious. Fortunately, the AIX/RT Base Operating System includes the command ***bffcreate*** that'll make things easier if you follow the guidelines below.

Preparing the Systems

You must select one system as the server machine for installation- and update files. The following steps describe what you need to do to prepare the server system and other systems.

1. On the server system, create two directories, `/usr/lpp.install` and `/usr/lpp.update`. The directories must have sufficient space for all the installation and update files you want to keep so you may want to define two ***minidisks***. Updates may require up to 10 megabytes each with an average around 3 megabytes.
2. Make sure your server has a `/tmp` directory with sufficient free space to hold the biggest installation or update file. If space is available in some other directory *bffcreate* can be told to use that directory for work files through the `-f` flag.
3. Increase the `ulimit` size for `root` on all systems. We suggest a value of or above `50000`. To change the `ulimit` value, use the `adduser` command and change the `Filesize` field for `root`.
4. Create the directories `/usr/lpp.install` and `/usr/lpp.update` on all non-server systems. If you have Distributed Services or Network File System installed no free space is required, otherwise the directories must have space enough for the largest installation or update file.

Creating Installation and Update Files

The program *bffcreate* takes a set of installation or update diskettes and creates a file in backup format. The file created can be used as input to the `installp` and `updatep` commands respectively. You run *bffcreate* on the server system.

To create a file from installation diskettes that contain only one program, use:

```
bffcreate -v
```

This will create a file named after the program on the installation diskettes in the directory `/usr/lpp.install` and will echo the file name to the console.

To create a file from installation diskettes that contain more than one program, use:

```
bffcreate -v -f /usr/lpp.install/xxxx
```

where `xxxx` is the file name you want `bffcreate` to create in the directory `/usr/lpp.install`. The file name is echoed to the console.

To create a file from update diskettes, use:

```
bffcreate -v -f
```

This will create a file named `updt.yyddd.nnn` (where `yyddd` is the year and day number, and `nnn` is a sequence number) in the directory `/usr/lpp.update` and will echo the file name to the console.

Installing and Updating

The following steps should be repeated on each system that needs to have a new program installed or an update applied, including the server.

On the Server: On the server you can issue the *installp* or *updatep* commands directly. Use the `-d` flag to select the proper file:

```
installp -d /usr/lpp.install/install.file
```

or

```
updatep -ac -d /usr/lpp.update/update.file
```

On Clients Using Distributed Services: If you have Distributed Services installed and want to install a new program, run the following sequence of commands:

```
mount -n server /usr/lpp.install /usr/lpp.install
installp -d /usr/lpp.install/install.file
unmount /usr/lpp.install
```

Similarly, to apply an update, use:

```
mount -n server /usr/lpp.update /usr/lpp.update
updatep -ac -d /usr/lpp.update/update.file
unmount /usr/lpp.update
```

On Clients Using Network File System: If you have Network File System installed and want to install a new program, run the following sequence of commands:

```
mount -n server -v nfs -o soft /usr/lpp.install /usr/lpp.install
installp -d /usr/lpp.install/install.file
unmount /usr/lpp.install
```

Similarly, to apply an update, use:

```
mount -n server -v nfs -o soft /usr/lpp.update /usr/lpp.update
updatep -ac -d /usr/lpp.update/update.file
unmount /usr/lpp.update
```

On Clients Using TCP/IP Directly: If you have neither Distributed Services nor Network File System installed but have TCP/IP active, you can install a new program by first copying the relevant installation or update file to the local directory and then proceed as for the server system. Use the TCP/IP *ftp* command to transfer the files.

To conserve space, erase the files from the non-server machine as soon as the program is installed or the update applied.

If `ftp` says the file is too large to be created on your system, edit the file `/etc/environment` and add the following line:

```
filesize=1000000
```

Then reboot the system and try again.

On Clients Using Tape Streamer: If you are not connected to the server but have a tape streamer on all machines, you can create the installation or update file directly on tape. For example:

```
bffcreate -v -f /dev/rmt0
```

To install, move the tape to the client systems and say:

```
installp -d /dev/rmt0
```

Faster Installation

To eliminate the sometimes annoying automatic reboot of AIX/RT when you install new options or apply updates that rebuild the kernel, (SNA Services, Distributed Services, SNA Services, etc.) do the following after installing the base diskettes and before *installp* (or prior to running *updatep*):

```
mv /etc/inuipl /etc/inuipl.good
cp /bin/sync /etc/inuipl
```

Now install AIX and/or apply all appropriate updates. After installing LPPs that would normally cause a system reboot, *installp* will simply exit quietly, and you can begin the next *installp* or *updatep*.

Obviously, you will need to reboot the machine before you begin to actually use any of the code installed into the kernel, but usually you'll wait until the entire AIX system is installed or all updates applied before using any of this new code.

When all code is installed and updated, perform the following steps:

```
mv /etc/inuipl.good /etc/inuipl
shutdown -rf
```

Reference Publications for This Chapter

The following publications are referenced in this chapter:

IBM RT Interface Program for use with TCP/IP, GC09-1214
Managing the AIX Operating System, SC23-2008
Using the AIX Operating System, SC23-2007
AIX Operating System Technical Reference System Calls and Extensions, SC23-2125

Asynchronous Communication

Asynchronous Terminal Emulation (ATE)

Overview

Asynchronous Terminal Emulation (ATE) allows an IBM RT user to log in to a remote system and use the facilities of that system from the IBM RT as though the user was working from a terminal directly connected to the remote system. ATE can use RS232C or RS422A connections either locally or via modem to a remote site. Local RS232C connections allow a maximum local distance of 15 meters (50 feet) between machines, whereas RS422A allows up to 1200 meters (4000 feet).

ATE allows an AIX system to appear to another system as either a VT100 terminal, or as the display currently being used. With ATE you could connect:

- To another AIX system
- To a System/36, System/38 or AS/400. The AIX system using ATE could be connected to an IBM 5208 ASCII-5250 Link Protocol Converter that is attached to a S/36, S/38 or AS/400. The IBM 5208 allows ASCII displays, PCs or RT PCs emulating ASCII displays and ASCII printers to attach to an IBM System/36, IBM System/38 or AS/400. Each twinaxially attached 5208 may connect up to seven ASCII devices. These are attached locally via RS422A or locally and/or remote via RS232C connection.
- To the IBM AS/400 ASCII Workstation Controller.
- To a S/370 HOST via the IBM 7170 Protocol converter.
- To a S/370 HOST via the IBM 3174 Cluster Controller with the Asynchronous Communications Adapter installed.
- To an IBM 9370 ASCII Communication Adapter.
- To a non-IBM ASCII host.

Hardware and Software Prerequisites

AIX/RT The Asynchronous Terminal Emulation program is part of the AIX/RT Operating System. The only hardware prerequisite is a free serial port. You may use the built-in serial port on the IBM RT floor-standing models, a 4-port RS232C or RS422A adapter card, an 8-port RS232C or RS422A adapter card or the PC/AT serial/parallel adapter card.

AIX PS/2 The Asynchronous Terminal Emulation program is part of the AIX PS/2 Operating System. The only hardware prerequisite is a free serial port. You can use the built-in serial adapter in the PS/2 or the Dual Port Asynchronous Adapter/A.

Installing and Preparing for ATE

The following steps tell you how to prepare for ATE communications:

1. Install the ATE software package if you did not do so already. Use the *installp* command.
2. The adapter card needs to be installed in an appropiate slot in the system unit, unless a built-in serial port is used.
3. Plug the RS232C or RS422A cable into the adapter card or the built-in serial port.
4. Add a `ttydev` to the system. A `ttydev` is a device description for the communications port and is added by executing the *devices* command. The *devices* command must be executed with ***superuser*** authorization or from a user-ID belonging to the system group.

   Select the terminal type you want to emulate with ATE and make any necessary detailed adjustments for the environment. The most common changes are line speed, parity settings, number of bits per character and whether the line is to be driven as a remote or local line. Use `BPC 8` and `no parity` if NLS (National Language Support) is required.
5. Once the device has been defined to the remote system, the ATE emulation package must be customized.

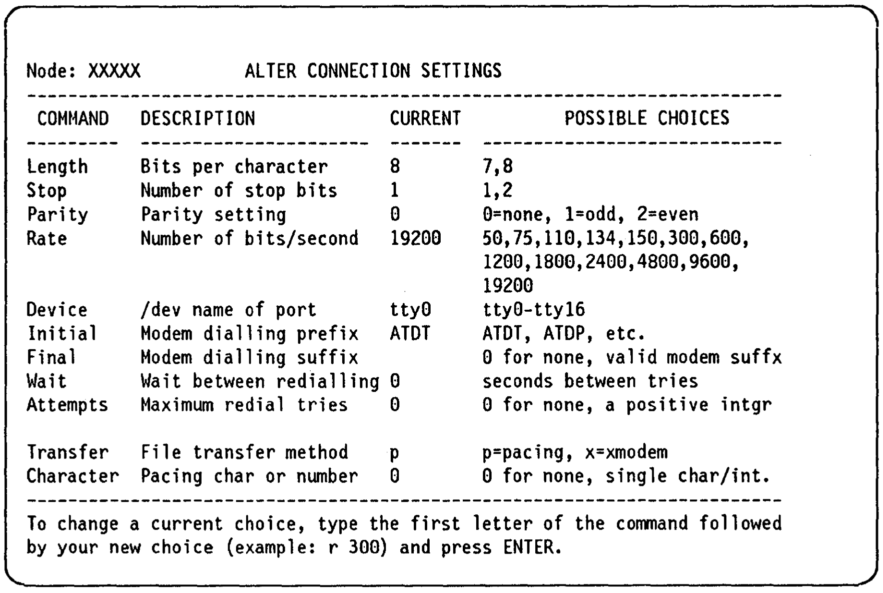
Customizing and Using ATE

To begin using or customizing ATE type the command *ate* at the AIX command line. This will start ATE and display the ***Unconnected Main Menu*** shown in Figure 1. As the name implies, this menu is displayed when the AIX system does not have a connection to the host computer. From the menu you can establish a connection to a remote host using either the `connect` or `directory` options. For local communication, select the `connect` option. The `directory` option is a listing of phone numbers to remote hosts that allow autodialing. This listing is stored in the file `/usr/lib/dir`.

```
$ ate

Node: host1          UNCONNECTED MAIN MENU
-------------------------------------------------------------------------
     COMMAND      DESCRIPTION
     -------      ----------------------------------------------
     Connect      Make a connection
     Directory    Display a dialing directory

     Help         Get help and instructions
     Modify       Modify local settings
     Alter        Alter connection settings
     Perform      Perform an Operating System command
     Quit         Quit the program
-------------------------------------------------------------------------
     The following keys may be used during a connection:
       ctrl b   start or stop recording display output
       ctrl v   display main menu to issue a command
    Use ctrl r to return to a previous screen at any time
-------------------------------------------------------------------------
Type the first letter of the command and press ENTER.
```

Figure 1. Asynchronous Terminal Emulation Unconnected Main Menu

Use the ***Alter*** command to change the default line specifications. The ***Alter Menu*** is illustrated in Figure 2 on page 19. The values displayed on the "Alter menu" must match the definitions for the port on the remote system.



```
Node: XXXXX          ALTER CONNECTION SETTINGS
--------------------------------------------------------------------------
 COMMAND   DESCRIPTION              CURRENT          POSSIBLE CHOICES
---------  ----------------------   -------   ----------------------------
Length     Bits per character       8         7,8
Stop       Number of stop bits      1         1,2
Parity     Parity setting           0         0=none, 1=odd, 2=even
Rate       Number of bits/second    19200     50,75,110,134,150,300,600,
                                              1200,1800,2400,4800,9600,
                                              19200
Device     /dev name of port        tty0      tty0-tty16
Initial    Modem dialling prefix    ATDT      ATDT, ATDP, etc.
Final      Modem dialling suffix              0 for none, valid modem suffx
Wait       Wait between redialling 0          seconds between tries
Attempts   Maximum redial tries     0         0 for none, a positive intgr

Transfer   File transfer method     p         p=pacing, x=xmodem
Character  Pacing char or number    0         0 for none, single char/int.
--------------------------------------------------------------------------
To change a current choice, type the first letter of the command followed
by your new choice (example: r 300) and press ENTER.
```

Figure 2. Asynchronous Terminal Emulation, Alter Menu

Press `ENTER` when all the parameters have been set to the correct values.

To establish a connection with the remote system, select the ***connect*** option from the unconnected main menu. At the prompt, type in the name of the port[1] defined in the "Alter Menu". Type the phone number of the connection for auto dialling, or the name of the port for direct connect, and press `ENTER`. To manually dial a number, just press `ENTER`. To redial the last number (0), type `r` and press `ENTER`.

If for some reason the *ate* program has been stopped abnormally, you may see the error message: "***Port is busy***". If this happens, look in the `/etc/locks` directory for a file named `ttyx`, where "x" is the number of the port. Remove this file with the command `rm /usr/locks/ttyx` before trying to connect again. Be sure to remove the correct file as you could remove one being used by another process (and another user).

A connection to the remote system should now be established. The login prompt of the remote system will be displayed ready for the user to log in. If a connection is not established, recheck that all line characteristics are defined correctly. If a connection is made but unrecognizable text is displayed, check that the line speed and parity are compatible between the ATE definitions and the remote system port definitions.

[1] This port must be disabled. It can be checked by invoking the *penable* command from an AIX shell. The port chosen must not be on the list of enabled ports. If it is, use the *pdisable* command to disable the port. For example: `pdisable tty0`

Once you are logged in, you can use all commands of the remote system as if you were logged in to that system from a directly attached terminal. Ensure that the terminal type is set correctly so that all of the terminal characteristics are correctly understood. If you're using the VT100 emulation function of ATE, the $TERM environment variable should be set to ***vt100***. If you use the ***native*** mode, the $TERM variable should be set to the type of terminal used, for example ***ibm6153*** or ***ibm3151***. To check that the terminal type is correctly set, use the *env* command to display all of the environment variables.

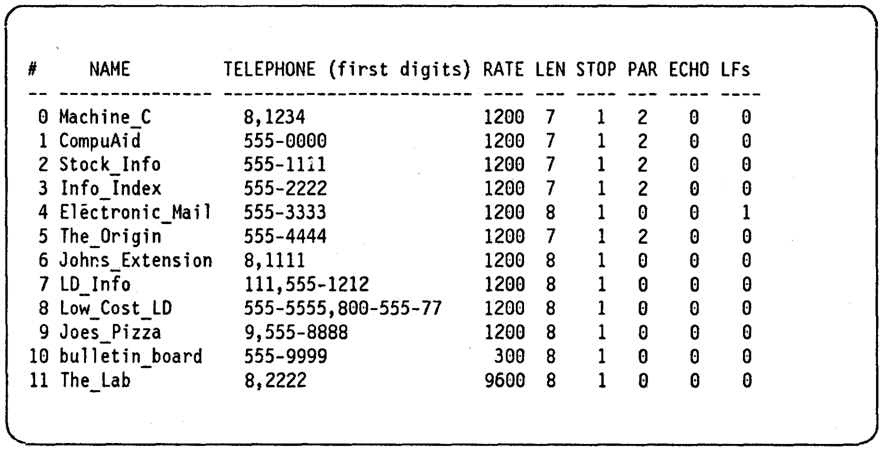
The procedures involved in setting up ATE remote communications are very similar to those for local ATE connections. The main differences are the attachment and definition for the modems. When defining the `ttydev` with the *devices* command, the following parameter must be set to the value shown for remote connections:

`dvam 1` (device attachment remote)

In addition to the adapter cards and cables necessary for local communication with ATE, modems are required for a remote connection. The primary menu (ATE Unconnected Main Menu) lets you establish a connection by using either the *connect* or *directory* command. For remote communications it is possible to use either option. *Connect* can be used to dial a remote number by directly entering the phone number of the system to call.

Dialing Directory

As an alternative, the ***Directory Menu*** can be used to connect to a remote host. The directory menu allows a user to retrieve telephone numbers of remote systems that have been previously defined to ATE. It simplifies the process of establishing remote communications to systems that are frequently used. The directory file not only contains telephone numbers but also the line characteristics. These characteristics must be correctly defined if a connection is to take place. If the remote machine is listed in the directory, select it by entering its directory reference number. The dial-up process should begin and a remote communication link will be established with the remote system. Figure 3 shows an example of a directory list.



```
#     NAME          TELEPHONE (first digits) RATE LEN STOP PAR ECHO LFs
-- --------------- ------------------------ ---- --- ---- --- ---- ----
 0 Machine_C         8,1234                  1200  7    1   2    0    0
 1 CompuAid          555-0000                1200  7    1   2    0    0
 2 Stock_Info        555-1111                1200  7    1   2    0    0
 3 Info_Index        555-2222                1200  7    1   2    0    0
 4 Electronic_Mail   555-3333                1200  8    1   0    0    1
 5 The_Origin        555-4444                1200  7    1   2    0    0
 6 Johns_Extension   8,1111                  1200  8    1   0    0    0
 7 LD_Info           111,555-1212            1200  8    1   0    0    0
 8 Low_Cost_LD       555-5555,800-555-77     1200  8    1   0    0    0
 9 Joes_Pizza        9,555-8888              1200  8    1   0    0    0
10 bulletin_board    555-9999                 300  8    1   0    0    0
11 The_Lab           8,2222                  9600  8    1   0    0    0
```

Figure 3. Asynchronous Terminal Emulation Directory List

The default dialing directory is in the file /usr/lib/dir. When you want to add new entries to the directory or change existing ones, you must edit the file and make the required changes.

ATE Default File

When Asynchronous Terminal Emulation is started it creates a default file called ate.def in the current directory. If you start ATE from a new directory each time, you can end up with a nice collection of such files, so don't. The ate.def file can be used to change the defaults for ATE, including the file name for the dialing directory. See the publication *Managing the AIX Operating System* for details.

File Transfer

Included in the ATE package are two file transfer programs. They use the ***XMODEM*** file transfer protocol and the ***PACING*** file transfer protocol.

The XMODEM file transfer protocol allows users to transfer data over the same link as used for ATE terminal emulation. The protocol is commonly implemented on many machines. This makes the XMODEM protocol particularly attractive for connections from AIX/RT and AIX PS/2 to non-IBM systems.

The XMODEM protocol is more comprehensive than many other file transfer protocols because XMODEM performs data integrity checks. No extra hardware or software is required to use XMODEM.

PACING is the alternative ATE file transfer protocol. It does not provide the same degree of checking as the XMODEM protocol does and should be used for text type files only. It should not be used to transfer program or non-text data files. The PACING protocol uses the same communications link as the terminal emulation and does not require any extra hardware or software.

Asynchronous Connection to IBM AS/400

You can connect from an AIX system with ATE to an ***IBM AS/400*** using the ***IBM 5208 ASCII-5250 Link Protocol Converter***. The IBM 5208 is an ASCII-5250 protocol converter that connects to the IBM AS/400 via twinaxial cable. It allows ASCII displays or systems emulating an ASCII display and ASCII printers to attach to the IBM AS/400. The ASCII devices are attached to the IBM 5208 locally via RS422A or locally and/or remote via RS232C connection. The IBM 5208 ASCII-5250 Link Protocol Converter supports several ASCII device types but shares only one with ATE: ***VT100***. Asynchronous Terminal Emulation (ATE) allows AIX systems to emulate a DEC VT100 terminal and connect to the 5208.

Hardware Prerequisites and Customization

Asynchronous communication can be performed using either AIX/RT or AIX PS/2. The hardware prerequisites are those listed in "Hardware and Software Prerequisites" on page 17. For details about customizing ATE, see "Customizing and Using ATE" on page 18.

Customizing the IBM 5208 ASCII-5250 Link Protocol Converter

Connect the IBM 5208 to the IBM AS/400 Workstation Controller using a twinaxial cable. You can customize the IBM 5208 with an ASCII terminal connected to one port of the IBM 5208. You can also use an AIX system with ATE connected to the IBM 5208 for customization. The IBM 5208 requires the following parameters for first time customization:

- Baud rates between 300 and 9600 bits per second.
- One of the following data bit and parity combinations:
  - 8 bit and NO parity
  - 7 bit and EVEN parity
  - 7 bit and ODD parity
  - 7 bit and SPACE parity
  - 7 bit and MARK parity
- You must use the RS232C interface with a direct (modem) cable in which all leads are going straight through from one connector to the other.

To configure the IBM 5208 press the Esc key then the Ctrl-F keys to get to the main menu of the IBM 5208. Press 1 for port configuration. The IBM 5208 gives you a choice of various ASCII terminals. Configure the port you want to use for the AIX connection as a DEC VT100 terminal. For our tests we selected the following parameters:

```
          Type  Profile  Baud   Parity  Data  Stop  Attachment
                         Rate           Bits  Bits
Portl:    DSP   VT100    19.2   NONE    8     1     DIRECT
```

Figure 4. Port Configuration of the IBM 5208 ASCII-5250 Link Protocol Converter

The parameters used in the port configuration of the IBM 5208 match those defined for ATE and the serial port configuration of the AIX system. If necessary, change the character set to match your national language. The following languages are available:

```
01. Austria               10. Italy
02. Belgium               11. Japan
03. Denmark               12. Norway
04. French Canadian       13. Portugal
05. Finland               14. Spain
06. Fr(Qwerty)            15. Spanish Speaking
07. Fr(Azerty)            16. Sweden
08. Germany               17. United Kingdom
09. Multinational         18. United States
```

Figure 5. Device Character Set of the IBM 5208 ASCII-5250 Link Protocol Converter

Select the save option. The IBM 5208 then copies the modified configuration to permanent storage. The IBM AS/400 automatically configures the IBM 5208 and the connected terminal to the system as a IBM 5291 display device. See Figure 6 and Figure 7 on page 23 for the resulting display device description. If all parameters are set correctly, the sign-on message of the IBM AS/400 now appears on your screen.

```
                   Display Device Description - Display

Device description . . . . . . . . . :   DEVD          DSP10
Device class . . . . . . . . . . . . :   DEVCLS        *LCL
Device type  . . . . . . . . . . . . :   TYPE          5291
Device model . . . . . . . . . . . . :   MODEL         2
Port number  . . . . . . . . . . . . :   PORT          5
Switch setting . . . . . . . . . . . :   SWTSET        0
Local location address . . . . . . . :   LOCADR
Online at IPL  . . . . . . . . . . . :   ONLINE        *YES
Attached controller  . . . . . . . . :   CTL           CTL01
Keyboard language type . . . . . . . :   KBDTYPE       USB
Drop line at signoff . . . . . . . . :   DROP
Character identifier . . . . . . . . :   CHRID         *SYSVAL
Allow blinking cursor  . . . . . . . :   ALWBLN        *YES
Auxiliary devices  . . . . . . . . . :   AUXDEV
```

Figure 6. IBM AS/400 ASCII Display Device Description (1 of 2)

```
                   Display Device Description - Display

Printer  . . . . . . . . . . . . . . :   PRINTER
Print file . . . . . . . . . . . . . :   PRTFILE       QSYSPRT
  Library  . . . . . . . . . . . . . :                   *LIBL
Maximum length of request unit . . . :   MAXLENRU
Text . . . . . . . . . . . . . . . . :   TEXT          CREATED BY AUTO-
                                                       CONFIGURATION
```

Figure 7. IBM AS/400 ASCII Display Device Description (1 of 2)

Using ATE With the IBM 5208 ASCII-5250 Link Protocol Converter

You can use an AIX system with ATE as if you were connected with a DEC VT100 terminal to the 5208. Because there is a difference between the IBM Enhanced Keyboard and the DEC VT100 terminal keyboard not all keys are available on the IBM Enhanced Keyboard. Not all functions of the IBM 5291 display are supported or allowed. The following functions can be used:

FUNCTION	IBM RT KEY(S)	FUNCTION	IBM RT KEY(S)
Attn	Ctrl A or Esc A	Cmd 5	Esc 5
Backspace	Backspace	Cmd 6	Esc 6
Cancel	Esc C or Ctrl C	Cmd 7	Esc 7
Clear	Esc L or Ctrl L	Cmd 8	Esc 8
Cursor Down	Cursor Down	Cmd 9	Esc 9
Cursor Left	Cursor Left	Cmd 10	Esc 0
Cursor Right	Cursor Right	Cmd 11	Esc -
Delete	Delete	Cmd 13	Esc !
Dup	Ctrl D	Cmd 14	Esc @
Enter	Enter	Cmd 15	Esc #
Erase Input	Esc I	Cmd 16	Esc $
Error Reset	Esc R	Cmd 17	Esc %
Field +	Esc P	Cmd 18	Esc ^
Field -	Esc M	Cmd 19	Esc &
Field Advance	Tab	Cmd 10	Esc *
Field Backspace	Esc Tab	Cmd 21	Esc (
Field Exit	Line Feed	Cmd 22	Esc)
Help	Esc h or Esc ?	Cmd 23	Esc _
Hex	Esc `	Cmd 24	Esc +
Home	Esc H	Command	Esc Esc
Insert	Esc Del		
New Line	Esc Line Feed		
Print	Ctrl P	Local Functions	
Roll Down	Esc D		
Roll Up	Esc U	FUNCTION	RT PC KEY(S)
Sys Req	Esc S		
Test	Esc T or Ctrl T	Brand Select	Esc Ctrl B
Cmd 1	Esc 1	Refresh Screen	Esc Ctrl A
Cmd 2	Esc 2	Toggle Indicators	Esc Ctrl W
Cmd 3	Esc 3	Access Configuration	Esc Ctrl F
Cmd 4	Esc 4	Terminal Disconnect	Esc Ctrl R

Figure 8. 5250 Functions on the IBM Enhanced Keyboard using ATE

To use the Esc key in combination with other keys, press the Esc key, and release it; then press the second key.[2]

Asynchronous Connection to OS/2

You can connect from Operating System/2 to an AIX systems using an asynchronous connection. OS/2 does not allow login from other systems using asynchronous connections.

Hardware Prerequisites

Asynchronous communication can be performed using either AIX/RT or AIX PS/2. The hardware prerequisites are those listed in "Hardware and Software Prerequisites" on page 17.

Customizing ATE for OS/2

You need to customize the serial port of the AIX system. We used the parameters shown in Figure 9 for the AIX system. These parameters must match those defined on the OS/2 system.

[2] If your ATE process ends abnormally and you want to start the connection to the IBM AS/400 again, you can get the message: "***port /dev/tty0 is busy***", and you cannot connect to the IBM AS/400 again. In this case check if the file /etc/locks/tty0 exists. Remove this file, then try the connect again.

```
tt      Terminal Type                 vt100
ae      Automatic Enable              true
dvam    Device Attachment Method      local
bpc     Bits per Character            7
nosb    Number of Stop Bits           1
pt      Parity Type                   none
om      Operation Mode                full
ixp     Include Xon/Xoff Protocol     true[3]
rts     Receive/Transmit Speed        19200
```

Figure 9. Serial Port Definition for Asynchronous Connection to OS/2

Customizing OS/2 for ATE

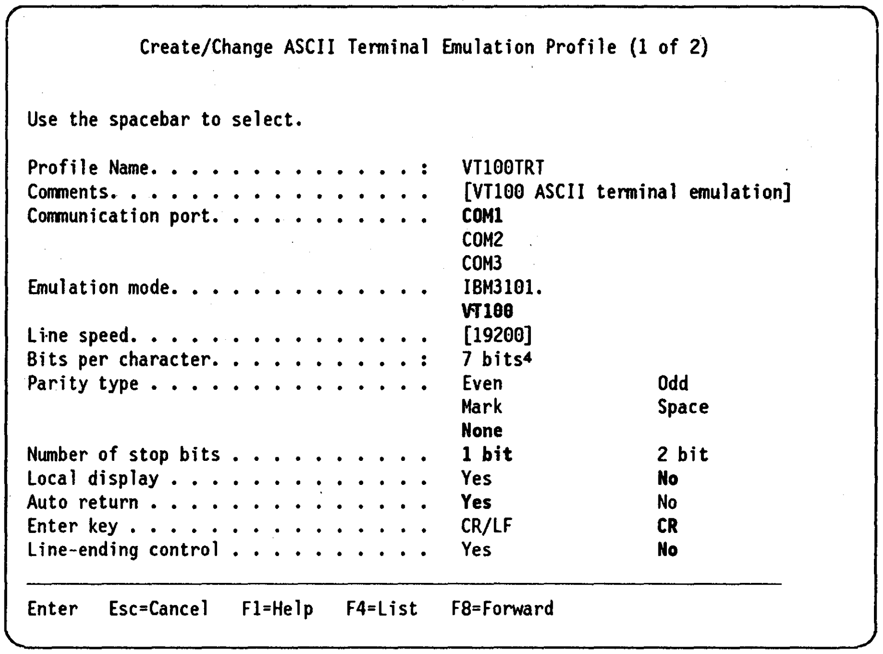
You need to tell OS/2 which device and device driver you want to use for the asynchronous connection. You do so by inserting a DEVICE statement in the C:\CONFIG.SYS file as shown in Figure 10. The device driver ASYNCDDB.SYS specified is valid for OS/2 Extended Edition Version 1.1.

```
DEVICE=C:\CMLIB\ASYNCDDB.SYS COM1
```

Figure 10. OS/2 Asynchronous Device Driver Definition

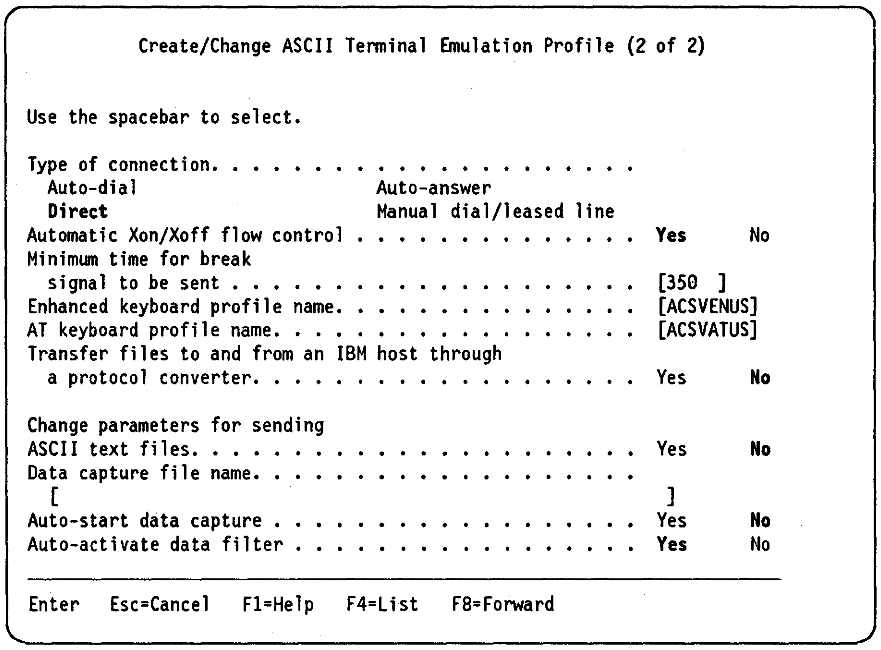
You can create your own ASCII terminal emulation profile by copying the default ***OS/2 Communications Manager*** terminal emulation profile to another name and change it. You can only use VT100 emulation from Operating System/2 to an AIX system so you are stuck with US English terminal emulation. We used the parameters shown in Figure 11 and Figure 12 to connect to the AIX system.

[3] If you are using a receive/transmit speed above 4800 bps you should use Xon/Xoff protocol, otherwise data may be lost during transfer.



```
          Create/Change ASCII Terminal Emulation Profile (1 of 2)


Use the spacebar to select.

Profile Name. . . . . . . . . . . . . :   VT100TRT
Comments. . . . . . . . . . . . . . . .   [VT100 ASCII terminal emulation]
Communication port. . . . . . . . . . .   [✓ COM1]
                                          COM2
                                          COM3
Emulation mode. . . . . . . . . . . . .   IBM3101.
                                          [✓ VT100]
Line speed. . . . . . . . . . . . . . .   [19200]
Bits per character. . . . . . . . . . :   7 bits[4]
Parity type . . . . . . . . . . . . . .   Even             Odd
                                          Mark             Space
                                          [✓ None]
Number of stop bits . . . . . . . . . .   [✓ 1 bit]        2 bit
Local display . . . . . . . . . . . . .   Yes              [✓ No]
Auto return . . . . . . . . . . . . . .   [✓ Yes]          No
Enter key . . . . . . . . . . . . . . .   CR/LF            [✓ CR]
Line-ending control . . . . . . . . . .   Yes              [✓ No]
______________________________________________________________________
Enter    Esc=Cancel    F1=Help    F4=List    F8=Forward
```

Figure 11. OS/2 ASCII Terminal Emulation Profile for ATE Connection (1 of 2). Selected options are shown in bold typeface.



```
          Create/Change ASCII Terminal Emulation Profile (2 of 2)


Use the spacebar to select.

Type of connection. . . . . . . . . . . . . . . . . . . . .
  Auto-dial                       Auto-answer
  [✓ Direct]                      Manual dial/leased line
Automatic Xon/Xoff flow control . . . . . . . . . . . . . .  [✓ Yes]     No
Minimum time for break
  signal to be sent . . . . . . . . . . . . . . . . . . . .  [350 ]
Enhanced keyboard profile name. . . . . . . . . . . . . . .  [ACSVENUS]
AT keyboard profile name. . . . . . . . . . . . . . . . . .  [ACSVATUS]
Transfer files to and from an IBM host through
  a protocol converter. . . . . . . . . . . . . . . . . . .  Yes         [✓ No]

Change parameters for sending
ASCII text files. . . . . . . . . . . . . . . . . . . . . .  Yes         [✓ No]
Data capture file name. . . . . . . . . . . . . . . . . . .
  [                                                            ]
Auto-start data capture . . . . . . . . . . . . . . . . . .  Yes         [✓ No]
Auto-activate data filter . . . . . . . . . . . . . . . . .  [✓ Yes]     No
______________________________________________________________________
Enter    Esc=Cancel    F1=Help    F4=List    F8=Forward
```

Figure 12. OS/2 ASCII Terminal Emulation Profile for ATE Connection (2 of 2). Selected options are shown in bold typeface.

After you changed the OS/2 Communications Manager Profile you must select the `Verify` option to verify the profile. Then stop the OS/2 Communications Manager and restart it.

Using the ASCII Terminal Emulation of OS/2 with the AIX systems

To connect from OS/2 to an AIX system, start ASCII terminal emulation from the OS/2 Communications Manager `Start Communications` menu. After a short time you'll be prompted for which profile you want to use. Type in the name of the profile you created or use the F4 key to list existing profiles, then choose the profile you want to use.

Issue the *penable tty#* command on the AIX systems you are using, where # is the number of the port with the connection to OS/2. Now the login prompt of the AIX system should appear on the OS/2 screen, and you can log in to the AIX system.

Note: When using AIX/RT after a succesful login you may see garbled data on your screen, and interaction with AIX/RT is no longer possible. This is because AIX/RT switches back to 8 bit characters after login with a valid userid. To prevent this add the following lines to the file `/etc/profile`:

```
if [ "$TERM" = "vt100" ]
then
   stty cs7
else
   stty cs8 -istrip
fi
```

Reference Publications for This Chapter

The ATE package is described in:

Using the AIX Operating System (AIX/RT), SC23-2007
Using the AIX Operating System (AIX PS/2), SC23-2024
Managing the AIX Operating System (AIX/RT), SC23-2008
Managing the AIX Operating System (AIX PS/2), SC23-2031

The IBM 5208 ASCII-5250 Link Protocol Converter is described in:

THE IBM 5208 Model 1 ASCII-5250 Link Protocol Converter User's Guide, XXXX-XXXX

The OS/2 Communications Manager is described in:

Operating System/2 OS/2 Communications Manager XXXYYYZZZVVV, xxxx-xxxx

[4] When using the VT100 emulation with OS/2 you can only use 7 bits per character.

Basic Networking Utilities (BNU)

The term ***BNU*** refers to a package of commands, directories and data files for communication among AIX and other UNIX-based systems over dial-up or direct lines or via local area networks using the TCP/IP protocol. BNU is typically used for AIX and/or UNIX connections that are long distance and relatively slow speed, using BNU on the AIX systems and BNU or ***UUCP*** on the UNIX systems.

Overview

In prior releases of AIX/RT, BNU was referred to as "UUCP". BNU is based on ***HoneyDanBer*** UUCP from UNIX System V Release 3. The first UUCP was made by the AT&T Bell Laboratories in 1976. It was improved in several later versions for UNIX System V Release 1 (SVR1) and System V Release 2 (SVR2).

An independent version of UUCP was developed at the University of California at Berkeley and became the basis for the versions shipped with Berkeley Software Distribution (BSD 4.x) as well as with DEC's Ultrix and Sun's SunOS operating systems. Sun actually does not ship the BSD implementation of UUCP, but an older version of its own. There are minor differences in particular manufacturers' implementations of UUCP.

With System V Release 3, AT&T began distributing a new version of UUCP that had been developed in 1983 by Peter ***Honey***man, ***D***avid ***A. N***owitz and ***B***rian ***E. R***edman. This version became popularly known as the ***HoneyDanBer*** UUCP (derived from the authors' names), but is called "Basic Networking Utilities" or BNU, in AT&T's official release of the product. BSD 4.3's UUCP is yet another significant update, incorporating some BNU features, but retaining more continuity with other Version 2 implementations.

BNU is largely backward-compatible with Version 2, so a UUCP network can contain both Version 2 and BNU sites. However, the names of the various configuration and control files have been changed. BNU systems will have the file: `/usr/adm/uucp/Systems`, older versions will not. BNU provides a conversion program *Cvt* to move UUCP command- and data files into the appropriate BNU directories. The BNU of the AIX systems is based upon BNU from UNIX System V Release 3 from AT&T.

UUCP stands for ***U***NIX-to-***U***NIX ***CoP***y. UUCP (as BNU) is a collection of programs designed so that UNIX systems can communicate with each other. BNU and UUCP include programs to:

- Transfer files between UNIX systems (*uucp*)
- Execute commands on a serving system (*uux*)
- Send mail to users on a serving system (*mail*).

When two hosts communicate using BNU or UUCP, one system initiates a call for a service (file transfer or remote execution) and the other system services this request. If configured for both functions, each of the two systems can initiate a call and serve a request. In the remainder of this chapter, we will use the term ***calling system*** to refer to a system that initiates a call, and ***serving system*** to refer to a system that serves a request.

Also, we will use the terms ***modem***, ***ACU*** (automatic call unit) and ***dialer***. A ***modem*** is a modulator-demodulator device for converting a string of bits on a serial line into electric signals for transmission across telephone and leased lines. Early modems were designed to be used in conjunction with an ordinary telephone. The call was dialed by hand to establish the connection. Another device referred to as an ***automatic call unit*** or programmable ***dialer*** is used to actually place the call automatically. A single dialer could service an entire bank of modems. The modern modems, so called "smart modems" have built-in autodial capabilities using software commands for dialing informations.

A program called *uucico* does most of the work transferring files or remote execution requests back and forth between systems. Another program called *uuxqt* is invoked on the serving system to process remote execution requests. In order for these daemons to do their job, several data files need to be in place, giving information about the systems to be called and the mechanism to be used to place the call.

The *uucp* program itself does not copy files from system to system nor does *uux* actually execute commands on a serving system. When a user invokes *uucp* or *uux* or sends mail to a user on a serving system, two things happen:

1. A work file containing information such as the name of the source file and the destination file, *uucp* or *uux* options and the type of request (send, receive, or execute) is created in the directory `/usr/spool/uucp`.
2. The *uucico* program is invoked to actually make the transfer. The *uusched* program periodically scans the spool directory for work files, and calls *uucico* only when a call needs to be made to the other system. *uucico* attempts to contact other systems and execute the instructions in the work files.

Installing BNU

Use the *installp* command to install Extended Services Program with "`uucp, ct and cu Support`". This requires about 900 blocks in `/usr`. The files for using BNU are located in the following directories.

- `/usr/lib/uucp` contains BNU administrative commands and commands used internally by BNU.
- `/usr/adm/uucp` contains the various BNU data files with information used to establish connections to remote computers and control access permissions.
- `/usr/bin` contains commands such as *uucp* and *uux* that can be used by ordinary users to request UUCP services.
- `/usr/spool/cron/crontabs/uucp` is a table used by the *cron* daemon to invoke various BNU functions at specified times. The functions it invokes are the scripts `uudemon.hour`, `uudemon.cleanu`, `uudemon.poll` and `uudemon.admin` in the `/usr/adm/uucp` directory.
- `/usr/spool/uucp` is where BNU creates temporary files in various work directories and also stores user requests.
- `/usr/spool/uucppublic` is known as the PUBDIR, and is a directory that anyone can access. Sometimes it is used to deliver files to users whose directories cannot be accessed.

Hardware Prerequisites

On the IBM RT you can use one of the built-in serial ports of the floor-standing models, a 4-port RS232C or RS422A adapter card, an 8-port RS232C or RS422A adapter card or the serial/parallel adapter card for asynchronous connections. You can use the Token-Ring adapter card, the Ethernet adapter card and the X.25 adapter card for TCP/IP connections.

For the PS/2 you can use the built-in serial port, a Dual Async adapter card, or the IBM Realtime Interface Co-Processor Multiport/2 adapter card with the 8-port RS232C interface board for asynchronous connections. You can use the IBM Token-Ring Network Adapter/A or the Ungerman-Bass NICps/2 Adapter for TCP/IP connections.

Customizing BNU

Required Steps

As stated in *Managing the AIX Operating System* in the section "Setting Up Remote Communications", you customize BNU by modifying the following configuration files:

```
/usr/adm/uucp/Systems          /usr/adm/uucp/Permissions
/usr/adm/uucp/Devices          /usr/adm/uucp/Dialcodes
/usr/adm/uucp/Dialers          /usr/adm/uucp/Poll
```

In addition, you'll have to modify other files, including shell scripts, to ensure that daemons perform required functions at times that are convenient and timely.

We shall not repeat the detailed instructions for customizing these files here. What we ***will*** do is suggesting a sequence of customizing steps, explaining when a certain step is required and pinpoint a few of the areas that are not too well described in *Managing the AIX Operating System*:

We shall first list the steps to customize your systems for a pretty much standard setup. The list will be followed by comments on some of the steps:

1. Prepare the password file for BNU.
2. Establish a physical communication link between your system and each of the systems you'll use BNU for. In case of dialout links, establish the connection between each system and its modem. For TCP/IP connections make sure that TCP/IP is running between the two systems.
3. Give your system a name by which it can be identified by BNU. For TCP/IP connections you must use the hostname defined for TCP/IP.
4. Customize primary configuration files:
   a. If your system will act as a calling system, you must create entries in the `/usr/adm/uucp/Systems` file for each of the serving systems your system will be calling.
   b. You must create entries in the `/usr/adm/uucp/Devices` file for each remote serving system your host will call.

   c. For each entry in the /usr/adm/uucp/Devices file referring to an ACU (dialer) that your system does not already know about you must specify dial instructions in the /usr/adm/uucp/Dialers file.

   d. You must update the /usr/adm/uucp/Permissions file to reflect what access the local system has to remote serving systems defined in /usr/adm/uucp/Systems and what access permissions you grant to remote calling systems.

   e. You may want to define common dial codes in the file /usr/adm/uucp/Dialcodes; however, specifying full phone numbers in /usr/adm/uucp/Systems is usually more efficient.

   f. You may want to check that the poll times given in the /usr/adm/uucp/Poll file are suitable and, if not, modify them.

5. You must uncomment the *crontab* entries in the file /usr/spool/cron/crontabs/uucp to activate the periodic scheduling of required BNU daemons.

6. Run *uucheck* to verify that your Permissions file is correct. If errors are reported, correct them. Rerun *uucheck* every time you change the Permissions file.

7. Run *uuname* to verify that all the systems your local system needs access to are known to your local system. If they are not, change the configuration files as required.

8. If you are using TCP/IP for BNU, uncomment the line in the file /etc/rc.tcpip that starts the *uucpd* daemon.

9. Shutdown your system and reboot.

10. After the system has been rebooted, verify the correct operation of each host system by using the *uutry* command (/usr/adm/uucp/uutry) or the *cu* command (/usr/bin/cu).

Special Considerations

Preparing the Password File

The publication *Managing the AIX Operating System* gives detailed information about what's required of the password file for BNU to run properly. In fact, the /etc/passwd files of AIX PS/2 and AIX/RT are both prepared for BNU in that they provide entries for a "model" user ID, ***(uucp)***. It's not recommended to use this model directly; you should create your own ID. By convention, the first such ID you create is called ***nuucp***.

If you need to assign varying access priveledges to different calling systems, you will need to create additional login IDs. The example in Figure 13 shows the "model" uucp UID and two working login UIDs: nuucp and xuucp. The working UIDs use there own directories and (like uucp) may have any UID and GID.

```
uucp:!:5:5:/usr/spool/uucppublic:/usr/lib/uucp/uucico
nuucp:!:209:5:/u/nuucp:/usr/lib/uucp/uucico
xuucp:!:210:5:/u/xuucp:/usr/lib/uucp/uucico
```

Figure 13. BNU User-IDs in /etc/passwd

On the IBM RT with the new security features of AIX/RT, you need to enable login for the UIDs used by BNU. The model entry for ***uucp*** in the /etc/security/passwd file ***does not allow login***. Figure 14 on page 33 shows that login is not allowed for the UID: uucp while the absence of the line: "***restrictions = nologin***" tells that login is allowed for the two working login IDs. If you do not define working login IDs, you ***must*** remove the line from the uucp user-ID.

```
uucp:
        password = 3wSjGhie6LP0Q
        restrictions = nologin

nuucp:
        password = EyIiTxjSlqb2s

xuucp:
        password = pQhqX7ioW.LWU
```

Figure 14. BNU User-IDs in /etc/security/passwd

Giving Your System a Name

Each system in BNU must have a ***node name***. For all connections other than TCP/IP, the node name is taken to be what the command *uname -n* returns. For TCP/IP, the node name is taken to be what *hostname* returns. It is strongly recommended that you assign identical names for these two. The node name returned by *uname -n* is taken from the value in /etc/master, while the hostname for use by TCP/IP is set with the hostname command.. On AIX PS/2, both names default to the ***site*** name entered during installation.

Changing the hostname on a system that is a *Yellow Pages* (YP) client, requires that you *kill* the process running the *ypbind* daemon while changing the hostname. It also requires that you change your hostname in /etc/rc.tcpip, /etc/hosts and any other files you may be using for *NFS* and *TCP/IP*. Then, you must change the hostname in the files used to build the YP data base on the YP master server, rebuild the YP data base and transmit the updated data base to all YP slave servers. Obviously, changing hostnames is not something you want to do every day.

The names of serving systems known to the calling system must be listed in /usr/adm/uucp/Systems for BNU to be able to communicate with them. Run the *hostname* command without any flags on each host to determine the name of that host. Use *uname -n* if TCP/IP is not installed.

The BNU Systems File

Notice, that you may define a system as a passive system which means that it can be called, but will never initiate calls. In this case, the system acts only as a serving system and needs no information in the /usr/adm/uucp/Systems file.

If your local system will act as calling system, the Systems file must contain a line for each serving system it will call. The format of entries in the file is shown in Figure 15. Examples of entries are shown in Figure 16.

```
    System_name    Time    Caller    Class    Phone    Login
```

Figure 15. Structure of the BNU Systems File

System_name	States the node name of the serving system. You can list the same node name more than once. Each additional entry represents an alternate communication path that BNU will use when trying to establish a connection to the serving system.
Time	Gives the time interval when the calling system can call the serving system. It consists of three subfields, the **day**, an optional **time** and an optional minimum **retry** period for calling when an attempt fails. For direct or TCP/IP connections, specify "Any" to allow calls at any time.
Caller	Specifies the type of device to be used for the call. This field is a pointer to a corresponding entry in /usr/adm/uucp/Devices. Insert the node name of the serving system for direct connections, "ACU" for dialed connections or "TCP" for TCP/IP connections.
Class	The line speed in bits per second. Insert "Any" if speed is negotiated or "-" for TCP/IP.
Phone	The dialer sequence that will be used by the dialer or ACU to call the serving system. Insert "-" for direct and TCP/IP connections.
Login	The login sequence for accessing the serving system as a BNU user. The systems login sequence is alternating "expect" and "send" strings. Strings are separated by spaces.

```
sys1 Any TCP - - in:--in: uucp word: uucppwd
sys3 Any TCP - - in:--in: uucp word: uucppwd
gatex Any gatex 9600 - "" \r\d\r\d\r in:--in: uucp word: uucppwd
```

Figure 16. Example of a BNU Systems File

When a user on your local host requests a BNU service, *uucico* is invoked. It first scans the /usr/adm/uucp/Systems file for the name of the serving system. Then it checks if it is a valid time to call. If so, it checks the caller and the class field and goes to the /usr/adm/uucp/Devices file to search for a device that matches. *uucico* then checks to see if a lock file exists for that device in /usr/spool/uucp/.Log, in which case the device is in use. If so, *uucico* checks to see if there is another device of the requested type and speed and uses it, if available. If no device is available, *uucico* returns to the /usr/adm/uucp/Systems file to see if there is another entry for the system in question. If so, the process is repeated. If not, the call is terminated. Once an available device is located, *uucico* uses the phone number for dial-up lines or calls *uucpd* for TCP/IP connections and tries to establish the connection with the information in the login field.

The BNU Devices File

The /usr/adm/uucp/Devices file contains information for direct links, automatic call units and network connections. See Figure 17 for the structure of the file and Figure 18 for an example file. Each entry must begin in column 1 of the file; otherwise, you will get the error message: "***No devices available***" when you try to use that device.

```
   Caller    Line    Line2    Class    Dialer-Token Pairs
```

Figure 17. Structure of the BNU Devices File

Caller The Caller field describes the type of link and may contain one of the following keywords:

ACU	For links made through a modem.
Direct	For a direct connection to the other system. If you use a direct connection this line must be followed by a line that specifies the serving system, which is directly connected to the calling system.
TCP	For a TCP/IP connection.
Node name	For a direct link to a particular system.

Line The device name of the port to be used for making the connection. For direct serial links and modems, this field will contain the name of the special file in /dev that corresponds to the serial port of the connection. Use "-" for TCP/IP.

Line2 This is an optional field that is used if the type field contains ACU and the ACU is an 801 type dialer, which is a device separate from the modem itself. For non-801 dialers and all other connections use "-".

Class This field contains the line speed of the device in bits per second if the keywords ACU or Direct are used. Use "Any" if the line will match any speed requested in the /usr/adm/uucp/Systems file. The entry in this field must match the class field in the /usr/adm/uucp/Systems file. Use "-" for TCP/IP.

Dialer-Token Pairs The remainder of the line contains pairs of dialer names and "tokens". Each pair represents a dialer and an argument to pass to that dialer.

```
TCP - - - TCP
Direct tty0 - 9600 direct
gatex tty0 - 9600 direct
```

Figure 18. Example of a BNU Devices File

Security in BNU, the Permissions File

Through the /usr/adm/uucp/Permissions file, BNU provides additional protection and much finer control for access of a serving system compared to prior releases of UUCP. See Figure 19 for the structure of the file and Figure 20 for an example.

```
LOGNAME=loginID   [options...]
MACHINE=serving_system:calling_system   [options...]
```

Figure 19. Structure of the BNU Permissions File

The /usr/adm/uucp/Permissions file has two types of entries:

- ***LOGNAME*** entries allow you to grant specific permissions for individual login IDs that are used when a calling system calls your local system. Only if you have entries defined in /usr/adm/uucp/Systems do you need LOGNAME entries.
- ***MACHINE*** entries allow you to specify permissions for individual serving systems that can be called from your local system. MACHINE entries give permission to user's on the calling system when they call a serving system. They do not give general permissions for that system.

```
LOGNAME=nuucp:xuucp READ=/u WRITE=/u REQUEST=yes SENDFILES=yes \
COMMANDS=all
MACHINE=sys2:sys1:gatex READ=/u/dieter WRITE=/u/dieter REQUEST=yes
```

Figure 20. Simple BNU Permissions File

Both types of entries may have option/value pairs. You can have as many of these option/value pairs as you want and can add entries for all or only some of the remote sites. For a description of the options, see the publication *Managing the AIX Operating System*.

LOGNAME	Specifies the login IDs that can be used by remote calling systems for log in to the local system when it acts as a serving system. All login IDs used by one particular remote calling system must appear in one and only one single LOGNAME entry.
MACHINE	Gives the name of the local calling system and the names of the serving systems that the local system is allowed to call. If you don't want to grant permissions to each system by name, the entry MACHINE=OTHER will assign permissions to any system not mentioned by name.

Access and Security Considerations: Giving a calling system the ability to copy files and execute commands on the serving system raises issues of system security. Fortunately, a lot of security measures are already in place. First of all, like all users, *uucico* must log in to the serving system. By assigning a password you can keep unauthorized users from logging in. Secondly, BNU's user-IDs in the /etc/passwd file do not receive a normal shell, but merely invokes another copy of *uucico*. The only work that can be done is that allowed by the slave (receiving) *uucico*. The system administrator has a few more mechanisms available to increase the level of security of the serving system:

- Creating additional /etc/passwd and /etc/security/passwd file entries to grant individual access to each calling system through different entries in the /usr/adm/uucp/Permissions file.
- Restricting local file access by calling systems, or requiring a call-back for certain system logins.
- Controlling the commands that calling systems can execute on the serving system.
- Controlling the systems that can forward files through the serving system and to which other system files can be forwarded.
- Assigning appropriate file access modes and ownerships to protect the BNU files (which contain such sensitive data as remote systems' phone numbers and login passwords) from outside users.

BNU allows you to define additional working logins for *uucico*. This allows you to define separate passwords for each calling system. You can also grant different levels of access on a system-by-system basis. Using separate working logins for each system also makes it easier to track which system has called you.

Make Sure the Links Work

The final step in installing a BNU link to another system is to test the connection to make sure that you have indeed configured it correctly. Before trying your first BNU request, you can try dialing the serving system with the *cu* command. You cannot use the *cu* command with TCP/IP connections, only with direct or dial-out lines.

Once you have successfully logged in with *cu* you are ready to test that *uucico* will be able to negotiate the chat script and successfully log in automatically. If you can't connect to the other system, BNU provides two shell scripts called ***uutry*** and ***Uutry*** for you. They will both invoke *uucico -x* for debugging the connection. The program *uutry* sends its output to the screen; so does *Uutry* but it also saves a copy in the file /tmp/system_name.

Testing with cu

When you have configured your system you should be able to call a remote system with *cu*[5] by typing:

```
cu system_name
```

cu will look up the details on how to contact the serving system just as BNU does. There must be a Direct entry in the /usr/adm/uucp/Devices file for the direct connection or an ACU entry if you want to use the *cu -l* command. You should now receive the login prompt of the remote system and be able to log in to the serving system. If you get a message like: "***No device available***" or "***Requested device unknown***", then you haven't got the /usr/adm/uucp/Devices file set up correctly.

[5] You cannot use *cu* with TCP/IP connections, only direct and dial-out lines can be used.

Testing with uutry

BNU provides a shell script called *uutry* that you can use for debugging the connection to a serving system. *uutry* invokes *uucico -x* for debugging the connection. The output will be sent directly to the screen of your system. You can also use *Uutry*, which does the same as *uutry*, but also sends the output to a file called /tmp/system_name. You normally issue the command:

/usr/adm/uucp/uutry system_name

where system_name is the name of the serving system you want to connect to. A bad connection could produce messages as shown in Figure 21:

```
mchFind called (sys2)
conn(sys2)
Device Type TCP wanted
Internal caller type TCP
tcpdial host sys2, port 540
getto ret 5
expect: (in:)
got ?
exit code 0
```

Figure 21. Using uutry: Incorrect Systems Entry

The attempt to contact sys2 failed because the entry for login to sys2 in the /usr/adm/uucp/Systems file was incorrect. After correcting the entry in the Systems file, you may receive the messages as shown in Figure 22:

```
mchFind called (sys2)
conn(sys2)
Device Type TCP wanted
Internal caller type TCP
tcpdial host sys2, port 540
getto ret 5
expect: (in:)
login:got it
sendthem (nuucp^M)
expect: (word:)
 Login incorrect.lost line errno - 0
close caller (5)
delock(-)
Call Failed: LOGIN FAILED
exit code 101
Conversation Complete: Status FAILED
```

Figure 22. Using uutry: Remote User-ID Unknown

In this case you reached sys2, but the user-ID nuucp did not exist on system sys2. After adding the user-ID nuucp to sys2 you may receive the messages as shown in Figure 23:

```
mchFind called (sys2)
conn(sys2)
Device Type TCP wanted
Internal caller type TCP
tcpdial host sys2, port 540
getto ret 5
expect: (in:)
login:got it
sendthem (nuucp^M)
expect: (word:)
 Password:got it
sendthem (nuucppwd^M)
exit code 0
```

Figure 23. Using uutry: Remote User-ID with nologin

We reached sys2, and the user ID nuucp exists, but the restrictions = nologin stanza was set in /etc/security/passwd for the User-ID so no login with user-ID nuucp is possible. After fixing this last problem we succeed (Figure 24):

```
mchFind called (sys2)
conn(sys2)
Device Type TCP wanted
Internal caller type TCP
tcpdial host sys2, port 540
getto ret 5
expect: (in:)
login:got it
sendthem (nuucp^M)
expect: (word:)
 Password:got it
sendthem (nuucppwd^M)
Login successful: System=sys2
msg-ROK
 Rmtname sys2, Role MASTER,  Ifn - 5, Loginuser - titus
rmesg - 'P' got Pgetx
wmesg 'U'e
Proto started e
*** TOP ***  -  role=MASTER, wmesg 'H'
rmesg - 'H' got HY
role=MASTER, PROCESS: msg - HY
HUP:
wmesg 'H'Y
cntrl - 0
send 00 0
,send 00 0
,exit code 0
Conversation Complete: Status SUCCEEDED
```

Figure 24. Using uutry: Successful Login

Now you can use the *uucp* command for sending a file from host sys1 to host sys2. We used uucp -r followed by uutry -r and got the result shown in Figure 25:

```
mchFind called (sys2)
conn(sys2)
Device Type TCP wanted
Internal caller type TCP
tcpdial host sys2, port 540
getto ret 5
expect: (in:)
login:got it
sendthem (nuucp^M)
expect: (word:)
 Password:got it
sendthem (nuucppwd^M)
Login successful: System=sys2
msg-ROK
 Rmtname sys2, Role MASTER,  Ifn - 5, Loginuser - titus
rmesg - 'P' got Pgetx
wmesg 'U'e
Proto started e
*** TOP ***  -  role=MASTER, Request: sys1!/u/titus/test -->
  sys2 !/u/nuucp (root)
wrktype - S
 wmesg 'S' /u/titus/test /u/nuucp root -dc D.0 644 root
rmesg - 'S' got SY
role=MASTER, PROCESS: msg - SY
SNDFILE:
-> 803 / 0.000 secs
rmesg - 'C' got CY
RQSTCMPT:
mailopt 0, statfopt 0
*** TOP *** - role=MASTER, Finished Processing file:
  /usr/spool/uucp/sys2/C.sys2N623c
wmesg 'H'
rmesg - 'H' got HY
role=MASTER, PROCESS: msg - HY
HUP:
wmesg 'H'Y
cntrl - 0
send 00 0
,send 00 0
,exit code 0
Conversation Complete: Status SUCCEEDED
```

Figure 25. Debugging File Transfer Using the uucp Command

Reference Publications for This Chapter

Basic Networking Utilities are described in:

Using the AIX Operating System (AIX/RT), SC23-2007
Using the AIX Operating System (AIX PS/2), SC23-2024
Managing the AIX Operating System (AIX/RT), SC23-2008
Managing the AIX Operating System (AIX PS/2), SC23-2031

IBM RT SNA Services

Although it is not the purpose of this document to give a comprehensive description of ***Systems Network Architecture*** (SNA), it is necessary to introduce a few concepts to understand how ***IBM RT SNA Services*** allows the IBM RT to be integrated into an SNA network.

Introduction to SNA

First, SNA is not a hardware or software product. It is an ***architecture***. This is perhaps best understood if you think of it as an overall scheme for communications systems, defining the structure of a communications network without imposing constraints on the nature of the devices of which it is composed. The structure of the network is defined and rules are laid down for its operation. However, the rules and definitions are designed for maximum flexibility so that a network may grow, and not be prevented from incorporating and taking advantage of future developments.

SNA is structured in layers to provide flexibility and minimize the extent to which users and applications need be concerned with the way in which data is transmitted over the network. The physical link between two points in the network (the wires and connections which provide a path for the passage of information) forms the bottom layer; the top layer is the application. Intermediate layers control data flow, the transmission protocol, the integrity of the data being transmitted, and so on. Figure 26 on page 42 gives a view of the SNA layers and what they mean.

Each layer communicates with adjacent layers only. The precise form of the information passed is left to the implementer of the network. From this it may be seen that it is relatively simple to enable SNA to run on different physical media such as Token Ring and X.25 Networks which need to use data link layers other than ordinary telephone lines.

As far as the implementer of SNA is concerned, he'll only need to change the part of the path control layer that interfaces to the data link layer so that it can use, say, Token-Ring LLC rather than SDLC. The other layers remain exactly the same and a terminal still communicates to the application on the host in the same way except over a Token-Ring network rather than an ordinary telephone line. Thus an SNA protocol can be used to communicate over a number of different physical media unlike other protocols such as BSC.

Similarly, it is relatively easy for a programmer to change any of the layers of SNA to extend their function without having to change any of the other layers.

SNA is essentially device-independent and imposes few limitations on physical devices. These may be attached directly to a processor channel, connected by means of telephone lines or on a Local Area Network. They may be displays, printers or another computer. The components of an SNA network are:

- Physical Units (PUs)
- Logical Units (LUs)
- Lines
- Systems Services Control Point (SSCP).

LAYERS	A N A L O G Y
APPLICATION	Data is created and/or updated. If you want to send a letter, this is where the content is created.
FUNCTION MANAGEMENT SERVICES (PRESENTATION SERVICES)	How is data formatted and presented? Does the sender and receiver use the same encoding, for example: ASCII or EBCDIC? Do sender and receiver speak the same language or must we translate?
DATA FLOW CONTROL (SESSION CONTROL)	Conversation protocols. Who is talking now? Who is talking next? We can't talk at the same time or information may be lost.
TRANSMISSION CONTROL	Coordinates the flow of messages and their acknowledgements. Specify in letter that you want a response (please return signed copy).
PATH CONTROL	What path is the information going to follow? A letter mailed from Denmark to Japan can follow different routes (via USA, the North Pole, Australia?). In SNA terms this is route control.
DATA LINK	How is the information enclosed to reach its destination safely? Paper envelope? Post card? In SNA: SDLC for telephone lines, QLLC for an X.25 network, LLC/MAC for Token Ring.
PHYSICAL	The physical transmission channel: Mailman, telephone, smoke signals? SNA does not enforce any; uses RS232, X.21, X.25, Token-Ring, ...

Figure 26. SNA Layers and Analogy

Physical Units

A ***physical unit*** (PU) is a component in the SNA network that can route information to other parts of the network. A PU must be able to implement the lower three layers of SNA. Physical units are classified into types, depending on their intelligence and capacity for routing SNA messages through the network:

Type 2 Node (PU.T2)	Cluster control unit (3x74).
Type 2.1 Node (PU.T2.1)	Cluster control unit and/or peer node. Usually a computer that can emulate a PU.T2 (for example, IBM RT, PS/2, PC, S/36, AS/400)
Type 4 Node (PU.T4).	Communications controller node (3720, 3725, 3745).
Type 5 node (PU.T5)	Host node.

Logical Units

One of the most important components of an SNA network is the ***logical unit*** (LU). To describe what a logical unit is we must first define the SNA term: ***end user***. In SNA terminology an "end user" is an end-point for the data that is sent across the network. In the case where a terminal talks to an application on a host, the terminal and the application will be SNA "end users" as the data goes no further than these on the network.

A logical unit then, is the component of the network that formats the data for the SNA "end user". For instance, at the terminal end it would format the data so that it fits on the screen (similar to the function of the *terminfo* file in AIX), and at the application end it would format the data for the application. When the two end points (LUs) have established the basis for exchange of information between them they are said to be in ***session***.

Like PUs, LUs are classified according to the type of formatting they have to do:

LU.0	Customer-defined session
LU.1	Terminal cluster (for example, Remote Job Entry (RJE) Terminal)
LU.2	3270-type display terminals
LU.3	Terminal printers (attached to 3270 controller)
LU.4	Terminal to terminal
LU.6.2	Advanced Program-to-Program Communications (APPC).

Lines

Lines are the physical media that connect the different components together in an SNA network. Examples are telephone lines, X.25 networks and Token-Ring local area networks.

Systems Services Control Point

The SSCP or ***Systems Services Control Point*** is the overall controlling function of a hierarchical SNA network. It is not used in a peer-to-peer SNA network. It controls all the sessions and information routes in the network, and all errors and problem notification are sent to it. It usually resides in the VTAM application on the S/370 host.

VTAM

To control the potentially huge network of physical and logical units that may exist in an SNA network, IBM S/370 hosts run a subsystem called ***VTAM*** (Virtual Telecommunications Access Method). This subsystem is essentially an "Operating System" for the network and controls the transmission of data between individual LUs. Extensions to VTAM provide network problem diagnosis functions.

VTAM must know the characteristics of all the devices for which it is responsible. Information is held in profiles configured into VTAM. The profiles are tables referred to as a "***VTAM listing***" or "***NCP gen***".

Node Type 2.1 (PU.T2.1)

The type 2.1 node is designed for peer-to-peer connections. In a peer-to-peer connection the nodes are not arranged in a hierarchical network and do not depend on a single controlling node. When a product implements a type 2.1 node it may also support attachment to an SNA subarea network (a hierarchical network) and in this case the product appears as a T2.0 node. Figure 27 on page 44 illustrates this.

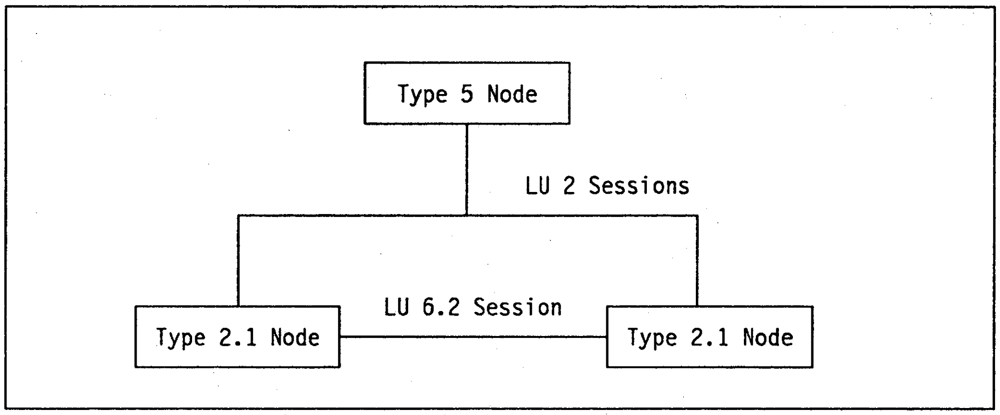


Figure 27. Type 2.1 Nodes in Hierarchical and Peer-to-Peer Network

The data link control layer in such a network can use a variety of protocols, depending on particular product implementations. On the IBM RT, for example, it may use SDLC for telephone lines, QLLC for X.25, LLC/MAC for Token-Ring, etc.

In a peer-to-peer network, one question arises: Who is in control of establishing sessions? In the hierarchical network the ***BIND*** that determines the session parameters (rules), is initiated from the ***primary logical unit*** (PLU), which is located in the type 5 node. The type 2.1 node can act as a PLU and can also act as a ***secondary logical unit*** (SLU). We say such a node is "PLU capable" as well as "SLU capable". This is illustrated in Figure 28.

Functions	NODE TYPES 2.0	2.1	4/5[6]
Parallel sessions		X	X
Able to send BIND (PLU capable)		X	X
Able to receive BIND (SLU capable)	X	X	X

Figure 28. Physical Unit Capabilities

Parallel sessions refer to the capability of the PU and LU to support multiple LU 6.2 conversations over a single LU-name/mode-name pair. An SNA ***conversation*** is the orderly exchange of data messages between two LU 6.2 LUs.

As the type 2.1 node is both PLU- and SLU capable, the only thing left is determining who sends the ***BIND*** for the sessions between the nodes. Most products implementing the type 2.1 node allows you to select between three "roles" when you define the logical link to another system: the PLU role, the SLU role and the ***Negotiable*** role. If two type 2.1 nodes are defined as negotiable, they will determine which one plays the PLU role at the time a session is estab-

[6] The markings for type 4/5 nodes are valid for a VTAM V3R2, NCP V4R3 and NCP V5R2 system. A type 5 node communicating with another type 5 node can have parallel sessions.

lished. In other words, for negotiable logical links the location of the PLU and the SLU is not predetermined.

What About BSC?

While SNA is a communications architecture defining specific protocols, ***Binary Synchronous Communication*** (BSC) is a generic term for certain kinds of data transmission. BSC is not a precisely defined protocol, and implementations of BSC tend to be device-specific. It is certainly a simpler communications method which is why it has only little of the flexibility of SNA. For instance, unlike SNA, it cannot easily be adapted to run on different physical media such as local area networks or X.25. In fact, it may only be used over telephone lines.

BSC implements a single, point-to-point synchronous communication between terminal and host. It does not have a layered structure but provides functions analogous to the physical- and data link layers of SNA. A host may support a number of simultaneous BSC communication links but, normally, terminals or other devices can't be interchanged; neither can dissimilar devices be "multi-dropped" on the same line, as they can with SNA.

Terminal emulation can overcome some of the potential problems of BSC since the "device" attached to the communications line is defined in software, thus allowing the emulation of whichever physical device is appropriate.

BSC Host Network Control Programs

In order to control a network of BSC connected devices, a host must run a network control program, just as it does for an SNA network. There are a number of different types in use. The most common ones are VTAM (Virtual Telecommunications Access Method), which can support both SNA and BSC communications, and EP (Emulation Program).

For these programs to know the characteristics of all the devices connected to the host, profiles which contain information about each of the devices are configured into the programs. These profiles are known as ***VTAM listings*** or ***EP listings***.

IBM RT SNA Services

IBM RT SNA Services is a set of programs and data files that are integrated with the AIX operating system and Virtual Resource Manager to allow an IBM RT to access an SNA network. Once implemented, SNA Services can be transparent to the user who need not be concerned about the details of its operation.

Application programs (often referred to as ***transaction programs*** in the SNA LU 6.2 environment) access the network through a device driver (`/dev/sna`) so that the network may be addressed almost like any other IBM RT I/O device. SNA Services must be told what the network looks like and what local and remote software will run on the network. This information must be provided in SNA Services Profiles. A general-purpose program (not only used with SNA) called the ***System Program Controller*** uses the information from these profiles to start a process in which to run the requested program.

Some IBM RT applications can access an SNA Network without using SNA Services. One example is the IBM RT 3278/79 terminal emulator. This program offers an LU Type 2 session with an SNA host without using SNA Services.

Installing SNA Services

IBM RT SNA Services must be installed using the *installp* command. A message warns that the kernel will be rebuilt, and the system re-started, when installation is complete. Once installed, SNA Services becomes an integral part of the AIX operating system, but must be configured before any communications can be established.

Customizing SNA Services

This section will describe in general how to configure SNA Services and especially which profiles need to be changed when configuring SNA Services for APPC/LU 6.2 communications. For a start, let's repeat:

- A ***data link***: Two network adapter cards communicating together form a data link.
- A ***session***: Two logical units (LU) communicating together form one or more sessions.
- A ***conversation***: Two application programs communicating together form a conversation.

When establishing communication between two systems, first establish the ***data link***, then the ***session*** and, finally, the ***conversation*** (see "Enabling Communication" on page 57). One or several conversations can be active at the same time over one session. Similarly, several sessions can be active over the same data link.

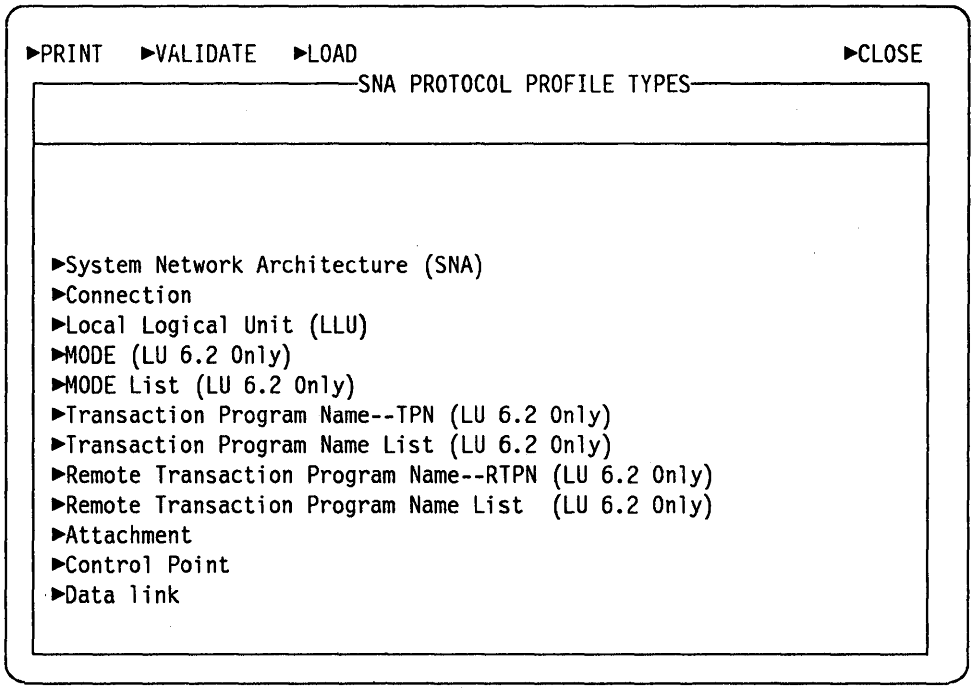
SNA Services contains a set of default profiles which can be copied and modified[7] using a full-screen program called ***snaconfig***. The *snaconfig* program must

[7] Is is *highly* recommended that SNA Services be deactivated with the command `stop -c sna` before altering the profiles. Altering the profiles without having stopped SNA Services can cause loss of information in the profiles and give you needless problems when trying to figure out what went wrong.

be used to set up the SNA Services Profiles. To start the SNA Services configurator:

1. Login as *root*
2. From the command line, type `snaconfig`
3. When the "Enter Node ID" panel appears, select Do.

This will give the panel shown in Figure 29.



```
►PRINT   ►VALIDATE   ►LOAD                                    ►CLOSE
                   SNA PROTOCOL PROFILE TYPES

 ►System Network Architecture (SNA)
 ►Connection
 ►Local Logical Unit (LLU)
 ►MODE (LU 6.2 Only)
 ►MODE List (LU 6.2 Only)
 ►Transaction Program Name--TPN (LU 6.2 Only)
 ►Transaction Program Name List (LU 6.2 Only)
 ►Remote Transaction Program Name--RTPN (LU 6.2 Only)
 ►Remote Transaction Program Name List  (LU 6.2 Only)
 ►Attachment
 ►Control Point
 ►Data link
```

Figure 29. Customizing SNA Services. The main menu of the SNA Configurator. Position the cursor on any field you don't understand, then press F12 to get help.

The *Attachment-*, *Control Point-* and *Data Link* Profiles refer to the ***physical unit*** (PU) and how it attaches to the network. The *Local LU Profile* contains information about the ***local logical unit*** (LU) on the IBM RT and the *Connection Profile* contains information about the LU on the remote system. This information is used when communication is established.

Each of these profiles will be linked to the *Attachment Profile*, so that they know through which physical unit (adapter) on the IBM RT to establish the connection. See Figure 30 on page 48.

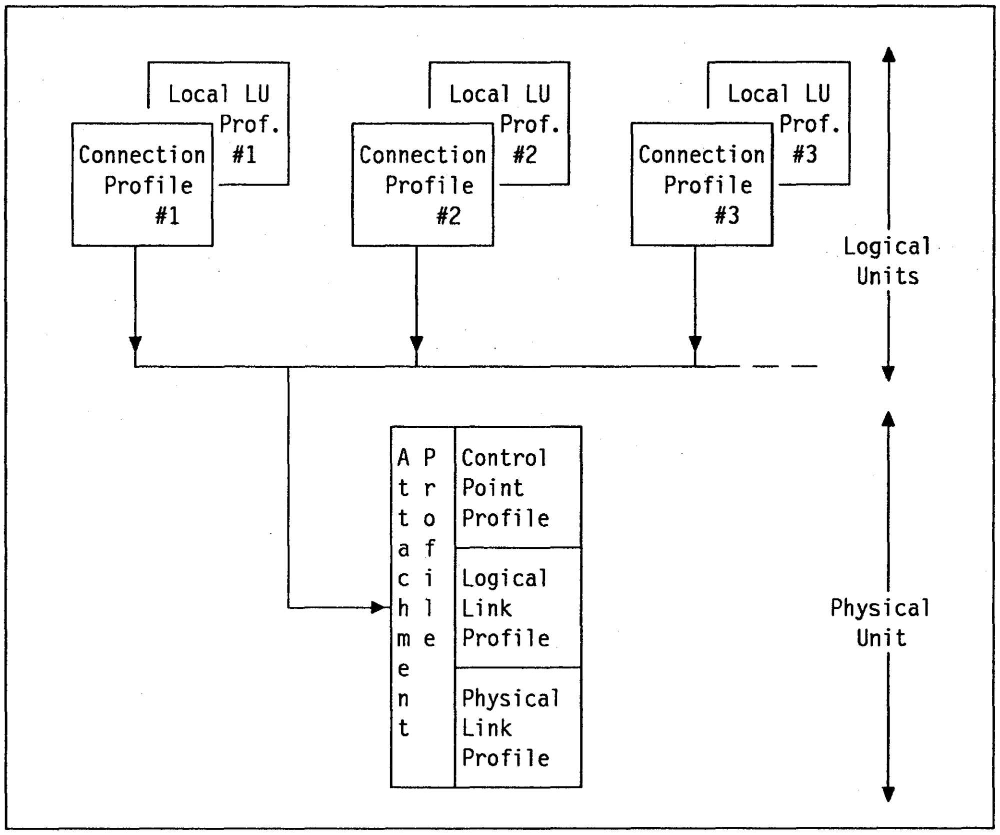


Figure 30. Relationship between SNA Services Profiles. This figure describes the relationship between the Local LU Profile, the Connection Profile and the Attachment Profile.

The following paragraphs describe each of the profiles that need to be customized. Not all parameters will be described in this chapter. Numerous samples of customizing parameters for various types of connections are listed in the appendixes of this publication. A full discussion of all the profiles is included in the *IBM RT SNA Services Guide and Reference* publication.

Physical Link Profile

This profile defines how the IBM RT physically attaches to the network. It corresponds loosely to the definition of the physical layer in the SNA seven-layer structure. Selecting `Data Link` from the main menu will bring you a menu of Logical- and Physical Link Profiles. A Physical link could, for example, be a Token-Ring physical link using the IBM RT Token-Ring Adapter or an RS232C physical link using the IBM RT Multiprotocol Adapter.

Data Link Device Name in this profile must match the data link name in the AIX/RT device configuration. For example, `trllc0` for the Token-Ring data link via the Token-Ring Adapter and `sdlcllc0` for the SNA/SDLC data link via the IBM RT Multiprotocol Adapter. In these profiles you also specify if you will be connected to a switched or non-switched line.

You need one Physical Link Profile for every data link (`trllc0`, `trllc1`, `ethllc0`, `sdlcllc0`, etc.) you use. You may have more than one Physical Link Profile for a data link but ***only one can be active at a time*** for a given data link. If you attempt to use a Physical Link Profile for a data link while that data link is already in use, your request is rejected with the message code (-727) and/or the message: "***Link is already active***".

In most cases, the Physical Link Profiles supplied with SNA Services are adequate. Use them if you can or copy suitable ones and modify them to match your setup.

Logical Link Profile

This profile defines the characteristics of the protocol used to communicate between the systems. It corresponds to the definition of the data link layer of the SNA seven-layer structure. Selecting `Data Link` from the main menu will bring you to a menu of Logical- and Physical Link Profiles. The `SDLC logical link` could, for example, be used if the physical link type is RS232C, Smart Modem or X.21. The `Token-Ring logical link` can be used if the physical link type is Token-Ring.

In the Logical Link Profile you specify if the station type is to be secondary, primary or negotiable. If the IBM RT is to be secondary, then the ***Local Secondary Station Address*** must be specified. If the IBM RT is to be primary, then a ***Remote Secondary Station Address*** must be specified in the Attachment Profile.

In most cases, and certainly for Token-Ring and Ethernet, the Logical Link Profiles supplied with SNA Services are adequate. If you can use them as they are, do so. If not, pick the ones that match your requirements closest, copy and modify them to suit your setup.

You'll need one Logical Link Profile for every link type you use: `TDEFAULT` for Token-Ring, `EDEFAULT` for Ethernet, etc.

Control Point Profile

The Control Point Profile identifies the physical unit (PU) associated with the IBM RT to the network. The `"XID Node ID"` field is relevant on dial-up (switched), X.25 and Token-Ring connections. If connected to another system via a leased (non-switched) line, the XID (Exchange Identification) is not being used. When SNA Services is used between two or more IBM RTs only, the XID is ignored. The XID value is a hexadecimal ID exchanged with the remote system when connection is established in order to avoid unauthorized access to the system. The XID is also used to exchange and negotiate the link level protocol parameters such as maximum i-field length, link transmit window, etc. The XID field is made up of two fields, concatenated together:

- ***XID block number***
- ***XID ID number***.

The XID block number makes up the first three hex digits of the parameter and defines the product type. For connections to VTAM on S/370 this corresponds to the IDBLK operand in the VTAM PU Macro.

Some system types have fixed XID block numbers. You must customize SNA Services to those XID block numbers to talk to such systems. A list of known XID block numbers follows:

017	PC 3270 Emulation Program Version 3
050	APPC/PC
056	IBM AS/400
05D	Operating System/2
05E	Workstation Program
061	Personal Communications/3270

The XID ID number (the last five hexadecimal digits) identifies the specific piece of equipment from all other similar ones on the network. For connections to VTAM this corresponds to the IDNUM operand in the VTAM PU macro.

Attachment Profile

This profile pulls together the information in the above three profiles to define the way the IBM RT is attached to the network. SNA Services uses this profile to open an attachment through the communications adapter to the network. For an SDLC link the station type entered in this profile is secondary, primary or negotiable. If ***primary*** is entered you must also insert the ***Remote Secondary Station Address***. This is used to address a specific attachment on a line, and each attachment on a multipoint line must have a unique ***station address***.

In a point-to-point environment the station address is normally set to X'C1'. X'C1' equals decimal 193, which should be entered in the profile.

For Ethernet and Token-Ring this profile must specify a ***call type*** of ***CALL*** or ***LISTEN***. Select CALL to indicate that the local station initiates a connection by calling another station. Select LISTEN to indicate that the local station does not initiate a connection but waits for a remote station to take the initiative. CALL corresponds to negotiable, LISTEN to secondary. Select ***AUTOLISTEN*** to indicate that a new "LISTEN attachment" should be started when the current "LISTEN" is satisfied.

Connection Profile

The Connection Profile describes the characteristics of a connection to a remote LU. It contains information relevant to the path control, transmission control, data flow control and presentation services layers of SNA. Together with the Local LU Profile, it is used to start a session between the local and the remote LU. Several things are specified in this profile. For example:

- The remote LU name
- The Mode List Profile name is referenced to make a link between the Connection- and the Mode Profiles
- Session concurrency is specified as ***single*** or ***parallel***
- The Remote Transaction Program List name is referenced to make sure which Remote TPN Profiles are valid for the connection.

Figure 31 on page 51 illustrates how the Connection Profile relates to the other SNA Services Profiles.

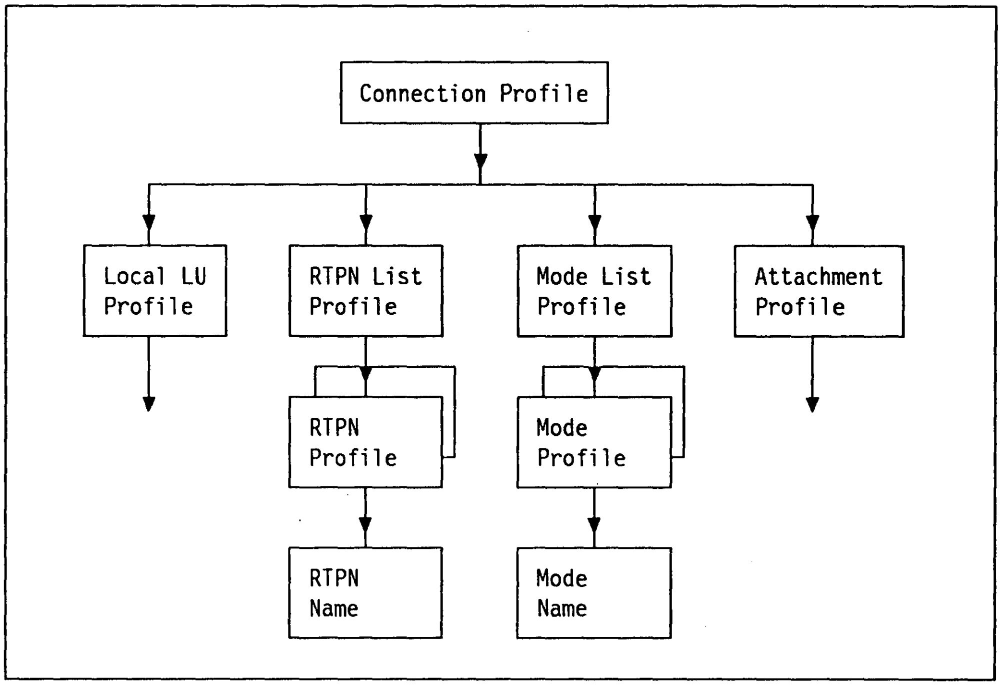


Figure 31. Relationship between the SNA Services Profiles. This figure illustrates how the Connection Profile relates to other profiles.

Transaction Program Profiles

LU 6.2 communication is based upon two programs exchanging information across a communications link. Normally, one program is started directly or indirectly by a user; this program ***initiates*** a ***conversation*** with a program at a remote system and, as one might expect, is usually referred to as the ***initiating program***, or the ***invoking transaction program***.

The program on the remote system is usually referred to as a ***remote transaction program*** or an ***invoked transaction program***. The initiating program uses an SNA mechanism called ***ATTACH*** to let the remote system know what remote transaction program to schedule. An IBM RT program must pass the name of the remote transaction program to the **snalloc** subroutine or corresponding system call.

The fact that one transaction is required on each side is reflected in the SNA Services Profiles. There is one profile set for the local transaction programs that can be scheduled from remote sites, and one set for programs that the local system can schedule at remote systems. Each consists of one or more transaction program name lists (or TPN Lists) pointing to transaction program profiles (or TPN Profiles) each describing one transaction program.

Local Transaction Program Profiles: The ***Local TPN List Profile*** points to one or more ***Local TPN Profiles*** and is itself pointed to from the Local LU Profile. The Local TPs are those programs that the local system will allow remote systems to schedule when using the LU name corresponding to the Local LU Profile pointing at the Local TPN List.

Observe that the name passed from the remote system when scheduling (attaching) a transaction program on your local system ***must match what you***

specify in tpn_name. The name can be entered in character or hexadecimal format when creating the Local TPN Profile. This name serves as a reference to whichever name your executable program happens to have.

Remote Transaction Program Profiles: The ***Remote TPN List Profile*** is pointed to from the Connection Profile and tells SNA Services what remote transaction programs may be scheduled at remote systems via a particular connection (and, hence, remote LU). The list points to one or more ***Remote TPN Profiles*** which are used to check the validity of allocation requests issued from your local system to a remote system.

Mode Profiles

The profiles describing the ***modes*** that can be used to establish sessions between local and remote LUs consist of the ***Mode List Profiles*** and the ***Mode Profiles***. The first is a list pointing to one or more of the latter.

Linked from the Connection Profile(s), the Mode Profiles describe the modes that are valid for the remote LU described in the Connection Profile(s). Remember, that a specific session is identified by the two LUs involved and the mode name they share. Consequently, several sessions may exist between two nodes if different mode names are used for each. Similarly, the same mode name may be used to establish sessions between one local LU and several remote LUs at the same time.

Local LU Profile

Not surprisingly, the Local LU Profile contains parameters that describe the characteristics of the local LU. The profile assigns a name to the local LU which may be used to establish LU-LU sessions with several remote LUs at any point in time. One LU-LU session may carry several LU 6.2 conversations.

For LU 6.2 connections you must specify the name of a Local TPN List Profile that points to Local TPN Profile(s). Only transaction programs pointed to from the Local LU Profile in this way may be scheduled from remote systems. Figure 32 on page 53 illustrates how the Local LU Profile and the local transaction profiles are related.

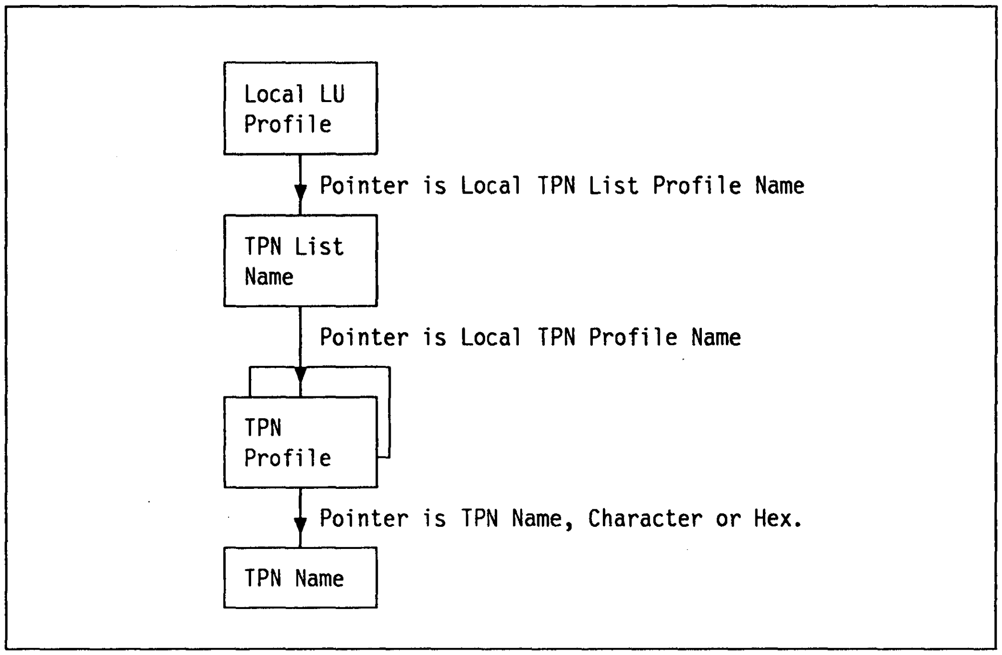

Local LU Profile
Pointer is Local TPN List Profile Name
TPN List Name
Pointer is Local TPN Profile Name
TPN Profile
Pointer is TPN Name, Character or Hex.
TPN Name

Figure 32. Relationship between the SNA Services Profiles

SNA Services Profile Hints

Configuring SNA Services is very simple and straightforward if your system has only one single SNA connection to the surrounding world. When you start configuring your system for multiple connections, though, things get more complex, often to the point where you end up with a set of profiles where ***nothing*** works; not even the things that used to work!

Very often, the reason for the problems is a lack of overview and a tendency to define more profiles than are actually required. It pays to spend a few moments away from the keyboard and do some planning ahead.

In Figure 33 on page 54 we have provided an overview of the SNA Services Profiles and the way they interact. We recommend that you make a similar drawing for your system before you start changing the profiles. Draw large boxes and insert names of profiles as you work through them manually.

You'll notice that four of the boxes in the figure are marked with the letters A through D. The following sections will address the special considerations that apply when customizing the marked profiles.

Before we do so, let's re-emphasize that SNA Services knows all the profiles, of course, but that the only profiles you can specify on the AIX *start* command are the ***attachment*** and the ***connection*** profiles. All other profiles are linked to one of these two profiles and the two are linked together.

Application programs know only connection profiles so the existence of attachment profiles is of no importance to them, as we shall discuss further in "Session Type" on page 56 below.

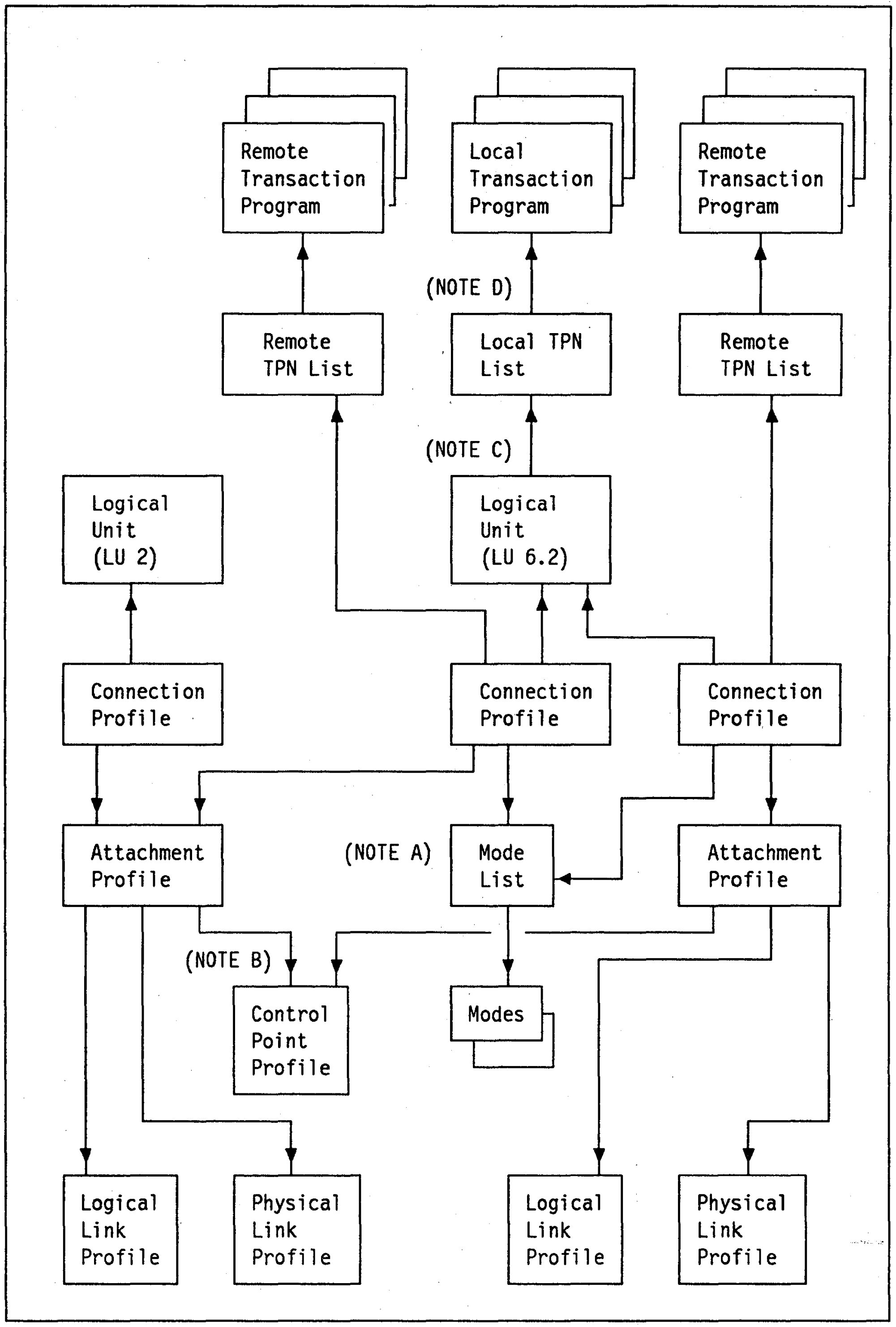


Figure 33. Overview of all SNA Services Profiles

Also, notice that the figure has three connection profiles. The leftmost one defines an LU 2 connection and looks different from the other two because it points neither to a Mode List Profile nor to a Remote TPN List.

Mode List Profile

As you'll see, Figure 33 has only one Mode List Profile (marked with ***A***). This list may point to one or more modes, but the list is shared by all LU 6.2 Connection Profiles. We suggest that you customize your system the same way.

The obvious advantage is that you have only one named set of modes to maintain across all LU 6.2 connections. Since an LU 6.2 connection is uniquely defined by the two LUs involved and the mode name they share, it is perfectly possible to have connections going to several nodes using one single mode name. Furthermore, this leads to all machines in the network using a common and limited set of mode names.

Mode names must be known by both partners for a session to be established so for every mode name you define for a given Connection Profile, the same mode name must be defined on the remote system described by the Connection Profile. So why define more than one? Actually, you probably shouldn't, except when the remote system does not support parallel sessions and you need more than one conversation active at a time between the two systems.

If the remote system supports parallel sessions, you may have more than one conversation active at the same time over a given LU-mode name pair, using parallel sessions. Normally, you should configure SNA Services to use parallel sessions for all LU 6.2 connections. Don't forget to define the maximum number of simultaneous sessions in the Mode Profile.

Control Point Profile

When two connections share a communication link, they might as well share the Control Point Profile and the profiles pointed to from lattergml. the Logical- and Physical Link Profiles. As you will see from Figure 33, our example uses only one Control Point Profile (Note ***B***). Often you need only one, since SNA Services appears as a PU 2.1 and can handle LU 1, 2, 3 and 6.2 through the same profile.

Sometimes, you do need more than one Control Point Profile; for example, when you want to talk to remote nodes with different XIDs or different network names. Thus, more than one Control Point Profile will often be required if you need to connect to more than one remote host, but you should try to keep the number of Control Point Profiles at a minimum.

If you have more than one Control Point Profile pointing to the same physical adapter through one or more Physical Link Profiles, you can't start attachments for more than one of those at any one time. If you try, your request is rejected with the error code (-727) and/or the message: "***Link is already active***".

Logical Unit Profiles

As seen from the SNA network, your LU 6.2 connections need have only one single logical unit name. It does not matter if the same LU name is used on different communication links. Figure 33 (Note **C**) points out the fact that the Logical LU Profile ***for LU 6.2*** (as opposed to the Logical LU Profile for LU 2) is pointed to by Connection Profiles for different communication links.

The sharing of one single Local LU Profile, again, can lead to drastic simplification of the total network configuration since every system is identified by one, and only one, unique logical unit name. This concept is often referred to as "CP=LU".

In Figure 33 you'll see that the Connection Profiles for LU 6.2 point to four other profiles:

- Attachment Profiles
- Mode List Profiles
- Logical Unit Profiles
- Remote Transaction Program List Profiles.

Since the two Connection Profiles point at different Attachment Profiles and, in turn, different Physical- and Logical Link Profiles, one can assume that the two connections use different communication links.

The two LU 6.2 Connection Profiles also point at different Remote TPN List Profiles. This probably means that not only do the connections use different communication links; they also go to nodes with different capabilities since they do not provide identical remote transaction programs.

To put it another way:

- Don't use identical copies of remote transaction program lists under different names. Only if there ***must*** be a difference should you use more than one remote transaction program list.

- Don't use multiple Attachment Profiles if the information they contain is identical or could be made identical.

Local TPN List Profile

In Figure 33 (Note ***D***) you'll see that there's only one Local TPN List. This would ideally be the case even if you had more than one Local LU Profile.

There's one legitimate reason for making more than one list of local transaction programs: that you want certain remote nodes to access only a limited set of the programs that can service incoming ***allocates***. If such restriction is not necessary, use only one local transaction program list.

Even when such restrictions do apply, they are more feasibly implemented through the remote transaction program lists on the remote nodes. Having only one list of local transaction programs keeps things simple for you.

Session Type

By nature, LU 6.2 invites you to define the session type as ***negotiable***. It's not easy to see, but for LAN connections this means that in the Attachment Profile you should specify the ***CALL*** option rather than the ***LISTEN*** option.

By specifying the CALL option you gain an additional benefit. Normally, you'd need to start the attachment for each of the connections you want to start. If you have, say, three Connection Profiles pointing to three different Attachment Profiles, and if the Attachment Profiles all use the same combination of Control Point Profile and Physical/Logical Link Profiles, then ***any*** of the Attachment Profiles will service ***all*** of the connections.

An application program usually starts by opening a specific connection. If the necessary control point and Logical/Physical Link Profiles are already active, then the attachment pointed to from the Connection Profile is not started. In fact, attempts to start it will be rejected by SNA Services because another Attachment Profile with identical pointers is already active and can be used.

Hence, by using the `CALL` option in the Attachment Profile, by sharing the LU name over all connections, by using only one mode list (and possibly only a single mode name) and by sharing control point and Logical/Physical Link Profiles, you can make your SNA Services Profiles simple and manageable.

Verifying SNA Services Profiles

When you have customized your SNA Services Profiles, do select the ***verify*** option of `snaconfig`. Be aware that the profiles as supplied with AIX/RT Version 2.2.1 contain an error which will prevent an error-free verification. You can ignore this error, but any change ***you*** make should verify correctly.

Enabling Communication

Once all the profiles have been defined communication can be enabled. This involves activating the communications link from both the local and the remote system. To enable communication on the IBM RT, use the following procedure:

1. Login as *root*
2. Type `start sna` to start SNA Services
3. Type `start /attachment/attachment_name` to start your attachment
4. Dial the number of the host if you are using a dial-up telephone line
5. If you are using a modem, watch the modem lights. The ***Receive*** and ***Transmit*** lights should flicker for an instant; then they should show a regular polling signal from the other system, echoed by the IBM RT.
6. Type `status -l sna`. This should show that the attachment has become active. For other than LU 6.2 connections it will also show that the connection for the attachment has become active. To start the connection with commands is normally not necessary using LU 6.2 communication, as the transaction program should activate the connection if it is inactive. If you want to start the connection manually, do so with the command:

   ```
   start /connection/connection_name
   ```

   Several Connection Profiles can be started using the same Attachment Profile. This is important to notice if you want several connections from an RT to another machine using only one attachment. For example if you want both LU 1/2/3 and LU 6.2 communication at the same time using only one attachment.

Note: These commands can be included in the `/etc/rc.include` file for automatic activation of the attachment when the machine is switched on. If Distributed Services is installed, the file `/etc/rc.ds` will have been configured to start SNA Services and at least one attachment.

Alternatively, the profiles may be activated from the ***snacontrol*** menu system. To activate the profiles this way, use the following procedure:

1. Login as *root*
2. Type `snacontrol`. This will bring up a full-screen panel where SNA, the attachments and the connections can be activated or deactivated, tested or interrogated.

The *snacontrol* facility includes some other functions:

- Display of the error log. This facility will invoke the *errpt* command and display the error log in the system. The error log is located in the directory `/usr/adm/ras`. Only the errors associated with SNA Services will be displayed.
- Sorting the display of attachments and connections. This makes the screen easier to read. You can sort the list according to status, so that all the active attachments and connections are listed at the top.
- Display status information on attachments and connections.
- Trace facility. The *snatrace* can be activated from the command line or from the *snacontrol* menu-system. The trace can give detailed information about the attachments. A command line example:

  ```
  snatrace -b -l attachment_name
  ```

  This will begin ("-b") a trace and give a long ("-l") trace listing. To end the trace:

  ```
  snatrace -e attachment_name
  ```

  The output from the trace is located in `/etc/lux/attachment_name`. This file is not in text format and must be formatted with the *trcrpt* command. For example:

  ```
  trcrpt /etc/lux/attachment_name
  ```

  This will format the trace output and display it on the screen.

Reference Publications for This Chapter

IBM RT SNA Services is described in:

IBM RT SNA Services Guide and Reference, SC23-2009

Other relevant publications are:

Systems Network Architecture, Transaction Programmer's Reference Manual For LU Type 6.2, SC30-3084
Systems Network Architecture, Architecture Logic For LU Type 6.2, SC30-3269
An Introduction to Advanced Program-to-Program Communication (APPC), GG24-1584

Distributed Services

Distributed Services (DS) is a licensed program that provides directory-, file- and resource sharing across networks of AIX systems. DS is currently available only under AIX/RT. DS has been announced for AIX PS/2 and IBM has announced its intentions to provide DS also under AIX/370.

Overview

AIX/RT Distributed Services Version 1.2.1 is the current version of DS. It is an automatic update from Version 1.2. Like earlier versions of DS it uses the SNA LU 6.2 protocol to provide its services. Though not officially supported on X.25 connections, *AIX/RT Distributed Services Version 1.2.1* will indeed run across all of the communications links supported by SNA Services.

The announced, but not yet available, *AIX/RT Distributed Services Version 1.3* will use the ***Internet Protocol*** (IP) rather than LU 6.2. DS as announced for AIX PS/2 also uses IP. Observe that because of this change of protocol, DS Version 1.3 does not talk to earlier versions of DS.

DS architecture allows a participating system to act as ***client*** and/or ***server***. A server allows other systems to access its files, and a client uses the files of other systems. *AIX/RT Distributed Services Version 1.3* will provide compatibility with Distributed Services of *AIX PS/2* and *AIX/370* and may be used either independently or in combination with the *AIX/RT Network File System*.

AIX Distributed Services is a comprehensive file-sharing system with superior facilities compared to most other systems in the marketplace. The highlights of Distributed Services are:

- Remote mount at directory or file level
- Distributed file and record locking to help preserve data integrity
- Inherited mounts
- High level of security
- Runs over Token-Ring, Ethernet and SDLC
- Remote print services
- Backup and restore across the network
- Application Programming Interface with Inter Process Communication messages over the network.
- Facilities for Single System Image (SSI)
- Application program server (code server) to simplify DS network administration and conserve disk space.

The publication, *Managing the AIX Operating System*, describes how to install, customize and use Distributed Services in detail in the chapter "Managing Distributed Services". You will not see the same information repeated here, but you will be provided with hints and shortcuts.

This chapter is a supplement to the "Managing Distributed Services" chapter and ***not*** a replacement. Its purpose is to add information that can assist you in selecting the options of Distributed Services that are of use to you and to help you customize DS.

Through the remainder of this chapter, we'll assume you have installed the Distributed Services diskettes using the `installp` command and that you've applied all updates that may be available by using the `updatep` command of AIX. Don't neglect the latter; you'll regret it long after. It is very important that all hosts in your network are at the same software level.

Hardware and Software Prerequisites

You must have one or more of the following communications adapters and the supporting software installed in each IBM RT running Distributed Services.

IBM RT Token-Ring Adapter and device driver
IBM RT Baseband Adapter (Ethernet) and device driver
IBM RT Multiprotocol Adapter (SDLC) and device driver
IBM RT IBM PC X.25 Communications Adapter and *x25c.*

For two IBM RTs to cooperate in a server/client relationship, both systems must have at least one communications facility in common as DS does ***not*** provide a gateway function in the LU 6.2 implementation.

Distributed Services Configuration Options

Distributed Services can be configured in a variety of ways, depending on your requirements. You'll have to decide how you want to use Distributed Services before you start the process of configuring it. Be aware, though, that a few restrictions apply:

- IPC semaphores and shared memory are not supported between remote systems.
- You can not share files across the bounderies of different Distributed Services networks (no gateway support).

Your Distributed Services Network can host a combination of the following configuration types:

- Full-function server/client system
- Single System Image (SSI)
- Code server
- Remote print server
- Remote backup and restore
- Node Table server.

Full-Function Server/Client System

If you customize all systems in the network for full-function server/client service, you can use all the functionality of Distributed Services:

- Transparent access to remote files and directories
- Local execution of remote applications and processes
- Centralized administration
- Queue sharing
- Data protection through SNA security mechanisms.

A full-function Distributed Services Network requires that you answer questions like:

- Which systems will be included in the Distributed Services network?
- Which systems will have access to which other systems in the network? Network access is defined by entries in a network Node Table.
- Which user can access files and applications on which other system in the network? User access is defined by entries in a User/Group Table.
- Which, if any, applications will be designed to communicate across the network using interprocess communication? This information is defined in the interprocess communications table.
- Which files and applications will be made available to which users and groups in the distributed network? Control over which files and applications that can be accessed is maintained locally and determined by standard AIX facilities.

Single System Image

Single System Image (SSI) is a simplified version of the full-function server/client configuration. SSI removes much of the hassle of setting up the profiles to ensure that access is restricted between machines. It will also take away some of the flexibility to configure cross-system access. Of course, this is because the whole idea of a Single System Image system is that everybody can access everything from any machine, subject to the standard AIX authentication mechanism. With an SSI system:

- Users can log on from any terminal on any system in the network
- Users still have to provide correct passwords when logging in
- File access is subject to standard AIX authentication mechanisms
- Users see the same file system structure no matter where they log in
- You have a system that's easy to install and maintain.

Code Server

The concept of a code server was developed to allow common programs to be installed on only one of the systems in a network. You can select one or a combination of the following two types of code server functions:

Active code server

Clients using an ***active*** code server run the programs they need by mounting the program libraries from the code server and loading the programs into its own memory across the network.

Passive code server

Clients using a ***passive*** code server do so by installing its own copies of programs from a prepared copy on the passive code server. Executing the programs once installation is completed is done from the local file system.

Remote Print Servers

There are two ways of defining remote printers in AIX:

Distributed Services remote printers

If you do not need to provide remote print services across network boundaries and if you do not use TCP/IP for other purposes, Distributed Services remote print services is the way to go. If you ***do*** use TCP/IP and want to do remote print from machines that are not using Distributed Services, or if your print server is on one network but some of the users of remote print on another, then go for TCP/IP remote print services. In an SSI configuration, the use of Distributed Services remote print servers require (see "The AIX Queueing System" on page 73) that you do ***not*** share the `/etc/qconfig` file.

TCP/IP remote printers

The advantage of the TCP/IP remote print services is that printers can be shared across network boundaries. For details about customizing TCP/IP remote print servers, see "Transmission Control Protocol/Internet Protocol (TCP/IP)" on page 165.

Remote Backup and Restore

This will make it possible for you to use a tape streamer connected to one node from another node.

Node Table Server

For large DS networks, a separate Node Table on each system is difficult to maintain. A Node Table server will help you to distribute changes in your configuration.

Remote Mounts

The principal use of Distributed Services is for mounting of remote files and directories over local files or directories. Usually, the local directories are empty directories, so-called ***directory stubs*** so that the mounting provides a ***logical extension*** of the local file system. However, nothing prevents mounts over local directories that already hold data.

To allow users access to local directories even when remote directories have been mounted over them, AIX/RT Distributed Services Version 1.2.1 automatically mounts all local file systems twice as seen in the simplified picture of an IBM RT file system in Figure 34 on page 63. Every branch of the local file system "tree" can be reached from the root directory and from the directory `/native`.

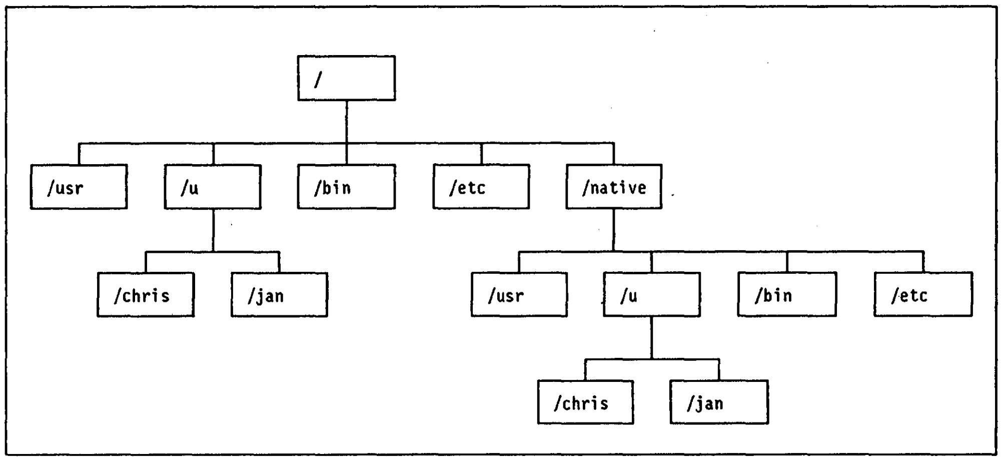


Figure 34. Simplified Local File System

Now consider a directory /u on a Distributed Services file server as shown in Figure 35. If we mount this directory over the local /u directory, we get the resulting file system appearance as shown in Figure 36.

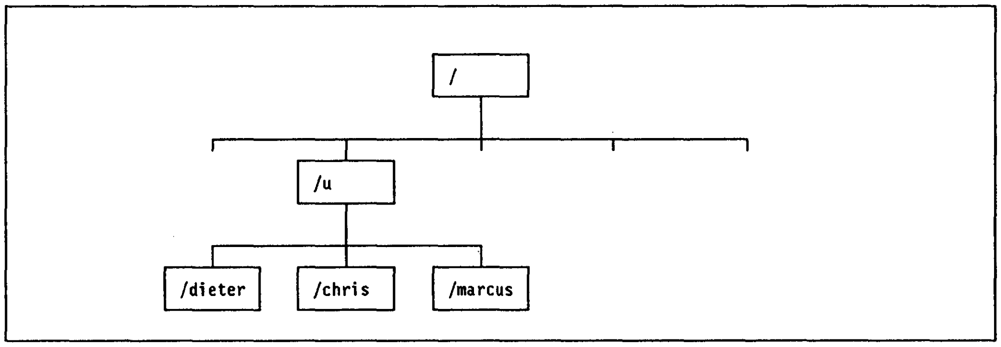


Figure 35. Part of Server File System

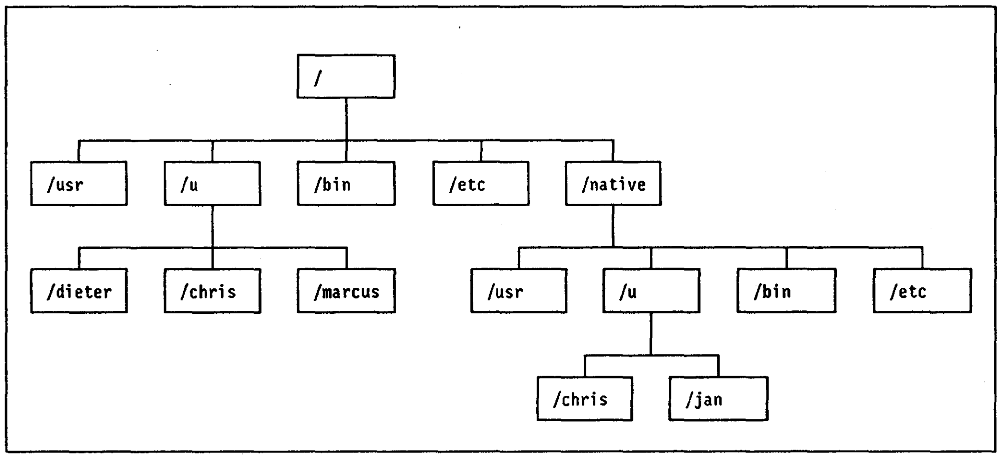


Figure 36. Resulting View of Local File System

As you'll see, the directory on the file server, including all its subdirectories and files, have replaced the local /u directory as seen from the root directory. However, the local /u directory can still be accessed through the path /native/u.

Establishing Connection

When a Distributed Services *mount* command is issued to mount a remote file or directory, AIX uses the ***Node Table*** to translate the node ***nickname*** into a ***node ID***. The node ID shares its name with a ***Connection Profile*** in SNA Services. Whenever the *ndtable* command is used to define a participating node, the corresponding Connection Profile is automatically added to the SNA Services profiles[8].

Distributed Services uses the Connection Profile for the remote node to start the ***attachment*** pointed to from the connection profile. Again, the Attachment Profile for a remote DS node is added automatically when the node is defined with the *ndtable* command and is named after the node ID of the remote node.

SNA Services establishes a connection using the named local profiles and the specifications in predefined Connection- and Attachment Profiles at the remote system. Figure 37 shows how DS uses the Node Table information to find the correct local profiles for Token-Ring.

In order for SNA Services to establish a DS connection, there must be a ***CALL*** node and a ***LISTEN*** node. On each system providing Distributed Services **server** functions, attachments for the supported network types must be started; usually this is done from the file `/etc/rc.ds`, which is executed at boot time. For an Ethernet connection the attachment name is **EDEFAULT**, for Token-Ring it's **KDEFAULT** (when the corresponding adapters are configured as the primary adapters of their kind).

`KDEFAULT` and `EDEFAULT` are defined as `LISTEN` nodes and wait for incoming calls from the network. The profiles also use the ***AUTOLISTEN*** option, meaning that there's always a derived copy of the attachment in starting status. For example, when `EDEFAULT` is in use, a new attachment is generated internally and gets the name `EDEFAULT1`. `EDEFAULT1` is the new attachment that'll be listening for request from the network.

Applying the Mount Request

Once the connection is established it remains active. DS is using the connection and session between the two nodes to apply the mount request.

This is done by changing the ***vnode*** (virtual inode) of the directory or file you mount over, so it no longer points to a local inode, but points to the remote file or directory. Of course, DS checks for authorization to do the mount and the existence of the directories or files on both systems and reports an error back if the mount can't be accomplished.

When the vnode substitution is in effect (that is, while the mount is active) Distributed Services will redirect all request for access to the local "mounted over" directory or file across the network to the remote mounted directory or file.

[8] Observe that the node ID is directly tied to the host's cpu. If the machine's processor is changed, *ndtable* should be run. It will automatically update the SNA Services profiles with the new node ID. All you need is to press enter at the first screen displayed by *ndtable*; then exit. No changes need be keyed in by the user.

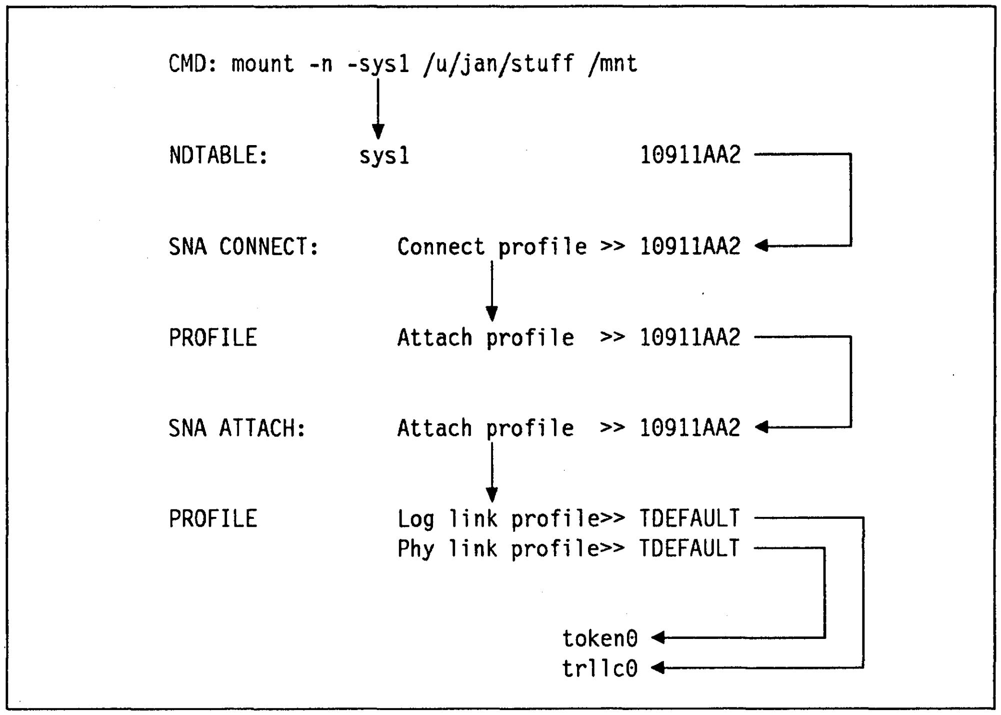

CMD: mount -n -sysl /u/jan/stuff /mnt
NDTABLE: sysl 10911AA2
SNA CONNECT: Connect profile >> 10911AA2
PROFILE Attach profile >> 10911AA2
SNA ATTACH: Attach profile >> 10911AA2
PROFILE Log link profile>> TDEFAULT
Phy link profile>> TDEFAULT
token0
trllc0

Figure 37. Resolving Mount Request

Customizing Distributed Services

No matter how you want to use Distributed Services, you must go through the basic install and customizing steps described in *Managing the AIX Operating System*:

1. If the communication adapter(s) you want to use are not already installed and configured, install adapters and use the diagnostics diskettes to check for memory, interrupt level and DMA assignment conflicts. Change the adapter settings to avoid conflicts, if required, and write down what the settings are for use later.
2. If the adapters are not already defined to your host, use the *devices* command to add the ***devices*** to the configuration.
3. If your adapter has not previously been used for SNA, you need to add a ***data link*** for each adapter you want to use for DS.
4. Modify the file /etc/rc.ds so that the attachments corresponding to the installed adapters are started. DS assumes EDEFAULT, so if you are not using Ethernet through a primary Ethernet adapter, you must comment the lines for EDEFAULT out.
5. In case you already have SNA Services active (even if active for an adapter not used by DS), before continuing with the customization, ***stop SNA Services*** and keep it stopped while you configure Distributed Services. To stop DS and SNA Services, use the following commands:

```
shutdown -df
stop sna
```

6. Do a basic customization of the systems on the network, using the information in the section "Configure a Basic Distributed Services System" in the publication *Managing the AIX Operating System*.
7. If you want to run Distributed Services over X.25 connections, refer to "Distributed Services over X.25" on page 319 for details about customizing SNA Services for such environment.
8. When you've finished the customization, start SNA Services and Distributed Services by typing:

```
        sh /etc/rc.ds
```

Since you will need information about the remote systems when customizing Distributed Services, it may be advantageous to have TCP/IP running so you can log in to the remote systems rather than walking around or calling people to ask questions they may not know how to answer. Also, since the programs you need to customize Distributed Services all have a user interface based upon *AIX Usability Services*, the keyboard overlay for *Usability Services* may come in handy.

Customizing for Basic Distributed Services

Two tables are necessary for the operation of Distributed Services, the ***Node Table*** and the ***User/Group Table*** which are maintained by the commands *ndtable* and *ugtable*, respectively.

Managing the AIX Operating System gives detailed instructions on how to create and maintain the two tables. We shall spare you yet another description. We shall ***not*** refrain from adding a few comments that we hope you'll find useful.

Node Table

With DS Version 1.2, when you customized Distributed Services for a Token-Ring connection, you would find an entry in the Node Table of every machine saying the data link type was ***Ethernet***. You could safely ignore this line. It was inserted automatically when installing Distributed Services, but was not used if you didn't use Ethernet. With DS Version 1.2.1 this line no longer appears.

Node Tables without nicknames are like programs without comments. Even in small networks, you'll soon find yourself doing *telnet* into the remote systems to sort out which node ID corresponds to which machine. So, ***do*** use nicknames for all nodes in the network; preferably the `hostname` of the machines to keep things simple.

User/Group Tables

Before you jump head-on into the mysteries of the User/Group Table and *ugtable*, work out on a piece of paper what it is you want. Proper planning can save you some big headaches later on. The important thing is to establish a system that allows you to adjust the tables for new users as these are (inevitably) added to the systems in your network, and to do so with the least possible work, not to mention the fewest possible errors.

If your intention is to provide one or a few ***data servers***, you can keep things relatively simple. It may be worth it to consider this possibility rather than adding disk to all machines as the disk space requirement grows.

Depending on your environment, you may want to forget about users and do all tables based upon groups. This would be feasible if your users work in groups that share all information. To accommodate new groups (where names are unimportant) you could start out by predefining, say, 20 extra (identical) groups on each system in the network. Adding users then means ***no extra work*** if you keep the users' login directories on each system's local disk.

Be aware, that as users are usually added to the systems in random sequence, there may be total anarchy in the allocation of numeric user IDs to user login names. If this turns out to be a major problem, do consider Single System Image, where the system maintains unique relations between numeric and alphabetic user IDs.

After you've made your plan, run *ugtable* on two systems, then start Distributed Services and work with remote mounts to check out if you've overlooked anything. You'd like to check:

1. That users have write access to their own data on remote machines no matter what numeric user ID they have.
2. That no user has unauthorized access to other users' data.
3. That nobody can run as superuser without logging in as such.
4. That access based upon group ***name*** rather than numeric group ID works as intended.

If it works, include another system and, again, check. To do the checking, you'll need to log in to each system using more than one login name on each. Make sure that you do not oversimplify, that is, run the checks from systems where there is no unique relation between numeric and alphabetic user IDs. Only after you have tested your plan on three systems should you take the final step and propagate it to all participating systems.

Deciding How to Mount

Between you and us, mounting is fun. For end users it should be transparent. So, don't expect (or allow) your end users to do other than authorized mounts. In practice, this means that no end user should ever have to use the *mount* command with the `-n` option. You may even want to change the attributes, owner and group of the *mount* command so only members of the `system` group can run the command.

You may select from a number of possible options to do the mounts automatically at system start-up time. The primary choice is whether you want to use one of the shell-script-based procedures that'll keep trying to mount if the desired mounts are not active. Check the comprehensive explanation of your options in *Managing the AIX Operating System* before you decide.

Usually, a simple approach is quite adequate. You base the mount upon stanzas in `/etc/filesystems` where you group remote mounts by logical use under descriptive names. You may or may not want to specify `mount = true`; the important thing is the grouping and the meaningful names of the groups. Then make a few simple shell scripts and tell the users to run them if they see problems with shared files. Consider the script in Figure 38:

```
wall Hey, everybody.  Remount is due in 20 seconds
wall Please wait
sleep 20
`pwd` > /native/$HOME/current.path
cd /
unmount -t engineer
mount -t engineer
cd < /native/$HOME/current.path
wall Okay, go ahead...
```

Figure 38. Sample Shell Script for Mount

Name the shell script `engineer` and make it executable with root authority. Then tell the users to type `engineer` if they encounter problems with the remote files. The shell script may not be very friendly in a multiuser environment but it's adequate when each system has only one user at a time. Notice that the script changes to the root directory while doing `unmount` and `mount`. Here's why:

The root directory is the only safe place to be over mounts, since the user may be positioned in a local directory that is now mounted over. Because of (presumably) the cacheing of disk, the user will continue to work in the local directory even though a mount over that directory has taken place. Switching to the root directory while doing mounts is good practice.

Customizing for Single System Image

The section "Creating a Single-System Image" in *Managing the AIX Operating System* tells you (almost) all you need to know when configuring your systems for Single System Image. There are a few comments, though:

/etc/rc.SSI

The above mentioned chapter says that the shell script that is run at system boot is `/etc/rc.DS`. The correct name is `/etc/rc.SSI`.

/usr/adm/ds.msg

The file `/usr/adm/ds.msg` is created as a log file and the system will save up to seven old generations under the same filename but with a suffix of `.n`, where `n` is a digit from 1 through 7.

DS-SSSI.README

Print and read the file `DS-SSSI.README` before you start the customizing of Single System Image. This file provides the latest updates and will give you the latest and most correct explanation of files and procedures you need to install SSI.

adminserver

Select one of the systems to be the administrative system. Don't forget that to follow the customizing procedure the, `/tmp` directory must have at least 300 blocks available. Copy the file containing the single system image sample files to `/tmp`. All required files are contained in `/usr/lpp/ds/samples/SSI.backup`, which is an archive file in backup format.

Now you only need to follow the instructions in the `DS-SSSI.README` file.

usr/adm/user.cfile

The file `usr/adm/user.cfile` is used when you add a new user to create a standard user directory. The "udir" parameter in this file defines the default $HOME directory for new users. Change this parameter to: `udir /u/node/`, where node is the nickname of the administrative system. If you have changed any of the other default values, such as the default login shell, change those values in this file also.

/etc/qconfig

As you may want to exercise a little more control over the print servers than the standard installation procedure provides, move the file `/tmp/SSI/etc/qconfig` to `/tmp/SSI/etc/qconfig.sav`. Then copy the original `/etc/qconfig` file to `/tmp/SSI/etc/qconfig`. Finally, choose between the two possible ways of doing remote print and update the file `/etc/qconfig` to reflect it:

TCP/IP print server

To use TCP/IP print servers, and provided you haven't installed this service already, install TCP/IP on all systems and change `/tmp/SSI/etc/qconfig` rather than `/etc/qconfig`.

Distributed Services print server

To use Distributed Services print server functions, you'll have to ensure that the file `/etc/qconfig` is not mounted. To do so, remove the file name from the two shell scripts `/etc/remounts.list` and `/etc/server.files`. Remove the `/tmp/SSI/etc/qconfig` file as it is not needed. Do **not** erase the original `/etc/qconfig` file.

Customizing a Code Server

This is the most complicated configuration, but it is also the best documented one. You'll need little advice after having read through the relevant sections of *Managing the AIX Operating System*, but you might find the following helpful.

Active Code Service

- A client using an active code server connects to the server at system startup and is dependent on the server for executing programs that are installed on the server only.
- At system start a client executes the *chngstate* command.
- The client uses remote mounts to mount several directories from the server over the corresponding directories on the client, for example:

```
/usr/bin
/usr/lib
/usr/lpp
/usr/include
/usr/man
/usr/pub
```

- A client may only connect to one code server at a time as it mounts program directories.
- If a primary code server is unavailable, the client may connect to an alternate code server.

- Each machine in the code server environment must have a base of software installed locally (VRM, AIX Base Operating System, SNA Services, DS and VRM Device Drivers).
- Complete copies of other licensed software may optionally be installed on the client to be executed from the local disk.[9]
- In a default environment each client has the same software installed, is connected to the same server and is "served" the same software.

Program Subsets

- Some licensed software can readily be used from a server in a DS environment by simple remote mounts, for example X-Windows. For such products, a full code server environment is unnecessary.
- Some licensed software cannot be used through simple remote mounts because they use:
  - Code in /etc (but /etc should not be remotely mounted)
  - Kernel device drivers (TCP/IP, Network File System, em78)

  Licensed programs with such special requirements provide program subsets, which are comprised of:
  - Code that needs to be resident on each client
  - An ***install*** script for the subset (for example /usr/lpp/nfs/liblpp.cp.a).
- In an active code service environment ***all*** licensed programs that include a program subset must be served from the server.

Planning for Active Code Service

- Refer to *Installing and Customizing the AIX Operating System*, Chapter 1.
- On the server, /tmp should be 35000 blocks.
- On the client, space must be allocated for program subsets. Information may be found in *Installing and Customizing the AIX Operating System* and also in the /README file.

Customizing Active Code Server

Follow the steps in *Managing the AIX Operating System*:

1. Install VRM, the Basic Operating System, DS and VRM Device Drivers (remember to change the size of /tmp and /dmp).
2. Backup VRM.
3. Configure with *devices, ndtable* and *ugtable*.
4. Edit /etc/rc - uncomment *chngstate* command
5. Backup the installed operating system.

Installing additional licensed software on the server:

1. Create directories (minidisks): /usr/lpp.install and /usr/lpp.update.

[9] A client in a code server environment cannot run local software without a series of remounts of native files (that is, the default environment is one where all code is "served").

2. Increase ulimit to allow for large files to be copied across the network. Change ulimit to 50000 or more.
3. Use *installp -b*
   - The "-b" flag runs the *bffcreate* command.
   - *bffcreate* creates a backup format program file and program subset file (stored in `/usr/lpp.install`).
   - Install from the program copy in `/usr/lpp.install`
4. Use *updatep -b*
5. Run *setrdperm*

Note: The *bffcreate* command stores a backup format image of the software in `/usr/lpp.install`. The *setrdperm* command runs *chmod o+r* on files for code service clients.

Customizing First Active Service Client

Follow steps in *Managing the AIX Operating System*:

1. Install VRM.
2. Install AIX from tape created when setting up server.
3. Run *ndtable*.
4. Install any licensed software that will be used in stand-alone mode.
   - Remote mount of `/usr/lpp.install`
   - Use *installp -d*, from mounted directory.
5. Update any licensed software that will be used in stand-alone mode.
6. Run *ugtable* for root, bin, sys, adm, etc.
7. Edit `/etc/codeserver/attach` to define primary and alternate code servers to the client.

Customizing Other Clients

Follow steps in *Managing the AIX Operating System*:

1. Backup a complete image of the first client.
2. Install VRM.
3. Install AIX from tape created in first set. Install any licensed software that will be used in stand-alone mode.
4. Run *ndtable* to update the SNA Services profiles for the processor you installed the tape on. If you don't, Distributed Services will fail to run properly[8].
5. Update any licensed software that will be used in stand-alone mode.
6. Run *ugtable* for root, bin, sys, adm, etc.
7. Edit `/etc/codeserver/attach` to define primary and alternate code server to the client.

Re-boot all machines to start the code server environment.

Starting a Client

chngstate: The *chngstate* command is issued from the /etc/rc script on a code server client when the client is started. Depending on a number of factors *chngstate* will ultimately run either /etc/rc.actvsrve to attach the client to the server in active service mode, or /etc/rc.standalone to bring up the client in stand-alone mode.

The first task of *chngstate* involves the validation and the subsequent processing of a file named /etc/codeserve/serverattach. This file specifies whether a system should be brought up in active-service mode and, if so, which server the client should attach to and how it should recover if that server is unavailable.

Once an active service client has successfully attached to a server it invokes the *chkcomp* command.

chkcomp checks for client and server compatibility before attempting to attach to the code server. If a program subset has not been installed on a client, then *chkcomp* will record its absence in a file /etc/codeserver/cs.compat. Also, if there are incompatibilities between the levels of the base software (in terms of the levels of installed software or updates), or if any software is installed on the client for standalone operation, then these will also be flagged in cs.compat. Finally, if the server of a program subset on a client is different from that of the program on the server, then this information will also be flagged.

After *chkcomp* has created the cs.compat file it passes control back to *chngstate*. If the incompatibilities between the server and client can be fixed by the system (and if the administrator has defined the "upgrade mode" of the client to be "automatic"), then *chngstate* will invoke either the *installc* or the *updatec* commands. These internal commands load program copies from the server's /usr/lpp.install or /usr/lpp.update directories.

It is important to note that the first time a client attaches to a server, a large cs.compat file, flagging all the software requiring program subsets to be installed on the client, will be created. If the upgrade mode is automatic, then the system will proceed to install each subset on the client.

serverattach: One of the configuration steps for an active service client involves the uncommenting of the /etc/codeserve/serverattach file stanzas and identification of the stanzas with the appropriate DS servers. The other parameters are:

state	which may be set to "active" or "stand-alone".
mode	that may be set to either "auto" or "manual". If "auto" is selected, then the client (through *chngstate*) will try to make its program compatible with the server automatically.
time	is the total time that the client will attempt to attach to the server. If the client cannot attach to the server during this period, then *cnhgstate* will process the next stanza.
interval	is the amount of time that the clients waits between attempts to attach to the server.

/etc/rc.actvsrvc: The final task of *chngstate*, when invoked from /etc/rc, is to run the /etc/rc.actvsrvc script.

This script uses the following *mount* command: `mount -t codeserve`. The system searches the client's /etc/filesystems file for stanzas with "type = codeserve" and then remotely mounts the corresponding filesystem from the server. The /etc/filesystems file that is shipped with AIX already contains a number of stanzas of type "codeserve". The administrator may comment out or add additional stanzas as appropriate.

If an administrator wishes to remount any of the client's native directories, then these should be specified in this shell script - for instance, if there is code that he wishes to make available only on a client, and not on its server.

Passive Code Service

- Makes use of "program copy files" in /usr/lpp.install, created by *bffcreate* or *installp -b*.
- Simply mount appropriate server directory at client:

```
mount -n m135 /usr/lpp.install /usr/lpp.install
```

- Then use "-d" flag on *installp* or *updatep*:

```
installp -d /usr/lpp.install/program
```

Final Remarks on Code Service

- Code service and SSI may coexist, but if either service is not available then the other service would not be affected. The most logical combination would involve a code server that was also an adminserver for the SSI. Some code service function may be provided through SSI.
- The ***Network 3270-PLUS*** and ***Network RJE-PLUS*** programs can't be used in a code server environment as they write into their directories during execution (you can use remounts on a client, however).
- ***VS Pascal, VS Fortran*** and ***CADAM*** cannot be used in a code server environment.
- For ***WHIP*** and Usability Services, refer to the /README file.

The AIX Queueing System

When the AIX *print* command is executed, it places a request for print jobs in a temporary file named /usr/lpd/qdir. This file contains control information for the print request (for example, time, date, user and node). If the print command is presented with a file containing a list of files (for example when such a list is piped to "print") to be processed by the printer backend, then the *qdeamon* will set up a temporary file in /usr/spool/qdaemon where the list is held until it can be processed by the backend. In this latter case, the file in /usr/lpd/qdir contains a reference to a temporary file holding the list in /usr/spool/qdaemon.

The qdaemon (once it has been notified of an addition to its queues) starts a backend process (at an appropriate time) to perform the actual "print" function. The backend prints the files listed in /usr/spool/qdaemon according to the information in /usr/lpd/qdir.

The /etc/qconfig file is used by both print and qdaemon. It describes the configuration of available queues and devices. Remote queueing is enabled through the appropriate customization of the /etc/qconfig files in a DS environment. The /etc/qconfig file is (obviously) an ASCII file, and therefore it would be inefficient for the system to frequently read such a file. For this reason, its information is held in a binary file named /etc/qconfig.bin. If the qconfig file is altered, the the *print* command will recognize the changes but ***only*** after you execute print -rr command to notify the qdaemon.

For more information see the section "Using Distributed Services to Provide Remote Queues" in *Managing the AIX Operating System*.

Distributed Services Remote Printers

Remote printers are easy to configure in a Distributed Services network, but they have two disadvantages:

1. They can not be reached through a gateway between two networks because Distributed Services (or, SNA Services, rather) does not support gateways.

   A solution is to connect all shared printers to the gateway machine where they can be reached from both networks. Of course, this only helps if you have only one gateway.

2. You can not have an SSI configuration in which /etc/qconfig is shared between the systems. That's because the printer server will not accept the remote printer definition in its /etc/qconfig file.

   You can still use DS remote printers in a SSI configuration if you decide to have a /etc/qconfig files on each node.

To install a remote printer, add the following queue stanza in /etc/qconfig on all ***remote*** nodes:

```
rlp:
        argname = -rl
        node = nodename
        rargname = p3812
```

nodename should be substituted with the printer server's real nodename. The word ***p3812*** must correspond to the ***argname*** stanza on the printer server that invokes a printer queue. In this example, there should be an entry like this in /etc/qconfig on the server:

```
lcl:
        argname = p3812
        device = dlcl
```

in which ***dlcl*** defines a printer. You can use any name for the *argname*, as long as the *rargname* corresponds.

When you have edited the /etc/qconfig file, issue the command *print -rr* to apply the updates. To print from a remote machine, use the print command like this:

```
print -rl filename
```

Notice that the -rl flag must match the entry in /etc/qconfig, so if you put ***rl*** in /etc/qconfig, you must use ***rl*** in the *print* command. If you put ***-rl*** in /etc/qconfig, you must also use ***-rl*** in the *print* command.

Remote Backup and Restore

The section "Using Distributed Services to Provide Remote Queues" in *Managing the AIX Operating System* describes how to add and use the remote backup facility. You only have to add a couple of stanzas in the remote and in the local system as described in "Defining Backup and Restore Queues". The print command is used for remote backup and restore but with a slightly different syntax.

Dynamic Node Table

One Node Table can be used as the source of Node Table entries for other nodes in a Distributed Services Network.

- The node holding the common Node Table is called the ***Node Table server*** and has the ***Master distributed network Node Table***.
- The Node Table server must be configured with ***dynamic node table*** .
- The Node Table server contains all information required to establish node-to-node communication between client nodes ***except*** the BIND password clients need to communicate with secured nodes.

Other nodes in the network can *mount* the server Node Table and thereby gain access to all nodes listed in the server Node Table.

- The ***Client distributed network Node Table*** is defined as a ***static Node Table***.
- The client Node Table contains an entry for the client and an entry for the server.
- If the server is a secure node, the client must define the BIND password for the server.
- Passwords for communicating with other nodes are defined using the *pwtable* command.

All options available for defining a static Node Table are valid for the dynamic Node Table.

Additional options available are:

- ***Prototype Connection Profile*** used to construct the Connection Profiles. Can use the system default or user-specified profiles.
- ***Prototype Physical Link Profile*** used to construct the physical link profiles. Can use the system default or user-specified profiles.
- ***Command/Selection Sequence*** used for either:
  - ***X.21 Selection Sequence***. A series of up to 255 ASCII characters that determine the destination of information on the network.
  - ***Smart Modem Command Sequence***. Up to 79 ASCII characters that control the modem's call establishment and data transfer options.

Distributed Services Commands

The following is a summary of commands related to DS:

bffcreate	Creates a program copy file from the specified distribution medium. Both *installp -b* and *updatep -b* use *bffcreate* to create program copy files. The command is part of the base AIX system but is normally used with DS.
chkcomp	Checks compatibility between a code server and an active code service client.
chngstate	Controls the attachment of a client to a server and, depending upon how the code service attribute file is customized, can perform most program installations or updates required to achieve compatibility between the server and the client.
dsipc	Used to install Distributed Services IPC key mappings into the kernel.
dsldxprof	Loads translate information from a flat file into the User/Group Translate Tables. *dsldxprof -d* deletes the ***pfsuidgid*** profile and then recreates it without any entries. This option is handled before the ***-f*** option if both exist.
dsstate	Alters the state of the Distributed Services kernel logic by allowing/blocking inbound/outbound requests and starting the Distributed Services kernel processes.
dsxlate	This command is intended for use by the system itself and, possibly, by the system administrator. The primary use of this command is internal to the system. It is called from the Distributed Services configuration operator interface to update the kernel when menus have been used to update the DS profiles, and it will usually be included in `/etc/rc` to load the kernel at startup time. *dsxlate* is used to install Distributed Services User/Group Translate Tables into the kernel. The translate information is stored in profiles managed by profile services.
installc	Installs complete programs and program subsets on a code service client.
ipctable	This command is intended for use by the system administrator to set up or change the IPC table entries.
ndtable	This command is intended for use by the system administrator to set up or change the Node Table entries.
pwtable	Used to define bind passwords (if needed) for connections between the client and other (non-server) systems on the network. This command will create the network node security table.
setrdperm	When running on the server, this command scans all files contained in the directories listed in `/etc/codeserve/csomdlist` for proper permission settings.
ugtable	This command is intended for use by the system administrator to set up and change the User/Group Table entries.

updatcc	Updates complete programs and program subsets on an active code service client.
writesrv	The *writesrv* program, which is run from `/etc/rc.include` at system start, provides the remote write capabilities. For more information, see *write* and *writesrv* in *AIX Operating System Commands Reference.*

Security

Under certain circumstances, some or all systems in a network may require an additional level of security. Additional security is provided through SNA Services, which allows you to restrict access to a particular node and its profile tables through the use of a DES-encrypted password.

Node security for nodes defined in static Node Tables is established by selecting the ***secure*** option while defining the Node Table or by modifying a previously defined static nodetable. After it has been defined, this password is required to access profile tables and must be supplied by any system that communicates with the secured node.

There are two steps to accomplish this:

1. The node to be secured runs the SNA utility *commopw* and specifies a 30- to 80-character password (can be a phrase with blanks). This password will subsequently be needed to access the *commopw* program as well as any of the profile tables on that node. This password, or security key generated from it, must then be defined on every system it communicates with.
2. The Node Table on each system that is to communicate with the secured node must include the secured node's password. This is accomplished through the *ndtable* utility by specifying the *secure* option in the security field associated with the secured node. Once the password is defined in the Node Table, the system can exchange files and programs with the secured node.

Problem Detection

- **Run Diagnostics!**
- If TCP/IP is installed, you can use the *ping* command to verify that the network hardware works properly.
- Use Token-Ring diagnostics if appropriate ("Appendix C" in *Managing the AIX Operating System*).
- Check error logs on both systems for link, SNA or DS messages.
- Check that correct device drivers are installed.
- Check the levels of AIX, SNA and DS software on both machines, including the levels of the updates and PTFs (Program Temporary Fixes) applied. `updatep -A` shows the currently applied updates.
- Check device definitions of hardware and data links.
- Check jumper settings on cards.
- Use SNA Services utility commands such as *status*, *linktest* and *snatrace*.

- Use the AIX tracing facility for DS.

Note: You can edit the `/etc/trcprofile` file in order to select specific DS events to trace. Tracing is enabled through the `trace` command and disabled through the `trcstop` command. The `trcrpt` command formats and displays the contents of the log file created by *trace*. Examples of how to trace DS events are given in *Managing the AIX Operating System*.

Tuning

There are, unfortunately, no hard and fast rules for tuning Distributed Services, but you will find a section at the end of the Distributed Services chapter in *Managing the AIX Operating System*. Also, Appendix I, "AIX/RT Performance Tuning" on page 535 contains information about tuning your AIX/RT system. Finally, for every new release of AIX, check the `/README` file where hints about tuning and other helpful information is passed from the development laboratories. The following parameters are candidates for changes when you tune a Distributed Services network:

- In `/etc/master`
  - **kbuffers**
  - **kprocs**
- Device buffers defined in `devices`
  - **nobodr**
  - **norbosr**
  - **nobibp**
- The *dsstate* options in `/etc/rc.ds`
  - **-cs** prevents cacheing of files that the local client reads from a server
  - **-ss** prevents cacheing of files on remote clients when using files on the local server
- Large or "busy" networks need special care. In `/etc/master` you may want to consider the following additional changes:
  - **dsnkprocs**
  - **nncb**
  - **maxnode**
  - **filetab**
  - **inodetab**
  - **callouts**

  These parameters affect the size of the `/UNIX` kernel. You might need to increase the paging space. Consult the `/README` file for the latest available information.

SNA Services Profiles

Distributed Services does a *sync* on open files every 60 seconds. This is no problem on modestly loaded systems but may impose significant extra load on busy networks. The *sync* is done by *cron*, and you may want to either change the interval or disable it, if your network is heavily loaded.

If you do so, you may want to change some values in the SNA Services profiles to prevent the connection from timing out. You will want to change the value "Stop connection on inactivity" to ***NO*** in the Connection Profile.

If you keep the value "Stop connection on inactivity" at ***YES*** in the Connection Profile, you would want to change the "Inactivity timeout value" in the Token-Ring Logical Link Profile to a value above the *sync* interval. Default is 48 seconds; change it to 64 seconds.

If your network has significant load, you should also increase "Response time-out" to 10 seconds and "Acknowledgement time-out" to 5 seconds. Both values can be found in the Logical Link Profile for Token-Ring. You may use other values, but the default values can easily become too small.

Application Programs

Distributed Services is transparent to application programs. They can access remote files just as if the files were local. Performance will normally not be significantly impacted when remote files are used, but in some cases, applications may handle file I/O in a way that is not very well suited for remote access.

If, for example, an application reads a large file in very small chunks, the overhead of the remote access can become excessive. VS Fortran, for example, can often generate code that reads data 4 or 8 bytes at a time. While impractical for remote files, it can render the application useless when used on remote files. Recompiling with Fortran 77 can often remove most of the overhead.

Also, applications should never use **getattr** or locking when not strictly required. Both functions can have significant impact on overall performance.

You should be aware, that whenever an application opens a remote file in write mode, cacheing on all client hosts is suspended. Obviously, an application should never open a file in write mode unless it intends to write. However, many applications violate this rule and get away with it when used on local files. When you use the same applications on remote files, and simultaneously from several hosts, you see significant performance degradation.

It is very important, that when you plan to use a standard application (a data base program, for example) with Distributed Services, you test it in a realistic environment before you decide. You may see superior performance with one server and one client. Adding just a few client hosts and testing the network with realistic loads may or may not tell a very different story. The important thing is to know before you invest time and money. The test may tell you to pick another solution or product while you can. If you wait for actual production to start and then see unsatisfactory performance, it may be too late to do something about it.

Fortunally, DS normally performs very well, so don't panic. All we want to say is, that large program packages you cannot modify yourself, should be tested in a realistic environment before you decide to use them with DS.

Reference Publications for This Chapter

Distributed Services and the DS commands are described in:

Managing the AIX Operating System, SC23-2008
Installing and Customizing the AIX Operating System, SC23-2013
AIX Operating System Commands Reference, SBOF-1814

APPC/LU 6.2 Communication

This chapter describes a set of APPC sample programs. The first section describes the programs and the following sections contain practical experiences running the sample programs.

APPC Sample Application

To demonstrate and allow readers of this publication to try program-to-program communication (APPC) via SNA LU 6.2, we've provided sample programs for various environments. The sample programs allow simple file transfers from an IBM RT to or from remote hosts, using whichever physical and logical connections are available for SNA LU 6.2 communications.

The sample APPC application is always initiated from the IBM RT by executing the program **snaftp**, an acronym for ***SNA File Transfer Program***. *snaftp* will schedule a ***remote transaction program*** on the remote host. The default name of the remote transaction program depends on the type of remote system.

Similarly, the *snaftp* program assumes default ***Connection Profile names*** depending on the remote host. The default remote transaction program names and corresponding Connection Profile names on the local IBM RT are shown in Figure 39, but may be changed by the user from the command line when executing *snaftp*.

Type of remote host	Default name of remote transaction program	Default name of local Connection Profile
AS/400	AS400RT	AS40062
AIX/RT	RTRT	RTRT
VM	VMRT	VM62
MVS	MVSRT	MVS62
IBM PC-DOS	DOSRT	DOS62
IBM OS/2	OS2RT	OS262

Figure 39. Default Names Used by the snaftp Program

The program *snaftp* and the remote transaction programs send and receive two distinct message formats:

1. Control messages
2. Data messages.

All messages are variable length, arbitrarily restricted to a maximum of 4k bytes, but programs should be designed to make it simple to change this size. The IBM RT program does so by defining the constant N_BYTES.

Whenever data is transferred in text format, the IBM RT *snaftp* program converts it to or from the representation used by the remote system. For IBM AS/400 and IBM System/370 this format is EBCDIC. For other systems it is ASCII. When control messages contain character codes, these codes are translated after the same rules.

Length and count fields are stored in S/370 binary format with the most significant bit in the highorder bit of the 4-byte fields ***except*** when talking to Operating System/2 and Disk Operating System machines; then the standard Intel format is used.

Message Formats

Control messages

Control Message Format									
CMT	FT	RF	STAT	PIL	IL	RL	BS	LM	MSG
1 byte	1 byte	1 byte	1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	Variable length

Field **Description**

CMT Control Message Type. The CMT-field can have the following values:

- **S** Request message. Requests that the remote system prepares to receive and store a file ***sent from*** the IBM RT. This message type is always sent by the IBM RT.
- **R** Request message. Requests that the remote system sends a file to be ***received by*** the IBM RT. This message type is always sent by the IBM RT.
- **M** Message type message. Sends status information from one system to the other according to the field ***STAT***. This message type can be sent by both sides.

FT File type. This field is only used for message types ***S*** and ***R***. The FT field can have the following values:

- **T** The file to be transferred is in text format.
- **B** The file to be transferred is in binary (untranslated) format.

RF File format. This field is only used for message type ***S***. The RF field can have the following values:

- **V** The file to be transferred must be stored on the remote system in variable record length format.
- **F** The file to be transferred must be stored in fixed record length format at the remote host.

STAT Status. This field is used only for message type ***M***. The S field can have the following values:

- **O** The capital "O" is used by either party to acknowledge the successful receipt and storing of a file and to acknowledge the receipt of a control message.
- **E** Used by either party to tell the partner that an unexpected condition occurred.

PIL Pad Information Length. The PIL field shows the padding added by the IBM RT. Padding of a file may be done by the IBM RT in the process of converting the IBM RT file format to the file format of the remote host. Padding should never be done by the remote host. Remote hosts should supply a zero in this field.

IL Info Length. The IL field gives the the total length of the file ***including*** padding. This information must be provided by the sender of the file. If the IBM RT sends the file, the length will be the file length as it should be stored at the remote system. If the remote system is the sender, the initial control message from *snaftp* does not contain the file size, but the remote system must supply the file size in its acknowledgement message.

RL Maximum/Variable Record Length. The RL field gives the maximum (in case of variable length records) or fixed logical record length. The remote host should obey the requested logical record length or return an error message if not possible. If the remote system is the sender, the initial control message from *snaftp* does not contain the record length. If the remote system is an IBM AS/400, it must return the record length in its acknowledgement message.

BS Block size. The BS field is ignored when the IBM RT receives files. When the IBM RT sends files, this field gives the requested block size to be used by the remote system when the file is stored. The requested block size must be obeyed by the remote host or an error message returned.

LM Length of Message. The LM field has the length of the file name (message types ***S*** and ***R***) or the length of any message returned in the field ***MSG***. The field must always be filled in and must have the value zero if no message text or file name is provided.

MSG Message. For messages of types ***S*** and ***R*** this field specifies the file name of the file on the remote system. For messages of type ***M*** the field supplies any error message text to be displayed (or otherwise disposed of) by the receiver of the message.

Data Messages

Data messages contain one or more field pairs, each consisting of a 4-byte length field and a variable length data field:

Data Message Format	
LD	DATA
4 bytes	Variable length

Field	**Description**
LD	Length of Data. Each LD field gives the length of the following data field. The field will hold the length of the data field even when fixed length records are transferred. Senders must ensure that lengths of data fields do not exceed the RL field and that they fit into the transmission block size (N_BYTES). If the remote system is an IBM AS/400, the data messages do ***not*** include the length field.
DATA	Data. The D field contains the data and has a length as specified in the preceeding LD field.

Logical Flow

Sending a File From the IBM RT

Figure 40 shows the logical flow of messages between the two systems involved in the file transfer of a file from the local (initiating) system to the remote system.

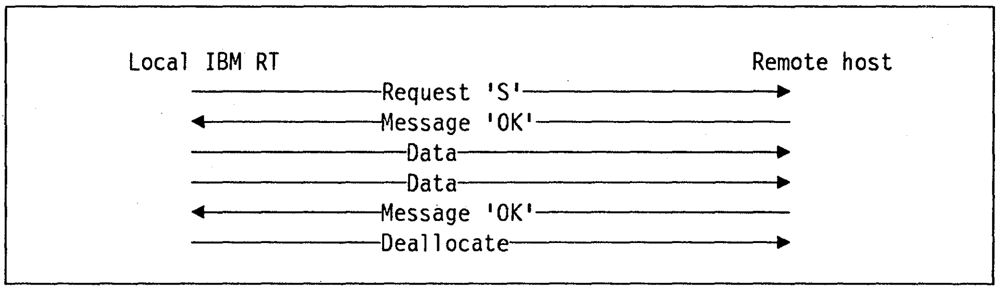


Figure 40. Sending a File to Remote System, Logical Flow

The flow is as follows:

1. The initial request contains all information the remote system needs to create the file, including file size, record format, record length and, of course, file name.
2. The remote host answers that it is ready for the file transfer.
3. The initiating host sends as many data blocks as required.
4. The remote host acknowledges that the file has been received and stored at disk.
5. The initiating host deallocates the conversation.

If, for some reason, the remote host is unable to receive and store the file, it should return an error message. If the file name sent to the remote system is invalid, for example, the flow would be as illustrated in Figure 41.

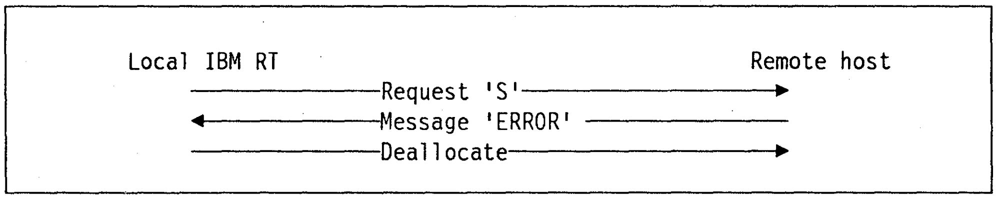


Figure 41. Remote System Detects an Error, Logical Flow

The flow is as follows:

1. The initial request contains all information the remote system needs to create the file, including file size, record format, record length and, of course, file name.
2. The remote host checks that the file name is valid (by trying to open the file, for example) and determines that the file name is invalid. As a result an error message is returned.
3. The initiating host notifies the user and deallocates the conversation.

Receiving a File on the IBM RT

Figure 42 shows the logical flow of messages between the two systems involved in the file transfer of a file from the remote system to the local (initiating) system.

```
Local IBM RT                                         Remote host
     ────────────────Request 'R'─────────────────────►
     ◄───────────────Message 'OK'─────────────────────
     ────────────────Message 'OK'────────────────────►
     ◄─────────────────Data───────────────────────────
     ◄─────────────────Data───────────────────────────
     ────────────────Message 'OK'────────────────────►
     ────────────────Deallocate──────────────────────►
```

Figure 42. Receiving a File From Remote System, Logical Flow

The flow is as follows:

1. The initial request contains all information the remote system needs to identify the file requested.
2. The remote host answers that it is ready for the file transfer and supplies the total file size and logical record length of the file.
3. The initiating host acknowledges the receipt of file size and logical record length.
4. The remote host sends as many data blocks as required.
5. The local (initiating) host acknowledges that the file has been received and stored at disk.
6. The initiating host deallocates the conversation.

Multiple Requests for File Transfer

Figure 43 shows the logical flow of two requests handled without deallocating the conversation. The example shows (first) a transfer from the local (initiating) system to the remote system, then (second) a file transfer in the reverse direction.

```
Local IBM RT                                         Remote host
     ────────────────Request 'S'─────────────────────►
     ◄───────────────Message 'OK'─────────────────────
     ──────────────────Data──────────────────────────►
     ──────────────────Data──────────────────────────►
     ◄───────────────Message 'OK'─────────────────────
     ────────────────Request 'R'─────────────────────►
     ◄───────────────Message 'OK'─────────────────────
     ────────────────Message 'OK'────────────────────►
     ◄─────────────────Data───────────────────────────
     ◄─────────────────Data───────────────────────────
     ────────────────Message 'OK'────────────────────►
     ────────────────Deallocate──────────────────────►
```

Figure 43. Multiple File Transfer Requests, Logical Flow

Notice that the ***deallocate*** is now replaced by another control message, namely the request to transfer another file. The design of the communications flow allows this to happen, but the current *snaftp* program does not implement it. Remote transaction programs should be prepared to handle multiple requests.

Hence, the remote transaction program must try to read another control message when a file transfer is completed. If another control message is received, the program should process it and then go back to check for further control messages. When the remote transaction program detects that while it is waiting for another control message, a ***deallocate*** has been issued by the initiating system, it should consider this a normal condition and terminate orderly.

Provided Remote Transaction Programs

With the current release of this publication, remote transaction programs are provided for the following remote systems:

1. IBM AS/400
2. Operating System/2
3. AIX/RT

Command Line Options

The program *snaftp* has the command line options listed in Figure 44. The options will be displayed by *snaftp* if you specify `snaftp -h` or `snaftp -?`.

```
-a : ASCII/EBCDIC and cr-lf conversion must be done
-h : This help text
-v : Variable record length file transfer
-c name : Connection profile name
-f name : Local file name
-r : Receive a file from remote host. Can not be used together with -s
-s : Send a file to remote host. Can not be used together with -r
-t name : Remote transaction program name
-V "remote_file_name" : Remote system is VM;
-T "remote_file_name" : Remote system is TSO;
-4 "remote_file_name" : Remote system is AS/400;
-A "remote_file_name" : Remote system is AIX;
-D "remote_file_name" : Remote system is DOS;
-O "remote_file_name" : Remote system is OS/2;
-b len : Physical blocksize at remote host
-l len : Logical record length (max variable length) at remote host
```

Figure 44. Command Line Options of the snaftp Program

The quotes around the remote file name are not strictly required for remote systems whose file naming standards do not allow blanks to be included in file names. See the second example below.

To transfer the AIX kernel from your local IBM RT to another IBM RT you can use the command:

```
snaftp -f /usr/sys/unix.std -s -A"/tmp/unix.bak"
```

Provided the connection is already active, the file transfer of an AIX kernel of almost a megabyte between IBM RTs and on Token-Ring takes about six seconds. This is close to a 50% utilization of a 4 Megabit Token-Ring considering that the time includes disk input/output and non-data traffic on the net.

To transfer a C Language source file from your local IBM RT to an OS/2 machine you could use a command as the following:

```
snaftp -f /u/chris/sample.c -sa -0 c:\c2\src\sample.c
```

This will transfer the C source file /u/chris/sample.c from the IBM RT and store it on the OS/2 machine as c:\c2\src\sample.c. Because of the "-a" option the text file format is converted from AIX to OS/2 format. You can send the file back to the IBM RT and compare the two:

```
snaftp -f /tmp/sample.c -ra -0"c:\c2\src\sample.c"
diff /u/chris/sample.c /tmp/sample.c
```

Hopefully the files are identical. Finally, here is an example of how you can send a text file to an IBM AS/400:

```
snaftp -f /u/fribert/sna.prof -sa -4"rtcomslib/snaprof"
```

A Word of Caution

Unfortunately, time has not allowed us to provide remote transaction programs for all the environments we'd have liked to cover. Similarly, time has not allowed the programs to be fully tested, let alone extended to do proper handling of every conceivable error condition. Occasionally, the programs may crash, especially if your input is wrong or when abnormal conditions arise. You'll have to live with that, or you must extend the programs.

You should not assume that we did everything right in the sample programs but if you are a newcomer to APPC programming, the sample programs should give you a good grasp of it. Don't be scared off by the complexity of the programs. They look complicated, but not because they handle SNA communications. The large majority of code lines are concerned with the other aspects of the application.

APPC Differences

Apart from learning what it's like to do APPC programming on an IBM RT, the sample programs give you a good opportunity to compare different environments. If you compare the IBM RT program rtrt.c to the OS/2 program OS2RT.C, you'll notice that the latter must take special actions when it wants to change the conversation state. In the IBM RT environment the applications programmer does not normally have to consider this.

There are advantages and disadvantages to this IBM RT way of doing things. However, when you make programs that talk LU 6.2 between IBM RTs it's generally an advantage.

APPC, Tested Environments

The following section will describe the customizing procedures for IBM RT communications using the sample LU 6.2 application *snaftp*. The described environments include the following:

- IBM RT to IBM RT communication via Token-Ring
- IBM RT to IBM RT communication via X.25
- IBM RT to IBM AS/400 communication via Token-Ring
- IBM RT to IBM AS/400 communication via SNA/SDLC line
- IBM RT to OS/2 communication via Token-Ring
- IBM RT to OS/2 communication via SNA/SDLC line.

For information on how to customize for X.25 communications, see "X.25 Communications" on page 309.

IBM RT to IBM RT Communication

To establish an LU 6.2 connection between two IBM RTs, the two systems must be customized. The following describes which steps are necessary in establishing the LU 6.2 connection and making the sample programs for file transfer work in this environment. The sample programs for the IBM RTs are listed in Appendix B, "LU 6.2 Sample Programs" on page 327.

1. A communications adapter must be installed in the IBM RTs, and device descriptions for the adapter and for the data link must be added to the systems using the *devices* command.
2. SNA Services must be configured on the IBM RTs.
3. A file transfer program must be written for the IBM RT. You need an initiating program (the program that initiates the file transfer) and a remote transaction program on the other IBM RT. In the sample programs the initiating program is called *snaftp* and the remote transaction program is called `RTRT`.
4. The physical connection between the systems must be established.
5. The attachment and connection must be activated.

The sample programs have been tested between two IBM RTs via Token-Ring and X.25 connections. They will run on an SNA/SDLC link if the SNA Services profiles are set up for this.

Customizing IBM RT SNA Services

Detailed listings of the SNA Services Profiles used to run the sample programs are shown in "Token-Ring Connection, RT PC - RT PC (LU 6.2)" on page 395.

Token-Ring LU 6.2 Connection

An overview of profile names is given in Figure 45 on page 89.

PROFILE TYPE:	PROFILE NAME (RT - RT)	DEFAULT PROFILE
Connection Profile	RTRT	CDEFAULT
Local Logical LU Profile Name	RTPC62	LDEFAULT
Mode List Name	RTPCM	MDEFAULT
Mode Profile Name	RTM	MDEFAULT
Mode Name	RT	
Local tpn List Name	RTLOCALLIST	TDEFAULT
Local tpn Profile Name	RTLOCAL	TDEFAULT
Local tpn Name	RTRT	tpn
Remote tpn List Name	RTRTREMOTELIST	RDEFAULT
Remote tpn Profile Name	RTRTREMOTE	RDEFAULT
Remote tpn Name	RTRT	rtpn
Attachment Profile	TRRTA	TDEFAULT
Control Point Profile	RTRTCP	CDEFAULT
Control Point Name	RTRTCP	
Data Link Profiles:		
Logical Link Type	TDEFAULT	TDEFAULT
Physical Link Type	TDEFAULT	TDEFAULT

Figure 45. SNA Services Profiles for RT-RT LU 6.2 Communication on Token-Ring. This figure gives a view of the profiles and the profile names used in the sample program setup for the Token-Ring LU 6.2 connection.

The two IBM RT LU names used in this setup are ***RTPC62B*** and ***RTPC62C***.

The following is a brief description of the major changes required to the default profiles in order to make the sample programs work. To change a default profile, copy it, give it a new name and then change it. For further information on customizing SNA Services refer to the *IBM RT SNA Services Guide and Reference* and "IBM RT SNA Services" on page 41.

- Data link profiles: Both the Physical- and Logical Link Profiles can be left as default for this setup.
- Control Point Profile: For the Token-Ring connection, an ***"XID Node ID"*** can be entered. In the sample setup we leave it at the default value, X'05C00000', but we could insert any value in the RTRTCP profile since SNA Services doesn't use this value for LU 6.2 communication between two IBM RTs. It can be quite convenient that the value is ignored this way since that allows you to have simultaneous connections from your local IBM RT to some other system type and to another IBM RT, even when the other system requires a special XID node ID.
- Attachment Profile: Two important parameters in this profile that must be set are ***call type*** and ***access routing***. The call type determines whether PLU-SLU (primary/secondary logical unit) will be negotiated or is fixed.
  - If you select a call type of "CALL" and access routing as "LINK ADDRESS" the field ***Remote Link Address Token-Ring*** must be filled in with the remote machine's adapter address. This can be the Token-Ring adapter burned-in address or a Token-Ring local administered address (LAA). On the IBM RT, an LAA can be configured with the *devices* command.

  - If you specify a call type of "CALL" and access routing as "LINK NAME" the field ***Remote Link Name Token-Ring*** must be filled in with the remote machine's node ID.
  - For a call type of "LISTEN" you don't need to identify the remote machine.

  If you select a call type of "LISTEN" you must start the attachment before you can process incoming allocates or execute an initiating transaction program. For a call type of "CALL" you must start the attachment before you can process incoming allocates, but you can initiate conversations directly from transaction programs without starting the attachment first.

- In the sample setup, the remote transaction program name contained in the Remote TPN Profile is called "*RTRT*" and is specified on the machine that initiates the file transfer with the `snaftp` program (or on both machines, if they both should be able to initiate a file transfer).
- The local transaction name is specified in the machine that receives a file transfer request from the other machine. Specify `RTRT` on both machines, if both should be able to receive a file transfer request.

  An important parameter in the Local TPN Profile is the ***maximum file Size***. If this is left to default, the largest file size that can be transferred is 51,200 bytes (100 blocks of 512 bytes). SNA Services will not warn you if this limit is reached so it should be set to the ***ulimit*** for the system. Type the command *ulimit* on the command line to see the current maximum file size the system can handle. Default value for ulimit is 8192 blocks.

- The mode name in the sample setup is `RT`.
- The Local Logical Unit Profile is used to describe the local LU by giving the LU a name and associating this LU with a Local TPN List Profile. In the sample setup, the two local LU names for the IBM RTs are **RTPC62B** and **RTPC62C**
- The Connection Profile is used to describe the remote LU, by entering its LU name and associating this remote LU with a Local LU Profile and a Remote TPN List.

When connecting two IBM RTs via Token-Ring, make sure that the SNA Services profiles on the two machines match, for example the local LU on one machine is the remote LU on the other machine. In the sample setup **RTPC62B** for one IBM RT and **RTPC62C** for the other IBM RT. Customizing the SNA Services Profiles for the two IBM RTs are almost identical procedures, so if the first IBM RT has been customized, you can save the profiles in a file using *snaconfig*. Use the *snaconfig PRINT* option and print the profiles to a file instead of a printer. Move the file to the other machine (using diskette, TCP/IP, or whichever method you prefer) and use the *snaconfig LOAD* option to load the profiles from the other machine. Then make the necessary changes.

X.25 LU 6.2 Connection

Complete profiles for LU 6.2 connection between two IBM RTs using X.25 are included in "X.25 Connection, RT PC - RT PC (LU 6.2)" on page 398. Figure 46 on page 91 gives an overview of profile names.

PROFILE TYPE:	PROFILE NAME (RT - RT)	DEFAULT PROFILE
Connection Profile	RTRT	CDEFAULT
Local Logical LU Profile Name	X25LOCAL	LDEFAULT
Mode List Name	RTPCM	MDEFAULT
Mode Profile Name	RTM	MDEFAULT
Mode Name	RT	
Local tpn List Name	RTRTLOCALLIST	TDEFAULT
Local tpn Profile Name	RTRTLOCAL	TDEFAULT
Local tpn Name	RTRT	tpn
Remote tpn List Name	RTRTREMOTELIST	RDEFAULT
Remote tpn Profile Name	RTRTREMOTE	RDEFAULT
Remote tpn Name	RTRT	rtpn
Attachment Profile	X25A	TDEFAULT
Control Point Profile	X25CONT	CDEFAULT
Control Point Name	RTPC1	
Data Link Profiles:		
Logical Link Type	X25L	TDEFAULT
Physical Link Type	X25P	TDEFAULT

Figure 46. SNA Services Profiles for RT-RT LU 6.2 Communication via X.25. This figure gives a view of the profiles and the profile names used in the sample program setup for the X.25 LU 6.2 connection.

The two IBM RT LU names used in this setup are ***RTPC2*** and ***RTPC1***.

The following is a brief description of the major changes required to the default profiles in order to make the sample programs work. To change a default profile, copy it, give it a new name and then change it. For further information on customizing SNA Services refer to the *IBM RT SNA Services Guide and Reference* and "IBM RT SNA Services" on page 41.

- Data link profiles: Both the Physical- and Logical Link Profiles had to be changed. Several options were changed, most noticably the ***Network User Address*** (NUA) for the local host.
- Control Point Profile: You should be able to use the default Control Point Profile for X.25 connections.
- Attachment Profile: You must specify the ***Network User Address*** (NUA) for the remote system, and you must select access routing as "LINK ADDRESS" if you select a station type of "primary" or "negotiable".

  If you select a station type of "secondary", you must start the attachment before you can process incoming allocates or execute an initiating transaction program. For a station type of "negotiable" or "primary" you must start the attachment before you can process incoming allocates, but you can initiate conversations directly from transaction programs without starting the attachment first.
- Profiles not mentioned above are identical to those for connection via Token-Ring.

IBM RT to AS/400 Communication

To establish an LU 6.2 connection between an IBM RT and an IBM AS/400 you need to customize both systems. The following describes the steps necessary to enable the LU 6.2 connection and making the sample programs for file transfer work in this environment. The sample program (AS400RT) for the IBM AS/400 is listed in "IBM AS/400 Remote Transaction Program" on page 362. The sample program for the IBM RT (snaftp) is listed in "IBM RT File Transfer Program (snaftp)" on page 327. The sample programs have been tested between the IBM RT and the IBM AS/400 via Token-Ring and SNA/SDLC connections.

The following steps are needed to enable LU 6.2 connections between an IBM RT and an IBM AS/400:

1. A communications adapter must be installed in the &rt, and device descriptions for this communication must be added to the system using the *devices* command.
2. SNA Services must be configured on the IBM RT.
3. A communications adapter must be installed in the IBM AS/400, and this communication must be added into the system using a ***line description*** and a ***controller description***.
4. A file transfer program must be written for the IBM RT. In the sample programs the initiating program on the IBM RT is called *snaftp.*
5. A file transfer program (remote transaction program) must be written for the IBM AS/400. The provided sample program is called AS400RT.
6. The physical connection between the systems must be established.
7. The attachment and connection must be activated.

Customizing IBM RT SNA Services

A detailed listing of the SNA Services profiles parameters used for running the sample programs against an IBM AS/400 are shown in Appendix C, "LU 6.2 Communication Profiles" on page 381.

Token-Ring LU 6.2 Connection

Figure 47 on page 93 gives a view of the profiles and the profile names used in the sample program setup for the Token-Ring connections to IBM AS/400.

```
PROFILE TYPE:             PROFILE NAME  DEFAULT PROFILE
                          (RT - IBM AS/400)
-------------------------------------------------------
Connection Profile Name   AS40062       CDEFAULT

Local Logical LU
Profile Name              RTPC62        LDEFAULT

Local Logical LU Name     RTPC62C

Mode List Name            RTPCM         MDEFAULT
Mode Profile Name         RTM           MDEFAULT
Mode Name                 RT

Local tpn List Name                     TDEFAULT
Local tpn Profile Name                  TDEFAULT
Local tpn Name                          tpn

Remote tpn List Name      AS400RTLIST   RDEFAULT
Remote tpn Profile Name   AS400RT       RDEFAULT
Remote tpn Name           AS400RT       rtpn

Attachment Profile        TRAS4A        TDEFAULT

Control Point Profile     RTAS400       CDEFAULT
Control Point Name        RTAS400

Data Link Profiles:
Logical Link Type         TDEFAULT      TDEFAULT
Physical Link Type        TDEFAULT      TDEFAULT
```

Figure 47. SNA Services Profiles for RT-IBM AS/400 LU 6.2 Communication on Token-Ring

The required changes to SNA Services default profiles are:

- Data link profiles: Both the Physical- and Logical Link Profiles can be left as default for this setup.
- Control Point Profile: For the Token-Ring connection, an ***"XID Node ID"*** must be entered. In the sample setup, X"05615177" is used in the RTAS400 Control Point profile (RT to IBM AS/400 connection). On the IBM AS/400 this parameter is in the line description EXCHID field.
- Attachment Profile: Two important parameters in this profile that need to be set, are **CALL type** and **access routing**. The CALL type determines whether PLU-SLU (primary/secondary logical unit) is to be negotiated or not. Access routing must be selected as "LINK ADDRESS", and ***Remote Link Address Token-Ring*** must be filled in with the remote machine's adapter address. This can be the Token-Ring adapter burned-in address or a Token-Ring local administered address (LAA). On the IBM RT, an LAA can be configured with the *devices* command.
- In the sample setup, the remote transaction name in the Remote TPN Profile is AS400RT.
- In the sample setup, the mode name in the Mode Profile is called RT.
- The Local LU Profile defines the the local LU name for the IBM RT as **RTPC62C.**
- The Connection Profile is used to describe the remote LU, by entering its LU name and associating this remote LU with a Local LU Profile and a Remote Transaction Profile List. In the sample setup, the IBM AS/400 remote LU name is **WTSCSL4.**

SNA/SDLC LU 6.2 Connection

Figure 48 gives a view of the profiles and the profile names used in the sample program setup for the SNA/SDLC connection to IBM AS/400.

PROFILE TYPE:	PROFILE NAME (RT - IBM AS/400)	DEFAULT PROFILE
Connection Profile	AS40062	CDEFAULT
Local Logical LU Profile Name	RTPC62	LDEFAULT
Local Logical LU Name	RTPC62B	
Mode List Name	RTPCM	MDEFAULT
Mode Profile Name	RTM	MDEFAULT
Mode Name	RT	
Remote tpn List Name	AS400RTLIST	RDEFAULT
Remote tpn Profile Name	AS400RT	RDEFAULT
Remote tpn Name	AS400RT	rtpn
Attachment Profile	SDLCAS4N	SDEFAULT
Control Point Profile	RTAS400	CDEFAULT
Control Point Name	RTAS400	
Data Link Profiles:		
Logical Link Type	SDLCAS4L	SDEFAULT
Physical Link Type	SDLCAS4P	RDEFAULT

Figure 48. SNA Services Profiles for RT-IBM AS/400 LU 6.2 Communication on SNA/SDLC Link

The following is a brief description of changes required to the default profiles for the sample programs. To change a default profile, just copy it, give it a new name and then change it. For further information on how to customize SNA Services, refer to *IBM RT SNA Services Guide and Reference* and "IBM RT SNA Services" on page 41.

- Data link profiles: Both the Physical- and Logical Link Profiles must be altered from the default. You must specify if the physical link is via a ***switched*** or a ***non-switched*** line and if the station type is to be secondary, primary or negotiable. In the sample setup, "non-switched" and "negotiable" are used.
- Control Point Profile: For a switched line, an ***"XID Node ID"*** must be entered. If you plan to use the same Control Point Profile for a Token-Ring connection and an SNA/SDLC connection to the IBM AS/400, an ***"XID node ID"*** must be entered since the link to the IBM RT is regarded as a switched line. In the sample setup, X'05615177' is used in the RTAS400 profile. On the IBM AS/400 this parameter is in the Line Description EXCHID field.
- Attachment Profile: Two important parameters in this profile are **station type** and **Remote Secondary Station Address**. If the IBM RT is to be Primary, the Remote Secondary address must be specified. If it is a point-to-point line, this is normally set to X'C1'. X"C1" equals decimal "193", which should be entered into the profile. If there is more than one attachment on the line, each attachment would need a unique address on the line.
- In the sample setup, the remote transaction name in the Remote TPN Profile is AS400RT.

- In the sample setup, the mode name in the Mode Profile is called RT.
- The Connection Profile is used to describe the remote LU and associate it with a Remote TPN List Profile. In the sample setup, the IBM AS/400 remote LU name is **WTSCSL4**.

Customizing the IBM AS/400

This section describes the customizing required on IBM AS/400 for LU 6.2 communications LU 6.2 via Token-Ring and SNA/SDLC. Detailed customization profiles for the IBM AS/400 are listed in "AS/400 Network and System Descriptions" on page 381.

The IBM AS/400 objects used during the tests are:

- Network attributes
- Mode description: RT
- Subsystem description: QCMN
- Job description: QGPL/QDFTJOBD
- Token-Ring network line description: RTAPPCTRL
- SDLC line description: RTAPPCNSL
- APPC controller on Token-Ring network: RTAPPCTRC
- APPC controller on TSDLC line: RTAPPCNSC.

The Network Attributes used to run the sample programs are included in "AS/400 Network and System Descriptions" on page 381.

Hints on Matching Parameters

These hints are based on practical experiences but may not be useful in every installation.

- IBM AS/400 is configured as a network node (APPN node type *NETNODE).
- IBM AS/400 local control point name must match IBM RT Remote LU name/CP name in the SNA Connection Profile.
- IBM AS/400 local network ID must match the IBM RT network name in the SNA Connection Profile.
- The mode descriptions and names must match those defined in the IBM RT Mode Profile.
- IBM AS/400 remote CP name (RMTCPNAME) must match the IBM RT Local LU Name specified in the IBM RT SNA Services Local LU profile.
- Subsystem description: The QCMN subsystem description is used.
- To be able to remotely start a transaction (program) on IBM AS/400 without specifying the name of the library where the program is stored, there must be a communication entry in the subsystem either by name or by type (*APPC) pointing explicitly to a job description (QDFTJOBD in the sample setup) that contains the name of the library where the remotely started program resides. The default value (*USRPRF) in the job description parameter of the communication entry can result in a message: "***Program Start Request Rejected***" with reason codes 605,1502 (program not found, procedure not found).

- Line and controller descriptions: A complete listing of the SDLC and Token-Ring line descriptions and controller descriptions used can be found in Appendix C, "LU 6.2 Communication Profiles" on page 381.
- If a Network ID other than the one contained in the IBM AS/400 network attributes is defined on the IBM RT then, the RMTNETID parameter in the IBM AS/400 controller description must match the IBM RT network name in the IBM RT SNA Connection Profile.

  Since IBM AS/400 is defined as APPN capable *YES in the controller description, the corresponding APPC device description is created by autoconfiguration after the controller is varied on and becomes ACTIVE and should not be manually configured.

The LU6.2 Application

A complete list of the programs and ICF file used in the sample setup can be found in "IBM AS/400 Remote Transaction Program" on page 362. The programs include comments that describe each step in detail. The programs and files used in the sample application are:

CRTRTICFF:

CL program used to create the RTICFF ICF file and perform an ADDPGMDEVE every time the ICF file is recreated.

RTICFF:

ICF file used to perform READs and WRITEs on the communication line.

CTLMSG record contains the layout of the control message received from and sent to the IBM RT.

RECDATA is used to receive and send data records. The largest record length expected to be sent or received is defined. VARLEN(fldlen) DDS keyword is used when sending data to assure that only the actual record length will be received.

Indicator 30 is turned on when Detach is received from the IBM RT, which corresponds to the IBM RT SNA Services system call ***"snadeal"*** with the parameter `DEAL_FLUSH (deal_str structure)` received from the IBM RT.

AS400RT RPG program:

This is the main program and performs the receive and send functions upon reception of a control message from the IBM RT indicating to do so.

ASRTTX CL program:

This program is called by AS400RT when receiving a send request (IBM AS/400 must receive a file) from the IBM RT. This program performs some syntax and object existence checking and builds the OVRDBF command to override the disk program file name to the name of the file that will be transmitted by the IBM RT.

ASRTRX CL program:

This program is called by *AS400RT* after receiving a receive request (IBM AS/400 must transmit a file) from the IBM RT. This program performs some syntax and object existence checking and builds the OVRDBF command to override the disk program file name to the name of the file that will be transmitted to the IBM RT.

The library list used by the target IBM AS/400 program depends on the job description used. In the simplest case (if all defaults are used), the communications entry used points to a job description ***(*USRPRF)***, which points to whatever is specified in the user profile of the user ID used ***(QUSER)***. The user profile points to the job description ***(QDFTJOBD)***, which points to the initial library list ***(*SYSVAL)*** that points to the system value ***QUSRLIBL***.

If you use a non-default setup, you must include *AS400RT* in whichever library your setup is using. In the default setup, *AS400RT* must be included in ***QUSRLIBL***.

Experiences connecting to IBM AS/400

The transfer of text files from the IBM RT to the IBM AS/400 involves the translation of ***tab*** characters from the value X'09' on the IBM RT to the defined EBCDIC standard X'05' on the IBM AS/400. When such text file is displayed on the IBM AS/400 the tab characters will be displayed as the character "▸", which is consistent with the behavior of DisplayWrite on the IBM AS/400. During the tests we did not find a way to make the IBM AS/400 expand the tabs automatically.

If you want the exact same looks of a text file after sending it to an IBM AS/400, you should expand the tabs in the IBM RT file locally prior to sending the file.

PC/DOS Communication

Advanced Program-to-Program Communication program for the IBM PC and PS/2 (APPC/PC) is a software package which supports the SNA application program interface (API) LU 6.2 (APPC) and node type PU 2.1. It allows program-to-program communication on IBM Token-Ring Network, SDLC and selected IBM PC Network communication links. Both mapped and basic conversions are supported.

No tests have been conducted with, and no sample programs written for, APPC/PC.

OS/2 Communication

There are various methods of connecting an OS/2 based machine into an AIX environment. One method is to use APPC/LU 6.2 communication. The sample program *snaftp* provided in this publication has been tested between an IBM RT running AIX/RT 2.2.1 and a PS/2 running OS/2 Extended Edition 1.1. Tests were made with both Token-Ring and SNA/SDLC connections. To run the sample programs you must:

- Customize SNA Services in the IBM RT.
- Customize OS/2 Communications Manager in the PS/2.

Sample SNA Services and OS/2 Communications Manager profiles are provided in this publication. Here is a list of where to find the profiles:

- See "Operating System/2 Common Profiles" on page 402, "Operating System/2 Token-Ring Profiles" on page 404 and "Operating System/2 SDLC Profiles" on page 407 for the OS/2 Communications Manager profiles.

- See "IBM RT Profiles for OS/2 Token-Ring" on page 409 and "IBM RT Profiles for OS/2 SDLC" on page 413 for the IBM RT SNA Services profiles.

  The *snaftp* sample program is listed in "IBM RT File Transfer Program (snaftp)" on page 327.

  The OS/2 remote transaction program *OS2RT* is listed in "Operating System/2 Remote Transaction Program" on page 369.

Reference Publications for This Chapter

IBM RT SNA Services are described in:

IBM RT SNA Services Guide and Reference, SC23-2009

Other related publications are:

Systems Network Architecture, Transaction Programmer's Reference Manual For LU Type 6.2, SC30-3084
Systems Network Architecture, Architecture Logic For LU Type 6.2, SC30-3269
An Introduction to Advanced Program-to-Program Communication (APPC), GG24-1584

3270 Emulation and Remote Job Entry

The vast majority of users of IBM System/370 mainframe computers are accessing the mainframe by way of a ***3270-type*** terminal. The term "3270-type terminal" refers to two main series of IBM terminals, the (original) IBM 3270 terminals and the (newer) IBM 3170 terminals. The primary terminals of the 3270 series are the IBM 3278 (monochrome) and the IBM 3279 (color) terminals. The basic functions of these terminals are supported by all terminals in both series and constitute the minimum subset of functionality for 3270-type terminals.

The newer IBM 3170 terminals all provide the basic subset of functions but have different extended functions. The IBM 3180 has extended functions like scrolling, the 3192 is a follow on of the IBM 3180 with enhancements like printer port and setup mode. The 3193 is a high resolution monochrome terminal for image display with image data compression. These enhanced functions are ***not*** supported by the terminal emulation products for AIX.

The family of terminals is often referred to as "IBM 327x" terminals, or simply as "3270's". The 3278 and 3279 terminals have no built-in processing capability. They can't be attached directly by cable to a mainframe computer, but need a terminal controller at the end of the telecommunication line or the channel to manage the data traffic to and from the computer.

Terminal controllers are computers in themselves and are often called ***cluster controllers*** because they control a cluster of terminals. These controllers are commonly known by the family name "3x74". Older controllers are of the IBM 3274 series; newer ones of the IBM 3174 series. Both series come in a great variety of models with different capabilities. The terminal controllers in a typical SNA network are connected remotely via a communication link to a 37xx communication controller or are locally attached to a System/370 channel. A 3x74 is connected at the remote or terminal end of the link and manages a cluster of 3270 terminals attached to that line. It acts as a multiplexor for the terminals.

Communication Controllers

A 37xx is commonly located at the host end of the communication link, and one or more will manage all of the lines connected to that host. These controllers are used by the host to handle all the terminal I/O on behalf of the host. We use the notion ***37xx communication controller*** to refer to a family of communication controllers known as the IBM 3705, IBM 3720, IBM 3725 and IBM 3745. They can connect to asynchonous, BSC, SDLC, X.25, X.21 and Token-Ring lines. The oldest model (the IBM 3705) does not support Token-Ring connections.

The 37xx communication controllers are programmable units and need a ***control program*** to perform their functions. In most cases, the control program is the ***Network Control Program*** (NCP) which can control BSC and SDLC lines. Other control programs are used for other link types: For X.25 the Network Packet Switching Interface (NPSI) program is used and for Token-Ring the NCP Token-Ring Interface (NTRI) is used. The 37xx is normally connected to the host via a /370 channel, but can also be connected remotely to another 37xx, depending on what model you use.

Integrated Adapters

Some host systems, for example the IBM 9370, offer the capability to directly attach the terminals to a so-called ***integrated adapter***. Terminals connected to integrated adapters appear to be directly attached to the host via coaxial cables. However, those configurations still can be viewed as if the terminals are attached to a control unit; the function of the control unit is "hidden" in the integrated adapter and the microcode. We will not cover those integrated ports or channel attached control units in this chapter; we will concentrate on remote connections.

Communications Software

At the "host" end of the communications line, software controls the network of terminals connected to the host. The 37xx communications controller runs a control program, most likely the Network Control Program (NCP). This hardware/software combination will act as an interface between the host computer and the one or many communications lines attached to it. On the host computer runs a program that controls all the physical and logical units in its domain. In a typical SNA network (see Figure 49) this program is the Virtual Telecommunication Access Method (VTAM). It has the function of the system services control point (SSCP) in its domain.

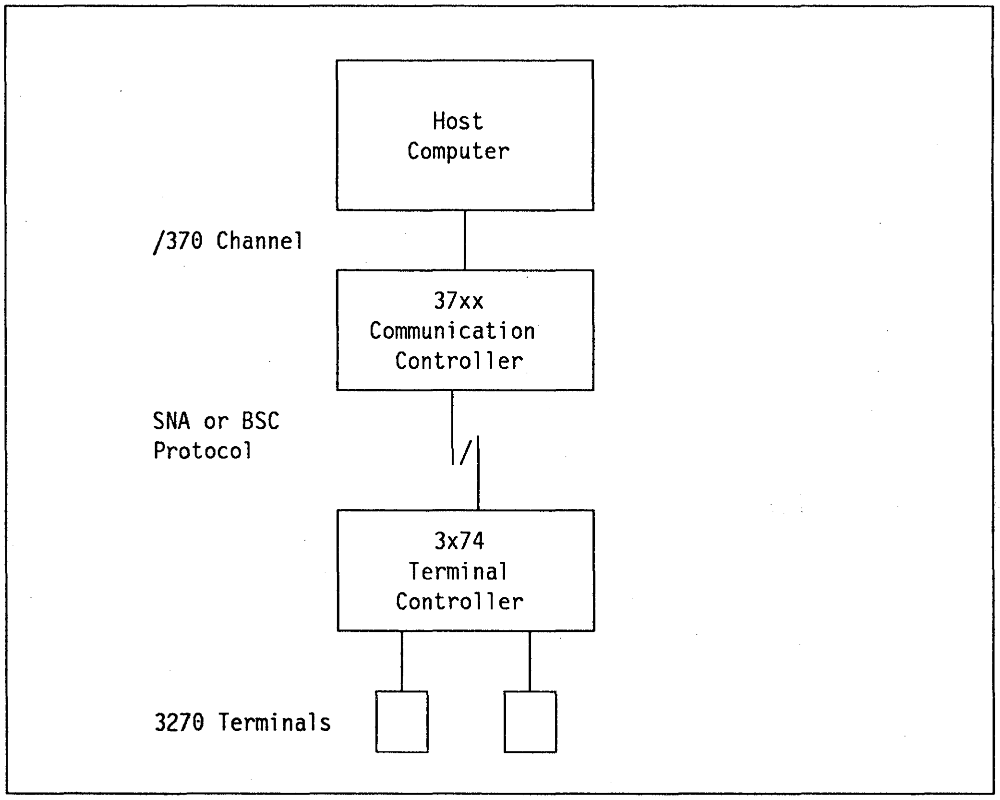


Figure 49. Typical Host to Terminal Configuration

The protocol governing the form of the data stream on the communications line will be either SNA or BSC; the significance of this will be explained later.

Moving Towards Terminal Emulation

Large numbers of computer configurations of this type, with variations in hardware and software, have been installed for many years. In more recent times, users have requested facilities that are not provided by a 3270-terminal. Many, for example, have installed small computers in locations remote from the mainframe computer. They often wish to use data stored on the mainframe as input to applications running on the smaller, local computer.

It is often expensive and time-consuming to change the hardware or software of a mainframe computer, but relatively cheap and simple to make a small computer behave like a terminal type already recognized by the mainframe. This has given rise to various forms of ***terminal emulation***, where the local computer can generate or receive a stream of data in such a way that the mainframe's communications controller is unaware that the device at the other end of the line is not a known terminal. In its simplest form, terminal emulation works in ***control unit terminal (CUT)*** mode, where the computer emulates the essential characteristics of the terminal it is replacing, and continues to rely on the nearby terminal controller for screen formatting and controlling the data stream on the communications line.

The presence of a computer replacing the terminal raises the possibility that it takes over some of the functions of the terminal controller as well. For example, rather than the computer emulating just one terminal, it may emulate several terminals (or have several sessions with the host), but still rely on the terminal controller to communicate with the host. This is known as ***distributed function terminal (DFT)*** mode, but is limited to a maximum of 5 sessions by the 3x74 microcode.

Finally, the local computer may replace the terminal controller with all its terminals and printers altogether. This is known as controller emulation. In all these types of emulation, the local computer may also provide additional features over the original terminal such as retrieval from and transmission to the host of relatively large volumes of data (file transfers).

The IBM RT may be used for any one of these types of terminal emulation.

Remote Job Entry

Some terminals are not confined to keyboards and displays. Before the advent of the interactive computing environment we are used to today, IBM introduced the IBM 2780, IBM 3780 and IBM 3770 ***remote job entry (RJE)*** terminals. These terminals consisted essentially of a cluster of card readers, card punches, printers, etc. which enabled the user to submit a "job" over a communications line for remote (batch) processing on a mainframe computer. The later 3770-series terminals also included a console so batch jobs on the host could be initiated from the terminal. The protocols used by these terminals became a de-facto standard for RJE, and is still used today.

The arrangement for connecting RJE terminals to an IBM host is similar to that of 3270-terminals, except that there is no need for a terminal controller (3x74) although a terminal controller may be used. Figure 50 on page 102 illustrates a simple RJE communications arrangement.

As you can see, a small computer, with appropriate software, could provide all the facilities needed to perform this function. Software is available which enables the IBM RT to emulate all of the functions described above. This will be described in later chapters.

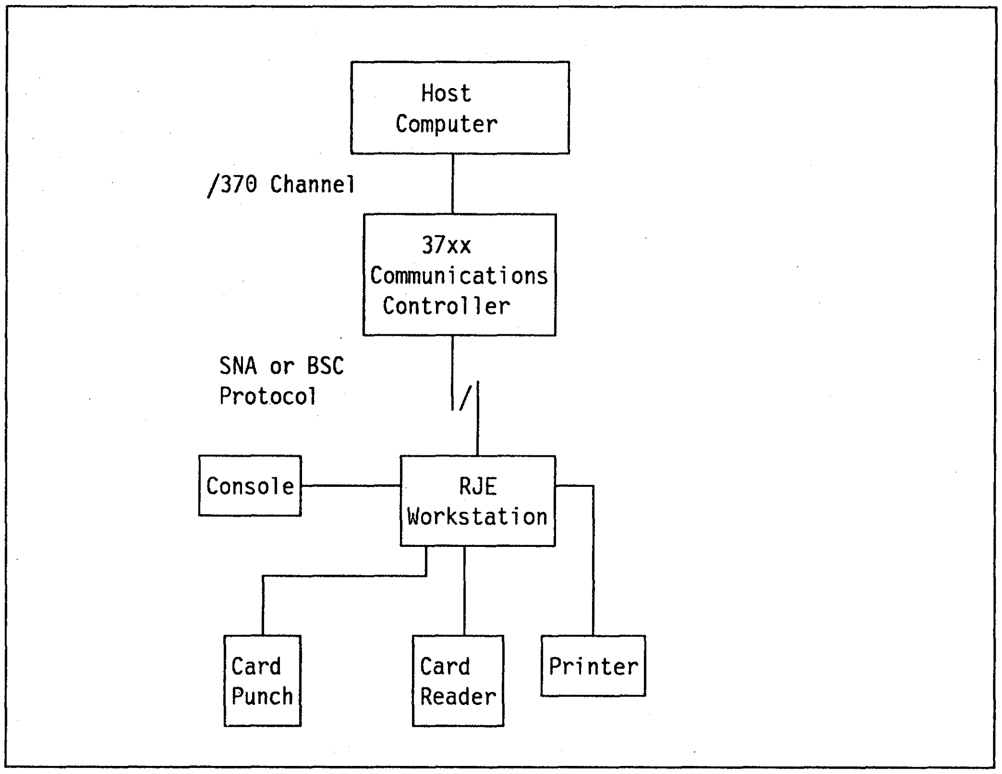


Figure 50. A Simple Host to RJE Configuration

Preparing For Terminal Emulation

Just as a "terminal" has to be defined to the host computer, the installation of an IBM RT for terminal emulation requires that the local (software) configuration of the IBM RT conforms with the definitions of the network at the host site. When you want to install one of the IBM RT terminal emulation or RJE products, therefore, it is essential that the team responsible for administering the host computer are involved at an early stage. More specifically, the people that you need to contact at the host site are:

- ***A network administrator*** - to provide information about the network, and to help with problem diagnosis if necessary.
- ***A systems programmer*** - to provide information about the host itself, and about the batch systems if you are connecting an RJE workstation.

The information you need to get from these people to start with is:

- The types of modem supported on your line.
- A host definition listing for your line. On an SNA network this will be a ***VTAM listing*** (see "VTAM" on page 43 for details). On a BSC network it may be called by any of several names depending on the host (see "BSC Host Network Control Programs" on page 45 for details).
- The ***SSCP ID***[10] of the host that you are communicating with. This is a number that identifies a particular host on the network and is usually available from a systems programmer.

To help you understand what options are available when communicating with a host and to understand the information given to you by the people administering the host, the following sections will go into a little more detail about the host environment.

Host Subsystems

The term "host subsystems" is often used to refer to the individual components of the software running on the host computer. A single host may run several subsystems which often have different characteristics. For example, a single machine may run both an interactive and a batch subsystem at the same time. Examples of the types of subsystems that may be active on a host include:

Interactive This type of subsystem is designed to directly communicate with users in "real time". Examples of interactive subsystems include Time Sharing Option (TSO) under MVS and Conversational Monitor System (CMS) under VM. These subsystems provide ***command interpreters*** that allow users to enter commands at a terminal and to receive the output from these commands at their displays as the commands execute.

Batch This type of subsystem accepts jobs to be executed without the need for an immediate response to the user's interactive display (batch processing). Jobs can be executed at a time convenient from a system point of view, perhaps when demand for the host machine is not so high. Examples of batch host subsystems include the Job Entry Systems (JES2 and JES3) under the MVS operating system.

Transaction The third type of host subsystem is the transaction processing system. This subsystem type is interactive in the sense that responses to user transactions are usually returned to the user terminal immediately. They differ from the TSO and CMS substems in that the user communicates with application programs rather than with command interpreters. Examples of such subsystems are the Customer Information Control System (CICS) and Information Management System (IMS).

Others Other subsystems, which may not be visible to the average user, may also be operating within a host. Such subsystems may be "system" functions. VTAM (Software supporting a communications network), for example, is supported as a host subsystem and is used to connect terminals to one of the other subsystems.

Each host subsystem is accessible to the end user through an application identification, often referred to as the "***APPLID***" of that subsystem. If multiple host subsystems are to be accessed, then each host application's identification must be established and noted. Some networks have a "front-end" menu system which simplifies application selection at the time communication is established.

[10] This only applies for SNA networks.

Sessions

The term ***session*** is used to describe the logical linking of the user's terminal with the host application program (host subsystem). A fairly simple concept when a single terminal accesses a single application, but considerably more complex when the "terminal" is a local computer, capable of emulating a number of terminals having concurrent access to a number of host subsystems. Each eligible session must be defined to the host as though it were utilizing its own terminal.

Multiple Active Sessions

Terminal emulators typically provide more functions than are available to users of the terminals they are emulating, simply because of the capabilities of the computer on which the emulator software is running. It is therefore appropriate to define the sessions to be emulated differently from the way you define "real" IBM 3270 terminals.

For example, the additional functions provided by Network 3270-PLUS include:

Multi Session Up to six host sessions can run on each physical display running Network 3270-PLUS (including ASCII displays).

Printing Host printer output can be directed as follows:

- Spooled output: One physical printer on the IBM RT can be used for both local and host print.
- Dedicated output: One physical printer on the IBM RT can be dedicated to serve one host printer. This is useful for security reasons where a single printer might need to be secured in a locked room.
- File output: The output from a host printer can be directed to an ordinary AIX file.

Example Scenario: The above can be illustrated by assuming a case in which an IBM RT is to be used by a number of persons, three of which require access to the host system. The person using the console might wish to utilize the multi-session feature to its full extent, having six sessions active. Users with ASCII displays could also have six sessions each, but two of each will suffice for this example. Thus a total of 10 host display sessions would be needed (6+2+2).

Assuming that:

1. The IBM RT has two physical printers, and that Printer 1 is used by the AIX printer queuing system (the spooler), and Printer 2 is to be dedicated for confidential output from the host.
2. The three users requiring access to the mainframe plan to use applications under three different host subsystems, X, Y and Z.

The printer definitions would need to be something like this:

PRT_X Virtual printer for use by subsystem X, output sent to the AIX print spool for Printer 1.

PRT_Y Virtual printer for use by subsystem Y, output sent to the AIX print spool for Printer1.

PRT_Z	Virtual printer for use by subsystem Z, output sent to the AIX print spool for Printer 1.
PRT_A	Virtual printer for use by user A, output sent to an AIX file. For example: `/u/A/output`
PRT_B	Virtual printer for use by user B, output sent to an AIX file. For example: `/u/B/output`
PRT_C	Virtual printer for use by user C, output sent to an AIX file. For example: `/u/C/output`
PRT_conf	Confidential host output directed to a dedicated printer (Printer 2 in this case).

The total need for host-defined printers would be seven, even though there are only two "real" printers on the IBM RT. The above scenario would need ten host-defined terminal sessions and seven host-defined printer sessions.

Communication Lines

This section will concentrate on the characteristics of the physical link between the host site and the local IBM RT. There are many characteristics that can be associated with such a link; the ones which directly affect the installation and setup of the IBM RT are given below. First, terminals may communicate with a mainframe using a number of different types of link:

Telephone lines

The most common way to connect terminals to a mainframe is with a telephone line available from your telephone supplier. These lines come in two forms:

- A Switched Line, where the communications path is built over a publicly available network each time the link is established (by using an ordinary telephone). Other users may establish paths across the same network, and the quality of the lines may vary.
- A Leased Line, which is permanently connected (for all practical purposes), and the quality of the link may be guaranteed by the organization providing the service. There are two main types of leased lines available:
  - Analogue lines use ordinary telephone wires and are used for relatively low speed communications.
  - Digital or X.21 lines are higher quality lines and can be used at higher speeds.

  Leased lines are preferred where the transmission of large volumes of data, or links of long duration are required, as they are more reliable.

Token-Ring Networks

If the terminals and the mainframe are in the same building, the terminals may attach to the mainframe using a Token-Ring local area network. This gives extremely high quality and high speed connections. By using Token-Ring bridges, Token-Ring networks can be extended beyond the capabilities of a single ring, either through local bridges to nearby rings, or through ***split bridges*** to remote Token-Ring networks.

X.25 Networks
In some cases terminals may be connected to the mainframe using X.25 Networks. These are high quality networks that use a special protocol and are used exclusively for transferring computer data. See "X.25 Communications" on page 309 for more details about X.25.

Multidropped Links
Where a leased line is used, terminal emulation programs allow the line to be shared, by assigning each machine on the line a unique address. This is called the polling address, and one is assigned for each terminal or other device. Even if only one device is using the line, it must still have an address assigned. This is usually hexadecimal X'C1' in the case of a single SNA device, and X'40' for a BSC device.

Other considerations include:

Line Speed
The line speed of a connection is the rate at which data is sent along the connection, expressed in bits per second (bps). In theory, the greater the line speed, the better the network response time. However, the maximum line speed possible may be limited in practical terms by such factors as the quality of the line and modem in use. Typically, switched lines support communications up to 2400 bps, analogue leased lines will support up to 19200 bps, digital X.21 leased lines support up to 64,000 bps (though not available in all countries) and Token-Ring networks run at 4,000,000 or 16,000,000 bps.

Modems
If not connected by coaxial cable to a 3x74 terminal controller, a suitable modem must be connected between the emulated terminal, or terminal cluster, and the communications line. With digital lines or X.25 networks the network supplier will typically supply the modem. The selection of suitable modems is beyond the scope of this document, but note that the modems chosen at either end of the communications line must be compatible and support the same speed of communications.

Protocols
Earlier in this chapter, the general IBM terminal/host environment was described, with a note that either an SNA or BSC protocol would be employed. Both protocols are supported by the emulation products discussed in this document. The significant implications of each are discussed in the chapters which follow.

IBM 3270 Emulation Products

The following chapters describe the available products for IBM 3270 emulation and RJE:

"IBM RT 3278/79 Emulation Program" on page 107
"Workstation Host Interface Program" on page 111
"Network PLUS" on page 131

The TCP/IP *telnet* command also provides IBM 3270 emulation. For details see several sections of the chapter "Transmission Control Protocol/Internet Protocol (TCP/IP)" on page 165.

IBM RT 3278/79 Emulation Program

The ***IBM RT 3278/79 Emulation Program*** enables the IBM RT to emulate either an IBM 3278 Display Model 2 or an IBM 3279 Color Display Model 2A or S2A from the IBM RT display screen and keyboard. The *IBM RT 3278/79 Emulation Program* also provides a means of transferring files between a host computer and the IBM RT. A single terminal is emulated on the IBM RT console, and the terminal session may be run in a virtual terminal on the console display. A subshell for execution of AIX commands may be invoked from the emulator session. The subshell can be used to run AIX commands and must be used to invoke the file transfer programs (*emrcv* and *emsend*).

System/370 Connection

Hardware and Software Requirements

- The 3278/79 Adapter must be installed in an appropiate slot in the IBM RT.
- The IBM RT must be connected to an IBM 3x74 Cluster Control Unit, a 4361 Display/Printer Adapter, a 4361 Workstation Adapter or a 9370 Workstation Adapter.
- The IBM IBM RT AIX Operating System Licensed Program, Version 1.1 or higher must be installed on the IBM RT.
- For file transfer operation a "Host File Transfer Program" is required.
  - For VM/CMS: IBM 3270 File Transfer Program, 5664-281.
  - For MVS/TSO: IBM 3270 File Transfer Program, 5665-311.
  - Updates - Contact your local support center and request an update tape that contains PTF UR90118 for VM/CMS or PTF UR20455 for MVS/TSO.

Installing and Customizing

The *IBM RT 3278/79 Emulation Program* is configured for National Language Support (NLS) during the installation process. You will be prompted to select one of the 16 supported languages. This is normally the only customizing required for the IBM RT 3278/79 Emulation Program. The language selected when installing the program must match the language selected when VRM was installed. The installation procedure for the *IBM RT 3278/79 Emulation Program* rebuilds the AIX kernel and reboots the IBM RT.

Using the IBM RT 3278/79 Emulation Program

The following is a brief summary of the use and capabilities of the IBM RT 3278/79 Emulation Program:

- The emulator is started with the command ***em78*** or *open em78* from the IBM RT console. The *em78* command takes several flags; use `em78 -?` to display help information about the command line flags.
- The subshell function is invoked by pressing `CTRL-C` on the keyboard. This gives an AIX shell with a modified prompt: "***emshell*** >". To return to the 3270 emulation sessions, press `CTRL-D` or enter the command *exit* and press the `ENTER` key.

- The ability to capture a host screen and save it in an AIX file or send it to a printer (the "-r" or "-s" flags). These functions are invoked by pressing either CTRL-ScrLck or CTRL-PrtScr once the emulator is started. The CTRL-ScrLck sequence replaces the print file with the new screen image. The CTRL-PrtScr sequence appends the new screen image to the print file.
- The ability to transfer a file to/from a System/370 host running either VM/CMS or MVS/TSO and with the Host File Transfer Program installed (IND$FILE). (The *emsend* and *emrcv* commands).

  If the IBM RT is connected to an IBM 3174 Cluster Controller or to an IBM 9370 Workstation Adapter, a ***file transfer bit*** must be set in the 3174/9370 WSA microcode. This is because the IBM RT *IBM RT 3278/79 Emulation Program* emulates a CUT-Type terminal (Control Unit Terminal). This is question 125 in the microcode generation, bit number 6. This bit does not need to be set for DFT-Type terminals (Distributed Function Terminal) that are used for host file transfer.
- The ability to specify a program to be executed every time the subshell is invoked (the "-c" flag).
- The ability to redefine the default keyboard and color layout in a profile and use this profile when the emulator is invoked (the "-k" flag). For example, if it is desired only to press one key to do a **reset** on the 3270 emulation screen (rather than the default: ALT-ESC), this can be done in the following way:
  - Take a copy of the /usr/lib/em78/emdefs.p file to the home directory.
  - Modify this profile. The following maps the Scroll Lock key to do the "reset" function. The line in the emdefs.p file says:

    ```
    k125        k32_72        reset
    ```
  - Run the utility ***emkey***. This utility takes the input file (default: emdefs.p) and creates an output file (default: emkeys.o).
  - Start the emulator with the "-k" flag. For example:

    ```
    em78 -kemkeys.o
    ```
- The *IBM RT 3278/79 Emulation Program* Version 1.1.1 has a file transfer application program interface (API). This enables a programmer to write a program that integrates the file transfer utilities into its structure. Then, file transfer can be invoked directly from the application. This application could, for example, include a new user interface for the file transfer functions.

To stop the emulator and remove it from memory, press CTRL-D twice at the 3270 session screen. This will also remove the lock files that the program keeps in /etc/locks. These files are: /etc/locks/em78 and /etc/locks/xfer.

Using TCP/IP telnet

With the TCP/IP command ***telnet***, it is possible to do a remote login from one IBM RT on a LAN to another IBM RT on the LAN. Invoking *em78* in a telnet session is not described in the documentation for the product and not officially supported, but it works. It is, in fact, possible to establish a telnet session from one IBM RT console (not from an ASCII terminal) to another IBM RT on the LAN, then login as a normal user and invoke the emulator. As this has not been through a formal test, this information is given "as is, no warranty", should you ever want to try. Remember that only ***one*** session at a time can use the *IBM RT 3278/79 Emulation Program* and only from a ***High Function Terminal*** (HFT).

IBM AS/400 Connection

The IBM AS/400 supports attachment of the 3x74 SNA/SDLC Cluster Controllers using the *3270RA* (Remote Attach) programs on the IBM AS/400. Using this facility, an IBM RT running the *IBM RT 3278/79 Emulation Program* can be connected to an IBM 3x74 that is connected via an SNA/SDLC link to an IBM AS/400.

The IBM AS/400 does not have a file transfer program for 3270-attached terminals (like the System/370 IND$FILE program), so file transfer is not possible. File transfer to the IBM AS/400 is discussed in "IBM RT to AS/400 Communication" on page 92.

Reference Publications for This Chapter

The IBM RT 3278/79 Emulation Program is described in the publication:

IBM RT 3278/79 Emulation Program, P/N 84X0681 (SV21-8032)

Workstation Host Interface Program

The ***AIX Workstation Host Interface Program*** is designed to allow 3270 terminal emulation and file transfers in a ***non-SNA*** environment. Originally designed to utilize the IBM System/370 Host Interface Adapter on the IBM RT, Workstation Host Interface Program now also supports connections from AIX/RT and AIX PS/2 via 3278/79 Adapters.

Overview

The *AIX Workstation Host Interface Program* (WHIP) supports IBM 3278 and IBM 3279 terminal emulation from AIX/RT or AIX PS/2 consoles and attached ASCII terminals. WHIP provides 3270 Emulation in ***DFT*** (Distributed Function Terminal) mode between an AIX system and an IBM System/370 host via the Advanced 3278/79 Emulation Adapter in the IBM RT or the ***3270 Connection Adapter*** in the IBM Personal System/2. In an AIX/RT environment there is additional support for a connection via the ***IBM 5088 Communications Controller*** and the ***IBM System/370 Host Interface Adapter*** installed in the IBM RT. Figure 51 gives a schematic view of the hardware configuration.

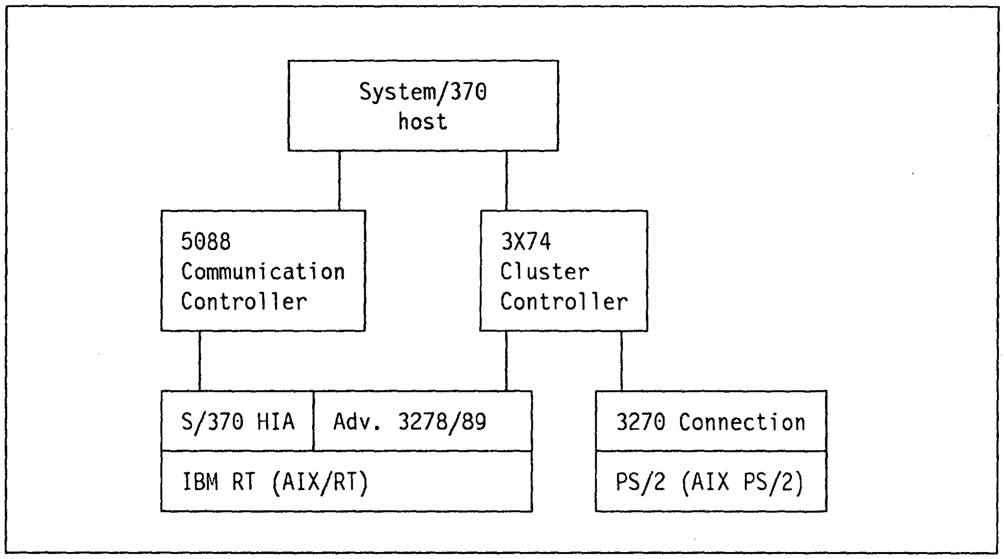


Figure 51. WHIP Configurations. WHIP connections to a System/370 host from AIX/RT and AIX PS/2.

WHIP Prerequisites and Dependencies

Many components must interact when using WHIP. We shall go through the requirements and dependencies in the following order:

- IBM RT Hardware Prerequisites
- IBM Personal System/2 Hardware Prerequisites
- Software Prerequisites
- Host Operating System Requirements
- Requirements in the Terminal Control Unit

IBM RT Hardware Prerequisites

Adapter One or both of the following adapters:

- An *IBM System/370 Host Interface Adapter* (HIA) and necessary cables for attachment to the *IBM 5088 Graphics Channel Control Unit* (GCCU) with the *3270 MSA Feature* installed, or an *IBM 5088 Remote Cluster Controller Model 1R or 11R*.
- An *Advanced 3278/79 Emulation Adapter* and cable for attachment to an IBM 3174 or IBM 3274 Cluster Controller or a 9370 Workstation Adapter. Local non-SNA and remote BSC communication links are supported.

Monitor IBM 6153, IBM 6154, IBM 6155, IBM 5151, IBM 5154 or IBM 5081 Display.

Terminal IBM 3161 in 3161 mode, IBM 3162 in 3161 mode, IBM 3163 in 3161 mode, IBM 3151 in 3161 mode or DEC VT220.

Printer For IBM 3287 printer emulation you can use IBM 4201 Proprinter or IBM 3852 Color Jet Printer. For screen print, any printer supported by AIX/RT can be used.

IBM Personal System/2 Hardware Prerequisites

Adapter *IBM 3270 Connection Adapter* and appropiate cable for attachment to an IBM 3174 or IBM 3274 Cluster Controller or a 9370 Workstation Adapter.

Monitor IBM 8503, IBM 8512, IBM 8513, IBM 8514 or IBM 5154 Display.

Terminal IBM 3161 in 3161 mode, IBM 3162 in 3161 mode, IBM 3163 in 3161 mode, IBM 3151 in 3161 mode or DEC VT220.

Printer Any printer supported by the AIX PS/2 Operating System.

Software Prerequisites

AIX/RT: IBM AIX/RT Version 2.2.1 or subsequent releases. For AIX/RT 2.2.1 you should install the latest update to WHIP. Request the update from your local support center.

AIX PS/2: IBM AIX PS/2 Version 1.1 or subsequent releases.

AIX options: Administrative support is required for both AIX/RT and AIX PS/2. If you intend to use the application program interface under AIX PS/2, the extended programming support and text services support options of the AIX Extended Services are required prior to the installation of WHIP. For AIX PS/2, the development toolkit and one of the compilers supported by the API must be installed. If you intend to use telnet for remote access to WHIP, TCP/IP must be installed and customized.

S/370 Host: The IBM 3270 File Transfer Program *(IND$FILE)*:

- For VM/CMS: IBM 3270 File Transfer Program, 5664-281.
- For MVS/TSO: IBM 3270 File Transfer Program, 5665-311.
- Updates - Contact your local support center and request an update tape that contains PTF UR90118 for VM/CMS or PTF UR20455 for MVS/TSO.

The file transfer program with the updates applied is much faster than the original program. In some cases, up to four times as fast.

Note: The minimum VM level required on the host to allow a file transfer via *pvm*, is service level 417 for VM/SP4 and VM/HPO 4.2, and service level 503 for VM/SP5.

Host Operating System and API: The application program interface is designed to operate in the following host environments:

- TSO/E (P/N 5665-285) under the following operating system software:
  - MVS/SP-JES2 Version 1.3 or 2 (5740-XYS or 5740-XC6)
  - MVS/SP-JES3 Version 1.3 or 2 (5740-XYN or 5665-291)
  - MVS/XA Data Facility Product Version 2.1 or later (5665-284).

  The TSO Assembler Prompter (5734-CP2) must be installed on the TSO host to support the application program interface (API).
- CMS under the following operating system software:
  - VM/SP Version 3 or subsequent releases (5664-167)
  - VM/SP HPO Version 3 (5664-173)
  - VM/VTAM Version 3 or subsequent releases (5664-280/A)
  - VM/XA SP Release 2.

Host Operating System Requirements

The following section is provided as an aid to host system programmers in determining additional host configuration parameters that may be required to support WHIP.

Requirements in MVS/TSO and VM/CMS

MVS I/O SYSGEN and ***VM nucleus DMKRIO I/O*** definition requirements are as follows:

The AIX system must be defined as an IBM 3278 or IBM 3279 display device. To MVS in the IODEVICE macro; to VM in the RDEVICE macro. There is no difference between the IODEVICE or RDEVICE parameters for an AIX system with WHIP and a regular 3278/79 display terminal. Sample IODEVICE and RDEVICE definitions of an AIX system as an IBM 3278 and IBM 3279 display terminal are listed in Figure 226 on page 517 and Figure 227 on page 517.

The 3x74 or IBM 5088 controller must be defined to the host operating system as an IBM 3274 cluster controller. A sample VM RCTLUNIT definition of a 16-port IBM 5088 Model 1 and a 32-port 3274 is shown in Figure 228 on page 517.

Requirements in MVS and VM VTAM Version 3

The AIX system must be defined in a LOCAL statement in a VTAM major node. There is no difference between the parameters in the LOCAL statement for an AIX system with WHIP and a 3278/79 terminal. The LOCAL statement in the local non-SNA major node contains a MODETAB parameter. This parameter must be the name of a valid VTAM logon mode table. The logon mode table consists of one or more MODEENT macros, each with a different name specified by the LOGMODE parameter and each specifying a different set of session parameters, for example, the screen size for the session.

A VTAM local non-SNA major node definition containing LOCAL statement definitions for four AIX systems, two as 3278/79 Model 2's, and two as 3278/79 Model 3's are listed in Figure 229 on page 518.

The IBM-supplied default logon mode table is named ISTINCLM. An example of an alternative logon mode table is shown in Figure 230 on page 519. In the MODEENT macro, it is important to set bit 8 of the PSERVIC (Presentation Services) operand to "ON" (the ***Query-bit***), if file transfer support is required. For example:

```
        MODEENT ...,PSERVIC='0080...,
or
        MODEENT ...,PSERVIC='00C0...,
```

The bufsize parameter for the VM/VTAM IOBUF storage pool must be increased from 64 bytes to at least 256, 285, or 286 bytes to run file transfer operations and API applications efficiently and without problems on the AIX system. The IOBUF bufsize is the size in bytes of each buffer in the IOBUF storage pool.

WHIP comes with an API sample program, called ***g32_sampl***, that transfers one megabyte of memory to and from the host in 1K, 2K, 4K, 8K, 16K, and 32K messages. Experiences show that *g32_sampl* terminates abnormally when the message size reaches 8K, if the IOBUF setting of bufsize is 64 bytes. You can run the original *IND$FILE* with its message size of 2K without any problems when the bufsize is 64 bytes. However, the updated *IND$FILE* will use a message size of up to 16K, and therefore can cause a problem if bufsize is left at 64 bytes.

It should be noted that the user running WHIP can change the WHIP message size to accommodate the host configuration. The environment variable, ***H3270MAXBUF***, specifies the maximum message size allowed by the host for both file transfer and API applications.

The presence of SDLC or BSC lines coming into VTAM through a communications adapter places additional requirements on bufsize. Here is a list of the minimum recommended VM/VTAM IOBUF bufsize with and without communication adapters:

Minimum IOBUF bufsize	Type of communication
256	no SDLC or BSC lines
256	SDLC lines
285	BSC lines
286	BSC lines and channel-attached SNA devices

Requirements in the Cluster Control Unit

Some controller customization is required the AIX system running the WHIP 3278/79 emulator, the WHIP 3270 file transfer or the WHIP API:

- 3274 Considerations:
  - The 3274 must be customized to include DFT support.
  - The 3274 must have Microcode Configuration Support-D.

  - If the AIX system uses extended data stream[11] processing, respond with "1" to customization question 160.

- If the AIX system is to support multiple emulator sessions, the controller must be customized for multiple interactive sessions (DFT), which is currently done when answering questions 116 and 117.

- An AIX system that emulates a DFT type terminal does not require a downstream microcode load diskette (DSL) in the controller.

- IBM 3174 and IBM 9370WSA: You do ***not*** need to set bit 6 ON (file transfer bit) in the microcode customization question 125 (Miscellaneous Feature Options). This bit should only be set to ON, if CUT (Control Unit Terminal) type terminals on the same controller want to do file transfer. WHIP emulates a DFT type terminal.

The IBM 5088 controller requirements for the IBM RT running the WHIP 3270 emulator, the WHIP 3270 file transfer or the WHIP API are:

1. The maximum and minimum control unit channel addresses of IBM RT's attached to the IBM 5088 Model 1 or Model 2 must be set in the IBM 5088 DIP switches and the ***HIA*** parameters ***lbond*** and ***ubond*** must be within the max/min IBM 5088 DIP switch values. The maximum RT control unit channel address is set as an 8-bit binary value in the IBM 5088 DIP switches U48-8 through U48-1.

2. The lLocal speed of the IBM RT IBM 5088 cable attachment must be specified the same in the WHIP system parameter ***lsped*** and in IBM 5088-1 or IBM 5088-2 switch U26-6.

The HIA parameters are defined using the AIX *devices* command.

Examples on how to set the switches on the IBM 5088 controller are given in the *IBM AIX Workstation Host Interface Program User's Guide and Reference Manual* and in *IBM 5088 Models 1, 2, and 1R Graphics Controller Maintenance Information*.

A status program called ***panel20*** is provided with the WHIP program. It can display the status of the customized HIA ports. Use it as an installation verification tool after you have installed the *IBM System/370 Host Interface Adapter* and the IBM 5088 Controller and use it as a diagnostics tool if you have problems getting your 3270 emulation sessions to work.

WHIP Terminal Emulation

Now, enough of the small talk. We'll now take you quickly through the installation steps for *Workstation Host Interface Program* terminal emulation and give you an idea of the functions you can use it for.

[11] Release 3 of VM/SP requires APAR 20521 for Extended Data Stream to work without causing a VM system abend.

Installation

The WHIP software is installed using the *installp* command. On an IBM RT you must also install the appropriate device driver for the device you are using. Then apply any updates available for the system. Under AIX PS/2 you may have to remove the file `/usr/lpp/whip/inst_updt.save` before you can apply updates to Workstation Host Interface Program. After installing it is necessary to add a device description for the adapter to the AIX system. Use the *devices* command to add the device description:

- On the IBM RT, add the device ***aea*** for the *Advanced 3278/79 Emulation Adapter* and/or add the device ***hia*** for the *IBM System/370 Host Interface Adapter*.
- On the IBM Personal System/2, add the device ***3270c*** for the *3270 Connection Adapter*.
- If using a DFT connection, set the parameter ***mnonid*** to the number of sessions needed (1 - 5).
- If using an *IBM System/370 Host Interface Adapter* connection, set the ***lbond*** and ***ubond*** parameters to the lower and upper boundaries of the available HIA device addresses. The ***ubond*** parameter must be greater than or equal to the ***lbond*** parameter.

WHIP Emulator Functions

The following section gives an overview of the functions included in the WHIP software. This is not a complete description. Please refer to the publications listed at the end of this chapter.

3270 Emulation as a DFT Terminal

WHIP provides 3278/79 emulation for the AIX system and supports attachment to an IBM 3174/3274 Cluster Controller or an IBM 9370 Workstation Adapter as a distributed function terminal. Only non-SNA attachment is supported, either a local connection via non-SNA channel attachment or a remote connection via a BSC communication link. Up to five sessions is supported when emulating a DFT type terminal.

Printer Emulation Support

An AIX system printer can emulate an IBM 3286 Model 1 or 2 or an IBM 3287 Model 1 or 2. This means that you can direct your print from the host system to your local AIX printer, as the printer is assigned its own host address.

Multiple Concurrent 3270 Emulator Sessions

With AIX/RT, a maximum of 21 logical emulator sessions are supported: Up to 16 via the IBM System/370 Host Interface Adapter and up to 5 sessions on a DFT connection via the IBM Advanced 3278/79 Emulation Adapter. These 21 concurrent emulator sessions may be divided between an IBM RT system console and attached ASCII terminals and printers. Support is provided for up to six concurrent sessions per attached ASCII terminal, up to six concurrent sessions per virtual terminal on the IBM RT console, and up to a total of 21 concurrent sessions on an IBM RT console. Using AIX PS/2, a maximum of 5 logical emulator sessions are supported via the IBM 3270 Connection Adapter. On both systems, sessions can be a mix of display and printer sessions.

Note: WHIP 1.1 will not support more that one session configured as a slow device (printer) using the Advanced 3278/79 Emulation Adapter connected to a remote 3274 BSC control unit.

ASCII Terminal Support

Supported ASCII terminals.:

- IBM 3161 in 3161 mode
- IBM 3162 in 3161 mode
- IBM 3163 in 3161 mode
- IBM 3151 in 3161 mode
- DEC VT 220
- A PC running an appropiate ASCII emulator.

Remote Access to 3270 Emulation via TCP/IP

Terminals, including appropriately configured personal computers, remotely connected via TCP/IP to the AIX system, may utilize the 3270 emulation feature of WHIP via ***telnet***. Remember to add a pseudo terminal device on the WHIP server for each telnet session.

WHIP File Transfer Program

WHIP contains a file transfer program, ***fxfer***, to do file transfers to and from a System/370 host. Functions include recovery following unexpected shutdowns, file transfer restart, asynchronous as well as synchronous file transfers and ASCII/EBCDIC translations. The program communicates with the *IBM 3270 File Transfer Program (IND$FILE)* on the host.

application program interface

WHIP provides an application program interface (API) that allows you to write programs to interface with the host system in different modes. One mode, API to 3270, allows an AIX program to manipulate a user's host session by sending key strokes to the host computer and inspecting the screens of data that are subsequently displayed. Another mode, API to API, allows an AIX program to send data back and forth to a corresponding host API application. The API can be accessed in AIX from C Language, Pascal, or FORTRAN programs and both the VM and MVS Operating Systems are supported.

National Language Support

Support is provided for the 13 National Language Keyboards supported by the AIX Operating Systems.

Supported by X-Windows

Users may run the WHIP Program from AIX X-Windows 2.1.

Enhanced Data Stream Support

The data stream support allows up to seven colors. The display features include reverse video, underline and blink. These features are supported for color consoles including remote consoles connected via telnet.

3278/79 Model 2, 3, 4 and 5 Emulation

3270 emulation provides support for display sizes of:

- 24 rows x 80 columns
- 32 rows x 80 columns
- 43 rows x 80 columns
- 27 rows x 132 columns.

Utility Programs

WHIP provides utility programs to help the user setup an appropiate environment. These are:

e789kdef	Allows the user to alter the default keyboard layout.
e789cdef	Allows the user to remap the seven host colors and four highlighting attributes.
e789paex	Allows the user to view and change the ASCII-to-EBCDIC and EBCDIC-to-ASCII translation tables used by the emulator, the file transfer program and the application program interface.

Using the 3270 Emulation Program

When used without command line arguments, the ***e789*** command initiates a single 3270 emulation session. Multiple sessions can be initiated at the same time by specifying the number of sessions on the command line. A 3270 emulation session is an AIX process with an opened virtual terminal and an opened link address. The initial state of the emulation session is the same as that of a real 3270 terminal that has just been powered on.

If *e789* is invoked without parameters, default parameters are read from the file `/usr/lib/whip/.whiprc`. When the emulator is started, the type of terminal is checked. Depending on the type of terminal, certain input arguments are chosen to run the emulator. These values can be overridden by making a user profile and/or by entering a parameter to the *e789* command.

Use `e789 -h` to display help information about the possible *e789* command line options.

If invoking the *e789* command from X-Windows, remember to add a pseudo terminal device with the *devices* command.

Because of the many options understood by the *e789* command, most of these can be specified in a user profile in "`.whiprc`". When the emulator is invoked, parameters are read from this profile. A default profile is provided and is located in the `/usr/lib/whip` directory. Move a copy of this profile to your `$HOME` directory and work with the copy. For example:

```
        cp /usr/lib/whip/.whiprc  $HOME/.whiprc
```

Here is a list of the profile entries:

P3270DEV	Specifies the communication path used by the emulator, for example `/dev/aea0`, `/dev/hia0` or `/dev/3270c`. The actual session path is set in an environment variable called ***H3270DEV*** For example: `H3270DEV=/dev/aea0/00`
H3270LANG	Specifies the default language for ASCII/EBCDIC translations.

H3270INDFIL	Specifies the default ASCII string for the file transfer module (*IND$FILE*) in the host. This parameter depends upon the ***H3270LANG*** parameter.
H3270MAXBUF	Specifies the maximum buffer size supported by the file transfer command and API applications.
E789_MOD	Specifies the 3278/79 Model number. Select: • 2 - for 24 lines x 80 columns • 3 - for 32 lines x 80 columns • 4 - for 43 lines x 80 columns • 5 - for 27 lines x 132 columns. When you invoke the emulator, make sure that the physical dimensions of the screen corresponds with the 3270 model chosen. If the screen (or X-Windows window) is too small, you will get an error saying: "***789-050, Display space too small to emulate 3278/79-X***", where "X" equals the value in the ***E789_MOD*** parameter.
FXFER	Specifies the default file transfer direction. Specify ***up*** for upload and ***down*** for downloading files. This can be overwritten with the *e789* parameters "***-u***" or "***-d***".
H3270QTIME	Specifies the length of time (in minutes) that the file transfer daemon remains active, awaiting additional file transfer requests.
H3270RTIME	Specifies the length of time (in minutes) that the file transfer daemon should attempt recovery.
H3270HOST	Specifies the default HOST operating system - CMS or TSO.
H3270LID	Contains the default logon ID (user ID). It is not a required parameter.
P1	A printer stanza for 3270 printer emulation. Specifiy one stanza for each emulated printer.

Differences in the Emulator Execution

Terminal emulation on an ASCII terminal in IBM 3161 or DEC VT220 mode and via *telnet* differs from that on the high function terminal in the following aspects:

- The Operator Information Area (OIA) of the emulated terminal is not automatically displayed on an IBM 3161 and DEC VT220. A key combination (default is `Ctrl-X`) can be used on the IBM 3161 and the DEC VT220 to display and then remove the OIA from the twenty-fourth line (the command line) of the display. This "pop-up" feature of the OIA exists because the OIA occupies the last line of the display on an IBM 3278 terminal (line 25) but the IBM 3161 and DEC VT220 have only 24 display lines.
- The default keyboard for the DEC VT220 does not support the PA3-key.
- The placement of keys on the DEC VT220 and the IBM 3161 differs significantly from that of the default keyboard of the high function terminal.
- Emulator speed is decreased on an IBM 3161 and a DEC VT220 as compared to an AIX console.

- National language support is not automatically provided for the emulator executing on a DEC VT220 or from terminals using telnet. To get national characters on the IBM 3161 you must:
  - Install WHIP with support for your national language.
  - When configuring the ***tty*** port in the AIX system with the *devices* command, set the following parameters:

    ```
    pt none  (parity = none)
    bpc 8    (Bits per Character = 8)
    nosb 1   (Number of stop bits = 1)
    ```

  - Setup the environment for the IBM 3161 using the ***setup key*** on the IBM 3161 keyboard. Again, specify no parity, 8 databits and 1 stopbit. Set the SEND field to ***PAGE*** and the INSERT field to ***SPACE***.
  - When the IBM 3161 user logs on to the AIX system, the *stty* command must be used. Example:

    ```
    stty -istrip -ignpar -cs8 imap ibm3161-C omap ibm3161-C
    ```

    If you want this automatically done when logging on to the AIX system, then use an editor to edit the file `/etc/ports`. A sample entry for the port definition is shown in Figure 52

```
/dev/tty0:
        enabled = true
        term = ibm3161
        parity = none
        speed = 19200
        logmodes = istrip+ignpar+cs8
        imap = ibm3161-C
        omap = ibm3161-C
```

Figure 52. Port Profile for Using IBM 3161 with WHIP

The WHIP Program is shipped with default keyboard mappings that allow the user to execute *e789* from the AIX system console, an IBM 3161 or a DEC VT220 without National Language Support. Default keyboards are mapped so that commonly used functions are placed in the same position on all three keyboard types, for example the PF-keys. A user can redefine the WHIP keyboard by using the ***e789kdef*** command.

Stopping The Emulator

To exit the WHIP emulator orderly, use the `Ctrl-D` key sequence twice from the 3270 emulation session screen. If for some reason there is a problem in the system and the emulator, file transfer or an API application won't stop or have terminated abnormally, you have a choice of using one or more of the following methods to clean things up:

- Use the *e789cln* command to clean up IPC (Inter Process Communication) resources that remain after abnormal termination. If you have other programs running, then be careful when you use this command or it will clean up your entire system, depending on your permissions, of course.
- If nobody but you is using the system, you can use the *shutdown* command to reboot the system, for example: `shutdown -fr`.

- Use the shell script in Figure 53 on page 121. This program kills the necessary processes and removes two message queues. Before you can use the program, you must know the message queue numbers (xxxxxxxx, yyyyyyyy). Use the *ipcs* command to display the message queues before and after starting the emulator and take a note of the two message queues used on your system when the emulator is active.

```
kill -9 `ps -e|awk '/ e789$| e789x$| dfxfer$/ {print $1}'`
ipcrm -Q xxxxxxxx
ipcrm -Q yyyyyyyy
```

Figure 53. E789KILL - Cleanup Shell Script for WHIP 1.1 Emulator on the IBM RT

Here is an example of two message queue numbers:

```
ipcrm -Q 630209de
ipcrm -Q 630002bd
```

WHIP File Transfer Program

The WHIP ***fxfer*** file transfer program allows you to transfer files from the host to the AIX system and vice versa. *fxfer* has the following capabilities:

- The file transfer may be initiated by using an AIX command or by a program using the file transfer programming interface. It may even be executed as a background process.
- The host operating system may be either VM/CMS or MVS/TSO. The file may be transferred from the AIX system to the host (upload) or from the host to the AIX system (download) and the transfer files may contain binary data or text (with or without ASCII or EBCDIC translation).
- File transfer may be either synchronous or asynchronous to the operator or the invoking program. If the file transfer is asynchronous, additional asynchronous file transfers may be initiated without waiting for the first transfer to complete as each request is placed in a queue.
- The status of the file transfer operation may be reported to a terminal, a file if invoked by the operator or to a program if invoked by a program.
- The file transfer may be interrupted before completion. When interrupted, the state of the transfer is saved. An interrupted file transfer operation may be restarted without loss of data. If the host communication is lost or disconnected, under certain conditions, the file transfer attempts to recover by reconnecting and logging back into the host. Once the host session is re-established, the file transfer is retried from the start.

The file transfer is handled by two modules: ***fxfer*** and ***dfxfer***. The *fxfer* module processes the file transfer requests, forms file transfer request queues, and sends the requests to the appropriate *dfxfer* process. The *dfxfer* module is the file transfer background process (or file transfer daemon) which performs the file transfer. Each *dfxfer* process is associated with one host session (or logon ID). Multiple file transfer requests to the same logon ID on the host are queued to the same *dfxfer* process.

Using the File Transfer Program

The file transfer program is initiated by invoking the AIX *fxfer* command. The ***-h*** option (`fxfer -h`) can be used to display a help screen that describes the options of the command. The program can be initiated by issuing commands from any terminal running the AIX shells (sh or csh) or from a terminal emulation session using an AIX subshell. When invoked from a subshell, the file transfer is called an ***explicit*** file transfer because it's assumed that the session to use is the already active session that the subshell was invoked from.

When invoked from an AIX shell, the file transfer is said to be ***implicit*** and it will attempt to do an automatic logon to the host. To tell the file transfer program what host session to use, you must set the environment variable ***H3270DEV***. The example below says device ***aea0*** and session ***00***.

```
H3270DEV=/dev/aea0/00
export H3270DEV
```

Note that if the environment variable is set when doing an explicit file transfer, *fxfer* is fooled into believing that it should do an implicit file transfer. In that case, a logon is attempted to the session given by the environment variable, which may not be what you want.

When using the *fxfer* command, AIX file names must be in the normal AIX format. The host file names must conform to the host naming convention which must be one of the following formats:

For VM/CMS:	***"filename filetype filemode"***
For MVS/TSO:	***"datasetname*** [***(membername)/password***]"

datasetname is either a sequential data set or a partitioned data set.

(membername) is the name of one of the members in the directory of a partitioned data set. The partitioned data set must exist.

/password is required if password protection was specified for the MVS/TSO data set.

File Transfer Examples

Create a WHIP user profile in your home directory. You can copy the default profile from `/usr/lib/whip/.whiprc`. Set the profile variables to your most commonly used configuration. The following is an example of one such configuration:

FXFER=up	/* transfer direction is upload */
H3270QTIME=2	/* dfxfer stays active for 2 minutes */
H3270RTIME=30	/* recovery time is 30 minutes */
H3270HOST=CMS	/* host operatingsystem is VM/CMS */
H3270LANG=USA	/* translation language is USA ASCII/EBCDIC */
H3270LID=usr_logonid	/* sets logon ID */

This configuration sets the default recovery time to 30 minutes. If there is a host problem in the middle of a transfer, the *dfxfer* daemon will reattempt to access the host without user intervention. If successful, *dfxfer* transfers the file from the beginning. If unsuccessful, it saves the file transfer status and any other queued requests in a ***restart file*** to be transferred later. Restart files used are `$HOME/i_fxfer.r` and `$HOME/x_fxfer.r`.

Examples using the file transfer program: When using the *fxfer* command, always specify the source file name first. Below are examples of the use of *fxfer*.

- This example assumes the previous user profile is set and the file `aixfile` exists in the current directory. The file transfer uploads `aixfile` to a VM host and translates it to EBCDIC using the USA translation table.

```
fxfer -t aixfile "test file a"
```

- This example downloads the file `user.file` from the MVS/TSO host to the AIX system. If `aixfile1` exists, it is replaced with the downloaded file from the host.

```
fxfer -d -r -H TSO "user.file" aixfile1
```

- This example uploads /u/aixuser/rtfile2 from the AIX system to MVS/TSO. It allocates 50 units of space. The unit of space is 1000 bytes. It adds 10 units of space each time the allocated space is filled.

```
fxfer -uH TSO -S 50,10,1000 /u/aixuser/aixfile2 "user.file1"
```

- This example downloads the VM/CMS host file `"test file a"` and appends it to the AIX file `aixfile` in the current directory. It translates the host file from EBCDIC to ASCII using the USA translation table. The status output is placed in the `stat.out` file. The file transfer is performed implicitly and the user logon ID is provided by the "*-x*" option. The user is prompted for the password.

```
fxfer -datl USA -f stat.out -x my_id -H CMS "test file a" ./aixfile
```

- This example uploads the AIX system file `aixfile1` to the VM/CMS host file `"test1 file a"`. It overwrites the host file if it exists. The "*-v*" option displays the number of bytes transferred and the elapsed time. The status output is displayed on the console. If the host file does not exist, the host file maximum logical record length is set to 132 bytes. The host file record format is variable. No translation is performed.

```
fxfer -urv -L 132 -V -H CMS aixfile1 "test1 file a"
```

Status Messages

The status of a file transfer is reported to the user when the file transfer is completed. Use the "*-v*" flag on the *fxfer* command to get status information during the file transfer. Here is an example of the status information given when the file transfer is completed:

TRANSFER STATUS:	message text
BYTE COUNT:	xxxxxxxx
SOURCE FILE:	filename
DESTINATION FILE:	filename
CREATED AT:	(time format)

WHIP Application Program Interface (API)

Installing the API

Before the API programs can be installed on the host, several prerequisite procedures must be completed:

- IBM 3270 File Transfer Program must be installed on the host.
- The WHIP installation must be successfully completed and the program must be operable.
- A user ID must be available on the host system, and this user must have access to the host file transfer program *IND$FILE*.

When these procedures have been completed, the WHIP API may be installed on the host.

Installing the API on MVS/TSO

To install the API on an MVS/TSO system you must perform the following:

1. Set your current directory to `/usr/lib/whip/mvs` by entering:

   ```
   cd /usr/lib/whip/mvs
   ```

2. Establish a WHIP 3270 emulation session by entering `e789` on the AIX system console.
3. Logon to the MVS/TSO host.
4. Start an AIX subshell by pressing `Ctrl-C`.
5. Start installation of the API with the command *instalapi*.
6. The *instalapi* program performs the following:
   - Uploads a *"g32catal.clist"* command file.
   - Catalogues a WHIP API library in the user catalog.
   - Uploads the API files. A status message is displayed as each file transfer occurs. In case of file transfer failure, the installation procedure terminates.

     The following files are transferred:

     ```
     Source Files        Destination Files
     ------------        ----------------------------
     g32alloc.mac        G32API.MACLIB.ASM(G32ALLOC)
     g32data.mac         G32API.MACLIB.ASM(G32DATA)
     g32dlloc.mac        G32API.MACLIB.ASM(G32DLLOC)
     g32read.mac         G32API.MAXLIB.ASM(G32READ)
     g32stat.mac         G32API.MAXLIB.ASM(G32STAT)
     g32write.mac        G32API.MAXLIB.ASM(G32WRITE)
     g32alloc.txt        G32API.TXTLIB.LOAD(G32ALLOC)
     g32data.txt         G32API.TXTLIB.LOAD(G32DATA)
     g32dlloc.txt        G32API.TXTLIB.LOAD(G32DLLOC)
     g32read.txt         G32API.TXTLIB.LOAD(G32READ)
     g32stat.txt         G32API.TXTLIB.LOAD(G32STAT)
     g32write.txt        G32API.TXTLIB.LOAD(G32WRITE)
     g32sampl.asm        G32APPL.ASM(G32SAMPL)
     g32test.asm         G32APPL.ASM(G32TEST)
     ```

- Compiles *G32SAMPL* and *G32TEST*.

The message *"**Installation of WHIP API library is complete**"* signifies the successful installation of the API library.

7. Terminate the AIX subshell by pressing `Ctrl-D` or typing `"exit"` and pressing the `ENTER` key. This returns you to the emulator session. On the host screen, a WHIP API status will be displayed.
8. The API functions can be verified by starting an AIX subshell by pressing `Ctrl-C`. Start the API test program by entering: `g32_test`. The program displays the message:

   *"**WHIP HOST API INSTALLED AND OPERATIONAL**"*
9. The *g32asm.clist* file is provided for compiling a WHIP API application program on the host.

   For example:

   ```
   exec g32asm 'g32test'
   ```

Steps for Installing the API on VM/CMS

To install the API on a VM host you must:

1. Set your current directory to `/usr/lib/whip/vm` by entering:

   ```
   cd /usr/lib/whip/vm
   ```
2. Establish a WHIP 3270 emulation session by entering `e789` on the AIX system console.
3. Logon to the VM/CMS host.
4. Start an AIX subshell by pressing `Ctrl-C`.
5. Start installation of the API by entering the command *instalapi*.
6. The program *instalapi* performs the following:

   - Uploads the API files. A status message is displayed as each file transfer completes. In case of file transfer failure, the installation procedure terminates. The following files are transferred:

     ```
     Source Files        Destination Files
     ------------        -------------------
     dispax.txt          DISPAX TEXT A
     dispio.txt          DISPIO TEXT A
     g32api.mac          G32API MACLIB A
     g32api.txt          G32API TXTLIB A
     g32sampl.asm        G32SAMPL ASSEMBLE A
     g32test.asm         G32TEST ASSEMBLE A
     g32asm.exe          G32ASM EXEC A
     ```

   - Creates `DISPAX` and `DISPIO` module files.
   - Compiles `G32SAMPL` and `G32TEST`.

   The message *"**Installation of WHIP API library is complete**"* tells you that the installation of the API library is complete.
7. Terminate the AIX subshell by pressing `Ctrl-D` or typing `exit` and pressing the `ENTER` key. This returns you to the emulator session. On the host screen, WHIP API status will be displayed.

8. The API functions can be verified by starting an AIX subshell, pressing `Ctrl-C`. Start the API test program by entering the command `g32_test`.

   The program displays the message:

   "WHIP HOST API INSTALLED AND OPERATIONAL"

9. The "`g32asm exec`" file is provided for compiling a WHIP API application program on the host. For example: `g32asm g32test`

Using the Application Programming Interface

The application program interface (API) provides high level access to the communication link and for program to program communication between an AIX application and a host application. The API consists of an object library on AIX and a MACLIB, a TXTLIB, and two I/O modules (VM only) on the host S/370 computer. In order for a host application to communicate with an AIX application, both must use the provided API commands. The API may be used in the AIX, VM/CMS, and MVS/TSO environments. In AIX, lower level access is provided via an interface to the 3270 terminal emulator.

An API session to a host may be in one of three modes:

API/API	This mode is used when an AIX application uses the API to communicate with a host application that also uses the API.
API/API_T	This mode is similar to the API/API mode, but messages are translated between ASCII and EBCDIC by the AIX API.
API/3270	This mode is used when an AIX application uses the API to communicate with a host application that assumes it is dealing with a 3270 terminal.

API Program Functions

The following functions are provided through the API:

- Program to program communication. AIX applications may communicate with host applications in a VM/CMS or MVS/TSO environment.
- Write and read functions for transmission of messages between the AIX and S/370 host applications. Messages are simple byte strings containing arbitrary data and may be up to 64000 bytes in length.
- Optional data translation between EBCDIC and ASCII in messages.
- Automatic logon to the host and initiation of a host application or a file transfer.
- Session control functions are provided to start and stop sessions, and to set the mode of the session.
- Session status includes error information and presence of incoming data.
- In addition to the message interface, the API provides an interface to the terminal emulator so an AIX application can invoke a host application that assumes it is communicating with a 3278/79 terminal.

Automatic Logon/Logoff Procedures

Some utility programs are provided to assist the API programmer in developing automatic logon procedures:

logform	A form for the user to record logon and logoff information. See the file /usr/lib/whip/logform.
genprof	Generates a profile for autolog.
genlaf	Generates a C program from a LAF (Logon Assist Feature) script. LAF is a pseudo programming language. LAF makes it easier (and faster) to develop applications, as it is a pseudo programming language where you only specify ***what*** you want your program to do, not ***how*** to do it.
mtlaf	Creates a test program for a LAF script.
tlaf	Runs test programs created by *mtlaf.*
fxlaf	Links the *fxfer* command with a LAF script.
apilaf	Links an API application with a LAF script.

When developing an autolog procedure, you can use the sample LAF scripts provided with WHIP as a starting point. They are called *g_log.vm* for VM/CMS and *g_log.mvs* for MVS/TSO. Three autolog profiles are provided as well: *SYSvm1, SYSvm2 and SYStso.*

Instead of going directly to the C programming interface, it is recommended for the first time API programmer to use the following procedures:

- Take a copy of the sample LAF scripts, read them and modify them. There are lots of comments in the samples.
- Prepare the LAF script with the *mtlaf* command.
- Test the LAF script with the *tlaf* command.
- If it is working, convert the LAF script to a C program using the *genlaf* command and provide additional code to suit your environment.

The above autolog procedures assume that the emulator was already activated with the *e789* command. If you should want the emulator to be started every night to do a file transfer (perhaps for backup reasons), you will need to use the ***crontab*** facility to start a program (that starts the emulator) in the AIX system at a specific time, or you can make your own program that starts the emulator at a specific time. Using a program to start the emulator is a little complex, so here is a list of steps your program should follow if you want it to start the emulator.

- Establish a controlling terminal and then start the emulator.

  Each process in AIX is associated with a process group and has a controlling terminal and each process group has a process leader. For example, when you log in, the shell (sh) process is the process group leader and any process descendants are in that process group. The terminal you are typing on is the controlling terminal. If you want to start the emulator using a program, then you will have to establish a controlling terminal. To establish a controlling terminal, use the ***setpgrp*** system call in AIX. This is described in *AIX Programming Tools and Interfaces.*

- Establish contact with a user ID on the host, using the autolog features of the API.
- Do the actual file transfer.
- Log off the host.
- Remove the emulator from memory. See "Stopping The Emulator" on page 120 for further information on how to stop the emulator.

API Sample Programs

WHIP is shipped with API sample programs to aid users in developing API applications.

g32_sampl	evaluates API performance
g32_test	verifies API installation
g32_3270	Sample API/3270 mode application

The AIX applications are written in the C language and host applications are written in assembly language. The C and assembly language source programs, as well as AIX executables, are located in the AIX directories:

/usr/lib/whip/vm	for VM/CMS systems
/usr/lib/whip/mvs	for MVS/TSO systems

The *g32_sampl* program is an API application that can test API performance and verify the automatic logon facilities. The executable code is linked with the autolog procedure so the program can be executed programmatically (implicitly) or as a child of the emulator (explicitly).

Execution of *g32_sampl* can take from 10 minutes to one hour. The *g32_sampl* program transfers data between the AIX system and the host system. A control window allows the user to monitor the progress of data transmission between the AIX system and the host system. The window shows the buffer number in process, the maximum number of buffers for a particular session, and the direction of message transmission.

The *g32_test* program is a test program to verify functionality of the API library after it has been installed with the *instalapi* command. This application also demonstrates how an application interfaces with the API library when programmatic logon is not needed.

The *g32_3270* program demonstrates how to use the API/3270 subroutines. It performs the following functions:

1. Uploads a host program called *g323270*
2. Assembles *g323270* on the host system
3. Executes *g323270* on the host system
4. Displays the output of the program on the AIX display.

An additional sample program is provided in this publication. It is called *sendkeys* and it sends character strings from the AIX system to the 3270 session screen. The sample program is listed in "WHIP API Sample Program (sendkeys.c)" on page 520. The *sendkeys* program can send one or two character strings to the 3270 session, but can easily be modified to send more. When the string is sent to the 3270 session screen, *sendkeys* emulates the 3270 ENTER key. This can be used, if a user wants to start a job in the 3270 session from an AIX program.

Reference Publications for This Chapter

The *Workstation Host Interface Program* is described in:

> *IBM AIX Workstation Host Interface Program, User's Guide and Reference Manual*, SC23-2060

Network PLUS

The Network PLUS series of programs provides an AIX/RT system with remote job entry (RJE) and 3270 terminal emulation functions in SNA or non-SNA environments. Network PLUS consists of four destinct program packages. You install one or more of these packages depending on your requirements.

Overview

Before starting the installation, you will need to decide which of the 3270-PLUS communication products provides the emulation that you require. The following is a brief description of the capabilites of each.

Network 3270-PLUS (SNA):
Operates in conjunction with IBM IBM RT SNA Services to emulate a 3x74 Cluster Controller with associated terminals and printers, attached to an SNA network. Uses the IBM RT Multiprotocol Adapter, X.25 Adapter, or Token-Ring Adapter (depending on the type of media used to connect to the host) to provide an overall total of 32 concurrent sessions on the console and attached ASCII terminals; up to a maximum of six on each.

Network 3270-PLUS (BSC):
When used with the IBM IBM RT Multiprotocol Adapter and a suitable modem, this version emulates an 3274 Cluster Controller and associated terminals, using the Binary Synchronous Protocol (BSC). Provides an overall total of 32 concurrent sessions on the console and attached ASCII terminals. As an alternative, this version may be used in DFT mode with the IBM Personal Computer 3278/79 Emulation Adapter (short card only) and a coaxial cable to a 3x74 Control Unit to provide a total of 5 sessions. This product can not be used with the X.25 Adapter or the Token-Ring Adapter.

Network RJE-PLUS (SNA):
Employs IBM RT SNA Services to emulate an IBM 3770-Series RJE Workstation attached to an SNA network. With the IBM RT Multiprotocol Adapter and a suitable modem, an X.25 adapter, or a Token-Ring Adapter, up to six RJE sessions with emulated card readers, card punches and printers can be defined.

Network RJE-PLUS (BSC):
Uses the IBM RT Multiprotocol Adapter and a suitable modem to emulate IBM 3770-series RJE (HASP) workstations, or IBM 2780 or 3780 Workstations. Again, it can not be used with the X.25 Adapter or the Token-Ring Adapter.

Network 3270-PLUS emulates the following 3270 devices:

SNA Controllers:	IBM 3274, Model 51C with Configuration Support C, Release 49.0 or equivalent. IBM 3276 Model 12.
BSC Controllers:	IBM 3271 Model 2, IBM 3274 Models 21C, 31C, 41C, 51C, 61C, IBM 3275 Model 2 and IBM 3276 Model 2.

Terminals:	IBM 3277 Model 2, IBM 3278 Model 2 and IBM 3279 Model 2A. Use the latter for sessions from an IBM RT console with a color monitor.
Printers:	IBM 3286 Model 2 and IBM 3287 Models 1 and 2.

Network RJE-PLUS emulates many of the functions of the following devices:

SNA Version:	IBM 3776 models 1, 2, 3 and 4, IBM 3777 models 1, 3 and 4.
BSC Version:	IBM 2780, IBM 3780 and IBM 3777-2 (HASP).

Table 5. Summary of IBM RT Host Communication Products with Network PLUS

Software Product	**SNA**	**BSC**	**Adapter**
Network 3270-PLUS	3277, model 2 3278, model 2 3279, model 2A 3286, model 2 3287, models 1, 2 3274, model 51C 3276, model 12 Max 32 Sessions		X.25
	3277, model 2 3278, model 2 3279, model 2A 3286, model 2 3287, models 1, 2 3274, model 51C 3276, model 12 Max 32 Sessions		Token-Ring
	3277, model 2 3278, model 2 3279, model 2A 3286, model 2 3287, models 1, 2 3274, model 51C 3276, model 12 Max 32 Sessions	3277, model 2 3278, model 2 3279, model 2A 3286, model 2 3287, models 1, 2 3271, model 2 3274, models 21C, 31C, 41C, 51C, 61C 3275, model 2 3276, model 2 Max 32 Sessions	Multiprotocol
		3278, model 2 3279, model 2A DFT Max 5 Sessions	Short PC 3278/79
Network RJE-PLUS	3776, models 1, 2, 3, 4 3777, models 1, 3, 4 Max 6 Sessions		X.25
	3776, models 1, 2, 3, 4 3777, models 1, 3, 4 Max 6 Sessions		Token-Ring
	3776, models 1, 2, 3, 4 3777, models 1, 3, 4 Max 6 Sessions	2780 3780 3777-2 (HASP)	Multiprotocol

Hardware Requirements

1. For ***remote*** connection to a host using a ***modem***.

   - An IBM 37xx Cluster Controller or an IBM 4300 or IBM 9370 Integrated Communications Adapter must be installed at the host site. If an X.21 line is used, the controller must be customized with a special X.21 port.
   - A Multiprotocol Adapter (feature no. 4762) must be installed in an appropriate slot in the IBM RT.[12]
   - A modem that supports the Multiprotocol Adapter and matches the bit rate of the host must be attached to the remote (IBM RT) end of a leased or switched communications line.

     In the case of an X.21 line a special type of modem, called a Network Terminating Unit (NTU), supplied by the network provider, must be attached to the communications line.
   - A *Modem Cable RS232C (16 pin)*, feature no. 4812 if you are using an RS232C port on the Multiprotocol Adapter, or a *Modem Cable - X.21*, feature no. 4816 if you are using the X.21 port on the Multiprotocol Adapter, must be used to connect the adapter to the modem.
   - The IBM RT must have at least 2MB of memory for Network 3270-PLUS (SNA) or Network RJE-PLUS (SNA), and at least 1MB of memory for Network 3270-PLUS (BSC) or Network RJE-PLUS (BSC).

2. For ***remote*** connection to a host over an ***X.25*** network.

   - An IBM 37xx Controller or IBM 9370 Integrated Communications Adapter with an X.25 port must be installed at the host.
   - An X.25 Adapter must be installed in an appropriate slot in the IBM RT.
   - A line and subscription to the (an) X.25 network in your country.
   - A modem, which is supplied by the X.25 network provider, must be attached to the remote (IBM RT) end of the communications line. The modem must have an ***X.21 bis*** interface.
   - A *PC Communications Adapter Cable X.25*, feature no. 2067 to connect from the adapter to the modem.
   - At least 2MB of memory must be installed on the IBM RT.

3. For local ***Token-Ring*** connection to a host.

   - An IBM 37xx Controller with a Token-Ring Interface Connector or an IBM 3x74 with a Token-Ring Adapter must be installed on the host network.
   - A Token-Ring Adapter (feature no. 3797) must be installed in an appropriate slot in the IBM RT.
   - At least 2MB of memory must be installed on the IBM RT.

[12] If you are installing the Multiprotocol Adapter into a IBM RT model 115 , 125, 130 or 135, you must change the DMA channel to channel 5; otherwise, the Multiprotocol Adapter ***conflicts*** with the EESDI-Adapter of the disk. To do this remove the jumper in the center of the adapter. See Figure 237 on page 532 for the placement of the jumper.

4. For ***DFT*** connection to a host using a 3x74.
   - A 3x74 must be installed at the remote (IBM RT) site.
   - A port on the 3x74 must be customized as a DFT port.
   - A 3278/79 Adapter (feature no. 5789) must be installed in the IBM RT.

Software Requirements

- AIX Version 2.2 (or later) must be installed on the IBM RT.
- The Multiprotocol device driver, the 3278/79 DFT device driver, or the Token-Ring device driver from the VRM device driver diskette, must be installed on the IBM RT. If you want to install SNA communications over an X.25 network, you must also install the "X.25 Communications Support Program" (Program no. 5601-179).
- The PC/Host File Transfer and Emulator Program (program number 5665-311 for MVS/TSO, 5664-281 for VM/SP, 5798-DQH for CICS and 5666-316 for VSE/SP 2.1) often known as the ***IND$FILE*** program, must be installed at the host.
- If you want to customize SNA communications over an X.25 network, you will also require the Network Packet Switching Interface (NPSI) running on the IBM 37xx that you are connecting to.

Disk Requirements

Table 6. Network PLUS: Disk space requirements in disk blocks

Program Name	**/**	**/vrm**	**/usr**
SNA Services	5250		5920
VRM device drivers			
Multiprotocol Adapter	520		100
3278/79 Adapter	220		50
Token-Ring Adapter	475		950
Network RJE-PLUS(BSC)	4		4350
Network RJE-PLUS(SNA)	4		4350
Network 3270-PLUS(BSC)	4		5200
Network 3270-PLUS(SNA)	4		5100

Network 3270-PLUS

Network 3270-PLUS permits the IBM RT to communicate with an IBM host as though it were a cluster of 3270-type devices. To the host, the IBM RT appears as an IBM 3x74 controller, the console and terminals attached to the IBM RT appear as IBM 327x terminals, and the IBM RT printer looks like an IBM 328x printer. This functional equivalence is illustrated in Figure 54 on page 135.

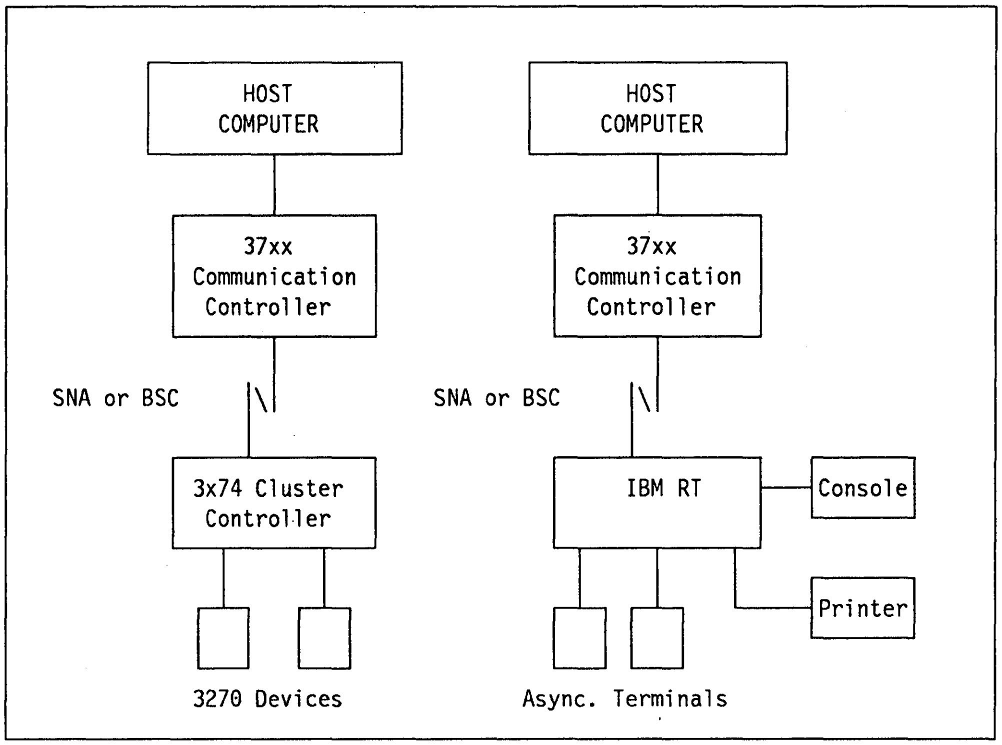


Figure 54. IBM 3x74 and IBM RT Functional Equivalency

The above illustrates the general CUT mode case. Network 3270-PLUS (BSC) also allows an IBM RT to be connected by coaxial cable to an IBM 3x74 cluster controller in DFT mode, in which case the IBM 3x74 controller is ***not*** emulated, and host to printer operations are ***not*** supported. See Figure 55.

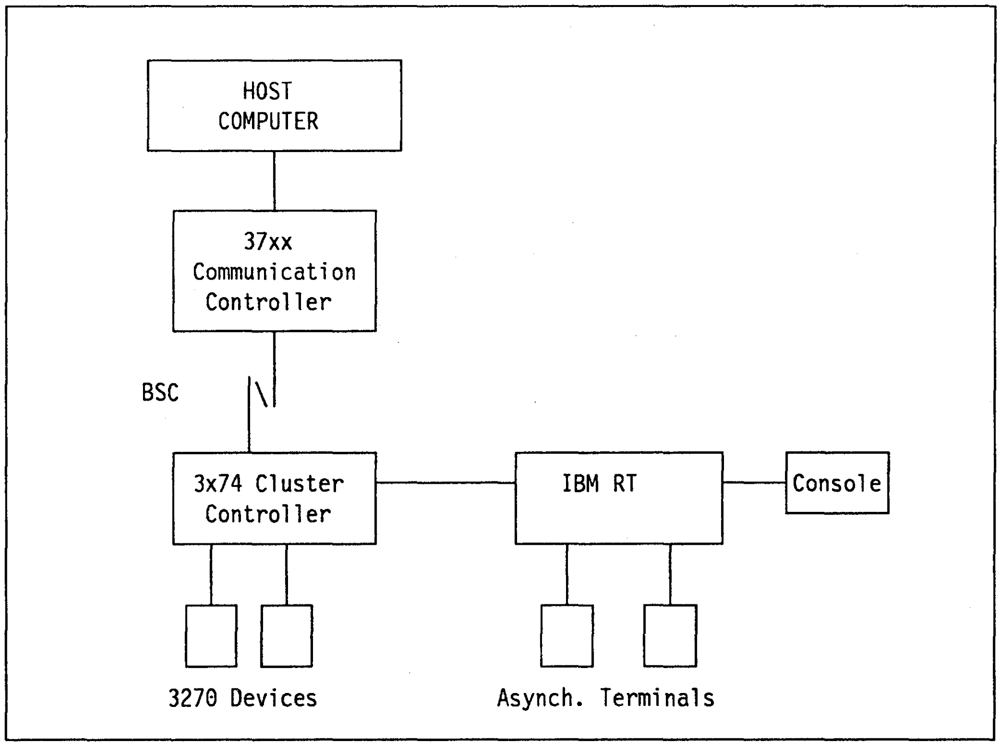


Figure 55. Connection of a IBM RT to a Host in DFT Mode.

Network 3270-PLUS (SNA) and ***Network 3270-PLUS (BSC)*** are very much alike in terms of the functions they provide and they will be described together. Where differences exist between the two versions, they will be noted in the text.

Both versions of Network 3270-PLUS consist of three components:

- Controller emulation
- Terminal emulation
- Printer emulation.

With the IBM RT emulating an IBM 3x74 cluster controller you may have up to 32 concurrent sessions, in the range from 2-129[13] for the SNA version, and up to 32 concurrent sessions (numbered 0-31) for the BSC version. Terminal users can access the host computer as if they were working at directly attached IBM 3270-type terminals of the emulated types. Host screens can be printed directly on a printer attached to the IBM RT or transferred to an AIX file. The Network 3270-PLUS printer emulator allows a printer attached to the IBM RT to be used in place of a host attached IBM 3286 or IBM 3287 printer. It can spool host initiated printer output to a file whether a printer is attached to the IBM RT system or not. Printer emulation operates independently from terminal emulation, and can continue to serve an attached printer also when the terminal emulator is inactive, and the IBM RT engaged in purely local processing.

Network 3270-PLUS (SNA) and IBM RT SNA Services

Before examining the Network 3270-PLUS product in detail, it is important to understand the relationship between the SNA version of Network 3270-PLUS and IBM RT SNA Services. Each piece of software is quite separate, is independently customized, and runs in its own environment. IBM RT SNA Services provides a gateway to an SNA network. This gateway could be utilized by a number of different terminal emulators, of which Network 3270-PLUS is one example. In contrast, the BSC version may be thought of as being a single, "point-to-point" connection.

Installation Procedure for Network 3270-PLUS

What follows is an overview of the complete installation procedure for IBM RT Network 3270-PLUS. The individual steps will be discussed in more detail later. The steps shown in parentheses may be ignored when installing the BSC version.

1. Install the necessary hardware.
2. Install the required device drivers and define the adapters with the *devices* command.
3. (Install SNA Services).
4. (Customize SNA Services).
5. Load the Network 3270-PLUS software onto the IBM RT disk using the *installp* command.
6. Customize the Network 3270-PLUS software.
7. Parse the software configuration.

[13] Note that in an SNA environment addresses 0 and 1 are reserved by SNA.

8. Shut down, then restart the IBM RT.
9. (Start SNA Services).
10. Make sure the line is enabled from the host end.
11. Establish the connection to the host mainframe[14] .
12. Load the Network 3270-PLUS code with the *s3270admin install* command for the SNA version or with the *b3270admin install* command for the BSC version.
13. Start a host session.

For the steps 1 and 2, see Appendix H, "IBM RT Hardware Installation" on page 531, and for steps 3 and 4, see "Customizing SNA Services" on page 46. We will start at step 5:

Installing Network 3270-PLUS.

1. Login as "root".
2. Install the Network 3270-PLUS program using the *installp* command.
3. Follow the prompts to install the software on the fixed disk.

   In the BSC version of the product, you will be asked if you are installing `mpdp` or `dft`. Select `mpdp` for a remote connection to the host and `dft` for DFT connection to an IBM 3x74. You will be asked which `dft` or `mpd` device you are customizing. Enter the name of the device you added with the *devices* command earlier. If you selected `mpdp` a further menu appears, asking you to enter a ***bsc port***. Unless you have already installed Network RJE-PLUS (BSC), select `/dev/bsc0`. This is the default minor device in the BSC parameters profile `/usr/lpp/r/comm/bsc/profiles/BSCparms.prof`.

Customizing Network 3270-PLUS

The software must now be customized to suit the requirements of your particular installation. There are five different configuration files for each version:

- Configuration Profile
- Terminal Option Set Profile
- Printer Option Set Profile
- Protocol Parameters Profile:
  - PH Parameters Profile for the SNA version
  - BSC Parameters Profile for the BSC version
- VTS Profiles.

[14] With Network 3270-PLUS (BSC) connection to the host may be made any time after step 11.

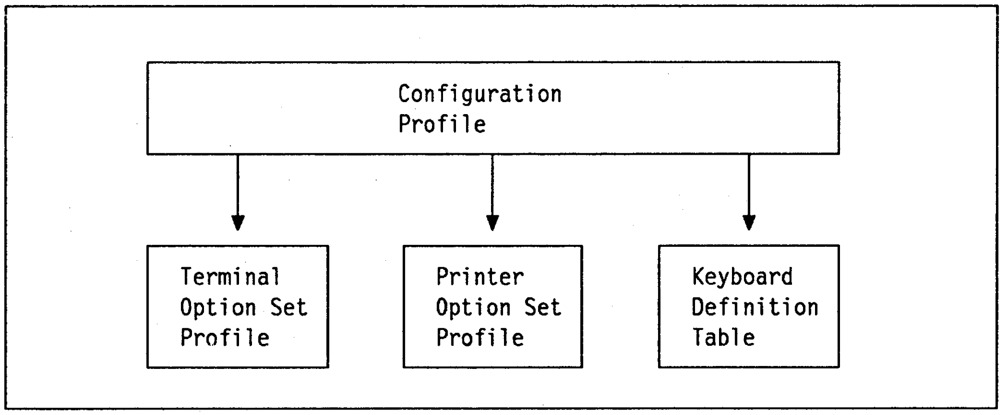

Configuration Profile
Terminal Option Set Profile
Printer Option Set Profile
Keyboard Definition Table

Figure 56. Relationships Between Some Network 3270-PLUS Profiles

Each profile contains keywords, followed by parameters in parentheses. Some of these parameters are the names of further profiles.

It is not the purpose of this document to provide an exhaustive description of all the configuration options possible through these profiles. The publications provided with the software describe every option in detail. The text which follows should provide a general overview of how to customize, but the reader should refer to the appropriate Network 3270-PLUS publication for more detailed information. Default profiles are provided in each case, and it may be possible to use many of those without modification. When modifying a default profile, make a copy for backup purposes, then use an editor to make the necessary changes.

Configuration Profile

For the SNA version this profile is in:

```
/usr/lpp/r/comm/sna/3270/profiles/sc3270.prof
```

For the BSC version in:

```
/usr/lpp/r/comm/bsc/3270/profiles/sc3270.prof
```

This is the "master" configuration profile. It describes how the emulated controller will appear to the host and the kind of terminals and printers connected to it. The profile holds a series of keywords, followed (in parentheses) by parameters defining the protocol to be used, the emulated controller type, the specific characteristics of terminals and printers to be emulated, and so on.

The default profile contains entries for a number of devices and features. Each is activated by having a colon in front of the entry. Use an editor to make any additional entries you may require.

Important

For the SNA version of Network 3270-PLUS it is ***very important*** that the configuration entries in this profile are consistent with the equivalent SNA Services profiles. The ***:terminal(x)*** entries in the configuration profile always search for the corresponding connection profiles in SNA Services identified by the name ***LUx***, where ***x*** is the corresponding terminal number. If the connection profile with this name ***LUx*** is not specified, you will not be able to start a terminal emulation with the address ***x*** to the host.

The most important keywords in this profile are as follows:

:Protocol Define as `SNA` for Network 3270-PLUS (SNA) and `BSC` for Network 3270-PLUS (BSC).

:ControllerType Define as `3274` or `3276` depending on how the controller is defined on the host. See the NCP or Emulation Program (EP) listing (also see "VTAM" on page 43).

:Terminal The definition of a terminal associated with a session. There should be one entry for each terminal session defined on the host, each linked with a particular LU number (SNA) or terminal address (BSC). Notice that the default Configuration Profile includes a set of default terminal parameters followed by two examples of specific terminal definitions. One of these `:Terminal(DEFAULT` is shown as being of the "default type", with the default parameters set and in the SNA version the `:Terminal(2)` follows the characteristics specified with the default terminal. Terminal 17 has its own terminal option file specified. Careful study of these default examples (which are repeated in Appendix A of the *Network 3270-PLUS User's Guide*) should make clear how they need to be amended to suit the needs of your installation.

You could, for example, choose to add definitions for two more terminals:

```
:Terminal(3)

:Terminal(4)
```

These would assume the default characteristics defined earlier in the profile, as no more specific parameters have been supplied. If you need to define one or more terminals which differ from the majority, then supply the appropriate parameters. For example:

```
:Terminal(5, ON, 3278, 2, 1920, 24, 80)
```

These parameters specify that emulated terminal ***5*** is a ***3278***, model **2**, whose ActiveFlag is to be set to ***on***. It is a device with a display of ***24*** rows of ***80*** characters and a buffer size of ***1920*** (24x80) characters.

:Printer The equivalent definition for an emulated printer, linked with its own LU number (SNA) or printer address (BSC). Similar defaults and examples are provided.

Note: With Network 3270-PLUS (SNA) terminal and printer addresses may only vary from 2 to 129. Address 1 is always reserved for RJE devices by the host. With Network 3270-PLUS (BSC) addresses may vary from 0 to 31 for remote connection to the host or from 0 to 4 for DFT connection.

If adding keywords, be sure to enter them before the `:End` keyword, which marks the end of the profile. In addition to specific configuration parameters, the `:Terminal` and `:Printer` keywords are also followed by parameters which point to other types of profiles:

- Terminal Option Set Profile
- Printer Option Set Profile.

The function of these will be discussed briefly, but in most cases you needn't make any changes to them.

Terminal Option Set Profile

The default Terminal Option Set Profile for SNA is in:

```
/usr/lpp/r/comm/sna/3270/profiles/TermOpts.prof
```

For the BSC version it's in:

```
/usr/lpp/r/comm/bsc/3270/profiles/TermOpts.prof
```

The profile holds a set of parameters which can invoke additional emulator features for each of the terminals to be emulated, including:

- ScreenToPrinter
- ScreenToFile
- ReadAhead
- Browse.

The Online Change feature makes it possible to override these options during a 3270 session. A separate Terminal Option Set Profile could be defined for each session, but it is more likely that users will prefer to share a common profile.

Printer Option Set Profile

The default Printer Option Set Profile for the SNA version is held in the file:

```
/usr/lpp/r/comm/sna/3270/profiles/PrtOpts.prof
```

The BSC version of the profile is in:

```
/usr/lpp/r/comm/bsc/3270/profiles/PrtOpts.prof
```

The profile describes all the printers to be emulated. Important keywords are:

:OutputPath	The destination of the output (SPOOL, FILE, or PRINT).
:OutputFile	The file to which the emulated printer output is to be directed, before it goes to the spool queue or directly to the printer.
:QueueName	The name of the spooler queue to be used by the emulated printer.
	Note: If you specify `lpq` to output the file to the AIX spooler,

you can only print on the ***default*** printer. You cannot specify that output should be routed to some specific system spool queue.

Parameters Profile

This profile is unique to the individual communications environment. In the SNA version of Network 3270-PLUS, it's called the "PH Parameters Profile" and is kept in:

```
/usr/lpp/r/comm/sna/profiles/SNAParms.prof
```

The BSC version is called "BSC Parameters Profile" and kept in the file:

```
/usr/lpp/r/comm/bsc/profiles/BSCParms.prof
```

In the case of SNA, much of the control of the communications link is done by IBM RT SNA Services, and so the SNA Parameters Profile has only one keyword, relating to the logging of diagnostic information. It will not normally need to be changed.

The BSC equivalent holds a number of keywords and related parameters to customize the Binary Synchronous Communications link:

:Line Parms There are a number of possible parameters associated with this keyword, including:

Connection	Must be set to `Multi-Point`.
Line	Select `leased` for a leased line connection, or `switched` for a dial up connection to the host.
RTS	Specifies whether the "Request to Send" modem signal should be on when connected or only while transmitting data. This parameter should be set to `PERMANENT` only when the line is full duplex. In half duplex mode it must be set to `SEND`.
Configuration	Signifies whether the local computer is a `PRIMARY` or `SECONDARY` station. Normally, the host is the primary and the IBM RT is the secondary so almost always `SECONDARY` is correct.
CallDirection	Normally `OUT`, unless you expect the host to initiate the call.
Autocall	Indicates whether the call will be dialed or answered automatically with `AUTO`, manually with `MANUAL` or nothing or both with `NONE` specified.
Clocking	Specify `EXTERNAL` if the modem provides the clocking; `DTE` if the IBM RT is to generate clocking signals. Today's modems provide the clocking, so `EXTERNAL` is usually the right choice.
BaudRate	Selected if `DTE` clocking required. Parameter values are `600, 1200, 2000 or 2400`.

LinkProtocol	Select `Smartmodem` if using a Hayes Smartmodem or compatible, or `RS232C` if another type of modem is being used. If you want to use the X.21 interface, select `X21`.
DialData	The dial string that needs to be sent to the smartmodem. For example, ATDT <tel. no.> to tone-dial the telephone number, <tel. no.> to auto-dial the telephone number.

:Control Unit There are a number of possible parameters associated with this keyword, including:

Mode	Select `3274` or `3276` depending on which controller emulation you want to use.
PollAddress	A unique polling address in hexadecimal. This is almost always X'40', unless a multi-dropped line is being used. Consult the network administrator.
MinorDevice	Selects a specific hardware port. For a DFT connection you can use `dft0` - `dft4`. For a BSC connection you can use `bsc0` - `bsc3`.

VTS Profiles

The Virtual Terminal System (VTS) profiles make it possible to use Network 3270-PLUS from different terminal types by describing the keyboard configurations of the terminals. For the SNA version they are held in the directory:

```
/usr/lpp/r/comm/sna/3270/profiles
```

For the BSC version in:

```
/usr/lpp/r/comm/bsc/3270/profiles
```

Each terminal type will have a source file, for example: `ibm3161.s3270` and a corresponding parsed file, for example: `ibm3161.term`. There are profiles provided for the IBM 316x terminals, the IBM 615x and IBM 515x consoles, and the DEC VT100 and VT220 terminals. If you use other terminals types than these, you will need to create new VTS profiles. See Appendix F of the relevant Network 3270-PLUS publication.

The default profiles contain entries for a variety of devices and features. Each is activated by a colon (:) in front of the entry. Use an editor to make any additional entries you may require (using the examples as models) and deactivate any entry not required by deleting the colon (:).

For a complete description of the significance of each keyword and parameter, refer to Chapter 8 of the *Network 3270-PLUS User's Guide*.

Parsing the Network 3270-PLUS Configuration

After the configuration profiles have been customized for a particular host connection, they must be parsed to check them for errors and to "compile" them for use by the Network 3270-PLUS software. To parse the configurations, type:

For the SNA version:

```
s3270admin parse configuration
```

For the BSC version:

```
b3270admin parse configuration
```

If you are connecting via the Multiprotocol Adapter, you must shut the IBM RT down and restart it after you have parsed the configuration files so the customized microcode will be loaded into the Multiprotocol Adapter.

Starting Network 3270-PLUS

Assuming that the configurations parsed without errors, you are now ready to start Network 3270-PLUS.

Note: If your console display has color, and you wish to use a 3279 (colour 3270 terminal) session on it, 3279 colours can be invoked at this stage by the following procedure:

1. Type `SetColor 5154` or `SetColor 6154`, as appropriate for your display.
2. Type `Ctrl-D` to log out.
3. Log in to re-set the display with new attributes.

IBM 3279 color emulation can be de-activated by typing: `SetColor off`.

If you use the SNA version of Network 3270-PLUS, SNA Services must be started before you can start Network 3270-PLUS. Use the *start sna*[15] command. Then start the emulator with the command:

For the SNA version:

```
s3270admin install
```

For the BSC version:

```
b3270admin install
```

This loads the program into memory. In the SNA version, the communications link (the SNA Services attachment) is activated at this point. If you are using a dial-up (switched) line, it will ask you to dial the number. Then the modem lights will flicker for a moment and a regular polling signal should be seen. In the BSC version, the software is just loaded into memory at this point ready for communication. As soon as you establish communication with the host or the host enables the line, communication should commence, and the modem lights

[15] You may want to put this command into one of the `/etc/rc...` files so that it is executed at start-up time. If DS is installed, SNA Services will be started from `/etc/rc.ds`.

should register a steady polling signal. Now you can start a session with the host using the command:

For the SNA version:

```
s3270 start n
```

For the BSC version:

```
b3270 start n
```

Where "n" is the number of one of the sessions (LU's) or BSC addresses specified during the configuration phase. At this point, a copyright screen should be displayed, followed by the Status Banner. Then, if all is well, you should see a "Logon Screen" from the host, and the host session will be active.

Note: Whenever any of the Network PLUS products are terminated abnormally (for example when the system is powered off without shutting down Network 3270-PLUS) several lock files are left on the system and Network 3270-PLUS will not restart properly. It is thus good practice to remove these before you start Network 3270-PLUS again. The procedure for doing this is described on page 11 of the "Memorandum to Licensees" at the front of the relevant Network 3270-PLUS publication.

Using Network 3270-PLUS

Once you have established a host terminal session, you can switch to a ***Multi-Session Menu*** to control host sessions. This is achieved by selecting the `FINISH/SUSPEND` key:

- From the console, using the `End` key
- From an ASCII terminal, using the `Esc-f` key combination.

Console users can also switch to another virtual terminal (`Alt-Action`) to execute AIX commands.

The Multi-Session Menu

This menu allows the user to:

- View the status of all open sessions
- Move between sessions
- Suspend a session and return to the AIX shell
- Terminate individual or all sessions.

Up to six sessions may be open at any one time. Only one may be active at a time, however. Others are automatically suspended when a new active session is selected. The Multi-Session Menu can be accessed either from the AIX shell by typing `s3270 menu` or from a 3270 emulation session with the `FINISH/SUSPEND` key. The menu will display:

- The LU number of each active or suspended session
- The session status (active or suspended)
- The date/time each session was suspended.

At the bottom of the screen the available functions are displayed:

- Exit/Suspend
- Finish LU

- Resume LU
- Start New LU
- Resume New LU
- Finish All LUs.

Menu selections are made by moving the cursor up or down to the required session entry, and then either pressing the appropriate PF key, or moving the cursor left or right to the desired function. If an option is selected inadvertently, it may be cancelled by pressing the `Pause` key.

Other Session Options: The following commands can also be invoked during a host session if previously enabled in the Terminal Option Set Profile. Many command options may be varied during the course of a session by means of the ***Online Change Menu***, selectable by pressing the `Online Change (Alt-F12)` key.

Screen ReadAhead	This option does not need to be specifically activated from the session; it is invoked when the emulator is started, and automatically fetches the next read-only screen from the host session and holds it until the user requests it.
ScreenToFile	Allows screen images to be sent to a local file for further processing. Invoked by pressing the `ScreenToFile (Alt-f)` keys. The destination file can be changed through the Online Change menu, by pressing `Online Change (Alt-F12)`.
ScreenToPrinter	Screen images can be directed to a print spooler for printing. As with `ScreenToFile`, the destination can be changed through the Online Change menu.
Cursor-Select	Where the application calls for a light-pen, this may be simulated using this feature. Place the cursor on the appropriate field, and the `Cursor-Select` key `(Alt-F8)` toggles on and off.
Browse	This feature allows the user to review a batch of screens processed earlier in the 3270 session. The feature is enabled and the number of screens to be stored is established during software configuration. The `ScreenToFile` and `ScreenToPrinter` features can be used while browsing, and the screen buffer remains in place until the end of the session. Invoke Browse by pressing `Prev Page (Esc-p)`. Use `Prev Page` and `Next Page (Esc-n)` to move back and forth within the screen buffer, and `Cancel (Ctrl-Home)` to return to the current screen.

File Transfer

A separate ***File Transfer Menu*** is used to up- or download files to and from the host. To display the menu, press `File Transfer (Alt-t)`. The following options are selectable from the menu:

PF1	Transfer (send and receive) files. This displays the ***File Transfer Command Screen***, described later.
PF2	Allows you to view the emulator screen during file transfer.

PF3	Aborts file transfer.
Help	Displays help screens. (Alt-h)
Cancel	Cancels the last activity, and returns you to the previous screen (Pause).
Finish/Suspend	Displays the Multi-Session screen. File Transfer continues in the background. You can return to File Transfer to monitor progress (End).
Exit	Returns you to the operating system shell. File transfer continues in the background and can be monitored through use of the *resume* command. (*s3270 resume* for the SNA version or *b3270 resume* for the BSC version.)

File Transfer Command Screen

This is the screen where you enter the file transfer command, the form of which will vary depending on the host environment. To send a file to the host:

TSO:	*SEND rt_file host_dataset_name* [*host_membername*] [*options*]
VM/CMS:	*SEND rt_file host_filename host_filetype* [*host_filemode*] [*options*]

Similarly, to receive a file from the host:

TSO:	*RECEIVE rt_file host_dataset_name* [*host_membername*] [*options*]
VM/CMS:	*RECEIVE rt_file host_filename host_filetype* [*host_filemode*] [*options*]

It is advisable to issue the CMS command `SET MSG OFF` before commencing VM file transfer to avoid the possibility of messages transmitted to your terminal corrupting the file being transmitted. As a "real" IBM 3270 has no way of interpreting the stream of data received from the host, no provision is made to separate a message from other data in the stream. The file transfer program may "hang", because the message has put the terminal in "MORE..." mode. Or the data stream was corrupted, and the file transfer will terminate because the checksum of a record was incorrect.

Not all the file transfer options can be described here, but remember that data on IBM mainframes is stored in EBCDIC, whereas AIX files are ASCII. You will need to use the *ASCII*, *CRLF* and *NL* options to take care of code conversions. See the appropriate Network 3270-PLUS User's Guide for details.

Information Displays

The Banner (Alt-PF11) keys alternately displays and removes a display of the status of the current session at the top of the screen.

If the Network 3270-PLUS software needs to send a message to the user, this will appear as a window on the display. If this message concerns an error condition which locks the keyboard, this can be unlocked by pressing the RESET (ScrollLock) key.

Leaving Network 3270-PLUS

Before ending emulation, you should first log off from all host sessions. Then, the emulation can be ended as follows:

For the SNA version:

```
s3270admin shutdown
```

For the BSC version:

```
b3270admin shutdown
```

A message will be displayed confirming that the emulator is being shut down. Wait for the "***shutdown successful***" message, which may take a while, as a number of processes may have to be terminated gracefully. Then you will be asked if you wish to shut down the system. Respond as appropriate.

Note: If you are running the SNA version of the emulator, SNA Services will still be active. If you no longer need to have SNA Services active, type: `stop -c sna` to stop SNA Services.

Network RJE-PLUS

With Network RJE-PLUS the IBM RT becomes, in effect, an IBM 2780, 3780 or 3770-series RJE workstation capable of submitting jobs to a remote mainframe computer for batch processing on a "first-in, first-out" basis. The emulated workstations consist essentially of a cluster of card readers, card punches and printers, and, in the case of the 3770-series workstations, a console. The RJE workstations are connected to the communications controller at the host without using a cluster controller as with 3270 terminals. This gives you one RJE session with the host. However, with the 3770 version of these workstations, the terminal can use up to six sessions (LU's) with the host simultaneously. This means that the terminal appears to the host as several "virtual" terminals, and thus can submit or receive several batch jobs at the same time. Up to six RJE sessions may be active simultaneously with 3270 sessions. See "SDLC leased Line to SNA System/370 Host" on page 419.

Note: IBM 2780s and IBM 3780s may be connected "back-to-back". Computers emulating these terminals can thus also be connected "back-to-back" and this has become a commonly used way of transferring files between computers. In this way two IBM RT's running Network RJE-PLUS (BSC) may transfer files to each other or to other computers emulating these RJE workstations.

Network RJE-PLUS offers features beyond those available on the emulated workstations, including:

- Commands to simplify the submission and deletion of jobs and to determine their status
- Job control, device monitoring and queue manipulation through an interactive menu
- Customizing through text profiles
- Routing of host output to selected IBM RT devices and files
- Online help.

Once again, there are SNA and BSC versions, with essentially similar characteristics. Both versions relate closely to their equivalent 3270 emulation counterparts.

Installation Procedure for Network RJE-PLUS

What follows is an overview of the complete installation procedure for Network RJE-PLUS. The individual steps will be discussed in more detail later. The steps shown in parentheses may be ignored when installing the BSC version.

1. Install the necessary hardware.
2. Install the required device drivers and define the adapters with the *devices* command.
3. (Install SNA Services).
4. (Customize SNA Services).
5. Load the Network RJE-PLUS software onto the IBM RT disk using the *installp* command.
6. Customize the Network RJE-PLUS software.
7. Parse the software configuration.
8. Shut down, then restart the IBM RT.
9. (Start SNA Services).
10. Make sure the line is enabled from the host end.
11. Make the connection to the host mainframe[16].
12. Load the Network RJE-PLUS code with the *srjeadmin install* for the SNA version or with the *brjeadmin install* command.
13. Start a host session.

As can be seen, the procedure is essentially the same as that for Network 3270-PLUS. For the steps 1 and 2, see Appendix H, "IBM RT Hardware Installation" on page 531, and for steps 3 and 4 see "Customizing SNA Services" on page 46. We will start at step 5.

Customizing Network RJE-PLUS

Software configuration for Network RJE-PLUS closely follows the procedure described in section "Network 3270-PLUS" on page 134. There profiles are eligible to be changed[17] for each version:

- Configuration Profile
- Protocol Parameters Profile
- VTS Profiles.

Each profile contains keywords followed by parameters in parentheses. Default and example values are provided and may be used as models for your own entries. Make copies for backup purposes and use an editor to make any addi-

[16] With Network RJE-PLUS(BSC), the connection to the host may be made any time after step 11.

[17] Other, existing profiles are ***NOT*** to be changed.

tions or changes. The procedure for customizing the Network RJE-PLUS profiles is the same for the two versions, except that different keywords vary.

Configuration Profile

The Configuration Profile for the SNA version is in the file:

```
/usr/lpp/r/comm/sna/rje/profiles/scRJE.prof
```

The BSC version profile is in:

```
/usr/lpp/r/comm/bsc/rje/profiles/scRJE.prof
```

This is the "master" profile. It describes the way in which the emulated RJE workstation(s) appears to the host and the number of printers, card punches and card readers. The profile holds a series of keywords followed (in parentheses) by parameters defining the characteristics of the various RJE devices assigned to each workstation, the specific workstation type to be emulated and so on.

The main point about this profile is that it must mirror the way the RJE terminal is defined at the host. You should pay particular attention that the ***LogonString*** used and the number of emulated devices match those defined in the ***JES Parameters profile*** on the host. See "SDLC leased Line to SNA System/370 Host" on page 419. The following keywords are common to each version; those unique to each are described later:

:Input Parms

Compression	Indicates whether blanks are to be compressed. Only valid in 3780 (BSC) emulation.
Tabchar	The character to be used as the tab character for IBM RT files to be sent to the host. Usually, this may be left as the default. For example, the ASCII 09 value represents the Horizontal Tab Character (HT).
Tablength	As for Tabchar.
EndofRecord	The character to be used as the end of record character on the IBM RT. Usually, you can leave this as the default. For example, the newline character, ASCII value decimal 10.
MaxInputSessions[18]	Set this to the number of host sessions (LU's) being used (1-6).
MaxElementsInChain[18]	Leave as default.

:Console

Two parameters define the size for the Live Console Browse storage file and, optionally, an encrypted password string. Leave as default initially.

[18] Not relevant in Network RJE-PLUS (BSC).

:Log Parms

Each parameter specifies which logging option is to be set on or off. Initially, leave all ***off***.

:Output Device

This stanza defines the types of devices attached to the RJE workstation (or to each logical RJE workstation in the SNA version). There should be one stanza for each device.

Medium	The type of device emulated (PRINTER, PUNCH(SNA) or CARD(BSC)).
DeviceNumber	In the case of multiple devices of the same type, the address of the device.
DestinationID	The name that you want to call this output device. Note that this should tie up with the corresponding `DestinationID` parameter in the `:Destination Control` keyword.

:Destination Control

This keyword specifies the local logical device which is to be the destination for output from the host. Each stanza should link up with an `:OutputDevice`. Parameters include:

DestinationID	Same as `DestinationID` for corresponding `:OutputDevice`.
OutputRouting	FILE, PRINTER, or SPOOLER.
OutputPath	Output filename for FILE or PRINTER.
QueueName	If `OutputRouting=SPOOLER` select `lpq`.
Translation	Do you want ASCII-EBCDIC translation?

Note that, although the `Destination Control` keyword is common to both SNA and BSC versions, the former has an additional parameter. `FCBName` is used to specify Forms Control Block definitions.

:End

As before, this parameter is required to signal the end of the configuration profile.

SNA Differences: The requirements of the SNA environment dictate that certain unique items of configuration information be provided:

:LU Spec

The specification of each of up to six Logical Units. At least one must be defined.

Address	The logical address of the session that your RJE workstation is using.
Autologon	Do you want this session to be immediately started when you start Network RJE-PLUS?
LogonID	A string that is a pointer to the `:Logon` stanza where the host logon string you wish to use is defined. See below and "SDLC leased Line to SNA System/370 Host" on page 419.

:Logon

This has two associated parameters:

LogonID	A string that corresponds to the LogonID in the :LU Spec keyword.
LogonString	The actual logon string that needs to be supplied to the host to logon. It will usually be of the form LOGON APPLID(JES2) DATA(RMTx) where RMTx is the logon name of this RJE workstation.

:Forms Control Block

This was mentioned during the earlier description of the :Destination Control keyword. A number of values control the formatting of text on the specified output devices. Does not normally need to be changed.

BSC Differences: Although the configuration of Network RJE-PLUS (BSC) is essentially similar to the SNA version, it does not, of course, require that IBM RT SNA Services be customized. Additionally, the BSC version can be set to emulate the 2780 and 3780 Workstations, as well as the 3770 Workstation.

In conformity with BSC practice, a session is referred to as an "address" rather than an "LU", and some unique keywords are required:

:Mode	The parameter associated with this keyword can be 2780, 3780 or HASP (3777), and specifies which workstation type is to be emulated.
:Signon	Where a specific string is necessary to gain access "signon" to a host application subsystem, that string is supplied as a parameter to this keyword.
:Send Buffer	The emulation software needs to know the size of the buffer the host subsystem uses for incoming data. This keyword parameter specifies the buffer size as a number of bytes.
:Card Readers	The number of "logical" card readers. In 2780 or 3780 mode, the number will always be 1; in HASP (3777) mode, it may be any number in the range 1-7.
:Outbound Default	Only applicable when 2780 or 3780 are being emulated, this keyword parameter specifies the default destination (PRINTER or PUNCH) to be used for host output.

Protocol Parameters Profile

The Protocol Parameters Profile is unique to the individual communications environment. For the SNA version it's called ***SNA Parameters Profile*** and is stored in:

```
/usr/lpp/r/comm/sna/profiles/SNAParms.prof
```

For the BSC version it's named ***BSC Parameters Profile*** and kept in:

```
/usr/lpp/r/comm/bsc/profiles/BSCParms.prof
```

These profiles are almost identical to the corresponding Network 3270-PLUS profiles. Again, the SNA Parameters Profile needs not be changed except to enable logging of diagnostic information. The BSC Parameters Profile has only one parameter that differs from its Network 3270-PLUS, counterpart, namely:

:Control Unit Parameters now include:

Mode Select 2780, 3780, or HASP depending on the type of RJE workstation you are emulating.

VTS Profiles

The Virtual Terminal System (VTS) profiles define the characteristics of the terminals used on the IBM RT by Network RJE-PLUS. In Network RJE-PLUS they reside in the directories:

For the SNA version in:

```
/usr/lpp/r/comm/sna/rje/profiles
```

For the BSC version in:

```
/usr/lpp/r/comm/bsc/rje/profiles
```

Each terminal type has a source file (for example `ibm3161.srje`) and a corresponding parsed file (for example `ibm3161.term`) associated with it. Profiles are provided for the IBM 316x terminals, the IBM 615x and IBM 515x consoles and DEC VT100 and VT220 terminals. If you are using terminals types other than these you must to create further VTS profiles for them. See Appendix F of the relevant Network RJE-PLUS publication for details.

Parsing the Network RJE-PLUS Configuration.

Once customized, the profile must be parsed. For the SNA version:

```
srjeadmin parse configuration
```

For the BSC version:

```
brjeadmin parse configuration
```

Then shut down the IBM RT and restart it.

Starting Network RJE-PLUS

Assuming that the configurations parsed without any error, you are now ready to start up Network RJE-PLUS. To start the SNA version, first start SNA Services with the *start sna*[15] command. Then start the emulator by typing:

```
srjeadmin install
```

For the BSC version start the emulator with:

```
brjeadmin install
```

This loads the program into memory. In the SNA version, the communications link (the SNA Services attachment) is activated at this point. If you are using a dial-up line, the program will also ask you to dial the number at this point. The modem lights should flicker for a moment and then a regular polling signal should be seen. In the BSC version, on the other hand, the software is just loaded into memory at this point, ready for communication. Any time after this, you may establish communication with the host. The modem lights should then flicker and register a steady polling signal from the host.

Note: Whenever any of the Network PLUS products are terminated abnormally (for example, when the system is powered off without shutting down Network RJE-PLUS) several lock files are left on the system and Network RJE-PLUS will not start properly. It is thus good practice to remove these before you start Network RJE-PLUS again. The procedure for doing this is described on page 11 of the "Memorandum to Licensees" at the front of the relevant Network RJE-PLUS publication.

You may want to have a password for Network RJE-PLUS to prevent unauthorized use. If so, the superuser may create or change it by typing:

For the SNA version:

```
srjeadmin password
```

For the BSC version:

```
brjeadmin password
```

You will be prompted to enter the password twice to ensure accuracy. Note that the first password, active when the software is received, is a null string ("").

Using Network RJE-PLUS

The workstation emulation is started by submitting a job for remote processing with the following command:

For the SNA version:

srje submit [*-flag* ... *-flag*] *filename*

For the BSC version:

brje submit [*-flag* ... *-flag*] *filename*

where you can specify several optional flags. The filename identifies the AIX file to be sent to the host computer.

Valid flags are:

-d followed by date (optional) and time for the job to be submitted, in the form `mm/dd/yy hh:mm`. For example: `-d 07/01/90 15:00` would be interpreted as "submit job at 3 p.m. on July 1, 1990".

-t specifies "Transparent Mode". Data in the file, including what might otherwise be seen as control characters, is to be read as a string of bytes without interpretation. In transparent mode no compression is done, which will have an effect on performance. Specify this option only if you want to transfer binary files.

-r followed by the record size in bytes. For example: `-r 96` signifies a 96-byte record. Valid sizes are 1-128, the default assumed is 80. Oversize records are split into two records.

-x "expedite". This places a job at the head of a queue which is normally served on a "first-in, first-out" basis.

With Network RJE-PLUS (SNA) the status of queued jobs may be checked by typing:

```
srjeadmin status
```

For the BSC version with:

```
brjeadmin status
```

This displays a list of all jobs on the queue, showing:

- Job Number (assigned by Network RJE-PLUS).
- Date or time the job was added to the queue, in the form hh:mm or mm/dd.
- The login ID of the operator who submitted the job.
- The name of the AIX file submitted (truncated at 44 characters).
- The job status, which may be:
  - "Active": In process of transmission to the host.
  - "Expedited": At the head of the job queue, awaiting processing.
  - "Queued": Awaiting processing, in sequence.
  - (Date and time): Awaiting date and time for submission.

Jobs may be deleted from the queue one at a time or in groups with:

For the SNA version:

srje delete <job number> [job number ... job number]

For the BSC version:

brje delete <job number> [job number ... job number]

The entire queue can be purged with the commands:

For the SNA version:

srjeadmin purge

For the BSC version:

brjeadmin purge

The latter should be used with great care, as purged jobs cannot be recovered. The normal AIX rules concerning ownership and authority apply.

Note: The Network RJE-PLUS SNA version 1.01 has a bug. After submitting a job to the JES system Network RJE-PLUS, SNA does not send the End Bracket as required by the SNA session control protocol. This causes VTAM not to send the print output to the printer because it is waiting for the End Bracket. You need to press any key on the Live Console to send a new Begin Bracket and End Bracket to VTAM. This causes VTAM to send the print output to the IBM RT printer. This bug is fixed by Rabbit Software Corporation and you can get a patch diskette version 1.14 that fixes this problem.

Ending Emulation

In a similar fashion to its Network 3270-PLUS counterparts, Network RJE-PLUS (SNA) emulation is terminated by typing:

```
srjeadmin shutdown
```

Network RJE-PLUS (BSC) is terminated with:

```
brjeadmin shutdown
```

National Language Support (NLS)

The Network 3270-PLUS programs do not support national languages other than US english. Translation between ASCII and EBCDIC is done through two translation tables in the file `/usr/lpp/r/comm/sna/3270/bin/3270DH` for the SNA version and in the file `/usr/lpp/r/comm/bsc/3270/bin/3270DH` for the BSC version. One table changes 7-bit ASCII characters to EBCDIC characters, the other one changes the EBCDIC characters to 7-bit ASCII characters. For EBCDIC, US code page 037 is used, for ASCII the lower half of US code page 437 is used. Because the European countries need their national languages when communicating with a host customized to that national language, we will give you some experiences of users who created their own solutions.

There are two possibilities to do this.

1. You can change the VTS profiles for the console and the ASCII terminals. This is done fairly quickly but does not take effect for the printers and file transfer. However, you may change the characters for the printer and for file transfer as described in "Changing the qconfig File for Printer NLS Support" on page 158.
2. You can change the two mapping tables in the `3270DH` file. The easy way is to transfer the file to a PC-DOS machine and make the changes with DOS debug or a hex editor. This solution is recommended only for experienced users, and it may take you quite a while before your tables work properly.

Changing the VTS Profile for NLS Support

The following is an example of how to change the VTS profiles for the console and the ASCII terminal to get support for the German national characters. This may be done in a similar way for the national characters of other countries.

The VTS profiles `ibm3151.s3270` for the console and `ibm3161.s3270` for the ASCII terminals, which can be found in the directory `/usr/lpp/r/comm/sna/3270/profiles` for the SNA version and in `/usr/lpp/r/comm/bsc/3270/profiles` for the BSC version, will be modified as shown in Table 7 on page 156.

Table 7. Mapping of the ASCII Characters for German Language Support with Network 3270-PLUS

German Character	EBCDIC Code (hex)	ASCII Code (hex) to be translated to	ASCII Code (hex) used with Console	Change mapping in VTS Profile	ASCII Code (hex) used with ASCII Terminals	Change mapping in VTS Profile
ä	C0	84	7B	yes	84	no
ö	6A	94	7C	yes	94	no
ü	D0	81	7D	yes	81	no
ß	A1	E1	7E	yes	E1	no
Ä	4A	8E	5B	yes	8E	no
Ö	E0	99	5C	yes	99	no
Ü	5A	9A	21	yes	21	yes
!	4F	21	5D	yes	5D	yes
§	7C	F5	40	yes	F5	no

Before you modify the profiles, make sure Network 3270-PLUS is stopped. Then:

1. Create a backup copy of the original VTS profile.
2. Modify the VTS profiles as described in Figure 57 on page 157 and Figure 58 on page 158.
3. Parse[19] the modified console profile with the command:

```
Termparse -v ibm5151.s3270 ibm5151.term
```

4. Parse the ASCII terminals profile with:

```
Termparse -v ibm3161.s3270 ibm3161.term
```

Next time you start the Network 3270-PLUS emulation, the modified profiles will be used.

[19] Parsing a profile is analogous to compiling a program. The parsing process must complete without errors, otherwise the changes you made are not valid. If you want to create VTS profiles with new names, be sure to add the new name in the profile `ttyprof.s3270`.

```
# @(#)ibm5151.s327        1.9
#
# vts terminal profile for Rabbit SNA 3270-PLUS on an ibm5151 (RT PC console).
   .
   .
   .
```

Insert the following modification statements before :Normalset

```
  :InByte ( 84, 7b )  modificication for ä
  :InByte ( 94, 7c )  modificication for ö
  :InByte ( 81, 7d )  modificication for ü
  :InByte ( e1, 7e )  modificication for ß
  :InByte ( 8e, 5b )  modificication for Ä
  :InByte ( 99, 5c )  modificication for Ö
  :InByte ( 9a, 21 )  modificication for Ü
  :InByte ( 21, 5d )  modificication for !
  :InByte ( f5, 40 )  modificication for §

:NormalSet
```

Insert the following modification statements after :Normalset

```
           character output translation...

  :OutByte ( 7b , 84 )  modification for ä
  :OutByte ( 7c , 94 )  modification for ö
  :OutByte ( 7d , 81 )  modification for ü
  :OutByte ( 7e , e1 )  modification for ß
  :OutByte ( 5b , 8e )  modification for Ä
  :OutByte ( 5c , 99 )  modification for Ö
  :OutByte ( 21 , 9a )  modification for Ü
  :OutByte ( 5d , 21 )  modification for !
  :OutByte ( 40 , f5 )  modification for §
   .
   .
   .
```

Remove the colon from the following three statements:

```
           graphic character rendition...

   OutByte ( "[" ,        bd  )  LEFT BRACKET displays as CENT SIGN
   OutByte ( "^" ,        aa  )  SHIFTED 6 displays as LOGICAL NOT SIGN
   OutByte ( "]" ,        b3  )  RIGHT BRACKET displays as VERTICAL BAR
   .
   .
   .
:End; of ibm5151.s3270
```

Figure 57. Modifying VTS Console Profile for German National Characters

```
# @(#)ibm3161.s327        1.8
#
# vts terminal profile for Rabbit SNA 3270-PLUS on an ibm3161 or ibm3163
   .
   .
   .
Insert the following modification statements before :Normalset

  :InByte ( 5D, 21 )  modificicication for Ü
  :InByte ( 21, 5D )  modificicication for !

:NormalSet

Insert the following modification statements after :Normalset

           character output translation...

  :OutByte ( 21 , 5d )  modification for Ü
  :OutByte ( 5d , 21 )  modification for !
   .
   .
   .
Remove the colon from the following three statements:

           graphic character rendition...

   OutByte ( "^" ,         "l"  )  SHIFTED 6 displays as LOGICAL NOT SIGN
   OutByte ( "[" ,         "L"  )  LEFT BRACKET displays as CENT SIGN
   OutByte ( "]" ,         "x"  )  RIGHT BRACKET displays as VERTICAL BAR
   .
   .
   .
:End; of ibm3161.s3270
```

Figure 58. Modifying VTS ASCII Terminal Profile for German National Characters

Changing the qconfig File for Printer NLS Support

You can change the ASCII characters for the printer by modifying the /etc/qconfig file as shown in Figure 59 on page 159. The idea is to replace the default printer backend with a shell script that first translates the characters and then passes the translated file to the default printer backend. In our example, output to the printer will be redirected to a shell script named /u/titus/redirect.script shown in Figure 60 on page 160.

For German NLS you need to translate the characters as shown in Table 8 on page 159. The translate strings used in Figure 61 on page 160 may also be used for file transfers by using the translate command *tr* in a similar way for the file to be transferred.

Note:

1. The modified printer stanza for lp0 is used for all printing via this queue. If you use it to print AIX files, those files will be translated as well which may not produce useful prints.

2. The translation from "!" to "Ü" should be done before the translation from "]" "!", to make sure that the original "!" is not also translated to "Ü".

3. The special characters "~", "@", "[", "]", "{", "}", "\" and "|" should ***not*** be used in the original print file.

The US translation table in the file /usr/lpp/r/comm/sna/3270/bin/3270DH translates the German special characters into certain US characters. For example, As shown in Table 8, the German character "ä" (EBCDIC X'C0') will be translated to "{" (ASCII X'7B'). The correct ASCII representation of "ä" is to ASCII X'84'. We need to do the reverse translation by the *tr* command in the shell script.

Table 8. Mapping of the ASCII Characters for German Language Support with Network 3270-PLUS

German Character coming from the Host	EBCDIC Code (hex) used by the Host	Character translated to by Network 3270-PLUS	ASCII Code (hex) used by Network 3270-PLUS	ASCII Code (hex) to be translated to by the file script	German Character after Translation by the file script
ä	C0	{	7B	84	ä
ö	6A	|	7C	94	ö
ü	D0	}	7D	81	ü
Ä	4A	[	5B	8E	Ä
Ö	E0	\	5C	99	Ö
Ü	5A	!	21	9A	Ü
ß	A1	~	7E	E1	ß
!	4F	]	5D	21	!
§	7C	@	40	F5	§

You modify the /etc/qconfig file like this:

1. Change the stanza for the printer ***lp0*** file /etc/qconfig as shown in Figure 59.

2. Create the shell script /u/titus/redirect.script as shown in Figure 60 on page 160. You can give the script file any name you want, but it must match with the entry in /etc/qconfig. The character strings used by the *tr* command to perform the retranslation to the German national characters are shown in Figure 61 on page 160.

3. Modify the permission code of the file /u/titus/redirect.script to executable, by typing:

```
chmod +x /u/titus/redirect.script.
```

```
lp0:
         argname = lp0
         friend = false
         discipline = fcfs
         device = dlp0

dlp0:
         file = /dev/lp0
         backend = /bin/sh /u/titus/redirect.script
```

Figure 59. Modified qconfig File for Printer Character Translation

The file script /u/titus/redirect.script looks as shown in Figure 60.

```
tr "string1" "string2" <$1 >/u/titus/redirect.tmp
/usr/lpd/piobe -pname=lp0 -device=/etc/ddi/sprinter /u/titus/redirect.tmp
```

Figure 60. The Shell Script for Printer Character Translation

The character strings used by the *tr* command are shown in Figure 61.

```
string1 = !~]@'\133''\134'{'\174'}

string2 = Üß!§ÄÖäöü
```

Figure 61. Character Strings used by the tr Command

Note: The characters "[", "\", and "|" will be interpreted as control characters by the *tr* command and cannot be used. Therefore, you have to use the octal code of these characters preceeded by a "\" character. This means, use:

\133	instead of [
\134	instead of \
\174	instead of |

Changing the ASCII/EBCDIC Translation Tables

This section gives you an example of how to change the ASCII/EBCDIC translation tables in the /usr/lpp/r/comm/sna/3270/bin/3270DH file for Icelandic character support with Network 3270-PLUS.

The problem for the Icelandic character support is that Network 3270-PLUS uses a 7-bit code page. Characters with an ASCII value above 127 are not allowed. Of the two translation tables in /usr/lpp/r/comm/sna/3270/bin/3270DH one of translates 7-bit ASCII codes to EBCDIC, the other translates EBCDIC codes to 7-bit ASCII. The two translation tables only contain 128 characters, but to support Icelandic characters, 256 characters are needed.

Solution

First of all, the tables were changed to reflect the fact that the Icelandic EBCDIC code page is different from the US code page. For example, we had to move the character "!" to a different position in the EBCDIC code page. The US EBCDIC code page is 037, the Icelandic EBCDIC code page is 871. After modifying the tables to fit EBCDIC code page 871, the following changes were made:

1. Icelandic national language characters are mapped to 7 bit ASCII characters. To do that 20 special US characters are sacrificed. Most of these special characters are seldomly used, anyway. The substitution is shown in Table 9 on page 161.

   The translation of the special characters to Icelandic characters is done by using the :lnbyte keyword in the file ibm5151.s3270 as shown in Figure 62 on page 162. The same :lnbyte keyword sequences can be used for ASCII terminals.

2. The redefined 7-bit characters are then translated to code page 871 as shown in the table in Figure 63 on page 164. This table maps characters in

the range 0 to 127 to the full 256-character range supported by AIX/RT. It was necessary to change the position of the twenty redefined characters shown in Table 9 on page 161.

3. When an EBCDIC character is returned, it is changed back into a 7-bit ASCII character as shown in the table in Figure 64 on page 164. This table maps characters in the range from 64 to 255 to the range from 0 to 127. The table "mirrors" the changes shown in Figure 63 on page 164.

4. Finally, the 7-bit characters are translated to the correct Icelandic characters in ASCII code page 850 by using the `:Outbyte` statement in the file `ibm5151.s3270` as shown in Figure 62 on page 162. The same `:Outbyte` keyword sequences can be used for ASCII terminals.

Table 9. Mapping of the ASCII Characters for Icelandic Language Support with Network 3270-PLUS

Icelandic Character	**Special US Character sacrificed**
I'D'	@
I'P'	[
Á	\
Æ	]
Ó	^
É	`
Í	{
Ó	|
Ú	}
I'Y'	~
á	^O
é	^N
í	^P
ó	^R
ú	^T
I'y'	^U
I'd'	^X
I'p'	^V
æ	^W
ö	^Y

The expression "I'x'" in the above table stands for an Icelandic character, which can not be printed here. It means the following:

I'D' = eth Icelandic capital
I'd' = eth Icelandic small
I'P' = thorn Icelandic capital
I'p' = thorn Icelandic small
I'Y' = Y acute capital
I'y' = y acute small

The translation of the special US characters to the Icelandic special characters is shown in Figure 62.

```
# @(#)ibm5151.s327        1.9
#
# vts terminal profile for Rabbit SNA 3270-PLUS on an ibm5151 (RT PC console).
   .
   .
   .
:NormalSet
```

Insert the following modification statements after :Normalset

```
          character output translation...

  :OutByte ( 40 , "I'D'" )  modification for eth Icelandic capital
  :OutByte ( 5b , "I'P'" )  modification for thorn Icelandic capital
  :OutByte ( 5c , "Á" )  modification for Á
  :OutByte ( 5d , "Æ" )  modification for Æ
  :OutByte ( 5e , "Ö" )  modification for Ö
  :OutByte ( 60 , "É" )  modification for É
  :OutByte ( 7b , "Í" )  modification for Í
  :OutByte ( 7c , "Ó" )  modification for Ó
  :OutByte ( 7d , "Ú" )  modification for Ú
  :OutByte ( 7e , "I'Y'" )  modification for Y acute capital

  :OutByte ( "\^n" ,  "é" ) modification for é
  :OutByte ( 0f , "á" )  modification for á
  :OutByte ( 10 , "í" )  modification for í
  :OutByte ( 12 , "ó" )  modification for ó
  :OutByte ( 14 , "ü" )  modification for ü
  :OutByte ( 15 , "I'y'" )  modification for y acute small
  :OutByte ( 0b , "I'd'" )  modification for eth Icelandic small
  :OutByte ( 18 , "I'p'" )  modification for thorn Icelandic small
  :OutByte ( 16 , "æ" )  modification for æ
  :OutByte ( 19 , "ö" )  modification for ö
   .
   .
   .
```

Figure 62 (Part 1 of 2). Modifying VTS Console Profile for Icelandic National Characters

```
Insert the following modification statements after :InByte for NEW LINE is']'

  :InByte ( "I'D'" , "@" ) modificication for eth Icelandic capital
  :InByte ( "I'P'" , "[" ) modificication for thorn Icelandic capital
  :InByte ( "á" , 5c ) modificication for Á
  :InByte ( "Æ" , "]" ) modificication for Æ
  :InByte ( "Ö" , "^" ) modificication for Ö
  :InByte ( "É" , "`" ) modificication for É
  :InByte ( "Í" , "{" ) modificication for Í
  :InByte ( "Ó" , "|" ) modificication for Ó
  :InByte ( "Ú" , "}" ) modificication for Ú
  :InByte ( "I'Y'" , "~" ) modificication for Y acute capital

  :InByte ( "é" , "\^n" ) modificication for é
  :InByte ( "á" , 0f ) modificication for á
  :InByte ( "í" , 10 ) modificication for í
  :InByte ( "ó" , 12 ) modificication for ó
  :InByte ( "ú" , 14 ) modificication for ú
  :InByte ( "I'y'" , 15 ) modificication for for y acute small
  :InByte ( "I'd'" , 0b ) modificication for eth Icelandic small
  :InByte ( "æ" , 16 ) modificication for æ
  :InByte ( "I'p'" , 18 ) modificication for thorn Icelandic small
  :InByte ( "ö" , 19 ) modificication for ö
   .
   .
   .
:End; of ibm5151.s3270
```

Figure 62 (Part 2 of 2). Modifying VTS Console Profile for Icelandic National Characters

The same modifications as in `ibm5151.s3270` need to be made for the ASCII terminal profiles.

The expression "I'x'" stands for the Icelandic character, which can not be printed here. It means the following:

I'D'	= Icelandic "eth" capital
I'd'	= Icelandic "eth" small
I'P'	= Icelandic "thorn" capital
I'p'	= Icelandic "thorn" small
I'Y'	= Y acute capital
I'y'	= y acute small

The two modified translation tables in the file `/usr/lpp/r/comm/sna/3270/bin/3270DH` are shown in Figure 63 and in Figure 64. They are shown as displayed by a hex editor.

```
22C30                            00 1E 00 00 19 60 60 60         .....```
22C40  60 60 15 60 0C 0D 51 45  55 60 CE 60 DE 8D C0 D0  ``.``..QEU`.`....
22C50  79 A1 60 60 1C 60 1E 60  40 4F 7F 7B 5B 6C 50 7D  y.``.`.`@0.{[1P}
22C60  4D 5D 5C 4E 6B 60 4B 61  F0 F1 F2 F3 F4 F5 F6 F7  M]\Nk`Ka........
22C70  F8 F9 7A 5E 4C 7E 6E 6F  7C C1 C2 C3 C4 C5 C6 C7  ..z^L~no|.......
22C80  C8 C9 D1 D2 D3 D4 D5 D6  D7 D8 D9 E2 E3 E4 E5 E6  ................
22C90  E7 E8 E9 4A 65 5A 5F 6D  71 81 82 83 84 85 86 87  ...JeZ_mq.......
22CA0  88 89 91 92 93 94 95 96  97 98 99 A2 A3 A4 A5 A6  ................
22CB0  A7 A8 A9 75 EE FE AD 60
```

Figure 63. Modified ASCII/EBCDIC Translation Table for Icelandic National Characters

```
22D00                            20 2D 2D 2D 2D 0F 2D 2D          ----.--
22D00  2D 2D 5B 2E 3C 28 2B 21  26 0E 2D 2D 2D 10 2D 2D  --[.<(+!&.---.--
22D10  2D 2D 5D 24 2A 29 3B 5E  2D 2F 2D 2D 2D 5C 2D 2D  --]$*);^-/---\--
22D20  2D 2D 2D 2C 25 5F 3E 3F  2D 60 2D 2D 2D 7B 2D 2D  ---,% >?-`---{--
22D30  2D 18 3A 23 40 27 3D 22  2D 61 62 63 64 65 66 67  -.:3@'="-abcdefg
22D40  68 69 2D 2D 2D 15 2D 2D  2D 6A 6B 6C 6D 6E 6F 70  hi---.---jklmnop
22D50  71 72 2D 2D 2D 2D 2D 2D  2D 19 73 74 75 76 77 78  qr-------.stuvwx
22D60  79 7A 2D 2D 2D 7E 2D 2D  2D 2D 2D 2D 2D 2D 2D 2D  yz---~----------
22D70  2D 2D 2D 2D 2D 2D 2D 2D  16 41 42 43 44 45 46 47  ---------.ABCDEFG
22D80  48 49 2D 2D 2D 2D 12 2D  17 4A 4B 4C 4D 4E 4F 50  HI----.-.JKLMNOP
22D90  51 52 2D 2D 2D 2D 14 2D  2D 2D 53 54 55 56 57 58  QR----.---STUVWX
22DA0  59 5A 2D 2D 2D 2D 7C 2D  30 31 32 33 34 35 36 37  YZ----|-01234567
22DB0  38 39 2D 2D 2D 2D 7D 2D                           89----}-
```

Figure 64. Modified EBCDIC/ASCII Translation Table for Icelandic National Characters

Reference Publications for This Chapter

Publications relevant for Network 3270-PLUS are:

IBM RT SNA Services Guide and Reference, SC23-2009
Network 3270-PLUS (SNA) User's Guide, SC23-0825
Network RJE-PLUS (SNA) User's Guide, SC23-0828

Transmission Control Protocol/Internet Protocol (TCP/IP)

Only two vendor independent protocols are generally accepted by many vendors and users. They are ***Open System Interconnect*** (OSI) and ***Transmission Control Protocol/Internet Protocol*** (TCP/IP). Although most users and vendors would agree that OSI will be the strategic multivendor protocol, the interest and growth of TCP/IP is expected to continue because of the relative maturity of TCP/IP compared to OSI. Since it is expected that users will be able to migrate to OSI, investment in TCP/IP continues.

Overview

TCP/IP is a set of communication protocols that have evolved since the late seventies. TCP/IP has been used in environments requiring multivendor connectivity and communications. Major users have been agencies of the U.S. Department of Defense and many of the major universities. In these cases, it is used both within an agency or university as well as between them. More recently, as the number of vendors supporting TCP/IP has increased, a larger number of users have started examining the use of TCP/IP as a general purpose multivendor connectivity option.

Included in TCP/IP are connectivity functions for both local area networks (LANs) and wide area networks (WANs). This includes a routing capability to route across both the local- and wide area network. A standardized addressing procedure is used for the major TCP/IP networks to insure uniqueness of addresses, thus permitting connectivity between enterprises. This collection of TCP/IP interconnected networks is known as the ***Internet***. Given the proper authority, a user on any of these standard TCP/IP networks can communicate to users on any of the other networks.

In addition to the connectivity functions, there are some higher level communications applications such as terminal emulation, file transfer, and electronic mail (among others) that are a part of TCP/IP.

It is possible with TCP/IP to have connectivity between different vendors' systems but not to have all the communication applications a user may require. Each vendor's implementation and the set of functions supported must be compared by the customer to insure their requirements are met. The terminal emulation function is a case in point as IBM supports both full screen 3270 operation as well as line-by-line operation, while many other vendors will only support line-by-line mode.

Protocol Layering

The term TCP/IP is often used as a generic name for a suite of protocols, all based upon the ***Internet Protocol*** (IP). IP defines the protocol for transmission of packets across a network, including the precise contents of packets, packet processing rules and error handling. IP does not guarantee delivery of packets. This task is left to protocol layers on top of IP.

The basic packet is called a ***datagram***. When passing through the network, datagrams may be fragmented and de-fragmented, depending on the capabilities of the hosts that are handling the datagrams on their way to the destination host. The IP layer of hosts on an Internet are capable of forwarding datagrams with no need for higher levels of protocols to interfere or even know about it.

IP is normally considered the second layer of the Internet Protocol model. The lowest level is the physical transport layer, which is dependant on the physical media used to connect the hosts. In the IP level, it is totally transparent what physical connection datagrams travel over. When datagrams are forwarded via several hosts they do indeed very often travel across different types of connections. The user can use any physical transmission approach including both wide area- and local area networks. Whatever is used, however, must be supported by all the devices that will communicate with each other. The most common protocols at this level have been X.25 and Ethernet. IBM supports these as well as Token-Ring and, on some implementations, asynchronous connections.

No TCP/IP implementation can get away with defining only one protocol layer on top of IP. A basic set of next-level protocols must be provided if the common set of functions is to be provided. These third-layer protocols include:

1. ***Transmission Control Protocol*** (TCP) for reliable packet delivery
2. ***User Datagram Protocol*** (UDP) for unreliable, connectionless delivery of packets
3. A series of ***Gateway Protocols*** to handle the routing of datagrams across gateways
4. ***Internet Control Message Protocol*** (ICMP) that allows for the exchange of control (and error) messages between the IP layers of hosts.

In this publication, we shall not discuss the capabilities of each of the third-layer protocols but merely state that they are implemented on all IBM implementations of TCP/IP. We shall also point out that the term TCP/IP, as used in this publication, refers to all these protocols. We shall only distinguish between them when actually required.

Internet and Internet Routing

The Internet Protocol (IP) layer of TCP/IP provides for a routing capability between networks and between systems on each network. This can be on a local- or a wide area basis. It can be between different parts of the same enterprise or between enterprises. Every IP host has a unique Internet address, which is a combination of a centrally assigned network ID and a locally administered local host address. This permits the routing of messages between, as well as within, enterprises as all hosts have unique addresses.

Internet addresses are of the form www.xxx.yyy.zzz with each field being an 8 bit binary number but usually specified as 4 decimal numbers. So that all users do not have to know these internal Internet addresses, a higher level naming method is used called ***domain naming***. These higher level names can be looked up in ***domain nameservers*** to find the actual Internet address. When messages are actually sent across the network, only the Internet address would be used.

Internet addresses consist of two parts. One is the network address and the second is the host ID. Companies that expect to use TCP/IP only internally to their own company may assign their own network and host ID's. If an organization expects to communicate with other enterprises over one of the TCP/IP networks that are collectively called ***Internet***, the enterprise needs a unique network address.

Both Internet standard network addresses and domain names are assigned by SRI[20] to insure uniqueness and thus permit routing between major enterprises. For example, IBM as an enterprise has several Internet network addresses assigned and has a domain name of "ibm.com". The IBM domain is a part of the domain directory called "com" which contains all commercial users. "edu" and "gov" are two other major user communities. Within IBM there would be a directory at "ibm.com" that would have the Internet addresses for locations within "ibm.com" such as "austin.ibm.com" or "raleigh.ibm.com". Within either the "ibm.com" directory or a separate directory at "austin.ibm.com" there would be entries for the hosts at each location. The domain nameserver function, which is supported in most IBM implementations of TCP/IP can search using these nested directories to find an Internet address.

Addresses within a network must be administrated by the local network owner. The network address assigned by SRI can be further subdivided within the enterprise to a specific local area network within the enterprise. Then the hosts (that is, an IBM RT, an ES/9370, an IBM Personal System/2) are assigned an Internet address and a domain name within the local network.

TCP/IP Architecture

TCP/IP is a peer-to-peer network. All systems, whether they are a large host system or an intelligent workstation, have the same network characteristics. Large host systems will generally have a more complete set of functions than intelligent workstations, but they are all peers from the network's perspective.

As we have already discussed, the TCP/IP architecture is based on a layered protocol. It consists of four layers as shown in Figure 65, where some common protocols are shown at the third layer.

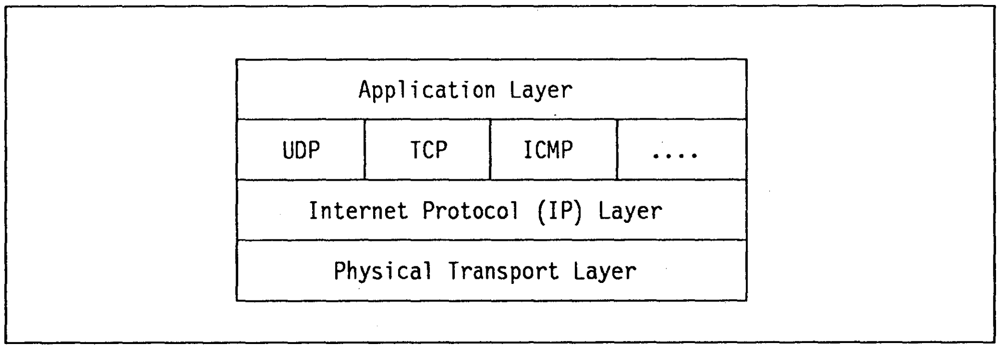


Figure 65. TCP/IP Protocol Layers

The lowest layer is the physical and is not actually specified by TCP/IP. At the next level is the Internet Protocol (IP). This level provides for routing of ***packets*** from one device to another. This could be from one local device to another, or it could be across a wide area network with many intervening networks that

[20] DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025

must be crossed. At the third level, we shall discuss two protocols, TCP and UDP, and the application layer protocols using each.

Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) provides for reliable transmission of data. At the lower layers, all transmissions are in individual packets. The TCP level involves both error checking and the disassembly/reassembly of packets to/from logical messages.

Above the TCP level are several application level protocols. Certain key protocols are implemented in nearly all TCP/IP implementations, while others are implemented in only a few. The application protocols, ***TELNET*** (terminal emulation), ***SMTP*** (Simple Mail Transfer Protocol), and ***ftp*** (File Transfer Protocol) have been available for several years. There are also a number of newer applications that IBM supports such as ***rexec*** (remote execution) and ***rlogin*** (remote login).

User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) provides for delivery of datagrams without providing the guaranteed delivery service of TCP. Protocols based upon UDP must implement their own means of checking for delivery, retransmission and error recovery, if so desired.

The application layer protocols that are based upon UDP include ***DOMAIN*** (Domain Name Protocol for the cross referencing between Internet addresses and host/domain names), ***TFTP*** (Trivial File Transfer Protocol for simple file transfers), ***X-Windows*** (remote graphics display), ***NFS*** (Network File System) and ***YP*** (Yellow Pages).

Network Management

The architecture defines no network management function and there is no defined methodology for forwarding network management alerts. However, when TCP/IP is used on a Token-Ring, all the normal Token-Ring network management functions are available as Token-Ring network management is independent of the higher level protocol. This applies as well to environments using an SNA Network as to the transport mechanism for TCP/IP.

TCP/IP Products

IBM has TCP/IP product offerings for many operating environments. TCP/IP is an integral part of the AIX/RT and can be ordered as separate products for other environments as follows:

- TCP/IP for AIX PS/2, program number 5713-AEW
- IBM TCP/IP for MVS, program number 5685-061
- IBM TCP/IP for VM, program number 5798-FAL
- IBM TCP/IP for PS/2, program number 5871-AAA
- IBM AIX Access for DOS Users, program number 5709-030
- IBM X-Windows for DOS Users, program number 5709-029.

TCP/IP Applications

Remote Login

telnet is a terminal emulation protocol that permits users to access applications in other systems. The basic telnet support includes line mode support, but there are also a number of full screen extensions that have been implemented. Whenever both client and server systems do not support the same full screen extensions, a line mode of operation will be used.

The *telnet* command permits a user to access applications on another system as if directly attached to that system. There is a client (user) and a server (remote host) relationship where the user's local host system must support the client portion and the target host must support the server portion. AIX, MVS and VM software products include client and server support, while TCP/IP for PS/2 provides only client support.

Most IBM telnet client implementations (AIX/RT, AIX PS/2, MVS, VM, and TCP/IP for PS/2) and servers (MVS and VM) support full screen extensions permitting 3270 access to MVS and VM. The protocol name associated with these extensions is called ***tn3270***.

rlogin (remote login) is another command that allows remote login between AIX and other UNIX based systems. Properly configured, *rlogin* allows remote login without supplying a password explicitly.

Simple Mail Transfer Protocol (SMTP)

SMTP, the ***Simple Mail Transfer Protocol***, is a protocol used for electronic mail with both client (sender) and server (receiver) portions. In most cases, users would use a mainframe or mid-range system for mail handling. Users of PC or IBM Personal System/2 workstations would typically use a terminal access method, (local, SNA, BSC, telnet) to gain access to MVS/VM hosts to create the mail and then send/receive the mail to/from the mail destinations.

The user does not directly use SMTP, but rather creates the mail to be sent and then uses the SMTP facility to send the mail. The user in an AIX environment would use the *mail* or *sendmail* commands or the series of subcommands of the *MH* (mail handler) system. The user in a VM environment would use either VM commands (note/sendfile) or the IBM Professional Office System (PROFS). In a VM environment, it is generally recommended that the user also use PROFS Extended Mail.

In an MVS and VM environment, users can send and receive mail across both TCP/IP and SNA/BSC networks. VM includes an interface to RSCS and MVS uses NJE. These systems may also be used as a gateway for mail that is created on another TCP/IP, VM SNA/BSC, or MVS SNA host. The user destinations for the mail may include users on both the TCP/IP and RSCS/NJE networks. This is possible because of the gateway function that permits mail to be sent over either TCP/IP and RSCS/NJE networks.

An associated protocol, the ***Post Office Protocol*** (POP), can be used to move mail from a UNIX or AIX system, which acts as "receiver" for a user. Workstation support such as TCP/IP for PS/2, which includes the sender capability, could use an alternate system as its mail receiver. This would be used when a user does not have an MVS or VM system available to be the mail recipient.

This user could use an SMTP client protocol at the workstation to create and send mail but use an AIX system as the mail recipient. The AIX systen would hold the mail until the user using POP requests that the mail be downloaded to the workstation. This is analogous to "general delivery " mail, where the mail is held until it is requested.

In a VM environment, the user has the option of using the VM commands note/sendfile or VM PROFS. The user can send mail to RSCS and TCP/IP users using the same commands. PROFS Extended Mail Facility should be used for added ease of use.

Users on other TCP/IP systems may send mail that can be received by TCP/IP for VM for users on that system. The VM TCP/IP system may also forward mail across the RSCS network if the destination address is for an RSCS node. Again, the remote TCP/IP user would send a copy to the VM TCP/IP system, which would forward the mail to the RSCS destinations.

Users sending mail across both RSCS and TCP/IP networks should use PROFS Extended Mail (5798-FBJ), which provides some ease-of-use features to facilitate sending mail across non-RSCS networks including interfaces to TCP/IP SMTP and OSI X.400. The PROFS Extended Mail program provides the ability to use TCP/IP user names in the note/sendfile and/or in the PROFS NAMES files. The appropriate TCP/IP and RSCS instructions will be added so that users will not have to add any additional information for messages or replies, making TCP/IP and RSCS differences transparent to the user.

On AIX systems, users have the ability to start a direct conversation with another user, using the *talk* command. This command opens a send and receive window on each user's terminal and displays the partner's message as it is typed.

File Transfer Protocol (FTP)

The ***File Transfer Protocol*** (FTP) provides a capability to access files and directories on remote hosts. Files may be transferred to/from the remote host. Password protection is provided as a part of the protocol.

With FTP there is a client/server relationship between the user initiating the request and the remote system the user is accessing. The server function may require the user to provide passwords. On AIX systems the standard password prompting is used, and FTP only works if the user at the remote system has a password defined. In TCP/IP for VM, passwords are defined at the VM minidisk level to provide user access control. Both in MVS and VM, RACF may be used to control access.

FTP includes functions for display and control so users with read/write access can display/define/delete files and directories. Protocol conversion between ASCII and EBCDIC can be performed. TCP/IP on MVS, VM, and AIX provide client and server functions, while TCP/IP for PS/2 has the client function only.

TFTP, ***Trivial File Transfer Protocol***, is another protocol which may be used for file transfer. TFTP is based upon UDP and does not include all the function of FTP such as security access control or display and control over directories. TFTP clients and servers are supported as a part of the TCP/IP for PS/2 product permitting exchange of files directly between IBM PCs and PS/2s.

X-Windows

X-Windows provides an application program interface (API) which allows a client program access to a bit-mapped, high resolution display connected to a system running an X-Windows server program.

The X-Windows API allows the development of code which is portable across operating systems and displays. An application using the X-Windows client function communicates with the X-Windows server function on the remote system.

X-Windows client and server function is supported on AIX/RT and AIX PS/2, while AIX/370 supports only the client function. For the VM environment, CMS applications using the X-Windows function will be written in C and require the IBM C for System/370 program offering (5713-AAH) and the IBM VS PASCAL Library (5668-717).

TCP/IP is used as the communication protocol between the client and the X-Windows server system. X-Windows is written to the UDP interface. PC DOS users may use the X-Windows for DOS Users program. This program includes the required TCP/IP networking functions as a part of the product.

Network File System (NFS)

The ***Network File System*** (NFS) provides file server support for the NFS 3.2 protocols developed by Sun Microsystems Inc. On all AIX systems both client and server function is supported. On VM and MVS, this support enables a VM or MVS system to act as a file server for other systems that have the NFS 3.2 client function installed. In VM, optional encryption of files requires IBM Information Protection System Cryptographic Programs for VM/CMS (5796-PPK) or a customer supplied encryption procedure. Neither MVS nor VM include the Network File System client function.

NFS is a separate feature of VM and MVS TCP/IP products. It is a separate product for all AIX implementations. Those products are:

- AIX/RT NFS, program number 5601-159
- AIX PS/2 NFS, program number 5713-AFG
- AIX/370 NFS 3.2, program number 5688-046.

Remote Execution

Several different protocols have been devised to allow users and applications on one system to invoke procedures and application execution on other systems. This can be useful for a number of environments, including the off-loading of many compute-intensive routines in engineering and scientific applications.

rexec, remote execution client, and *rexecd*, remote execution server (daemon) is one of these protocols. As an example in an AIX environment, this permits a user on a remote host to invoke a shell script and receive the results back at that remote host.

rsh (remote shell) allows the execution of commands directly at remote AIX systems without an explicit login.

AIX/370 and AIX Access for DOS Users both implement the *on* command that allows direct execution of a command at any host in the RCF cluster (AIX/370) or the host that serves the requestor (AIX Access for DOS Users).

AIX-unique Applications

A function provided only by AIX/370 and AIX PS/2 is the ***Transparent Computing Facility*** (TCF). It permits a cluster of up to 31 AIX/370 and AIX PS/2 systems to be constructed with central administration of user logons/passwords, as well as system resources across an entire cluster. The cluster appears as one system to the user. Systems may be dynamically added and removed from the cluster allowing flexibility. Files may be replicated to increase availability and performance. AIX manages the synchronization of the data, including the resychronization after a file has been restored to a cluster. TCF is also supported using a channel-to-channel protocol between AIX/370 systems.

Another AIX-unique function is the announced (but not yet available) Version 1.3 of Distributed Services that runs via IP, and may be used between single hosts and TCF clusters. Distributed Services allows transparent file access, print and batch server functions. Full record level locking is provided. One or more single system images may be created, permitting (but not requiring) the user to customize different systems views. DS 1.3 is nor compatible with earlier versions of DS.

IBM TCP/IP System Connectivity

IBM provides TCP/IP support for host operating environments, as well as support for IBM workstations. IBM supports a wide range of connectivity options in these environments.

AIX TCP/IP supports Ethernet, Token-Ring, Asynchronous and X.25 connections (the latter not for AIX/370) and may use integrated adapters on the IBM RT, the IBM Personal System/2 and ES/9370. The IBM 8232 LAN Channel Station is supported for channel attachment to host systems running AIX/370.

The MVS TCP/IP support includes Token-Ring, Ethernet, PC Network, and X.25 (including DDN). The IBM 8232 LAN Channel Station and IBM 3720, 3725, and 3745 Communication Controllers may be used on 3090, 4300 and ES/9370 systems using MVS.

In a VM operating environment, integrated adapters may be used on the ES/9370 for the Token-Ring, Ethernet, and X.25 including DDN. Channel attached devices may be used on 309x, 43xx, and the ES/9370 and include the IBM 8232 LAN Channel Station. TCP/IP for VM includes support for the IBM 7170 Data Acquisition Control Unit and IBM S/1 (supports DDN X.25).

TCP/IP for PS/2 supports Token-Ring, Ethernet, and PC Network. TCP/IP for PS/2 supports both PS/2 and IBM PC machines.

IBM 8232 LAN Channel Station

The ***IBM 8232 LAN Channel Station*** provides for attachment of local area networks to a host via a System/370 Channel. The LAN Channel Station unit supports up to two IBM 7532 Industrial Computers that each support up to two LAN connections to a host channel. IBM 8232 LAN Channel Station supports connectivity between IBM Token-Ring, PC Network, Ethernet and the 30xx, 43xx or 937x host processors. The TCP/IP products for MVS, VM, and AIX/370 provide users with the capability of participating in a TCP/IP network. The IBM 8232 LAN Channel Station may also be used to support interfaces using other networking protocols. Interfaces have been announced to use both MAP 2.1 (Manufacturing Automation Protocol) and PVM (Pass-Through VM).

The IBM 8232 LAN Channel Station permits S/370 host processors to channel-attach to various local area networks (LANs). This subsystem consists of:

1. IBM 8232 LAN Channel Station Model 1 or 2
2. IBM 7532 Industrial Computer Model 266
    a. 1 per IBM 8232 Model 1
    b. 2 per IBM 8232 Model 2
3. IBM LAN Channel Control Program

The IBM LAN Channel Support Program provides the required software for the IBM 8232 operation and interfacing to either MVS/VM TCP/IP or AIX/370. In addition, IBM PC DOS Version 3.3 (or above) is required for each IBM 7532. When using either an IBM Token-Ring or IBM PC Network, the IBM LAN Support Program is also required for each 7532 installed in the 8232 LAN Channel Station.

Each 7532 has its own channel attachment, so it is possible for an 8232 Model 2, which has two 7532s, to be attached to two systems with each having independent control over the LAN(s) attached to the respective 7532. In this way, the IBM 8232 Model 2 can be shared between any of the five host protocols/environments.

Each channel attached IBM host will require one or more of the following:

1. IBM TCP/IP for VM Program Offering (5798-FAL)
2. IBM TCP/IP for MVS Program Offering (5685-061)
3. IBM VM MAP 2.1 Program Offering (5798-FBG)
4. IBM VM/Pass-Through PC Connect Facility 1.0 (5799-CRJ).
5. IBM AIX/370 Program (5713-AFL).

The AIX/370 and TCP/IP for VM can also share a single 7532; however, two separate LAN adapters would be required as the adapters can not be shared.

AIX System Connectivity

AIX/370 using either the IBM 8232 or the IBM ES/9370 integrated adapters supports attachment to either an IBM Token-Ring or an Ethernet LAN. AIX/RT and AIX PS/2 both support the IBM Token-Ring and Ethernet using integrated adapters. AIX/RT and AIX PS/2 also support X.25 and Asynchronous interfaces.

Both AIX/370 and AIX/RT include TCP/IP as a base feature. TCP/IP for AIX PS/2 is a separate product. Several of the application functions are included in the base, but others are also packaged separately.

In addition to the AIX products, there are application functions that operate in an IBM PC DOS environment. IBM AIX Access for DOS Users users permits a PC DOS workstation user to access an AIX system. Some of the functions provided are VT100 terminal emulation, virtual print, and virtual disk. The AIX system must have the ***AIX DOS Server*** program installed. The AIX system and the DOS system may be interconnected using a TCP/IP network. IBM AIX

Access for DOS Users has the required TCP/IP interfaces as a part of the system. Both Ethernet and Token-Ring is supported. IBM X-Windows for DOS Users also can use both Token-Ring and Ethernet LANs.

TCP/IP for IBM Personal System/2

IBM Personal System/2 and IBM PC workstations can be attached to IBM Token-Ring, IBM PC Network, or Ethernet local area networks using TCP/IP. When a workstation is being used as an end user system, the workstation could be attached to any of the above LANs. The workstation could alternatively function as a router, which is a means of interconnecting one or more LANs. Depending upon the model of the workstation, up to 5 local area networks can be connected using a router.

MVS TCP/IP Connectivity

TCP/IP for MVS supports a wide range of connectivity options. The IBM 8232 LAN Channel Station may be used for support of local area networks including Token-Ring, Ethernet, and IBM PC Network (broadband).

The 37x5 Communication Controllers may be used for X.25 including use with the DDN networks and Public Data Networks, (PDN). The 37x5 NCP/NPSI support would be used when communicating directly to a X.25 network, such as one of the DDN networks. NCP/NPSI with SNA Interconnection XI could be used to communicate directly to an external machine using an X.25 interface.

IBM MVS (and VM) systems may also use the ***SNA Network Link*** function of IBM's TCP/IP products to communicate between TCP/IP systems using an SNA network.

VM TCP/IP Connectivity

TCP/IP for VM supports a wide range of connectivity options. The IBM 8232 LAN Channel Station may be used for support of local area networks. The IBM S/1 may be used in support of X.25 DDN. VM systems may also use the SNA Network Link to communicate to TCP/IP systems using an SNA network. The IBM ES/9370 provides additional connectivity options using the integrated adapters for Token-Ring, Ethernet and X.25 (including DDN and PDN, Public Data Network).

SNA Interconnection (XI)

At the enterprise level, most organizations have developed their own wide area network. In many cases, this network is based upon IBM's SNA architecture. The IBM SNA Interconnection (XI) function of the IBM Communication Controllers (3745, 3720, 3725) can be used to permit non-SNA communications traffic to use the same telecommunications facilities as the SNA applications.

The interface to IBM SNA Interconnection (XI) is an X.25 interface. This permits dissimilar systems to share a common physical network (the SNA backbone) as long as they support an X.25 interface.

The XI function permits an enterprise to build a single physical wide area SNA network and share that physical network with TCP/IP and/or other SNA application protocols. This allows the SNA backbone to consolidate physical networks and provide for Network Management of all wide area communication.

SNA Network Link

In addition to using SNA Interconnection (XI) to provide for consolidation of physical networks, it is also possible to use the ***SNA Network Link*** function of MVS and VM TCP/IP software. This permits MVS and VM host processors to route TCP/IP messages between them. The TCP/IP messages may have originated from a TCP/IP application on either system or may have been routed to it from another TCP/IP host system. The MVS or VM host systems may be either the originator, destination, or simply a router of the messages.

This facility permits each host to set up a logical SNA session with other host systems. These paired systems establish a communications path between them that the TCP/IP systems at each location can use to communicate between the "paired" locations. Each host can have multiple sessions. Each session represents another set of paired systems.

As an example, there could be four locations each with an IBM MVS or VM host. Each host could have three SNA sessions, one to each of the other locations.

At locations that do not have an IBM host, the SNA Interconnection (XI) function can be used to route information to an IBM host. That IBM host could use the SNA Network Link function to route to other systems in the network.

Bridges and Routers

When a TCP/IP host sends a message to another host on the same LAN, it can do so directly using the addressing method used by the LAN. If the host already knows that address, it can send the message immediately, but if the address is unknown, the requesting host must first use a LAN broadcast mechanism to find the LAN address that corresponds to a given TCP/IP address, (TCP/IP addresses are referred to as Internet addresses). TCP/IP hosts will cache addresses to reduce the frequency of broadcasts.

If there are multiple Token-Ring LANs, they can be interconnected using IBM bridges. The broadcast mechanism will locate the other host across those bridges and will also retain the route information (path information about bridges) to reduce the frequency of broadcasts.

Local Area Networks can also be interconnected using Routers. Routers operate using the TCP/IP Internet addresses so they are protocol dependent. Bridges are independent of the protocol because they operate using LAN addresses. The advantage of the bridge is that its protocol independency allows other protocols (for example Netbios, 3270 Emulation, LU6.2 and IEEE 802.2) to run across it as well. The Internet Router only supports the TCP/IP protocols.

When multiple LANS are used in establishments due to either the size of the environment or because of the use of mixed LAN technologies, there will be a backbone LAN that will serve as the transport between the local departmental LANs. Bridges and routers are used to interconnect the backbone LAN and the departmental LANs.

Heterogeneous Networks

In almost all cases when TCP/IP networking and application functions are used, there will also be other networking systems that are being used. TCP/IP will frequently be the networking protocol used to communicate between the various heterogeneous systems and networks.

Some of the aspects relative to TCP/IP and a heterogeneous network environment involve the following:

1. OSI and TCP/IP relationship
2. SNA and TCP/IP relationship
3. LAN Technologies - Token-Ring/Ethernet.

The following sections address the above subjects.

OSI and TCP/IP Architectures

The reason many users have an interest in TCP/IP is to provide connectivity in a multivendor environment. This is also the objective of the International Standards Organization (ISO) ***Open Systems Interconnect*** (OSI) reference model.

Although TCP/IP is currently more mature and has much wider vendor support, it is anticipated that most TCP/IP systems will eventually migrate to OSI. OSI will potentially offer several advantages in that there will be many more network and application components to choose from to satisfy a wider range of application requirements, and it is expected that there will be better vendor interoperability because of compliance testing. Because many of the TCP/IP networks are quite large and are both inter- and intra-enterprise, it can be expected that this migration may take many years.

TCP/IP and OSI have several comparable functional layers even though there is a varying number of layers in the two architectures. There are three different levels at which it is best to compare and contrast the two architectures (physical network, logical network, and application).

At the lowest levels, it should be possible to share the same physical networks (X.25, Token-Ring or Ethernet LAN) because many of the same protocols are used. This permits network or media sharing but does not provide any communication functions between TCP/IP and OSI systems.

At the logical network level, OSI's levels 3 and 4 (transport and network) are comparable to TCP/IP's "TCP" and "IP" layers. (NOTE: OSI includes many options at each level, but they are comparable when using a specific set of OSI options).

At the TCP/IP application level, there are again comparable options with specific OSI upper levels. Most comparable of these functions would be electronic mail (SMTP to X.400), file transfer (FTP to FTAM), and terminal emulation (telnet to VTP). In most cases, OSI will be richer in function than the TCP/IP equivalent.

TCP/IP and SNA

Unlike the comparison of OSI and TCP/IP, where users are most often concerned about eventual migration, in comparing TCP/IP and SNA it is most useful to examine coexistence. The use of gateways between the networks, the use of shared facilities that both can access, and the sharing of physical network resources are all possible.

Gateway functions:

Terminal Emulation	TCP/IP users can use the *telnet* command to access an IBM system, and then from that system use VTAM to access applications across the SNA backbone. Likewise, a SNA VTAM user can access a system that has both VTAM and TCP/IP, and from that system use *telnet* to access applications across the TCP/IP network.
Electronic mail	An IBM TCP/IP system can be an electronic mail gateway between an SNA and TCP/IP network.
File transfer	FTP can not be used to directly transfer files across an SNA network, but FTP commands can be executed from a command processor so procedures can be developed to transfer files from an TCP/IP system to a gateway system and then to a SNA system.

Shared resources: The Network File System for MVS and VM permits access to files that are in host format. These same files that are accessed using NFS can be accessed by other application users on the SNA network.

Shared Physical Network: SNA permits the sharing of the physical network resources using either SNA interconnection (XI) which is a program that can be used on IBM Communication Controllers (3745, 3725, and 3720), or using the SNA Network Link facility of MVS and VM TCP/IP. This would permit consolidation of physical networks.

The sharing of LAN resources is also possible. When using the IBM Token-Ring/IBM PC Network, both TCP/IP and other protocols may be used in the IBM PC and PS/2 workstations. Applications written to the Netbios interface, such as IBM LAN Program can coexist and share the same adapter. When using the IBM Token-Ring/IBM PC Network, the user may use the IBM LAN Manager and NetView/PC on the LAN. These network management products are independent of the higher level protocols that are used (TCP/IP, Netbios, LU6.2, or link level IEEE802.2 protocols) on individual workstations on the LAN.

LAN Technologies

IBM supports both the use of Ethernet and Token-Ring for the TCP/IP local network. In comparing the two LAN types, it is useful to examine both the architectures and the application functions that are supported. As LAN's get larger, it is more important to examine both the reliability and ability to recover automatically from failure conditions as a part of the total network management of the LAN.

The IBM Token-Ring guarantees a level of service to all users which an Ethernet cannot. The contention access system of CSMA/CD (Carrier Sense Multiple Access/Collision Detect) cannot guarantee a level of service to any station. The level of service is controlled by the number of active users, length of messages, length of network, and location of users.

Although standard Ethernet has a clock data rate of 10 megabits per second (mbps), it can not sustain a rate over 4 mbps for standard-specified distances and packet lengths. For some applications, it may be no more than 1 - 2 mbits. The reason it can not achieve an effective rate any higher is due to the CSMA/CD access method. This access method also has the characteristic that the higher the clock rate the lower the percentage utilization of the total bandwidth that is available for useful work. As such, it is not practical to consider running a CSMA/CD LAN at higher clock rates. The Token-Ring access method does not have this characteristic. A 4 mbps Token-Ring LAN can sustain data rates of 3.8 mbps or more.

The protocols that are used on the LAN's vary as well. Netbios is a very common protocol for PC to PC communication. TCP/IP has evolved as a very common protocol between all types of systems (hosts, mid-range, and intelligent workstations) on local area networks.

Another area of major difference between LANs is fault isolation, recovery, and network management capabilities. The basic design of the Token-Ring includes the ability for every station to generate an alert if either hard or soft errors occur between itself and its neighbor stations. Using this information along with topology information, which is automatically maintained, permits fault isolation to a single segment of the ring. Because of built-in recovery and redundancy, recovery is usually automatic as well.

The IBM LAN Manager supports the Token-Ring with the use of NetView/PC. This provides for the reporting of alerts to a NetView host as well as to the LAN manager.

AIX Family Implementations of TCP/IP

All AIX implementations of TCP/IP provide the same basic set of functions which can be subdivided in the following groups:

1. System administrator commands
2. TCP/IP servers (daemons)
3. End-user commands.

System Administrator Commands

The following system administrator commands are available on AIX implementations of TCPIP:

arp	Displays or changes the Internet address to hardware address translation tables used by the ***Address Resolution Protocol*** (ARP).
host	Shows the Internet address of a specified host.
hostid	Sets or displays the identifier of the local host.
hostname	Sets or displays the Internet name and address of the local host.
ifconfig	Configures network interfaces and their characteristics. The default command to configure the network interface in AIX PS/2 and AIX/370. In AIX/RT, the initial configuring of the network interfaces

must be done with the command *netconfig*. On subsequent executions of *netconfig*, this command invokes *ifconfig* for most functions.

netconfig	Configures network interfaces (adapters) to be used by TCP/IP according to the information in the /etc/net file. See the description of *ifconfig*.
netstat	Shows local or foreign addresses, routing tables, hardware statistics, and summary of packets transfered.
route	Adds or deletes information in the routing tables of a gateway. Routing information created with this command is known as ***static routing***.
ruptime	Displays status information about hosts that are connected to local networks and are running the *rwhod* daemon.
setclock	Reads the time from a timeserver on the network and sets the time and date of the local host accordingly.
trpt	Performs protocol tracing on sockets.

TCP/IP Servers (Daemons)

The following daemons provide the basis for TCP/IP on the AIX implementations:

gated Provides gateway routing functions. /etc/gated.conf is the configuration file for this daemon. *gated* or *routed* must be running on systems that provide routing functions to remote networks.

inetd A generic deamon that will invoke other servers (daemons) when requests for services are received. When *inetd* is started, it reads the file /etc/inetd.conf which specifies what servers should handle each service type and what protocol (TCP, UDP) is used for the service. The services described in the file must match those described in the file /etc/services where the relation between IP ports and service type is defined. When a request is received, *inetd* schedules the function that can handle the request based upon the information obtained from /etc/inetd.conf and uses the port number defined in /etc/services for IP packets. The *inetd* daemon is often called the ***super daemon***.

Many daemons that had to be started directly from the file /etc/rc.tcpip in earlier versions of TCP/IP are now started by *inetd*. To see a list of daemons started by *inetd*, check the file /etc/inetd.conf.

lockd A generic deamon that runs on both server and client systems and provides record and file locking for remotely mounted files. This daemon will be provided with Network File System for AIX/370, but not for other current or announced versions of NFS on AIX systems.

lpd Provides the server function for remote printing. This daemon will use two files:

1. /etc/host.equiv (names of hosts allowed to execute commands and print), and
2. /etc/hosts.lpd (names of hosts allowed to print only).

	/usr/spool/lpd is the directory where control information and data files for remote print are kept.
named	Provides the server function for the domain nameserver protocol. /etc/named.* and /etc/resolv.conf are files used by this daemon. The domain nameserver is available on all AIX implementations of TCP/IP and is the only nameserver function available on AIX/RT if Yellow Pages is not installed and configured for nameserver function.
portmap	Unlike most TCP/IP applications, RPC uses only one well-known port number. The *portmap* daemon listens on that port and maps RPC program numbers to port numbers on RPC servers. The *portmap* daemon is part of TCP/IP, but is currently required only for Network File System and any applications using RPC.
routed	Manages network routing tables. Needs information in /etc/gateways and/or /etc/networks. *gated* or *routed* must be running on systems that provide routing functions to remote networks. The *gated* server should be preferred over *routed* whenever possible.
rwhod	Sends broadcasts to all other hosts every 3 minutes to update a local data base with information about logged-in users and network status. It also listens to and answers such broadcasts when received from *rwhod* running on other hosts. This daemon is a real "time waster" and can steal considerable amounts of cpu ressource and network bandwidth. It should be used with extreme care. The data collected by *rwhod* can be displayed by *rwho* and *ruptime*.
sendmail	Provides the server function for sending and receiving remote mail. This daemon is not really part of TCP/IP but rather of Basic Networking Utilities (BNU).
statd	A generic deamon that runs on both server and client systems and assists *lockd* in providing record and file locking for remotely mounted files. This daemon will be provided with Network File System for AIX/370, but not for other current or announced versions of NFS on AIX systems.
tftpd	Server function for the Trivial File Transfer Protocol. To serve this protocol, the *tftpd* daemon must be uncommented in /etc/inetd.conf on the serving host.
syslogd	Server function for reading and logging system messages. This daemon uses the file /etc/syslog.conf to determine what logging to do and what log device/file to use.
uucpd	Server function for using the Basic Networking Utilities over an Internet network. See "Basic Networking Utilities (BNU)" on page 29 for more information.

End-User Commands

File Transfer

ftp	Allows transfer of files using the file transfer protocol (FTP). File transfers may take place between similar or dissimilar hosts. Numerous subcommands of *ftp* are available, the most common being:

cd	Change working directory on remote host
get	Read a file from remote host
lcd	Change working directory on local host
mget	Read several files from remote host
mput	Write several files to remote host
prompt	Toggle prompted mode
put	Write a file to remote host
quit	Terminate *ftp* session.

rcp	Transfers data between similar hosts. This command is a network implementation of the AIX *cp* command.
tftp	Implements the trivial file transfer protocol (TFTP). Its only function is to read and write files to and from a foreign host.

Remote Mail and Conversations

talk	Allows users on the same or different hosts to have an interactive conversation. *talk* opens both a send window and a receive window on each user's display. Each user is able to type into the send window while *talk* displays what the other user is typing.
mail	The command to invoke the basic mail facilities of AIX. The *mail* command is not part of TCP/IP but is often used to send or retrieve mail via TCP/IP. See "AIX Mail" on page 267 for details about the *mail* command.
MH	The mail handler (MH) commands complements the *mail* command. See "AIX Mail" on page 267 for information about the mail handler.

Remote Login, Command Execution and Printing

finger	Returns information about users on a specified host.
lprbe	Can be used directly by users to send print files to a remote host but is usually used as a printer backend by the *print* command in AIX.
ping	Sends an echo request to a network host to determine whether that host is operational and on the network. Flags to the *ping* command allows repetitive echo requests to be sent, the size of packets to be set, etc. The command can provide statistics about round-trip timings.
rdist	Maintains identical copies of files over multiple host. This command is only available on AIX PS/2.
rexec	Executes a single command on a similar foreign host.
rlogin	Allows a user to log in to a similar foreign host.
rsh	Used to open a shell that allows you to execute commands on a similar foreign host.
rwho	Displays status information for users on remote hosts when the remote hosts are running the *rwhod* daemon.

telnet The *telnet* command is a terminal emulation program that allows you to log in to remote hosts whether similar or dissimilar to your local host.

Installing TCP/IP on AIX

On all AIX platforms, TCP/IP is installed with the *installp* command. The following is an overview of the steps to install the product:

1. For AIX PS/2 and AIX/RT, insert the communications adapter you will be using (see Appendix H, "IBM RT Hardware Installation" on page 531 for details). Then run diagnostics to verify that there are no DMA-, interrupt- or address conflicts in your system. If you see any conflicts, correct the adapter settings using the reference diskette on a PS/2 or by moving jumpers on the adapter if an IBM RT. When there are no more conflicts, take a note of the adapters settings for use when you define the adapter to the AIX system with the *devices* command.
2. On the IBM RT, use the *installp* command to install the VRM device drivers for the adapters you'll be using.
3. Using the *installp* command, install TCP/IP from the distribution media. On AIX PS/2 and AIX/370, you'll be prompted for the ***Internet address*** and the primary network interface for TCP/IP. The host name of your host defaults to the name you specified when you installed the AIX system.
4. Use *updatep* to apply any updates to your system.
5. On an IBM RT, use the *devices* command to add the ***adapter*** you will be using for TCP/IP[21]. On AIX PS/2, you need only use the *devices* command if you will use more than one adapter for TCP/IP.
6. Run *devices* and add ptys (pseudo-terminals). You need to add the ptys if you want to use *telnet*, *rlogin*, *rsh* or *X-Windows* across the TCP/IP network.

   On AIX/RT and AIX/370, you need to add one set of pseudo terminals for *telnet* and another for *rlogin*, *rsh* and *X-Windows*. This is not required for AIX PS/2.

   AIX/RT For *telnet*, pseudo terminals must be defined with:

   ```
   ae true
   logger true
   tt hft
   ```

   The number of ptys you add determine the maximum number of *telnet* sessions remote hosts can have active to your host simultaneously. You may want to use the value dumb for tt if you expect *telnet* to be used from terminals on remote hosts rather than from the console of remote hosts. In that case, users running *telnet* from the console of remote hosts can execute the following commands (assuming the Bourne shell is used):

[21] On AIX/RT, although not required for TCP/IP you may also want to add the ***data link*** corresponding to the adapter so you won't forget if you use this adapter for SNA Services at a later time. It doesn't hurt TCP/IP so we recommend you follow this practice.

```
TERM = hft
EXPORT TERM
```

For *rlogin*, *rsh* and *X-Windows* you must use:

```
ae false
logger false
tt hft
```

AIX PS/2 You need to add ***asynchronous pseudo terminals*** *(ttyps)* and you should use:

```
ae false
logger false
tt hft
```

no matter what the use of the pseudo terminals is.

AIX/370 You need to add pseudo terminals. Define them using the rules you used for AIX/RT.

A Practical Example

We shall now take you through the steps to install and customize a small network with several different connection types:

- Token-Ring
- Ethernet
- X.25
- Asynchronous connection.

Using the following hosts:

- IBM RT ***m135*** with AIX/RT 2.2.1, two Token-Ring Adapters and one Baseband Adapter (Ethernet) to be used as gateway between three networks and as domain nameserver, print server, time server, Network File System server and Yellow Pages server. One Token-Ring Adapter is connected to the local Token-Ring network; the other is connected to another small Token-Ring network that acts as a logical extension of a remote Token-Ring network through a ***Split Bridge*** running on a PS/2.
- IBM RT ***chris*** with AIX/RT 2.2.1 connected to Token-Ring.
- IBM RT ***gatex*** with AIX/RT 2.2.1, one Token-Ring Adapters and one IBM PC X.25 Communications Adapter to be used as gateway between a remote host (***sys1***) and the local Token-Ring network.
- IBM RT ***sys1*** with AIX/RT 2.2.1 and an IBM PC X.25 Communications Adapter.
- PS/2 Model 80 ***dosnc*** with a Token-Ring Adapter and running either AIX Access for DOS Users or X-Windows for DOS Users.
- PS/2 Model 80 ***ps2nc*** with AIX PS/2 1.1, one Token-Ring Adapter and one Ethernet adapter. We will define this host as having a secondary hostname ***ps2nceth*** when addressed via Ethernet.
- IBM /9370 Model 60 ***aix370*** with AIX/370 (pre-release version) connected to the network via Ethernet. The same machine runs TCP/IP for VM via a Token-Ring connection using the host name ***vm9370***.
- IBM RT ***sys2*** with AIX/RT 2.2.1 and an asynchronous connection to the host ***sys3*** via modem eliminator.

- IBM RT ***sys3*** with AIX/RT 2.2.1 Baseband Adapter and an asynchronous connection to the host **sys2** via modem eliminator.

The configuration is shown in Figure 66.

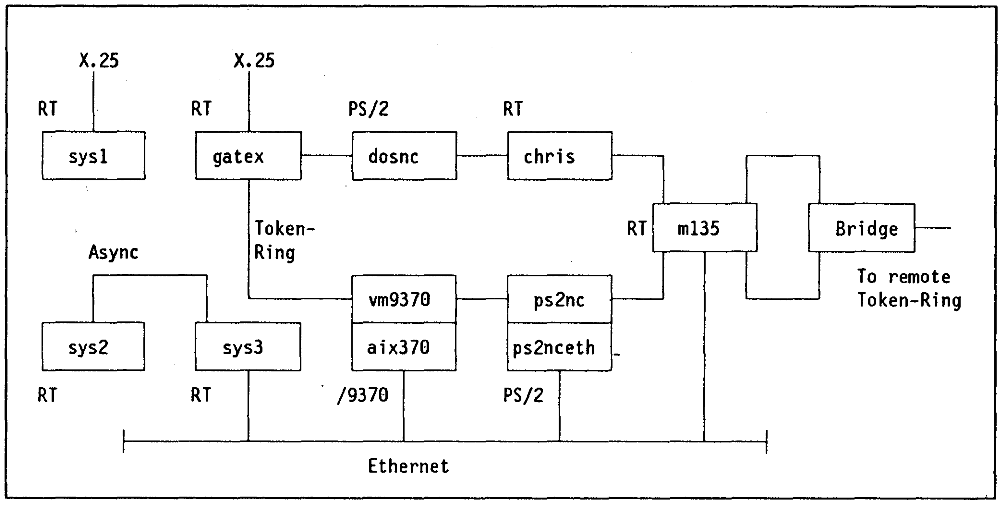


Figure 66. Sample Configuration for TCP/IP

We will do the customizing in steps, starting by defining the hosts connected to the local Token-Ring and Ethernet networks and setting up static routing between the two local networks. This first step will not define nameserver functions. Later we'll explain how to add interfaces for other network connections and how to implement nameserver- and gateway functions.

Please do ***not*** expect us to explain every aspect of the installation and customizing procedure. You must have the TCP/IP publication for each type of AIX host you want to use and consult that publication for details.

All installation and customizing should be done while logged in as ***superuser***.

Basic AIX TCP/IP Customizing

We have already ("Installing TCP/IP on AIX" on page 182) given you an overview of how to install TCP/IP on an AIX system. We will go a little more in detail with some of those steps but mostly concentrate on the further customizing of each machine.

The procedures described below take for granted that you want to use almost all of the TCP/IP functions available. If you don't want, say, print server functions, skip over the steps describing how to customize for print server, etc.

Customizing TCP/IP on AIX/RT

Since the host ***m135*** has Token-Ring and Ethernet adapters installed, let's start by customizing that host for all the LAN adapters.

Customizing Host m135

1. Two Token-Ring Adapters are installed in host ***m135***. They must be installed with different adapter jumper settings. One adapter acts as the primary adapter, the other as the secondary. The Baseband Adapter is installed with default jumper settings.

2. Use the *devices* command to add the three adapters as: `token0`, `token1` and `net0` respectively.

3. As a precaution[21], use the *devices* command to add the three data links: `trllc0`, `trllc1` and `ethllc0`

4. Run `devices` and add four pseudo terminals with `ae true` and `logger true` for use by *telnet* and add at least three pseudo terminals with `ae false` and `logger false` for use *rlogin*, *rsh* and *X-Windows*.

5. Edit the /etc/hosts file. You must insert entries for our local host and, since we are not using a nameserver in this first customizing, a line for each host on our local networks:

```
# Internet Address Hostname # Comments

127.0.0.1       localhost      # loopback (lo0) name/address

#       The local Token-Ring network
9.3.1.7         m135              #m135 on local Token-Ring
9.3.1.4         chris
9.3.1.6         gatex
9.3.1.12        ps2nc
9.3.1.15        dosnc
9.3.1.100       vm9370

#       The local Ethernet network
192.48.11.7     m135              #m135 on local Ethernet
192.48.11.3     sys3
192.48.11.12    ps2nceth
192.48.11.100   aix370

#       The remote Token-Ring network
129.35.22.32    m135              #m135 on remote Token-Ring
```

Figure 67. The /etc/hosts File on Host m135

The Internet addresses used are the ones of the ITSC in Austin. You must, of course, insert the appropriate Internet addresses for your local installation. Note the three different forms of addresses:

9.3.1.x	This is a ***Class-A*** Internet address because the first element has a value lower than 127. The initial **9** is the network address; the remaining elements of the address give the local address.
129.35.22.x	This is a ***Class-B*** Internet address because the first element has a value greater than 127 and less than 192. The network address is ***129.35*** so the last two elements can be used to identify a local host.

192.48.11.x This is a ***Class-C*** Internet address because the first element has a value greater than 191. The network address is ***192.48.11*** so only the last element can be used to identify a local host.

We shall later see how a particular Internet address class can be divided into subnets.

6. Edit the /etc/master file and change the following entries in the **sysparms** stanza:
   a. To retain the node name over kernel rebuilds, change the **node** name. On ***m135*** change to:

      ```
      node = "\"m135\""
      ```

   b. For each Ethernet, X.25, and 802.3 interface used for TCP/IP, add 1 to each of the **procs** and **kprocs** keywords values.
   c. For each Token-Ring interface that will be used for TCP/IP, add 2 to each of the **procs** and **kprocs** keywords values.
7. Edit the /etc/rc.tcpip file and do the following:
   a. Insert the node name in the *hostname* command:

      ```
      /bin/hostname m135 > /dev/console
      ```

   b. Because this host will be used as print server, uncomment the line that starts the *lpd* daemon.
8. For each host that may use ***m135*** as a print server, add a line to the file /etc/hosts.lpd. In our case, we want all local hosts to use the print server, so we edit the file like this:

```
chris
gatex
ps2nc
dosnc
vm9370
sys3
ps2nceth
aix370
```

Figure 68. The /etc/hosts.lpd File on Host m135

9. Edit the /etc/rc file and check that lines to run the /etc/rc.include shell script are uncommented.
10. Edit the /etc/rc.include file and uncomment the lines to run the /etc/rc.tcpip shell script when the system starts.

[22] On AIX/RT, the fields ***inetlen*** and ***r_inetlen*** should be changed to be no more than 1500 bytes. 1500 bytes is the maximum data portion of datagrams supported by TCP/IP on AIX/RT. The default values assume the fields are specified including headers, but actually the values are interpreted as the length of the data portion.

If you need to connect to non-IBM hosts, you should adjust the ***inetlen*** and ***r_inetlen*** values so they do not exceed those of the other hosts.

11. Edit the /etc/net file to insert stanzas for the three adapters we will be using. You'll see that the /etc/net file is where you specify the relation between Internet address and adapter: On the host ***m135***, we will use three adapters, so we need the following entries.

```
token0:
        netaddr = 9.3.1.7
        inetlen = 1500[22]
        r_inetlen = 1500
        localbroadcast = true

token1:
        netaddr = 129.35.22.32
        inetlen = 1500
        r_inetlen = 1500
        localbroadcast = false

net0:
        netaddr = 192.48.11.7
        inetlen = 1500
        r_inetlen = 1500
```

    The primary Token-Ring Adapter is used for the local Token-Ring network. Note the option ***localbroadcast***. It specifies whether link level broadcasts shall be propagated through Token-Ring bridges. We have said ***true*** for the local Token-Ring because we don't want broadcasts to go through any bridge that may be connected to the local ring.

    For the remote Token-Ring, we have specified ***false*** because we will need the ability to access the remote Token-Ring network through a Token-Ring Split Bridge.

12. Rebuild the kernel by executing the following commands, so that the changes to /etc/master take effect:

```
cd /usr/sys
make
mv /unix /unix.old
mv unix.std /unix
shutdown -rf
```

13. When ***m135*** has rebooted itself, TCP/IP is initialized and ready to use. With no other hosts customized yet, you can't do much, but you may want to check that TCP/IP is indeed running by saying:

```
ping m135
```

    The command will use the *loopback* entry in /etc/hosts to send a datagram to the local host, so the network interface is not tested, but you can see whether TCP/IP is active.

Customizing Host chris

To customize the host ***chris*** that has only a Token-Ring Adapter, use the steps used for host ***m135***. There are a few differences, though, so let's go through it:

1. Since only one Token-Ring Adapter is installed, use default jumper settings.
2. Add only one adapter: token0 and only one data link: trllc0[21] with the *devices* command.
3. Run devices and add pseudo terminals as for the host ***m135***.

4. Edit the /etc/hosts file and add a line for the local host and each remote host that ***chris*** must be able to use TCP/IP for. If you also want to use the ***time server*** and ***print server*** functions of TCP/IP, add a line for each so the file looks like this:

```
# Internet Address Hostname # Comments

127.0.0.1       localhost       # loopback (lo0) name/address

#        The local Token-Ring network
9.3.1.7         m135              #m135 on local Token-Ring
9.3.1.7         timeserver        #m135 as time server
9.3.1.7         printserver       #m135 as print server
9.3.1.4         chris
9.3.1.6         gatex
9.3.1.12        ps2nc
9.3.1.15        dosnc
9.3.1.100       vm9370

#        The local Ethernet network
192.48.11.3     sys3
192.48.11.12    ps2nceth
192.48.11.100   aix370
```

Figure 69. The /etc/hosts File on Host chris

Notice that ***m135*** is only defined on the local Token-Ring network and that the printserver and timeserver entries point to that host, again using the local Token-Ring.

5. Edit the /etc/master file as you did for the host ***m135***.

6. Edit the /etc/rc.tcpip file and make the following changes:

   a. Insert the node name in the *hostname* command:

   ```
   /bin/hostname chris > /dev/console
   ```

   b. Insert a *route* command for static routing to the local Ethernet which can only be reached through the host ***m135***:

   ```
   route add net 0 9.3.1.7
   ```

   Insert the command where advised in the file. Note that this format of the command defines a ***static default route*** meaning that all requests for Internet addresses outside the local network will be routed through the machine at Internet address 9.3.1.7. Since our local network is on a Class-A network, all requests bound for addresses that do not begin with **9** are routed through the host ***m135***.

   Provided we define the gateway host in the local /etc/hosts file we could specify the hostname of the gateway host rather than its Internet address.

   c. If you want the time to be set automatically from the time server during boot of your local host, add the following line as one of the last lines in /etc/rc.tcpip:

   ```
   setclock timeserver
   ```

7. Edit the /etc/rc file and check that lines to run the /etc/rc.include shell script are uncommented.

8. Edit the /etc/rc.include file and uncomment the lines to run the /etc/rc.tcpip shell script when the system starts.

9. Edit the /etc/net file and insert a stanza for Token-Ring Adapter as follows:

```
token0:
        netaddr = 9.3.1.4
        inetlen = 1500[22]
        r_inetlen = 1500
        localbroadcast = true
```

10. Because we want to use a remote print server, run the shell script /usr/lpp/tcpip/samples/prtsvr.inst to add the following stanzas to the /etc/qconfig file:

```
rp0:
        argname = rp0
        friend = TRUE
        device = drp0

drp0:
        backend = /usr/bin/lprbe -statusfile
```

11. Rebuild the kernel and reboot the system.

12. When ***chris*** comes back up, TCP/IP is initialized and ready to use. With ***m135*** already running TCP/IP, use the following command to check your customizing:

```
ping m135
```

Customizing Host sys3

To customize the host ***sys3*** for its Baseband Adapter, use the same steps as for host ***chris*** with a few exceptions:

1. We use a Baseband Adapter with default settings.

2. Add adapter: net0 and only one data link: ethllc0[21] with the *devices* command.

3. Run devices and add pseudo terminals as for the host ***chris***.

4. Edit the /etc/hosts file and add a line for the local host and each remote host that ***sys3*** must be able to use TCP/IP for plus ***time server*** and ***print server*** entries. The file could be:

```
# Internet Address Hostname # Comments

127.0.0.1       localhost      # loopback (lo0) name/address

#       The local Token-Ring network
9.3.1.4         chris
9.3.1.6         gatex
9.3.1.12        ps2nc
9.3.1.15        dosnc
9.3.1.100       vm9370

#       The local Ethernet network
192.48.11.7     m135               #m135 on local Token-Ring
192.48.11.7     timeserver         #m135 as time server
192.48.11.7     printserver        #m135 as print server
192.48.11.3     sys3
192.48.11.12    ps2nceth
192.48.11.100   aix370
```

Figure 70. The /etc/hosts File on Host sys3

Now, the host ***m135*** and the printserver and timeserver entries use the Ethernet Internet addresses since ***sys3*** is connected to the local Ethernet.

5. Edit the /etc/master file as you did for the host ***chris***.
6. Edit the /etc/rc.tcpip file and make the following changes:
   a. Insert the node name in the *hostname* command:

      ```
      /bin/hostname sys3 > /dev/console
      ```

   b. Insert a *route* command for static routing to the local Token-Ring which can only be reached through the host ***m135***:

      ```
      route add net 0 192.48.11.7
      ```

      This defines a ***static default route*** so all requests for Internet addresses other than the local network will be routed through the machine at Internet address 192.48.11.7. Since our local network is on a Class-C network, all requests bound for addresses that do not begin with **192.48.11** are routed through the host ***m135***.

   c. If you want the time to be set automatically from the time server during boot of your local host, add the following line as one of the last lines in /etc/rc.tcpip:

      ```
      setclock timeserver
      ```

7. Edit the /etc/rc file and check that lines to run the /etc/rc.include shell script are uncommented.
8. Edit the /etc/rc.include file and uncomment the lines to run the /etc/rc.tcpip shell script when the system starts.
9. Edit the /etc/net file and insert a stanza for Ethernet adapter as follows:

   ```
   net0:
           netaddr = 192.48.11.3
           inetlen = 1500[22]
           r_inetlen = 1500
   ```

10. Because we want to use a remote print server, run the shell script /usr/lpp/tcpip/samples/prtsvr.inst to update the /etc/qconfig file.
11. Rebuild the kernel and reboot the system.
12. When ***sys3*** comes back up, test the connection to the two systems already running with:

    ```
    ping m135
    ping chris
    ```

Customizing TCP/IP on AIX PS/2

In our example, we have only one AIX PS/2 host, namely the host known as ***ps2nc***. To illustrate how you can address a host using more than one Internet address and hostname, we have installed both a Token-Ring Adapter and a Ethernet adapter and will assign a different hostname to each adapter.

Customizing Host ps2nc

1. The Token-Ring Adapter must be installed before installing TCP/IP. After installation, boot with the reference diskette and configure the adapter into the system.
2. Now use *installp* to install TCP/IP. The Token-Ring Adapter and the primary Internet address are configured as we install TCP/IP, the Token-Ring Adapter is automatically added as a device and the file /etc/rc.tcpip is updated by the installation procedure.
3. Since we want to use static routing through the host ***m135***, edit the file /etc/rc.tcpip and add a *route* command where advised to in the file:

    ```
    route add 0 9.3.1.7 1
    ```

    Observe, that because we have two adapters installed and defined, this default route will only be used when we attempt to address networks with network addresses different from **9.x.x.x** and **192.48.11.x**. Access to the two LANs we are directly connected to does ***not*** go through the host ***m135***. Also, we could as well have given the *route* command the following looks:

    ```
    route add 0 192.48.11.7 1
    ```

    because we have direct access to the host ***m135*** through both adapters.
4. To add the second adapter and define it to TCP/IP, we must do the following:
    a. Shutdown the system and install the Ethernet adapter.
    b. Boot with the reference diskette and configure the adapter.
    c. Reboot AIX PS/2.
    d. Use *devices* to add the Ethernet adapter. The kernel will be rebuilt as part of the installation procedure, but don't reboot, yet.
    e. Edit the file /etc/rc.tcpip. Copy the line containing the *ifconfig* command. Then change the copied line to use the adapter ***net1*** and change ***$SITE*** to the secondary hostname, in our case ***ps2nceth***. On our host, the two *ifconfig* lines look like this:

        ```
        /etc/ifconfig tk0 $SITE
        /etc/ifconfig net1 ps2nceth
        ```

5. If you want the time to be set automatically from the time server during boot of your local host, add the following line as one of the last lines in /etc/rc.tcpip:

   ```
   setclock timeserver
   ```

6. Run devices and add four asynchronous pseudo terminals (ttyp) with ae false and logger false.

7. Edit the /etc/hosts file and add entries for remote hosts and the local Ethernet adapter. Also add print- and time server entries:

```
# Internet Address Hostname # Comments

127.0.0.1       localhost      # loopback (lo0) name/address

#        The local Token-Ring network
9.3.1.7         m135              #m135 on local Token-Ring
9.3.1.7         printserver       #m135 as print server
9.3.1.4         chris
9.3.1.6         gatex
9.3.1.12        ps2nc             #ps2nc on Token-Ring
9.3.1.15        dosnc
9.3.1.100       vm9370

#        The local Ethernet network
192.48.11.7     timeserver        #m135 as time server
192.48.11.3     sys3
192.48.11.12    ps2nceth          #ps2nc on Ethernet
192.48.11.100   aix370
```

Figure 71. The /etc/hosts File on Host ps2nc

   Just for the fun of it, the print server uses the Token-Ring address of ***m135*** while the timeserver uses the Ethernet address.

8. If you want to be able to use the *netconfig* command, edit the /etc/net file to add a stanzas for Ethernet adapter. The active stanzas in the file would then be:

   ```
   tk0:
           netaddr = 9.3.1.12

   net1:
           netaddr = 192.48.11.12
   ```

9. Because we want to use a remote print server, run the shell script /usr/lpp/tcpip/samples/prtsvr.inst to update the /etc/qconfig file.

10. Shut the system down and reboot.

11. When ***ps2nc*** is rebooted, TCP/IP is initialized and ready to use. Check the local hostnames with the following commands:

    ```
    host 192.48.11.12
    host 9.3.1.12
    ```

    If correctly customized, the two commands should return the two different hostnames defined for the host.

12. Check that everything works as intended, using the following commands:

a. From the local host, say:

```
ping m135
```

b. From hosts ***m135*** and ***chris*** say:

```
ping ps2nc
ping ps2nceth
```

Customizing TCP/IP on AIX/370

Our host ***aix370*** uses the Ethernet adapter on the IBM 9370 in our example. Let's see what we need to customize the host.

Customizing Host aix370

The following description assumes you are installing TCP/IP from magnetic tape as part of the AIX system.[23] We allow ourselves to completely ignore the (very likely) possibility of PS/2s participating as sites in an AIX cluster with the IBM 9370.

1. The host has two adapters installed. Both must be configured to VM.
2. Using *installp* install TCP/IP.
3. Run `devices` and add four pseudo terminals with `ae true` and `logger true` for use by *telnet*[24] and add at least three pseudo terminals with `ae false` and `logger false` for use by *rlogin*, *rsh* and *X-Windows*,
4. Edit the `/etc/systems` file and change the line `ipc43=0` to `ipc43=1` to enable IP support.
5. Edit the `/etc/hosts` file:

```
# Internet Address Hostname # Comments

127.0.0.1       localhost      # loopback (lo0) name/address

#        The local Token-Ring network
9.3.1.4         chris
9.3.1.6         gatex
9.3.1.12        ps2nc
9.3.1.15        dosnc
9.3.1.100       vm9370

#        The local Ethernet network
192.48.11.7     m135              #m135 on local Ethernet
192.48.11.7     printserver       #m135 as print server
192.48.11.3     sys3
192.48.11.12    ps2nceth
192.48.11.100   aix370
```

Figure 72. The /etc/hosts File on Host aix370

[23] The procedure described is based upon a pre-release version of AIX/370 and is not necessarely identical to the procedure you must follow when customizing the released version.

[24] It is generally discouraged to do remote logins to AIX/370 itself. An AIX cluster will normally have sites that are PS/2s running AIX PS/2. These sites handle remote logins more efficiently than the System/370 host can do because data stream handling one byte at a time is not what the mainframe is designed for.

You'll probably ***not*** want the time to be set from a time server, so only a print server entry is added to the file.

6. If you want to be able to use the *netconfig* command, edit the `/etc/net` file to add a stanzas for Ethernet adapter:

   ```
   net1:
           netaddr = 192.48.11.12
   ```

7. Edit the file `/etc/rc.tcpip` and change the line to run the *ifconfig* command to match the adapter you use. The first Token-Ring Adapter would be `tk0`; the first Ethernet adapter would be `ce0`. In our case, we insert:

   ```
   /etc/ifconfig ce0 $site
   ```

   On PS/2s participating in an AIX cluster we would have used `pg0` to specify the Ethernet adapter.

8. Because we want to use a remote print server, we must add stanzas to the `/etc/qconfig` file. The released version of AIX/370 may have the shell script `/usr/lpp/tcpip/samples/prtsvr.inst`; our version did not, so we changed the `/etc/qconfig` file with and editor, using the AIX/RT file as a model.

9. Shut the system down and reboot.

10. When ***aix370*** is rebooted, TCP/IP is initialized and ready to use. Check that everything works as intended, using the following commands:

    a. From the local host, say:

       ```
       ping m135
       ```

    b. From hosts ***m135*** and ***chris*** say:

       ```
       ping aix370
       ```

Installing and Customizing X-Windows for DOS Users

Installation and customizing of X-Windows for DOS Users is described in "X-Windows for DOS Users" on page 305. All we shall add at this point is to say that all hosts that will be running X-Windows client programs on our host ***dosnc*** should be defined to that host during the installation procedure. You will be prompted for the hostnames and Internet addresses of all such hosts and should specify all the hosts on the local Token-Ring network.

First, you are prompted for the default gateway host. For our configuration you should specify host ***m135*** since that host has access to all other, known hosts. Because ***dosnc*** is on Token-Ring, specify the Token-Ring Internet address of ***m135***.

Next you are prompted for other hosts. You may want to add hosts connected to Ethernet, but so far we haven't been able to run X-Windows client applications across the gateway host. Updates to X-Windows for DOS Users may change this.

Remote Printing with TCP/IP

We have already configured all systems to use the TCP/IP print server function on host ***m135***. The stanza added to /etc/qconfig on each system defined the queue ***rp0***. If you have no local printer on a host, you can move the two stanzas to the top of the /etc/qconfig file so you won't have to specify any print queue on the *print* command to print on the TCP/IP print server.

If you do have a local printer defined, you'd probably like the *print* command to default to that printer, so leave the /etc/qconfig file unchanged. To print on the remote printer you'd need to use the *print* command like this:

```
print rp0 file_to_print
```

The command is identical to *print* for a local printer, except for the ***rp0*** option. All options for the *print* command can be used. If you experience problems using remote printing, see "Remote Print Problems" on page 228.

Remote Login and Remote Command Execution

This section describes how to configure for and use remote login and remote command execution. If you followed the customizing steps outlined above, you have already added pseudo terminals for *telnet*, *rlogin* and *rsh*. If you didn't, you'll have to add three pseudo terminals to each system that shall accept *rlogin* and *rsh* and one or more on systems that shall accept remote logins with *telnet*. Remember, that on AIX/RT and on AIX/370 the pseudo terminals for *rlogin* and *rsh* must have:

```
ae false
logger false
```

The same settings must be used for all pseudo terminals on AIX PS/2 regardless of use. For use by *telnet* on AIX/RT and AIX/370, the pseudo terminals must be defined with:

```
ae true
logger true
```

The commands for remote login and remote command execution can be grouped after the requirements they place on configuration files on the requesting and/or serving host:

/etc/hosts.equiv: Checked by the daemons *rshd*, *rlogind* and *lpd*. If the requesting host is listed in the this file on the serving host, *rsh*, *rcp*, *rlogin* and remote print requests *(lprbe)* are accepted by the server. In the case of remote print, the requesting host could alternatively be listed in the file /etc/hosts.lpd.

If the requesting user is logged in with root authority and the remote host and user ID is not listed in the file $HOME/.rhosts on the serving host, *rsh* and *rcp* will be rejected and *rlogin* will prompt for a password

$HOME/.rhosts: When *rcp*, *rlogin* or *rsh* is used, and either the requesting user is logged in with root authority or the requesting host is not listed in the /etc/hosts.equiv file on the server, the serving host checks if the requesting host and user ID is listed in this file. If this is not the case, the commands *rcp* and *rsh* are rejected, while the requestor of *rlogin* is prompted for a password.

If the user ID of the requesting user exists on both hosts and the user ID is not overwritten from the command line, then the commands will succeed (*rlogin* without prompting for password) regardless of the presence of the files `/etc/hosts.equiv` and `/$HOME/.rhosts`.

$HOME/.netrc: This file is used to control the automatic login feature of *rexec* and *ftp*. This feature allows a user to use the commands without being prompted for user ID and password at the serving host. If an entry specifying the user ID and password to be used at the serving host is found in the file `$HOME/.netrc` on the ***requesting*** host, the prompting for user ID and password doesn't take place.

If the hosts are operating in controlled access mode or if an entry is not found in `$HOME/.netrc`, the requestor is prompted for the user ID and password to use on the serving system.

Note that the ***telnet*** command will always prompt the requesting user for the user ID and password to use on the serving system.

The control files and the commands that use them are shown in Figure 73.

Command	/etc/hosts.equiv	$HOME/.rhosts	$HOME/.netrc
telnet	no	no	no
rlogin	yes	yes	no
rsh	yes	yes	no
rcp	yes	yes	no
lprbe	yes	no	no
rexec	no	no	yes
ftp	no	no	yes

Figure 73. Files Controlling Remote Execution with TCP/IP

The directory `/usr/lpp/tcpip/samples` has samples of the `.rhost` and `.netrc` files that you may wish to use as a starting point when customizing.

Using the $HOME/.rhosts File

If, using our configuration as shown in Figure 66 on page 184, we want to allow the user ***jan*** to use *rlogin*, *rsh* and *rcp* from any AIX/RT system against the host ***m135*** without supplying a password, and if we also want to allow the user ***fribert*** to use the same commands from the host ***chris***, then the file `/u/jan/.rhosts` on ***m135*** should have the following lines:

```
sys1    jan
sys2    jan
sys3    jan
chris   jan
gatex   jan
```

Figure 74. Example of .rhosts File on Host m135

and the file `/u/fribert/.rhosts` on ***m135*** should look like this:

```
chris    fribert
```

Figure 75. Example of .rhosts File on Host m135

To execute the commands, the requesting users must be logged in as user ***jan*** or ***fribert*** respectively, or they must specify that user ID on the command line when entering the commands. Actually, since nither of the user IDs have root authority, they could also use the commands if the host ***m135*** had the five hosts listed in the file /etc/hosts.equiv.

The $HOME/.rhosts file should have permissions set to 600 (read and write by owner only) for security reasons and must be used by either root or the requesting user ID.

Using the /etc/hosts.equiv File

Users can use the commands *rlogin*, *rsh* and *rcp* without supplying a password if they are not logged in with root authority and provided the name of the requesting host is listed in the file /etc/hosts.equiv on the serving host. To allow all users on our Ethernet network to use these commands on host ***sys3*** you'd need to create the /etc/hosts.equiv file shown in Figure 76 on ***sys3***. Login with *telnet* always causes a prompt for user ID and password.

```
m135
ps2nceth
aix370
```

Figure 76. Example of /etc/hosts.equiv File on Host sys3

Using the $HOME/.netrc File

To execute without prompting for user ID and password, the commands *rexec* and *ftp* both require the file $HOME/.netrc on the ***requesting*** host to have information about user ID and password to use on the serving system.

For example, assume the user ***chris*** on host ***m135*** should be able to use the commands with superuser authority on host ***sys3*** and without supplying a password. Also assume that the user can use the commands with his own user ID on the hosts ***chris*** and ***gatex***. Then the file /u/chris/.netrc on host ***m135*** would contain:

```
machine sys3 login root password topsec
machine chris login chris password secret
machine gatex login chris password secret
```

Figure 77. Example of .netrc File on Host m135

The $HOME/.netrc file must have permissions set to 600 (read and write by owner only) and be owned by root or the requesting user to be valid. If this is not the case, the file is ignored.

Using the $HOME/.3270keys File

The *telnet* command uses one additional file when used for login in 3270 terminal emulation mode. The default file is /etc/3270.keys, but each user may have a customized version in $HOME/.3270keys.

The file is used to map the keyboard and to define the use of colors for 3270 emulation. A model file is supplied in /usr/lpp/tcpip/samples/3270key.rt on AIX/RT. Copy this file to your home directory as .3270keys and modify it if you need special keyboard mapping.

Peculiarities of telnet and ftp

For reasons we have not been able to explain, the *telnet* and *ftp* commands behave a little strangely with certain update levels of AIX/RT 2.2.1. Update IX1716 to TCP/IP removes some of these pecularities.

When using *telnet* from AIX PS/2 to AIX/RT and when the login user ID uses the bourne shell, the login never completes. You can overcome this by saying:

```
telnet -e none host_name
```

You can also eliminate the problem by editing two files: First, edit the file /etc/ports and insert the actual device type of your console display in the /dev/console stanza. For example, term = ibm6154

Secondly, edit the file /etc/profile and change the line saying: trap "" 1 2 3 to say: trap "" 2 3 and change the lines that set the terminal type environment variable to look like this:

```
if [ "$TERM" = "" ]; then
   TERM=`termdef`;
fi
```

When you use the *em* command of AIX Access for DOS Users to log into an AIX/RT host, you see the same problems as when logging in with *telnet* from AIX PS/2. You cannot specify options on the command line of *em* so only the second solution to the problem can be used.

When using *telnet* into AIX/370 from AIX/RT, the login does not complete. To overcome this, again use:

```
telnet -e none host_name
```

When using *ftp* to access a remote IBM RT, you'll be prompted for user ID and password, but the *ftp* prompt does not show up. All you need is to press ENTER one or more times to get the prompt. The number of times you must press ENTER depends on the length of your password.

Configuring for Nameservers

We have already defined our installation so that all three LANs are known. However, keeping the configuration updated has become quite a nuissance with hosts being added and removed all the time, so we've decided to introduce ***nameservers***.

On AIX/RT, the only nameserver protocol supported by TCP/IP is the ***domain name server*** protocol. You can still use the "***old style***" nameserver protocol on AIX PS/2 and AIX/370, but these systems also support domain nameservers. It is strongly recommended that you use the domain nameserver protocol whether you have IBM RTs in your network or not.

On systems with Yellow Pages installed, you have the option of using YP to provide nameserver service. We shall explain both ways to provide nameserver functions in the following sections. Before we proceed, we have to dwell with the concepts of ***domains, name resolution*** and ***reverse name resolution***.

TCP/IP Domains

In a small, isolated TCP/IP network, it's easy to identify all hosts with unique hostnames. As a network expands and, especially, when it gets connections to other, existing networks, things get more complicated. If you connect to ***Arpanet***, for example, you must qualify your local hostnames in some way.

The way is to use domain names. First, you'll have to define your own domain name. You must acquire a domain name that's unique to your organization and associated Internet addresses from SRI as described in "Internet and Internet Routing" on page 166. In our small configuration we use the domain name ***itsc.austin.ibm.com*** for the two local networks and the domain name ***austin.ibm.com*** for hosts on the remote Token-Ring network.

The full domain name of our host ***m135*** as addressed from any of the local networks is: ***m135.itsc.austin.ibm.com*** but when addressed from hosts on the remote Token-Ring it is: ***m135.austin.ibm.com***.

Name- and Reverse Name Resolution

Whenever a TCP/IP host needs to access another host, it must "look up" the address of that host. For example, if you type the command:

```
ping sys3
```

from the host ***chris***, then the local host must resolve the name into an Internet address, which determines what procedure to follow to actually send the request to host ***sys3***. We will refer to the process of obtaining an Internet address from a host name as ***name resolution***. If the remote host is on the same physical network as your local host, the Internet address found is mapped to the actual adapter address and the request sent directly.

If the Internet address points to a different physical network, the request is sent to the gateway machine where the name resolution process is repeated. This can continue across many gateways until the request can be delivered on a local interface or no route is found.

In some situations, it is required that an Internet address can be translated (resolved) to a hostname. For example, when you use remote printing, the print request carries the Internet address and the hostname of the requesting host. The requesting host is only allowed to print if the file `/etc/hosts.lpd` (or `/etc/hosts.equiv`) contains the hostname of that host.

The checking is done by trying to resolve the Internet address of the requesting host into a hostname and then comparing the result with the names in /etc/hosts.lpd. We will refer to the translation of an Internet address into a hostname as ***reverse name resolution***.[25]

Name resolution is done by the system call **gethostbyname**; reverse name resolution through the system call **gethostbyaddr**. Whenever a TCP/IP request is processed and the exact Internet address and/or hostname of the remote host is unknown or cannot be relied upon, TCP/IP issues one of the above system calls. In systems without YP services, the request is resolved from the local /etc/hosts file if a nameserver is not active, otherwise from the nameserver(s).

See "Network File System (NFS)" on page 247 for a description of name- and reverse name resolution with YP.

Domain Nameserver

We have already installed all hosts so they can communicate based on the host addresses given in the /etc/hosts files on each host. Now let's show what you need to change in the configuration files so all hosts can use the host ***m135*** as a domain nameserver.

A host in a network using TCP/IP nameservers can be:

- A ***primary*** nameserver whereto all name and reverse name resolution requests are first sent from hosts in the local domain.
- A ***secondary*** nameserver. Requests that can not be resolved by the primary nameserver are transferred to the secondary nameserver(s) in turn until it is resolved or until no more secondary nameservers are found.
- A nameserver ***client*** that uses the nameserver functions of the domain nameserver(s).
- A ***cache*** nameserver, either:
  - A primary or secondary nameserver that builds an internal cache of resolved hostnames and addresses as they get resolved. Entries in the cache are kept a specified time and then removed. If a resolution request can be resolved from the cache, it is, thereby reducing network traffic.
  - A nameserver client that provides nameserver service only for itself. Again, name- and reverse name resolution requests will be resolved from the cache, if possible. If not possible, the active nameserver for the domain is asked to resolve it and the result is stored in the local cache, available for future resolution requests.

We shall now explain how you must change the configuration files on all hosts in the network to use a domain nameserver. We will give examples for the following:

[25] Don't confuse "reverse name resolution" with ***address resolution***, which is the process of resolving an Internet address into whatever is required by the link level layer to actually access the remote host. This is taken care of by the ***Address Resolution Protocol*** (ARP).

- ***m135*** will be defined as a primary domain nameserver for the local domain (itsc.austin.ibm.com), pointing to a nameserver for the remote domain (austin.ibm.com) on the remote Token-Ring network.
- ***ps2nc*** will be defined as a domain nameserver client without a local cache.
- ***chris*** will be defined as a domain nameserver client ***with*** a local cache.

The remaining hosts can be configured using the same steps as described for these three.

Customizing the Domain Nameserver

The following steps summarize the procedure to customize. Details about the /etc/named.* files are given immediately after.

1. Edit /etc/hosts on ***m135*** to define all hosts with full domain name. Actually, you don't need entries in /etc/hosts for other than your local adapters on the domain nameserver, but the directory /usr/lpp/tcpip/samples provides shell scripts to create the nameserver configuration files from it, so it may be a good idea. Figure 78 shows how the file looks on our host ***m135***:

```
# Internet Address Hostname # Comments

127.0.0.1       localhost      # loopback (lo0) name/address

#        The local Token-Ring network
9.3.1.7         m135.itsc.austin.ibm.com m135 #m135 on local Token-Ring
9.3.1.4         chris.itsc.austin.ibm.com chris
9.3.1.6         gatex.itsc.austin.ibm.com gatex
9.3.1.12        ps2nc.itsc.austin.ibm.com ps2nc
9.3.1.15        dosnc.itsc.austin.ibm.com dosnc
9.3.1.100       vm9370.itsc.austin.ibm.com vm9370

#        The local Ethernet network
192.48.11.7     m135.itsc.austin.ibm.com m135 #m135 on local Ethernet
192.48.11.3     sys3.itsc.austin.ibm.com sys3
192.48.11.12    ps2nceth.itsc.austin.ibm.com ps2nceth # ps2nc on Ethernet
192.48.11.100   aix370.itsc.austin.ibm.com aix370

#        The X.25 network
192.48.12.1     sys1.itsc.austin.ibm.com sys1

#        The remote Token-Ring network
129.35.22.32    m135.itsc.austin.ibm.com m135 #m135 on remote Token-Ring
```

Figure 78. The /etc/hosts File on the Domain Server (m135)

   We have added an entry for the host ***sys1***. We shall later explain how to customize this host for connection via X.25.

   Notice how each host is defined with its full domain name first, followed by the plain hostname. If we use this file to create the Yellow Pages maps, entries will be added so that reverse name resolution requests will be resolved to both names by YP, and so that name resolution requests can be resolved from either the full domain name or the short name.

2. Edit the file /etc/rc.tcpip and make the following changes:

a. Uncomment the line that starts the *named* daemon. This daemon will process name- and reverse name resolution requests from ***m135*** itself as well as from other hosts that have ***m135*** as domain nameserver.

b. Change the *hostname* entry to include the domain name as well as the short name:

```
/bin/hostname m135.itsc.austin.ibm.com m135 > /dev/console
```

3. Copy /usr/lpp/tcpip/samples/named.boot to /etc/named.boot and edit it to fit your requirements.
4. Run the awk script /usr/lpp/tcpip/samples/hosts.awk to create the file /etc/named.data. Then edit the file to match your requirements.
5. Run the awk script /usr/lpp/tcpip/samples/addrs.awk to create the file /etc/named.rev. Then edit the file to match your requirements.
6. Create the file /etc/named.ca and edit it to match your configuration.
7. Create a file named /etc/resolv.conf as a zero length file, typing:

   ```
   touch /etc/resolv.conf
   ```

   The presence of this file tells the *named* daemon that this host is a domain nameserver. If the file is missing or is not zero bytes long, the *named* daemon will not start.

/etc/named.boot

In our case, we wanted to add a cache for resolved requests for addresses and names on the remote Token-Ring network, so we changed the file /etc/named.boot to what you see in Figure 79. The purpose of the file is to define the local domain name and to point to other configuration files.

```
; type  domain   source file or host
;
domain         itsc.austin.ibm.com
cache          austin.ibm.com                 /etc/named.ca
primary        itsc.austin.ibm.com            /etc/named.data
primary        in-addr.arpa                   /etc/named.rev
```

Figure 79. Domain Nameserver /etc/named.boot File

For details about the format of the file, consult the TCP/IP publication for the system you use.

/etc/named.ca

Figure 80 on page 203 shows our /etc/named.ca file that describes what resolved information should be stored in the cache.

```
;domain                     ttl  class  type   data
austin.ibm.com                1  IN     NS     bcroom.austin.ibm.com.
;host                       ttl  class  type   data
bcroom.austin.ibm.com.  3600000  IN     A      129.35.17.2
```

Figure 80. Domain Nameserver /etc/named.ca File

The file contains two comment lines (first and third line) and two active lines. You'll see that we want the cache to store information about the domain `austin.ibm.com`, which is the domain used on the remote Token-Ring network.

/etc/named.data

This file is used for ***name resolution***, that is, to translate host- or domain names into an Internet addresses. We have edited the file for readability and to add pointers to the nameserver on the remote Token-Ring network. The resulting file is shown in Figure 81 on page 204.

```
; /etc/named.data
@  IN  SOA  m135.itsc.austin.ibm.com.   chris.chris.itsc.austin.ibm.com. (
                     89.109   ; serial
                     3600     ; Refresh
                     300      ;Retry
                     360000   ; Expire
                     86400  ) ; Minimum
             IN NS m135
;
;       Names defined for the local host
;
m135         9999999 IN A  9.3.1.7      ; m135 on the local Token-Ring
             9999999 IN A  192.48.11.7  ; m135 on the local Ethernet
             9999999 IN A  129.35.22.32 ; m135 on the remote Token-Ring
timeserver   9999999 IN CNAME m135
printserver  9999999 IN CNAME m135
;
;       Ethernet addresses
;
sys3         9999999 IN A  192.48.11.3
ps2nceth     9999999 IN A  192.48.11.12 ; Same as ps2nc, but on Ethernet
aix370       9999999 IN A  192.48.11.100
;
;       Token-Ring addresses
;
chris        9999999 IN A  9.3.1.4
gatex        9999999 IN A  9.3.1.6
ps2nc        9999999 IN A  9.3.1.12
dosnc        9999999 IN A  9.3.1.15     ; X for DOS and AIX Access
vm9370       9999999 IN A  9.3.1.100    ; 9370 VM (not AIX/370)
;
;       X.25 addresses
;
sys1         9999999 IN A  192.48.12.1  ; x25 address
;
;       Remote name servers
;
austin.ibm.com.         9999999  IN     NS    bcroom.austin.ibm.com.
bcroom.austin.ibm.com.  9999999  IN     A     129.35.17.2
```

Figure 81. The /etc/named.data File on the Domain Server (m135)

The first entry in this file is `Start of Authority` (SOA) that spans several lines and defines parameters for the domain nameserver ***m135.itsc.austin.ibm.com***.

The entries saying `CNAME` define aliases for a host. In our case, aliases are defined for the `timeserver` and `printserver` names, both on host ***m135***. If these servers were placed on other hosts than the domain nameserver, the `CNAME` entries should point to that (those) other hosts, of course.

The last two lines point to a nameserver on the remote Token-Ring. This nameserver allows us to resolve hostnames on the remote network into addresses when the full domain name of hosts are given. An example:

```
host jensen.austin.ibm.com
```

/etc/named.rev

The /etc/named.rev file is used for ***reverse name resolution*** of Internet addresses into domain names. Again, we have edited the file to make it easier to read and to add a pointer to a nameserver for the remote Token-Ring network. Our file is shown in Figure 82.

```
; /etc/named.rev
;
; Define root domain
@  IN  SOA  m135.itsc.austin.ibm.com. chris.chris.itsc.austin.ibm.com. (
                    89.109   ; serial
                    3600     ; Refresh
                    300      ;Retry
                    360000   ; Expire
                    86400  ) ; Minimum
             IN NS m135
;
;       This local machine
;
7.1.3.9                 IN     PTR   m135.itsc.austin.ibm.com.
7.11.48.192             IN     PTR   m135.itsc.austin.ibm.com.
32.22.35.129            IN     PTR   m135.itsc.austin.ibm.com.
;
;       Token-Ring addresses
;
4.1.3.9                 IN     PTR   chris.itsc.austin.ibm.com.
6.1.3.9                 IN     PTR   gatex.itsc.austin.ibm.com.
12.1.3.9                IN     PTR   ps2nc.itsc.austin.ibm.com.
15.1.3.9                IN     PTR   dosnc.itsc.austin.ibm.com.
100.1.3.9               IN     PTR   vm9370.itsc.austin.ibm.com.
;
;       Ethernet addresses
;
3.11.48.192             IN     PTR   sys3.itsc.austin.ibm.com.
12.11.48.192            IN     PTR   ps2nceth.itsc.austin.ibm.com.
100.11.48.192           IN     PTR   aix370.itsc.austin.ibm.com.
;
;       X.25 addresses
1.12.48.192             IN     PTR   sys1.itsc.austin.ibm.com.
;
;       Nameservers
;
35.129.in-addr.arpa     IN     NS    bcroom.austin.ibm.com.
```

Figure 82. The /etc/named.rev File on the Domain Server (m135)

Entries in the file specify the Internet address in reverse order and point to the full domain name of the corresponding host. It is ***very important*** that all domain names specified ***are terminated by a period*** because the reverse name resolution process will add the local domain name to any name that is not terminated with a period. For example, if the line defining our host ***chris*** was not terminated by a period, resolution of the Internet address ***9.3.1.4*** would generate the answer:

```
chris.itsc.austin.com.ibm.itsc.austin.com.ibm
```

which is certainly not what we'd like.

The last line points to a nameserver on the remote Token-Ring. This nameserver allows us to resolve Internet address of the address range on the remote Token-Ring into fully qualified hostnames. For example:

```
host 129.35.18.8
```

Customizing a Cache-less Nameserver Client

A cache-less domain nameserver client does ***not*** run the *named* daemon. Rather, it defines to its local name- and reverse name resolution system calls that a domain nameserver is available and should be used in place of the information in the local `/etc/hosts` file. The presence of the file `/etc/resolv.conf` with a non-zero length is what does this.

To illustrate how to customize a cache-less nameserver client, we will use our host ***ps2nc***. To customize ***ps2nc***, we did the following:

1. Edited the file `/etc/hosts` so all unnecessary lines were removed. We also specified the full domain name for the local interfaces, but this is not really necessary.
2. Changed the *hostname* entry in `/etc/rc.tcpip` to include the domain name as well as the short name.
3. Created the file `/etc/resolv.conf`.

The resulting `/etc/hosts` file is shown in Figure 83.

```
# Internet Address Hostname # Comments

127.0.0.1       localhost      # loopback
9.3.1.12        ps2nc.itsc.austin.ibm.com ps2nc
192.48.11.12    ps2nceth.itsc.austin.ibm.com ps2nceth
9.3.1.7         printserver
192.48.11.7     timeserver
```

Figure 83. The /etc/hosts File on Cache-less Client ps2nc

The file `/etc/resolv.conf` for ***ps2nc*** is shown in Figure 84. It defines the domain name that our host belongs to and gives the Internet address of the domain nameserver.

```
domain          itsc.austin.ibm.com
nameserver      9.3.1.7
```

Figure 84. The /etc/resolv.conf File on a Cache-less Client

Customizing a Nameserver Client with Local Cache

A domain nameserver client with a local cache must run the *named* daemon. This requires the presence of at least one /etc/named.* file and that the file /etc/resolv.conf exists and is of zero length. To customize host ***chris*** as a domain nameserver client with local cache, you must do the following:

1. Edit /etc/hosts on ***chris*** to remove lines that are not required.
2. Edit the file /etc/rc.tcpip and make the following changes:
   a. Uncomment the line that starts the *named* daemon which will process name- and reverse name resolution requests from the local cache when possible and otherwise route requests to the domain nameserver.[26]
   b. Change the *hostname* entry to include the domain name as well as the short name:

      ```
      /bin/hostname chris.itsc.austin.ibm.com chris > /dev/console
      ```
3. Copy /usr/lpp/tcpip/samples/named.boot to /etc/named.boot and edit it to fit your requirements.
4. Create the file /etc/named.ca and edit it.
5. Create the file /etc/named.locdata and edit it.
6. Create the file /etc/named.local and edit it.
7. Create a file named /etc/resolv.conf as a zero length file, typing:

   ```
   touch /etc/resolv.conf
   ```

/etc/named.boot

Figure 85 shows the /etc/named.boot file on host ***chris***. It defines the domain name and points to three other configuration files.

```
; /etc/named.boot
;
domain          itsc.austin.ibm.com
cache           .                         /etc/named.ca
primary         itsc.austin.ibm.com       /etc/named.locdata
primary         0.0.127.in-addr.arpa      /etc/named.local
```

Figure 85. The /etc/named.boot File on Client with Local Cache

/etc/named.ca

This file defines the primary nameserver for the domain as shown in Figure 86 on page 208. Since ***chris*** is on the local Token-Ring network, we specify the Internet address of host ***m135*** as defined for Token-Ring. The file is used for name resolution when this can not be done from the cache. As names are resolved, the result is placed in the local cache.

[26] On AIX PS/2 there are no uncommented *named* lines, so just add: /etc/named & to the end of the file /etc/rc.tcpip.

```
; /etc/named.ca
.                              1   IN   NS   m135.itsc.austin.ibm.com
m135.itsc.austin.ibm.com   3600000 IN   A    9.3.1.7
```

Figure 86. The /etc/named.ca File on Client with Local Cache

/etc/named.locdata

The /etc/named.locdata file is used for ***name*** resolution when not possible from the local cache.[27] The file corresponds to the file /etc/named.data on the domain nameserver. As names are resolved, the result is placed in the local cache. Our file is shown in Figure 87.

```
; /etc/named.locdata
.                9999999  IN   NS   m135
```

Figure 87. The /etc/named.locdata File on Client with Local Cache

/etc/named.local

The /etc/named.local file is used for ***reverse name*** resolution when not possible from the local cache. The file corresponds to the file /etc/named.rev on the domain nameserver. As addresses are resolved, the result is placed in the local cache. Our file is shown in Figure 88.

```
; /etc/named.local
                                 IN   NS   m135.itsc.austin.ibm.com.
                             1   IN   PTR  localhost.
```

Figure 88. The /etc/named.local File on Client with Local Cache

named Data Base

The *named* data base is created from the /etc/named.* files when the daemon is started. If you need to recreate the data base after changing the configuration files, you can send the signal **SIGHUP** to the daemon. You do so by typing:

```
kill -1 process-id
```

where process-id is the process ID of the *named* daemon as displayed with the *ps* command.

To obtain a legible version of the data base in the file /usr/tmp/named_dump.db, send the signal **SIGINT** to the daemon by typing:

```
kill -2 process-id
```

[27] It is not clear to us why this file is required. However, without it, name resolution is not possible.

Using X.25 for TCP/IP

Since we've already inserted references to the host ***sys1*** as connected to X.25, now let's see how we install and customize the hardware and software on that host.

Customizing Host sys1 for X.25

To customize the host ***sys1*** that is to have only one communications adapter, the ***IBM PC X.25 Communications Adapter***, we must go through the steps used for host ***chris*** and a few more. Because we have a domain nameserver active, we customize ***sys1*** as a cache-less client right away:

1. Acquire an X.25 connection. The update level we were using for our tests required that the X.25 connections supported ***negotiable packet size***, and we had to ask for a change of the X.25 engine configuration. Negotiable packet size is used in some other countries (like the UK) but not available in others (like France). See below under the discussion of ***tranfile***.
2. Shutdown the host and install the IBM PC X.25 Communications Adapter in any slot.
3. Reboot the host and use *installp* to install the IBM 6150 X.25 Communications Support. Connect the adapter to the X.25 modem or X.25 engine port, whichever applies for your installation.
4. Use *updatep* and apply any updates that may apply to your system. For AIX/RT 2.2.1, and at the time of writing, the following updates to TCP/IP and IBM 6150 X.25 Communications Support are required: `IX02690`, `IX02691`, `IX02934`, `IX02966`, `IX02990`, `IX03024`, `IX03102`, `IX3103` and `IX03661`.
5. Run *devices* and add the X.25 adapter as `x25w0`.
6. Run `devices` and add pseudo terminals as for the host ***chris***.
7. Edit the `/etc/hosts` file and insert the appropriate lines. Our file looks like this:

```
# Internet Address Hostname # Comments

127.0.0.1       localhost      # loopback (lo0) name/address
192.48.12.1     sys1.itsc.austin.ibm.com
```

Figure 89. The /etc/hosts File on Host sys1

   Note that we use a Class-C address different from the one we use for the Ethernet.

8. Edit the `/etc/master` file as you did for the host ***chris***.
9. Edit the `/etc/rc.tcpip` file and make the following changes:
   a. Insert the node name in the *hostname* command:

   ```
   /bin/hostname sys1.itsc.austin.ibm.com sys1 > /dev/console
   ```

   b. Because X.25 does not support broadcasts, we can not use dynamic routing, so any route from *sys1* must be defined as static routes. In our configuration, the only way we can reach the rest of the hosts is through the host ***gatex*** that has a Baseband Adapter and an IBM PC X.25 Communications Adapter. Therefore, we must insert a *route* command for static routing through ***gatex***. This is done with:

```
route add net 0 192.48.12.6
```

If we were using the EGP protocol, we could actually use *gated* to provide dynamic routing, since EGP does not use broadcasts. This would be relevant if we were customizing a ***root server*** for the Arpanet.

10. Edit the /etc/rc file and check that lines to run the /etc/rc.include shell script are uncommented.

11. Edit the /etc/rc.include file and uncomment the lines to run the /etc/rc.tcpip shell script when the system starts.

12. Edit the /etc/net file and insert a stanza for X.25 as follows:

    ```
    x25w0:
             netaddr = 192.48.12.1
             inetlen = 576
             tranfile = /usr/lpp/x25w/tranfile
    ```

    Use the defaults for **inetlen** if they differ from the value shown above. The ***tranfile*** entry is required for access to the ***Public Data Network*** (PDN). If this entry is omitted or doesn't specify a file name, the X.25 interface is configured for the ***Defense Data Network*** (DDN).

13. Create the file /usr/lpp/x25w/tranfile and edit it so it defines the relation between the remote Internet addresss and the NUA of the remote hosts. All remote hosts you want to talk to must be listed, but it doesn't hurt to specify the local host as well:

```
#Internet addr  NUA              Packet size    Negotiable

192.48.12.1     3106001984          512         yes
192.48.12.6     3106008301          512         yes
```

Figure 90. Translation Between NUA and Internet Address

This file provides the translation between Internet address and the ***Network User Address*** (NUA) on an X.25 ***Public Data Network*** (PDN). The NUA in the file must match the NUA assigned to you by your local supplier of X.25 services, of course.

At the time this publication is available, an update (IX3361) should be available. The format shown in Figure 90 assumes that this update is applied. The update allows you to specify the maximum X.25 packet size to be used by TCP/IP in case you use negotiable packet size or the fixed packet size if not. It also gives you the choice between using negotiable packet size or not.

If fixed packet size is selected and a packet of different size is received, the packet is rejected. You should normally specify the same packet size for all NUAs, but there may be cases where you'd want to use different sizes. Packet size must be 128, 256, 512 or 1024.

14. Create the file /etc/resolv.conf for ***sys1*** as shown in Figure 91 on page 211 to be able to use ***m135*** as a domain nameserver.

```
domain          itsc.austin.ibm.com
nameserver      192.48.11.7
```

Figure 91. The /etc/resolv.conf File on a Cache-less Client

15. Rebuild the kernel and reboot the system.
16. When ***sys1*** comes back up, TCP/IP is initialized and ready to use.

Before you can use TCP/IP to access the other hosts, you must repeat the installation and customizing procedure for the host that'll be the gateway between the X.25 network and the local networks. In our case, this host is ***gatex***.

Configuring a Gateway

A TCP/IP ***gateway*** connects two physical networks so TCP/IP requests can be sent to and received across network boundaries. Our small configuration operates with several, separate networks:

```
                 9.x.x.x          The local Token-Ring network
                 129.35.x.x       The remote Token-Ring network
                 192.48.11.x      The local Ethernet network
                 192.48.12.x      The X.25 network
```

Figure 92. Networks in Sample Configuration

Static Routing

For all hosts to be able to communicate with any other host, gateways are provided. Our configuration has the following gateways:

```
                 Hostname         ------connects networks------

                 m135             9.x.x.x      and  129.35.x.x
                 m135             9.x.x.x      and  148.48.11.x
                 m135             148.48.11.x  and  129.35.x.x
                 gatex            148.48.11.x  and  148.48.12.x
```

Figure 93. Gateways in Sample Configuration

and so far all hosts have been defined so they use ***static routing*** through the use of the *route* command.

When you want to access the host ***sys1*** from one of the other, local machines your default route as defined thus far does not help us. All hosts on the local LANs point to the host ***m135*** as the default gateway. For requests to be routed further on from ***m135***, we must add yet another routing entry on ***m135***:

```
route add host 192.48.12.1 gatex.itsc.austin.ibm.com
```

This tells our primary gateway to route all requests for the ***host*** on the X.25 connection through the local host ***gatex***. If more hosts than ***sys1*** were connected only to the X25 network, and all were using the same Class-C network address, we could have said:

```
route add net 192.48.12.0 gatex.itsc.austin.ibm.com
```

to route all requests for the ***net*** that can contain the Class-C addresses **192.48.12.1** through **192.48.12.254**.

Now, only two hosts know how to reach ***sys1***: ***m135*** and ***gatex***. The first knows because of the *route* command; the latter knows because it itself has an adapter associated with the same Class-C address as ***sys1*** uses.

From any other host, requests directed at ***sys1*** will go through ***m135***. For hosts on the Ethernet network, this is the only way ***sys1*** can be reached. Hosts on the Token-Ring network have direct access through ***gatex***, though, so we are wasting valuable LAN bandwidth and gateway machine cycles. To go directly from the hosts on Token-Ring to the X.25 gateway host, we can add:

```
route add net 192.48.12.0 gatex.itsc.austin.ibm.com
```

to the /etc/rc.tcpip file of each host on the local Token-Ring.

The route Command

The *route* command differs slightly between the three AIX implementations. This will not be the case for future versions of AIX. The command as currently available on AIX/RT will be ported to all AIX platforms. This section describes some of the things you should know about the *route* command on AIX/RT. For further details, or when using the command on AIX PS/2 or AIX/370, please consult the TCP/IP publication for the system you use.

As we have already seen, the *route* command can be used to ***add*** new static routes to a host. Similarly, you can use the command to ***delete*** entries from the static routing table on a host. The format of the command to add or delete static routes is:

```
route add [net|host] destination gateway metric
```

If neither net nor host is given, host is assumed. When net is specified, destination should be given as an Internet network address with all four elements specified, but with zeroes in as many elements as dictated by the address class. Since we use all three Internet address classes, let's see how we would specify the destination:

```
For network address range          Destination network address

9.1.1.1      -  9.254.254.254              9.0.0.0
129.35.1.1   -  129.35.254.254             129.35.0.0
192.48.11.1  -  192.48.11.254              192.48.11.0
```

Figure 94. Network Address Specification with route Command

Since the *route* command only accepts network addresses at the Internet address level, it's not possible to specify static routing to subnets of a network.

When you add a route to a ***host***, you can specify the hostname or the Internet address of the destination host, but if you use hostname, that hostname must be known to the local host. If you specify Internet address, all elements of the address must be given, of course, since we are then specifying the route to one particular host rather than to a network.

The `gateway` specification can be given as the hostname (if known to the local host) or the full Internet address of the gateway host.

The *route* command takes flags, the most important of which is the command line option *-f* which will ***flush*** the routing tables on the local host. After you used this option, all static routes are deleted.

The netstat Command

You can use the *netstat* command to obtain information about various network-related tables on your local host. One important command line option is the flag *-r* which will display the routing tables of the host, whether static or dynamic. Figure 95 shows an example of the output of `netstat -r` on host ***m135***.

```
Destination     Gateway              Flags  Refcnt Use     Interface

localhost       localhost            UH     2      804     lo0
default         bcroom.austin.ibm.co UG     0      3901    token1
9.3.1           m135                 U      12     55425   token0
192.100.2       bingo.austin.inm.com UG     0      0       token1
192.48.11       m135                 U      6      17061   net0
192.48.12       gatex                UG     0      0       token0
aus.ibm.com     m135                 U      2      32663   token1
129.33          ausgat1.austin.ibm.c UGD    0      0       token1
129.34          ausgat1.austin.ibm.c UGD    0      63      token1
```

Figure 95. Displaying Routing Tables with the netstat Command

The meaning of the `FLAGS` column is:

U Indicates that the route is available ***(up)***.

H Present if route is to a single host rather than to a network.

G Means that datagrams for ***destination*** will be routed through the ***gateway*** specified. If this flag is not present, the local host is directly connected to the ***destination*** network.

D Tells you that this route was added ***dynamically***, that is, based on information received from some other gateway.

The *netstat -r* command is our first choice when we want to check why it's not possible to reach a host that we have been able to get to in the past. If the routing tables show the correct route, there's a huge probability of a network failure or that the remote host is failing or not ready to talk.

Another valuable option of the *netstat* command is the ***-v*** flag that'll display the VRM statistics for the device drivers used by TCP/IP. The statistics will tell you about network problems like short packets and CRC errors. It will also tell you if you are actually using the adapters since the counts will increase when you do.

Dynamic Routing

Just as it's manageable not to use domain nameservers in small networks, it's acceptable to use only static routing when you have a small number of networks to interconnect and when those networks a fairly stable. As the number of hosts and networks increases, or when the configuration is volatile, keeping static routes to networks other than your local network can become a burden. You may then want to introduce dynamic routing.

However, before doing so, make sure you understand the impact on network traffic. On the remote Token-Ring network in our configuration, for example, there are several thousand hosts. If every such host was to use dynamic routing, the load on the Token-Ring network could be increased to an unacceptable level. Why is this?

It's because dynamic routing is based upon daemons that exchange information about network routes periodically by sending broadcasts to each other. Depending on the options you use when you start the daemons, your system can act as a ***passive*** or an ***active*** gateway. An active gateway will supply routing information about it's local network by broadcasting that information across connections to other networks. A passive gateway will accept information from remote, active gateways but will not attempt to keep remote gateways updated.

AIX gives you a choice of gateway daemons. The recommended daemon is ***gated*** which supports more gateway protocols than the alternative ***routed*** daemon. If you need ***EGP (Exterior Gateway Protocol)***, which is required for interconnection with ***ARPANET*** and ***Milnet***, or the ***HELLO*** protocol you must use the `gated` daemon.

routed

Configuring `routed` for a small network as ours is simple. You need only edit two files:

```
/etc/networks
/etc/gateways
```

`/etc/networks` contains the network name database. When defining a network without connections to any of the "official" networks, you do not need this file. If you do connect to other networks, consult the TCP/IP publication for the AIX system you use.

You must have the file `/etc/gateways` if you want to use the *routed* daemon. In our network, for example, we could tell gateways on the remote Token-Ring network that to reach our X.25 host, they should go through the local gateway ***m135***.

If we do not provide this information to hosts on remote LANs, such hosts wouldn't know how to get to our ***sys1***. Local hosts know, because they have a default static route to the local gateway, which in turn has a static route to ***sys1***. Hosts on remote nets, though, will hopefully not have a default route pointing to ***m135***, so they don't know. To tell them, we must run ***m135*** as an active gateway, which means that we must start the daemon with the **-s** flag. The file /etc/gateways would be as shown in Figure 96 on page 215.

```
# <destination>  <name1>   gateway  <name2>  metric  <value>  <type>
net          192.48.12.0 gateway  9.3.1.6 metric    1      passive
```

Figure 96. The /etc/gateways File

Our Class-A address is officially assigned to IBM. So is the Class-B address we have been using. The Class-C addresses we use, however, are arbitrarily picked. We did so because we intended to keep those addresses hidden to the world. What happens when we start *routed* daemon with the above /etc/gateways file is that we tell all remote gateways how to get to our X.25 network.

Now assume that somebody else has defined a small network using the same Class-C addresses as we use. Remote gateways will be rather confused when they learn about two different routes to the same network. We may receive datagrams destined to the other network and vice versa. To overcome this problem and still allow hosts on the remote Token-Ring network to get to our X.25 network, we can do one of two things:

1. Define our host as a ***passive*** gateway.

   All that takes is to start the *routed* daemon with the ***-q*** (quiet) option. If a host on the remote Token-Ring sends a request for one of our Class-C networks, and if the request is not resolved by the nameservers on that network, a gateway on the remote net may eventually ask all the gateways it knows if they have a route to **192.48.12.x**.

   Our host would respond and say it does know a route and can handle the request. So may other gateways. Depending on the accumulated metric (the number of gateways involved) to get to the network, the gateway on the remote net will route the datagram through ***m135*** or some other gateway with a smaller, accumulated metric.

2. Obtain an official Class-C address.

   As simple as it sounds, this may not be easy, but it is the only correct way, of course. Then, if somebody else is using the Class-C address we have assigned to us, we can file an official complaint and force the other party to give up the use of our Class-C address.

   We would, of course, proud of our new Class-C address, want to tell everybody about it, so in this situation we'd like to let our gateway daemon run as an active gateway.

gated

The *gated* daemon can handle the RIP, the EGP, and the HELLO protocols at the same time. If you use the ***EGP*** protocol (which you probably don't since it's normally only used by root servers on Arpanet), you must:

- Obtain an ***autonomous system number*** for your gateway. Such numbers are assigned by SRI (see "Internet and Internet Routing" on page 166).
- You must know who your ***EGP neighbors*** are. Again, SRI must supply you with that information.

For installations in universities and commercial environments, the RIP protocol is normally used. This is also what we will be using to illustrate how you configure for *gated*.

The /etc/gated.conf file is the only configuration file used by *gated*. Figure 97 shows how we edited the file to define the route to our X.25 network.

```
EGP                     no
RIP                     yes     supplier
HELLO                   no
defaultegpmetric        1
#
#net    network-name   gateway  gw-name  metric  value  rip|hello|egp
net     192.48.12      gateway  9.3.1.6  metric    1    rip
#
#host   host-name      gateway  gw-name  metric  value  rip|hello|egp
#
#defaultgateway  gateway-name  rip|hello|egp  metric  value  passive|active
defaultgateway   129.35.17.2   rip            metric    1    active
```

Figure 97. The /etc/gated.conf File

We specified "***RIP yes supplier***" and "***EGP no***" to indicate our choice of gateway protocol. The keyword ***supplier*** tells *gated* that we want it to transmit routing information to remote gateways. If we didn't want this to happen, we'd have specified ***quiet***.

The option pointopoint combined with sourceripgateways (neither is shown in the figure) could have been used to restrict the remote gateways that *gated* should keep informed of routes.

We have two active lines in the routing section of the /etc/gated.conf file:

net
: Defines the route to our X.25 host. The same considerations as discussed for the *routed* daemon apply.

defaultgateway
: Tells our local gateway ***(m135)*** that if it cannot determine a route, it should use a gateway machine (with Internet address **129.35.17.2**) on the remote Token-Ring network as a gateway to everybody else.

Subnet Addressing

Our small configuration is not really that small. In fact, it is connected to the entire IBM TCP/IP network in the USA. We use two Internet network addresses officially assigned to IBM:

1. The Class-A address 9.x.x.x
2. The Class-C address 129.35.x.x

IBM has a large number of smaller networks all over the USA. All these networks must use the same two network addresses. How then, do we route between the remote networks at each site? The answer is ***subnet addressing***.

One of the advantages of using subnet addresses is that the rest of the world doesn't know or care. Every gateway outside IBM will know only two network addresses to get to the IBM network, namely the two shown above. Gateways and routers within the IBM network take care of the further routing.

Network Addressing

Before we dive into the mysteries of subnet addressing, let's see how routing is done from plain Internet addresses. The concept is simple.

Whenever a datagram must be sent to a given Internet address, the destination Internet address is compared to the Internet addresses of the local network interfaces of the source host. Depending on Internet address class, the destination host is assumed to be connected to a local network as follows:

Address class	On local network if
Class-A	First address element is equal
Class-B	First two address elements are equal
Class-C	First three address elements are equal

Actually, this is done by a logical "and" of a ***mask*** (derived from the the network class of each local interface) with the destination Internet address. If the result is equal to the network address (as opposed to the host address) of a local interface, the destination address is assumed to be on a network that can be reached directly via one of the local interfaces.

For example, on our host ***m135*** we have so far defined three local network addresses. When converted into binary representation they become:

```
    9.0.0.0  =  0000 1001 0000 0000 0000 0000 0000 0000
  129.35.0.0  =  1000 0001 0010 0011 0000 0000 0000 0000
 192.48.11.0  =  1100 0000 0011 0000 0001 0011 0000 0000
```

The corresponding ***masks*** are:

```
    9.0.0.0  =  1111 1111 0000 0000 0000 0000 0000 0000
  129.35.0.0  =  1111 1111 1111 1111 0000 0000 0000 0000
 192.48.11.0  =  1111 1111 1111 1111 1111 1111 0000 0000
```

As an example, let's see if we can reach the host address **129.35.18.8** through a local interface. We try first with the masks of the lower address class, then proceed with the next higher class if no match is found. The first step is:

```
   129.35.18.8  =  1000 0001 0010 0011 0001 0010 0000 1000
    "and" mask  =  1111 1111 1111 1111 1111 1111 0000 0000
        result  =  1000 0001 0010 0011 0001 0010 0000 0000
"xor" net addr  =  1100 0000 0011 0000 0001 0011 0000 0000
    difference  =  0100 0001 0001 0011 0001 0001 0000 0000
```

The difference is non-zero, which means that the interface does not have a direct connection to the destination host. Next, we repeat the process for the interface that has the Class-B address assigned:

```
   129.35.18.8  =  1000 0001 0010 0011 0001 0010 0000 1000
    "and" mask  =  1111 1111 1111 1111 0000 0000 0000 0000
        result  =  1000 0001 0010 0011 0000 0000 0000 0000
"xor" net addr  =  1000 0001 0010 0011 0000 0000 0000 0000
    difference  =  0000 0000 0000 0000 0000 0000 0000 0000
```

This time the difference is zero, so we can use the interface to reach the destination Internet address.

At this point, let's confess that the ***mask*** we were using really is a ***subnet mask***. What happens when you configure your network is that every network gets a default subnet mask assigned to it depending on the Internet address class. You can see this if you use the command *netconfig query* to display the status of the network interfaces as shown in Figure 98. For each interface a subnet mask is listed.

```
token0:
Internet Address: 9.3.1.7       inetlen: 1500   hardware type: IEEE 802.5
subnet mask: ff000000   remote inetlen: 1500   broadcast: local

net0:
Internet Address: 192.48.11.7   inetlen: 1500   hardware type: ethernet
subnet mask: ffffff00   remote inetlen: 1500

token1:
Internet Address: 129.35.22.32  inetlen: 1500   hardware type: IEEE 802.5
subnet mask: ffff0000   remote inetlen: 1500   broadcast: all rings
```

Figure 98. Listing Interface Status with netconfig Command. The output shown is from the server ***m135*** before subnet masks have been defined.

Subnet Addressing

When you understand the way network addressing is done, you start wondering why the mask necessarily has to be given in 8-bit chunks. The answer to your question is, of course, that it hasn't. The term we use to describe addressing where the mask is different from the default mask is ***subnet addressing***.

Assume we have a gateway machine on the remote Token-Ring network and that this gateway has an interface to the IBM site in Palo Alto. The Internet address range assigned to Palo Alto is **9.49.x.x** and the address range assigned to the Austin site is **9.3.x.x**. How will our gateway determine when to send datagrams across the link to Palo Alto?

The answer is that we must assign subnet masks to the interface on the gateway. However, at the Austin site we have several, smaller networks, all using the **9.3.x.x** address range. One such network is our local **9.3.1.x** net. Therefore, the gateway should not only filter requests based upon the first 16 bits of the address. The subnet mask must be defined as the default subnet mask of a Class-C network for everything to work correctly.

Now let's look at our local host ***chris***. Up to now, no subnet mask is defined, so the host will assume that a request destined for any **9.x.x.x** address can be reached via the local Token-Ring interface. If we say:

```
ping 9.49.13.5
```

and think we can get across the link to Palo Alto and contact a host there, we'll be disappointed. For this to work, we must supply a subnet mask for the host. We do so by modifying the stanza for the Token-Ring interface in the file /etc/net on the host. We change it as shown in Figure 99 on page 219.

```
        token0:
                netaddr = 9.3.1.4
                inetlen = 1500[22]
                r_inetlen = 1500
                subnetmask = 255.255.255.0
                localbroadcast = true
```

Figure 99. Subnet Mask for a Class A Network (AIX/RT)

You'll see that this corresponds to the default mask for a Class-C network. Some implementations of TCP/IP require the subnet mask to be given as hexadecimal values. The above mask would then be: FFFFFF00.

In AIX PS/2 and AIX/370 you'd normally specify a subnet mask by modifying the *ifconfig* command in the file /etc/rc.tcpip as shown in the example line for our host ***ps2nc*** in Figure 100.

```
        /etc/ifconfig tk0 $SITE subnetmask=255.255.255.0
```

Figure 100. Subnet Mask for a Class A Network (AIX PS/2)

If we now try to *ping* the host in Palo Alto again, we can do it. The route our request takes is:

1. It is determined that no local interface can reach the destination host, so the request is sent to ***m135*** because the default static route points to it.
2. Host ***m135*** has the same subnet mask for the **9.x.x.x** interface as host ***chris***, so it likewise finds that the destination host can not be reached via a local interface. It determines from the *gated* routing tables (see Figure 97 on page 216) that the default gateway is at Internet address **129.35.17.2** and forwards the request.
3. The host ***bcroom.austin.ibm.com*** at this Internet address also determines that it has no direct link to Palo Alto and forwards the request to a gateway that has.
4. The gateway with an interface to Palo Alto then sends the request across the remote link.
5. In Palo Alto, the gateway that receives the request may or may not have direct access to the destination host. If it has, the request is sent directly. If not, the process is repeated until the destination host is reached.

What we can learn from this is that for subnet masks to work as intended, ***all hosts*** in the network must use ***identical*** subnet masks for a given Internet network address. Hosts on other networks neither should nor can, since they do not have any interface that allows direct access to our network.

It is generally recommended that when assigning subnet masks for Class-A networks, you do so in 8-bit chunks. If possible, follow this recommendation. If you don't and want to connect to non-IBM hosts via TCP/IP, you may not be able to. Also, some application software may not work correctly.

Subnet Addressing for Class-B Networks

We have seen how to use subnet masks for a Class-A network. Now let us look at Class-B and Class-C networks. The Class-B address assigned to IBM is used at more than one site. For example, the network in Austin is assigned the address range **129.35.16.x** through **129.35.31.x**. Again, addressing is done through subnet addressing.

The subnet mask used is **255.255.240.0** which corresponds to X'FFFFF000' or the binary value:

```
1111 1111 1111 1111 1111 0000 0000 0000
```

This divides the Class-C address into 14 different subnets:

```
129.35.16.0,  129.35.32.0,  129.35.48.0, ...,  129.35.224.0
```

When we say 14 rather than 16 subnets, this is because no host should have an address that, when "anded" with the subnet mask would yield a zero result in any of the 8-bit chunks that are entirely or partly covered by the subnet mask. If we don't follow this rule, host addresses may be confused with broadcast addresses and we're in deep trouble.

Similarly, no host should be assigned an address that will yield an all-one result in any of the 8-bit chunks covered by the subnet mask when "anded" with the mask. For example, the address **129.35.244.5** when "anded" with the subnet mask, will give:

```
129.35.244.5  =  1000 0001 0010 0011 1111 0100 0000 1000
 subnet mask  =  1111 1111 1111 1111 1111 0000 0000 0000
      result  =  1000 0001 0010 0011 1111 0000 0000 0000
```

The section covered by the subnet mask in the third 8-bit group gives a result of all-ones so the address should not be used for any host.

Again, we have learned a lesson: Using subnet masks reduces the total number of Internet addresses we can use within a given Internet network address. Everything has its price.

On our host ***m135*** we have to define a subnet mask if we want to cooperate with the gateways on the remote Token-Ring network. So must every single host that has an interface that connects directly to the **129.35.x.x** net. We change the /etc/net file on ***m135*** so the stanza for the interface to the remote Token-Ring looks as shown in Figure 101.

```
token1:
        netaddr = 129.35.22.32
        inetlen = 1500[22]
        r_inetlen = 1500
        localbroadcast = false
        subnetmask = 255.255.240.0
```

Figure 101. Dividing a Class B Network into Subnets

Subnet Addressing for Class-C Networks

Even in a Class C network is it possible to use subnet masks. For example, a subnet mask of **255.255.255.240** will give you 14 smaller networks, namely:

```
192.48.11.16,  192.48.11.32,  192.48.11.48, ...,  192.48.11.224
```

It is recommended that subnet masks for Class-B and Class-C networks are assigned in 4-bit groups as we've done in the example above.

Subnet Addressing Applied

To illustrate the use of subnet addressing in a Class-B network, assume we have a gateway with interfaces as shown in Figure 102.

```
token0:
       netaddr = 129.35.17.32
       inetlen = 1500
       r_inetlen = 1500
       localbroadcast = false
       subnetmask = 255.255.240.0

token1:
       netaddr = 129.35.34.32
       inetlen = 1500
       r_inetlen = 1500
       localbroadcast = false
       subnetmask = 255.255.240.0
```

Figure 102. Defining a 20-bit Subnet Mask in /etc/net

This gateway is connected to two different subnets, namely 129.35.16.0 and 129.35.32.0. Also assume that a host with Internet address 129.35.17.4 on subnet 129.35.16.0 wants to send a packet to two hosts with Internet addresses 9.3.1.7 and 129.35.33.14, respectively. How does the requesting host and the gateway use the subnet mask?

First consider the request for address **9.3.1.7**. On the requesting host (with only one interface) this address will be logically "anded" with the subnet mask and then compared to the subnet address of the host:

```
    9.3.1.7  =  0000 1001 0000 0011 0000 0001 0000 0111
subnet mask  =  1111 1111 1111 1111 1111 0000 0000 0000
     result  =  0000 1001 0000 0011 0000 0000 0000 0000
```

This yields the subnet address **9.3.0.0**. The subnet address of our interface is **129.35.16.0**. Clearly, 9.3.0.0 is not identical to 129.35.16.0 so the destination host can't be reached via the interface. Therefore, our host must route the packet to a gateway that knows how to reach the destination subnet or net.

The only gateway the requesting host has access to is the host who's /etc/net file is shown in Figure 102. The request is sent to that gateway for further routing.

On the gateway the process is repeated for each interface. Since both have the same subnet mask, the result is again a destination subnet address of **9.3.0.0**. And, again, no local interface has a direct connection to such a subnet or net, so the request must be forwarded to the next gateway.

Now consider the request for address **129.35.33.14**. On the requesting host (with only one interface) this address will be logically "anded" with the subnet mask and then compared to the subnet address of the host:

```
129.35.33.14  =  1000 0001 0010 0011 0010 0001 0000 1110
 subnet mask  =  1111 1111 1111 1111 1111 0000 0000 0000
      result  =  1000 0001 0010 0011 0010 0000 0000 0000
```

This yields the subnet address **129.35.32.0**. The subnet address of our interface is **129.35.16.0** so we can not reach the destination host via the interface. The packet is thus routed to the gateway.

On the gateway the process is repeated for each interface. Since both have the same subnet mask, the result is again a destination subnet address of **129.35.32.0**. Our second interface has exactly this subnet address, so we can send the packet to the destination without going through any more gateways.

C2 Security with TCP/IP on AIX/RT

To convert your TCP/IP for operation in ***controlled access mode***, execute the ***/etc/securetcpip*** shell script by typing:

```
sh securetcpip
```

This is a one-way road to C2 Security, meaning that the only way to revert this is to reinstall TCP/IP. For further information about C2 Security with TCP/IP and controlled access mode, see the *Interface Program for use with TCP/IP* publication and the /etc/securetcpip file.

TCP/IP via Asynchronous Connection

Our configuration as shown in Figure 66 on page 184 includes an asynchronous connection between the hereto ignored host **sys2** and the Ethernet connected host ***sys3***. In our tests, the connection was established via a null-modem cable, but we shall now describe how to configure and use such a connection using ***Hayes 1200 Smartmodems***.

When TCP/IP is used via an asynchronous connection, it's said to be using a ***Serial Line Interface Protocol*** (SLIP). The connection is illustrated in Figure 103. We will assume that the host **sys2** will be the host to initiate the connection (the calling host).

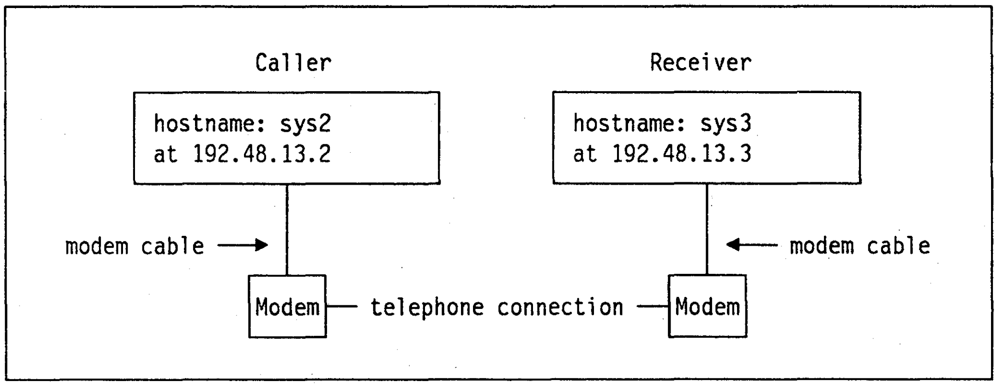


Figure 103. TCP/IP Asynchronous Connection

Customizing for SLIP

To customize the host ***sys2*** that is to be connected only through a serial interface, we must again go through the steps used for host ***chris*** and a few more. Because we have a domain nameserver active, we customize ***sys2*** as a cacheless client:

1. Install the modems and connect to the telephone lines. Then connect the modem to a serial port on the IBM RT. Use Asynchronous Terminal Emulation between the two hosts to ensure that the connection and modems are working properly.

2. Use *installp* to install TCP/IP.

3. Use *updatep* and apply any updates that may apply to your system.

4. Run *devices* and add a ***tty*** interface for the port you are using. Define the ports on the two hosts involved as shown in Figure 104.

```
Calling host (sys2)              Receiving host (sys3)

dvam      1                      dvam      1
bpc       8                      bpc       8
rts       1200                   rts       1200
sns       true                   sns       true
pt        none                   pt        none
ixp       true                   ixp       true
aa        false                  aa        true
ae        false                  ae        false
nosb      1                      nosb      1
om        full                   om        full
pro       dtr                    pro       dtr
```

Figure 104. Defining tty Ports for SLIP

5. Run `devices` and add pseudo terminals as for the host ***chris***.

6. Edit the `/etc/hosts` file and insert the appropriate lines. Our file looks like this:

```
# Internet Address Hostname # Comments

127.0.0.1       localhost       # loopback (lo0) name/address
192.48.13.2     sys2.itsc.austin.ibm.com
```

Figure 105. The /etc/hosts File on Host sys2

   Note that we use a Class-C address different from the one we use for the Ethernet and X.25 network.

7. Edit the `/etc/master` file as you did for the host ***chris***.

8. Edit the `/etc/rc.tcpip` file and make the following changes:

   a. Insert the node name in the *hostname* command:

```
/bin/hostname sys2.itsc.austin.ibm.com sys2 > /dev/console
```

b. Because SLIP does not support broadcasts, we can not use dynamic routing, so any routing from ***sys2*** must be defined as static routing. In our configuration the only way we can reach the rest of the hosts is through the host **sys3** that has a Baseband Adapter and a SLIP connection. Therefore, we must insert a *route* command for static routing through ***sys3***. This is done with:

```
route add net 0 192.48.13.3
```

c. On the host ***sys3*** we have an interface for a Baseband Adapter and would like that interface to be active even when the SLIP interface is not. We'd therefore want to run the script /etc/rc.tcpip when the host boots. The default file runs *netconfig* unconditionally. We'd like to change it so that only the non-SLIP interfaces are started automatically. To do this, change the script so that the *netconfig* command is executed as follows:

```
/etc/netconfig add -s net0
error=$?
if test $error -ne 0
then
   echo '/etc/rc.tcpip: /etc/netconfig can not add net0'
   exit 1
fi
```

This change will leave it to us to start the SLIP interface manually which is usually desirable, so that the boot process will not have to wait for a time-out because the interface will fail to initialize if both hosts are not ready.

9. Edit the /etc/rc file and check that lines to run the /etc/rc.include shell script are uncommented.

10. Edit the /etc/rc.include file and uncomment the lines to run the /etc/rc.tcpip shell script when the system starts if your host has more than one interface in use for TCP/IP. If not (as for **sys2**), do not uncomment the line that runs /etc/rc.tcpip.

11. Edit the /etc/net file and insert a stanza for the tty port as follows:

```
tty0:
        netaddr = 192.48.13.2
        inetlen = 1500
        r_inetlen = 1500
        dialstring = ATDT5555555O\r
        disconnect = false
        dstaddr = 192.48.13.3
```

inetlen and r_inetlen should be changed to 1500[22]. The entry shown is for the ***calling*** host. The dialstring is set to call the number 555-5555, which you would need to change to whichever telephone number you will be using. The code ATDT (modem commands) and \r (which causes the number to be dialed) are important.

The ***receiving host must not*** specify a dial string, so the stanza to be added on host **sys3** should be:

```
tty0:
        netaddr = 192.48.13.3
        inetlen = 1500
        r_inetlen = 1500
        disconnect = false
        dstaddr = 192.48.13.2
```

If the connection is a direct connection (without modems), the dial string should not be specified.

12. Create the file /etc/resolv.conf for **sys2** as shown in Figure 106 to be able to use ***m135*** as a domain nameserver.

```
domain          itsc.austin.ibm.com
nameserver      192.48.11.7
```

Figure 106. The /etc/resolv.conf File on a Cache-less Client

13. Rebuild the kernel and reboot the system.
14. When **sys2** comes back up, the SLIP connection will not be ready. We have to start it manually when we need it.

Before you can use TCP/IP to access the other hosts, you must repeat the installation and customizing procedure for the host that'll be the gateway between the SLIP network and the local networks. In our case, this host is ***sys3***.

The switch settings for the modem should be:

```
Caller (sys3)              Receiver (sys2)

Down: 2,4,6 and 8          Down:  2, 4 and 8
Up:   All others           Up:    All others
```

Figure 107. Hayes 1200 Smartmodem Switch Settings for SLIP

The meaning of these switch settings (which should help arrive at the proper settings for non-Hayes modems) are:

Switch 1	Up specifies that the computer supports RS-232C DTR lead (pin 20). Down specifies the computer does not support DTR lead.
Switch 2	Up specifies result codes are sent as English words. Down specifies that they are sent as single digits.
Switch 3	Up specifies that no result codes are sent. Down specifies that the codes are sent to the screen.
Switch 4	When up, Hayes 1200 echoes characters in the local command state. When down, the modem does not, unless half-duplex is selected and the modem is on-line.
Switch 5	Up selects auto answer of incoming calls on the first ring. Down indicates incoming calls will ***not*** be answered.

Switch 6 Up: When on-line, enables computer connected to the modem to determine if a carrier signal is from a distant modem by reading the status of the RS-232C Carrier Detect lead (pin 8). Down: Forces the computer to accept locally echoed characters and result codes from the modem. The RS-232C Carrier Detect lead (pin 8) will be logic ***true*** at all times even if there is no carrier signal.

Switch 7 Up for single line telephone installations connected to standard RJ11 jack. Down for multiline installations connected to an RJ12 or RJ13 jack.

Switch 8 Up disables modem command recognition. Down enables command recognition.

Using the SLIP Connection

At this point you are ready to establish the connection.

1. Make sure the ports on both RTs are disabled. If in doubt, disable the ports by typing:

   ```
   pdisable tty0
   ```

2. As already mentioned, we recommend starting and stopping the connection manually because the connection cannot occur if both systems are not up. If TCP/IP is not already started because we have initialized another interface, start and configure it by typing:

   ```
   sh /etc/rc.tcpip.
   ```

   If TCP/IP is already running, you can configure the SLIP interface by typing:

   ```
   netconfig add -s tty0
   ```

   Either way, ***you should hear the modem dial.***

Once TCP/IP is running, you can stop and start the SLIP connection by typing:

```
netconfig delete -s tty0  (to stop)
```

and

```
netconfig add -s tty0  (to start)
```

If you want to later use the modem on one of the machines for ATE or some other application (anyone but TCP/IP), you will need to stop the connection by issuing the commands on both machines.

Logging In to the Other Host

Now that the connection path is established, you can use the TCP/IP commands as for any other TCP/IP connection as long as you stay away from commands that use broadcasts.

You can use the commands from either host -- regardless of the fact that one host was the caller and the other was the responder. In fact, multiple "sessions" can be going on over the connection at the same time. For example, a user on host ***sys3*** can be logged into ***sys2***, and at the same time a user on host ***sys2*** can be logged into ***sys3***.

TCP/IP Hints

The following section is a loose collection of hints about the use of TCP/IP as a system administrator. It is not intended as an introduction to all the commands of TCP/IP. Only commands we have not already mentioned above are covered.

Starting and Stopping TCP/IP

On hosts with many daemons running it's a tedious job to stop and start TCP/IP interfaces. On AIX/RT, you can use the shell script ***/etc/tcp.clean*** to remove all daemons and disable all interfaces. The only thing this script doesn't do is to remove any routes from the routing tables. To stop TCP/IP and remove all static routes you'd say:

```
sh /etc/tcp.clean
route -f
```

Network File System and Yellow Pages depend on TCP/IP so if you are using those, you probably want to stop them before you stop TCP/IP. To do so, type:

```
sh /etc/nfs.clean
```

To restart TCP/IP, NFS and YP, say:

```
sh /etc/rc.tcpip
sh /etc/rc.nfs
```

Displaying and Changing Address Resolution Tables

The *arp* command has proved very useful for showing the address resolution tables as built in memory by the ***Address Resolution Protocol*** (ARP). To show the current tables type:

```
arp -a
```

The command also allows entries to be deleted from or added to the in-memory tables.

Verifying Name- and Reverse Name Resolution

To obtain the Internet address of a host, say **sys2** type:

```
host sys2
```

To obtain the hostname of a host, say:

```
host 192.48.11.2
```

The command is useful to check whether the active name- and reverse name resolution function knows a particular host. It is invaluable when checking that a domain or YP nameserver can resolve names and addresses as intended.

Error Logging Daemon

The *syslogd* daemon is normally uncommented in the file /etc/rc.tcpip and is thus started when TCP/IP is. It uses a configuration file: /etc/syslogd.conf to describe the action to take when log- and error messages are generated by TCP/IP. Actually, the daemon is used by other functions of AIX as well.

If you experience problems with TCP/IP, you may want to edit the /etc/syslogd.conf file. You'd want to increase the level of detail for log messages, and you'd want to send the output from *syslogd* to a file rather than discard discard the output or direct it to the console.

Refer to *AIX Operating System Technical Reference* for details about *syslogd* and its configuration file.

Remote Print Problems

We have already described how to define a print server and how to customize the print server clients. If you have not been able to get the print server to work, use the following check list:

1. Did you run the script `/usr/lpp/tcpip/samples/prtsvr.inst` on all the print server clients?
2. Did you specify the correct remote print queue when you issued the *print* command on the client hosts?
3. Is the *lpd* daemon running on the print server?
4. Is the printer defined to the print server as a locally attached printer?
5. Can the client find the print server? If you key

   ```
   host printserver
   ```

   on the client and this does not give you the Internet address of the print server, update your nameserver configuration files or `/etc/hosts` on the client with information about the printserver.
6. Can the print server resolve the Internet address of the client host to a hostname? If you type

   ```
   host x.x.x.x
   ```

   on the print server host, where "`x.x.x.x`" is the Internet address of the client, and the response does not yield the hostname of the client, update your nameserver configuration files or `/etc/hosts` on the print server to include information about the client host.
7. Does the hostname returned when you key

   ```
   host x.x.x.x
   ```

   on the print server match one of the hostnames in `/etc/hosts.lpd` or `/etc/hosts.equiv`? If not, insert the hostname of the client in one or both files.

TCP/IP for PS/2

A LAN with AIX systems may use TCP/IP to talk to a wide range of other systems. PC- or PS/2-based work stations, for example, will often be very attractive compared to ASCII terminals because they not only give all the functions of the terminals but also allow the vast number of PC-DOS applications to be used. In later chapters, we shall discuss the products X-Windows for DOS Users and AIX Access for DOS Users. This section gives you an overview of TCP/IP for PS/2.

The program package *IBM Transmission Control Protocol/Internet Protocol for the IBM Personal System/2*, abbreviated to ***TCP/IP for PS/2*** in this publication, operates under PC-DOS and provides mainly client TCP/IP functions. It can be used in conjunction with TCP/IP products that provide the corresponding server functions. TCP/IP for PS/2 can connect via Token-Ring, PC Network or Ethernet. Mainly client functions are implemented, but a few functions allow communication between two systems that both use TCP/IP for PS/2. Some of the implemented TCP/IP applications are:

CUSTOM	An easy to use full-screen configuration tool for TCP/IP for PS/2.
BIND	Maps keyboard and colors for *TELNET* 3270 emulation.
FTP/TFTP	File transfer programs.
LPR	Prints a file on a foreign host.
REXEC	Executes a command on a foreign host.
SETCLOCK	Sets the data and time from a timeserver.
TELNET	Remote login to a foreign host (IBM 3270 host or ASCII host).
ROUTER	Configures the TCP/IP for PS/2 machine as an Internet gateway.

Other applications are provided. Some of these will be covered in the following sections.

Hardware and Software Prerequisites

The TCP/IP for PS/2 product is designed to operate on any PC or PS/2 with a minimum of 256KB of memory. The following LAN adapters are supported:

- IBM Token-Ring Adapter II
- IBM Token-Ring Adapter/A
- IBM PC Network Adapter II
- IBM PC Network Adapter II/A
- IBM PC Network Baseband Adapter
- IBM PC Network Baseband Adapter/A
- Ungermann-Bass PC-NIC Adapter
- Ungermann-Bass NIC/ps2 Adapter
- 3Com Etherlink Adapter.

For use on an IBM Token-Ring or IBM PC Network, the *IBM LAN Support Program* is required.

Though some releases of the TCP/IP for PS/2 program can run on PC-DOS versions earlier than 3.30, it is recommended that PC-DOS 3.30 (or later versions) and *LAN Support Program*[28] is used rather than the older (*TOKREUI* and

[28] The *LAN Support Program* is not required in an Ethernet environment.

NETBEUI) drivers. On an IBM Personal System/2, you ***must*** use *PC-DOS 3.30* or later and *Lan Support Program*.

TCP/IP for PS/2 Installation

The TCP/IP for PS/2 package is installed from the distribution diskettes using the *INSTALL* command. This will give a list of choices on the screen:

```
* A - To copy distribution diskettes to administrator's system
* U - To copy distribution diskettes to user's system
* R - To register new user and copy user diskettes
* V - To verify registered user and reissue user diskettes
* G - To setup a gateway diskette (system administrator)
* ? - To get an explanation of this program
* Q - To quit the program
```

If you want to copy the distribution diskette to a hard disk, press `U` - don't press `ENTER` yet - then enter the drive and directory (for example C:\username) and press `ENTER`. This directory must exist before installation. To create the directory type: `mkdir c:\username`, before installing. Alternatively, the administrator can install the distribution diskettes onto a fixed disk and from this machine generate diskettes with the TCP/IP for PS/2 programs for several PCs or PS/2s. Separate customizing must be done on each machine using TCP/IP for PS/2.

TCP/IP for PS/2 Customizing

On a PC-DOS system, device drivers are installed through the configuration file ***CONFIG.SYS***. Additional initializing of the system is done from the start-up file ***AUTOEXEC.BAT***. TCP/IP for PS/2 supplies a device driver ***NETDEV.SYS*** that must be installed from `CONFIG.SYS` for TCP/IP for PS/2 to work and a resident program ***OPENMON.COM*** that allows the TCP/IP for PS/2 programs to coexist with other LAN Programs such as the *IBM LAN Program*.

The installation sequence of device drivers and the sequence of the commands issued from `AUTOEXEC.BAT` is very important. The following assumes you will be using a Netbios application at the same time as TCP/IP for PS/2 is used. If this is not the case, only use the underscored lines.

1. For the TOKREUI/NETBEUI drivers, use the following sequence of drivers in `CONFIG.SYS` and commands in `AUTOEXEC.BAT`. The example is for IBM Token-Ring Network:

   CONFIG.SYS:

   ```
   1. DEVICE=NETDEV.SYS
   ```

   AUTOEXEC.BAT:

   ```
   1. TOKREUI
   2. OPENMON
   3. NETBEUI
   4. A Netbios program that opens the adapter
   5. TCP/IP for PS/2 program
   ```

2. For the *LAN Support Program*, use the following order of execution in `CONFIG.SYS` and `AUTOEXEC.BAT`. It is important that the *OPENMON* program is executed before a Netbios program causes the network interface to be opened.

CONFIG.SYS:

```
1. DEVICE=NETDEV.SYS
2. DEVICE=DXMA0MOD.SYS
3. DEVICE=DXMC0MOD.SYS
4. DEVICE=DXMT0MOD.SYS O=N
```

The parameter ***O=N*** on the DXMT0MOD.SYS entry means that the Netbios programming interface should be loaded into the system, but the adapter should not be opened (Open_On_Load= No).

AUTOEXEC.BAT:

```
1. OPENMON
2. A Netbios program that opens the adapter
3. TCP/IP for PS/2 program
```

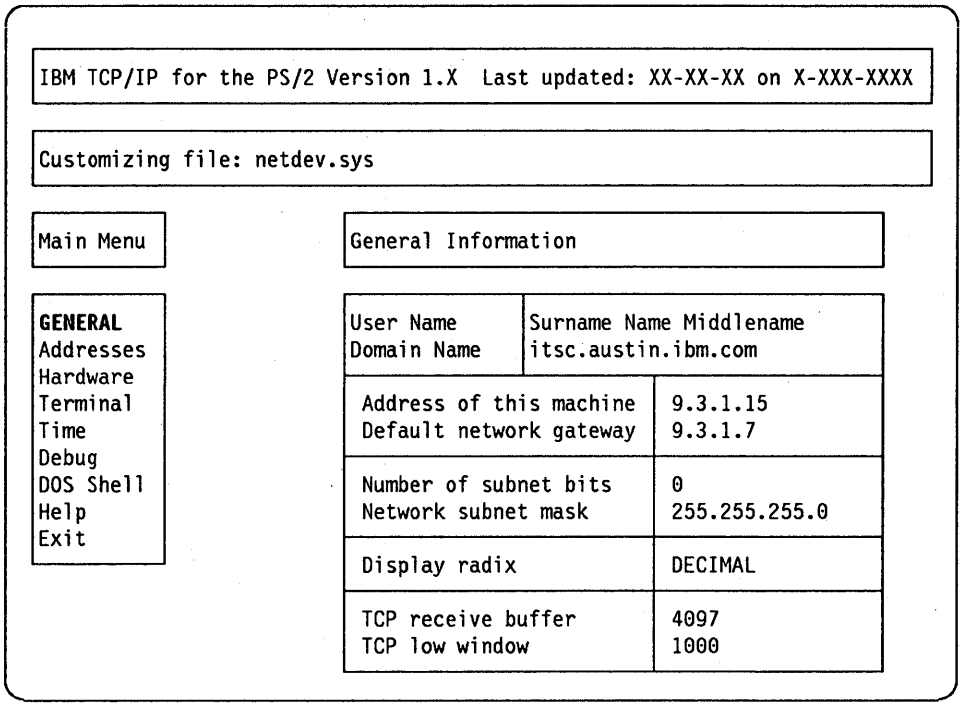
3. If the BIOS of the PC is dated prior to June 1985, an extra device driver should be installed from CONFIG.SYS. This device driver is called TIMERINT.SYS. To check if this driver is required, run the program BIOSDATE.EXE. Additional information can be found in the DXMINFO.DOC file. The three files mentioned are part of the *LAN Support Program*.

The next step is to customize the TCP/IP environment. This is done using a full-screen customizing program called ***CUSTOM***.

If the *CUSTOM* program is started without parameters, the program reads the current settings in memory and can be used to alter the current settings in memory. So, if TCP/IP for PS/2 is running and a customized parameter needs to be changed temporarily, this can be done without rebooting the machine. Normally the *CUSTOM* program would be started with:

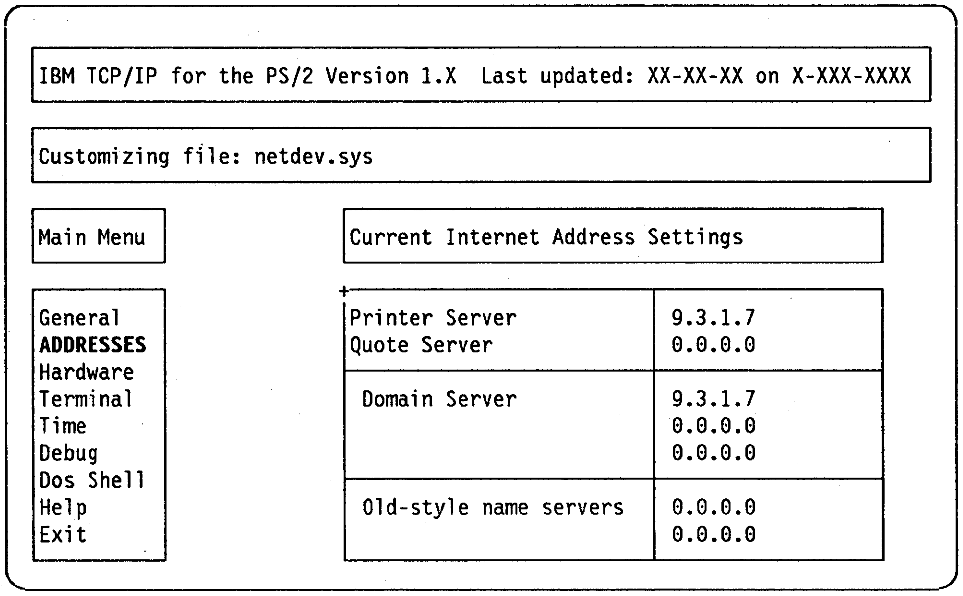
```
CUSTOM NETDEV.SYS
```

so that all the customizing parameters are taken from and stored in the NETDEV.SYS file. To make the parameters stored in NETDEV.SYS take effect, the machine must be rebooted with NETDEV.SYS in the CONFIG.SYS file. Figure 108 on page 232 and Figure 109 on page 232 shows two of the customizing screens of TCP/IP for PS/2.



```
IBM TCP/IP for the PS/2 Version 1.X  Last updated: XX-XX-XX on X-XXX-XXXX

Customizing file: netdev.sys

Main Menu              General Information

GENERAL                User Name               Surname Name Middlename
Addresses              Domain Name             itsc.austin.ibm.com
Hardware
Terminal                Address of this machine  9.3.1.15
Time                    Default network gateway  9.3.1.7
Debug
DOS Shell               Number of subnet bits    0
Help                    Network subnet mask      255.255.255.0
Exit
                        Display radix            DECIMAL

                        TCP receive buffer       4097
                        TCP low window           1000
```

Figure 108. Customizing Screen for TCP/IP for PS/2 (General)



```
IBM TCP/IP for the PS/2 Version 1.X  Last updated: XX-XX-XX on X-XXX-XXXX

Customizing file: netdev.sys

Main Menu              Current Internet Address Settings

General                Printer Server            9.3.1.7
ADDRESSES              Quote Server              0.0.0.0
Hardware
Terminal                Domain Server            9.3.1.7
Time                                             0.0.0.0
Debug                                            0.0.0.0
Dos Shell
Help                    Old-style name servers   0.0.0.0
Exit                                             0.0.0.0
```

Figure 109. Customizing Screen for TCP/IP for PS/2 (Addresses)

Use the cursor keys to go through the options on the main menu in the left side of the screen. Press ENTER to jump to the right part of the screen; press ESC to return to the main menu.

The screens shown are customized to fit the configuration we have become so very well acquainted with (Figure 66 on page 184). Our TCP/IP for PS/2 machine is connected via Token-Ring and has the Internet address 9.3.1.15. Notice that on TCP/IP for PS/2 we specify a ***static default route*** by filling in the field ***Default network gateway*** as shown in Figure 108. We have also added a subnet mask in accordance with the standards for our network as described in "Subnet Addressing" on page 218. In Figure 109 on page 232 we have entered the Internet addresses of our domain nameserver and print server.

Another (not shown) screen is called ***HARDWARE***. You must supply information about the communications adapter:

- Base I/O address
- DMA receive channel
- DMA send channel
- Interrupt vector number.

For most adapters, it is only necessary to know the Base I/O address and the Interrupt vector number. On PCs, these are usually set using "jumpers" or "switches" on the LAN adapter card, so check the switch/jumper settings for this information. On PS/2 Micro Channel based machines, this information is stored electronically in the system unit. Boot from reference diskette and choose "Show Configuration Menus". This will give the information needed for the *CUSTOM* program.

Note: The Token-Ring Adapter/A should use ***Interrupt Level 3*** when a 3270 Connection Adapter/A is also installed. The 3270 connection Adapter/A is fixed on Interrupt 2, which is the default for the Token-Ring Adapter/A. The default ***Base I/O Address*** for the Token-Ring Adapter/A is `X"CC00"`.

Yet another customizing screen is called ***TERMINAL***: The ***terminal settings*** are important if the PC or PS/2 is used for remote login to an ASCII host. This will be discussed in "TELNET Command" on page 235.

Other options from the main menu are:

- Specify the Internet address of a time server and tell TCP/IP for PS/2 to set the date and time on the local system over the network.
- Set debug options. A set of trace facilities that can be set to OFF or ON (default OFF).
- Invoke a DOS Shell. This function will start a secondary DOS command processor.
- Select the help screen that explains how to use the keyboard functions when customizing.
- Exit the program. The exit screen allows you to exit the *CUSTOM* program with or without saving the changes you have made.

When the customizing parameters have been stored in the *NETDEV.SYS* driver, the PC or PS/2 must be rebooted. After this, the TCP/IP for PS/2 application programs can be used.

TCP/IP for PS/2 Applications

This section describes some of the TCP/IP for PS/2 applications that differ from their counterparts in the AIX implementations of TCP/IP. For a detailed description of all applications, you must read the reference publication for TCP/IP for PS/2.

PING Command

This command sends an echo request to a foreign host and displays the foreign hosts response, if any. For example: `PING 9.3.1.7`. If you enter the *PING* command without parameters, it will cause TCP/IP for PS/2 to enter ***Server Mode***. This mooe gives you additional functions via subcommands. For example: The *"N"* command gives network statistics information.

FTP Command

The *FTP* command is used to transfer one or several files to/from a foreign host. The implementation is pretty much identical to that of AIX/RT. Observe, that in normal operation, TCP/IP for PS/2 does not have daemons running that allow file transfers to be initiated from other TCP/IP hosts.

LPR Command

The LPR command sends a text or graphics file to a print server on a foreign host. If you key:

```
LPR MYFILE.PRT
```

the file `MYFILE.PRT` is sent to the default print server. The default print server is found by first looking at the DOS environment variable `LPR_SERVER`. If the environment variable is not set, the print server defined in the *NETDEV.SYS* driver (if any) is used.

The flag **-s** can be used to explicitly specify what print server host to use. In either case, the file `/etc/hosts.lpd` (or `/etc/hosts.equiv`) on the print server must list your local machine's hostname or the print request will be rejected by the print server,

REXEC Command

The *REXEC* command is a remote execution client implemented using TCP protocol. For example, if a user wants to make a listing of files in the directory `/u/fribert` on a remote AIX system with hostname ***chris*** and using the password `jlpass`, the following command line could be used:

```
REXEC -schris -ufribert -pjlpass "ls -l"
```

The example below will execute the two commands *ps* and *df* on the AIX system and return the result to the DOS user's screen.

```
REXEC -ssys1 -udieter -pdhpass ps;df
```

You can omit the flags ***-s***, ***-u*** and ***-p*** by setting DOS environment variables. Use the following commands from the DOS command line or from a DOS batch file:

```
SET REXEC=sys1
SET USER=dieter
SET PWD=dhpass
```

When this is done,

```
REXEC ps;df
```

is identical to the second example above.

The *REXEC* command allows DOS redirection, so the output of the commands executed through *REXEC* can be sent to a DOS file or DOS device rather than to the display. Examples:

```
REXEC "ls -als /u/username" > username.lst
REXEC "ls -als /u/username" > lpt1:
```

TELNET Command

The *TELNET* command allows you to log on to a foreign host. This host could be an IBM System/370 host running TCP/IP for VM, or it could be an ASCII host (for example AIX/RT or AIX/PS2). To establish a telnet session with a remote host, type: `TELNET hostname` or `TELNET Internet Address`. Examples:

```
TELNET m135
TELNET 9.3.1.7
```

Once the session is established, the foreign host will send its login screen, so a user can log in to the host. Note that AIX hosts must have at least one pseudo terminal available for the connection to be established.

TCP/IP for PS/2 TELNET to an ASCII host emulates a ***Heath H19*** ASCII terminal. The Heath H19 is a line mode teletype terminal. Unfortunately, this terminal emulation does not support the full-screen applications on an AIX host. The *devices* and *minidisks* commands, for example, will not run correctly when used from TCP/IP for PS/2.

To make these full-screen commands work with Heath H19 emulation, you must add entries in the `terminfo` library of the AIX host. A `terminfo` file for the Heath H19 terminal is not provided with AIX, so it is up to the user to build a source `terminfo` file and compile it with the *tic* command. The following steps explain how you do this:

1. Create the source `terminfo` file. On AIX/RT call the file `h19.ti`. The file should look as shown in Figure 110.

```
h19|IBM TCP/IP Telnet Terminal,
        cr=^M,cols#80,lines#24,clear=\EE,ind=\ES,km,
        kf1=\ES,kf2=\ET,kf3=\EU,kf4=\EV,kf5=\EW,kf6=\EP,kf7=\EQ,kf8=\ER,
        kf9=\Ec,kf10=\Ed,kf11=\Ee,kf12=\Ef,
        kcuu1=\EA,kcud1=\EB,kcuf1=\EC,kcub1=\ED,
        cuu1=\EA,cud1=\EB,cuf1=\EC,cub1=\ED,
        cup=\EY%p1%' '%+%c%p2%' '%+%c,
        el=\EK,ed=\EJ,dch1=\EN,dl1=\EM,
        il1=\EL,sc=\Ej,rc=\Ek,smso=\Ep,rmso=\Eq,
        home=\EH,
        khome=\EE,kend=\EF,kpp=\EG,knp=\EH,
        ktab=^I,cbt=^O, dch1=\EN, kdch1=\177,kich1=\273,
```

Figure 110. Terminal Information File for Heath H19. This file has been tested on AIX/RT and AIX/PS2.

Figure 111 shows the meaning of some of the stanzas in Figure 110.

Key	Code	Key	Code
F1	\ES	Alt-a	\x9E
F2	\ET	Alt-b	\xB0
F3	\EU	Alt-c	\xAE
F4	\EV		
F5	\EW		
F6	\EP		
F7	\EQ		
F8	\ER		
F9	\Ec		
F10	\Ed		
F11	\Ee		
F12	\Ef		

Note: \E = Escape and \x = Hex

Figure 111. Meaning of Some Entries in Heath H19 Definitions

2. Set your ***TERMINFO*** environment variable as follows:

   ```
   TERMINFO=/usr/lib/terminfo
   export TERMINFO
   ```

3. Compile this source file using the AIX command *tic*, for example:

   ```
   tic -v h19.ti
   ```

   This will put the translated file in /usr/lib/terminfo/h/h19. The size of the total compiled entries must not exceed 4096 bytes.

4. Make sure the thus defined terminal type is used for the pseudo terminal when logging in from TCP/IP for PS/2. If a given user will only log in from the TCP/IP for PS/2 system, you could add the following lines in the file /u/username/.profile, where "username" is the user ID:

   ```
   TERM=h19
   export TERM
   stty -istrip cs8
   ```

   The last line enables 8-bit character support which is required for National Language Characters. If users may log in from several terminals, the user will need to issue the same three commands from the command line immediately after login.

5. Final customizing is done with the TCP/IP for PS/2 *CUSTOM* command. In the TERMINAL menu, the settings should be:

   - Backspace Character: **Backspace** (Not DEL)
   - Wrap on end-of-line: **ON**

Once these steps have been followed, the user will be able to use most of the commands on the AIX system. The *INed* editor, though, has its own terminal definition files in the def.trm file. Read the publication carefully, should you wish to copy and modify these terminal definitions. The *tdigest* command is used to compile the def.trm source code.

Other Functions Using TELNET

When a telnet session is established, other functions than terminal emulation are available. A subcommand menu can be invoked using the keysequence: `"Ctrl-]"`:

Some of the subcommands are:

```
* c - Closes the connection and exits from telnet
* U - Turns on status and clock on line 25
* u - Turns off status and clock on line 25
* a - Sends an "Are you there" inquiry to the target host
* I - Displays the PC Internet Address
* T - Enter TFTP Server Mode
```

One very useful function is the possibility to invoke the *TFTP* program for file transfer while a telnet session is active. This is one of the few ***server mode*** applications of TCP/IP for PS/2. TFTP enters this mode when you enable TFTP from the TELNET subcommand line with the command `T`. The actual file transfer is invoked by the TFTP service on the foreign host. When the file transfer starts, the TFTP server on TCP/IP for PS/2 asks the telnet user for permission to accept the request. The user must respond with `Ctrl-]y` or `Ctrl-]n`.

If you need to use PC-DOS functions while a telnet session is active, use `"Ctrl-]!"`. This invokes a nested DOS command interpreter so you can use DOS commands without stopping the telnet connection. Do not use network commands while running this nested DOS command interpreter.

If you need a connection to a S/370 host using the PC 3270 Entry Lever Emulation Program, start this emulation program first, then use the hotkey (`Alt-Esc`) to get to the DOS shell. From there, establish the telnet connection. Now you can use the `Alt-Esc` hotkey to jump between PC 3270 emulation and the telnet connection. Observe, that because the PC session of the PC 3270 Entry Level Emulator is suspended while in the 3270 session, your host will not respond to TCP/IP packets. Some hosts could time out if the telnet session is inactive for a longer period of time.

TFTP Server Mode

The *TFTP* server on TCP/IP for PS/2 can also be started when no telnet session is active. TCP/IP for PS/2 then enters ***server mode***. While in this mode, the PC cannot be used for anything else. In this mode, the PC can even be a print server, if the client TFTP (the remote host) specifies that file transfer shall take place to the DOS device ***LPT1***, rather than to a DOS file.

Mail System

TCP/IP for PS/2 provides mail commands that allow mail to be exchanged between TCP/IP for PS/2 and other TCP/IP hosts with mail server capabilities using the ***Post Office Protocol*** (POP). TCP/IP for PS/2 itself does not provide mail server (mailer) functions. Among the mail commands you can use are:

INC	Transfers/receives new mail from mail server
COMP	Composes and sends a new mail message
POST	Sends a mail message directly through SMTP
SEND	Sends mail and verifies that the mail server account exists
NEXT	Display the next mail message
PREV	Display the previous mail message

SCAN	Generates a one-line summary of each message
SHOW	Displays a mail message
FOLDER	Selects or lists contents of a mail folder
REFILE	Moves a mail message to another folder
RMF	Removes a mail folder
RMM	Removes a mail message.

Internet Router

When configured as an Internet router (gateway), the PC or PS/2 running TCP/IP for PS/2 is dedicated to this task. The machine must be an IBM PC/AT or a PS/2 with a 16- or 32-bit processor. Routing of packages can take place between more than two networks. TCP/IP for PS/2 as an Internet router supports the following protocols: IP, ICMP, GGP and HMP.

TCP/IP on IBM System/370 Hosts

IBM TCP/IP for VM (5798-FAL) and ***IBM TCP/IP for MVS (5585-061)*** are Program Offerings (POs) that allow the VM/SP, VM/SP HPO, VM/XA, MVS/370, MVS/XA or MVS/ESA Operating Systems to participate in a multi-vendor Internet network using the TCP/IP protocol set. The products allow a VM/CMS or MVS/TSO user to interface with other systems that have implemented the TCP/IP protocols. This connectivity includes the ability to transfer files, send and receive mail and log on to a remote host in a network of different systems and across network boundaries.

The TCP/IP for VM and TCP/IP for MVS programs use a System/370 channel or an IBM 9370 integrated adapter for connection to the selected network. The network protocols supported are IBM Token-Ring, Ethernet LAN, ProNET (only TCP/IP for VM) and DDN X.25, but only Token-Ring and Ethernet attachment will be discussed in this publication. The description in this publication applies to Release 1.2 of IBM TCP/IP for VM and to Release 1.0 of TCP/IP for MVS.

Hardware and Software Prerequisites

TCP/IP for VM and TCP/IP for MVS can communicate with AIX based systems using an IBM System/370 (IBM 30XX, IBM 43XX or IBM 9370) processor.

For the IBM 30XX and IBM 43XX series of processors, the connection to the Token-Ring and Ethernet LANs is via the IBM 8232 LAN Channel Station connected to a System/370 channel on one side and to one or more LANs on the other side (IBM Token-Ring, PC Network and Ethernet LANs).

For the IBM 9370 connection, the IBM 9370 Telecommunications Subsystem Controller and associated LAN adapters can be used to connect to the LAN:

- The IBM 9370 Token-Ring Subsystem Controller consists of an IBM 9370 Communications Processor (feature #6130) and an IBM 9370 Token-Ring Adapter (feature #6034)
- The IBM 9370 IEEE 802.3 Local Area Network Subsystem Controller consists of an IBM 9370 Communications Processor (feature #6130) and an IEEE 802.3 LAN Adapter (feature #6035).

Note: When using TCP/IP on the IEEE 802.3 LAN Adapter, this adapter cannot be shared with other applications.

IBM TCP/IP for VM is designed to operate with one of the following Operating Systems:

- VM/SP Release 4, or later (5664-167)
- VM/SP HPO 4.2, or later (5664-173)
- VM/XA SP Release 1, System/370 mode (5664-308).

IBM TCP/IP for MVS is designed to operate with one of the following Operating Systems:

- MVS/SP (MVS/370) Version 1 Release 3.5 (5740-XYS/XYN)
- MVS/SP (MVS/XA) Version 2 Release 1.3 (5740-XC6, 5665-291)
- MVS/SP (MVS/ESA) Version 3 Release 1.0 (5685-001/002).

TSO-E Release 1.3 is required for users of SMTP and TELNET.

Functional Description

IBM TCP/IP for VM and TCP/IP for MVS have implemented the following common functions:

- *File transfer* to remote terminals using the File Transfer Protocol (FTP). The file transfer command uses the same syntax as the AIX FTP commands, with exceptions as necessary due to the VM and MVS environments. It is possible to use a CMS or TSO command while in the FTP subcommand mode. You enter the FTP subcommand mode by invoking FTP without options from the CMS or TSO command mode.
- *Electronic Mail* using the Simple Mail Transfer Protocol (SMTP). TCP/IP also allows PROFS users to send and receive notes and files using SMTP.
- *Remote terminal access* using the ***telnet*** protocol.
  - If you use *telnet* from a VM or MVS host to another VM or MVS host, you can operate the remote host in full screen mode. Telnet from an AIX/RT or AIX/PS2 console into a TCP/IP for VM or TCP/IP for MVS host will support full screen mode.
  - If you use telnet to connect from a VM or MVS host into an ASCII host, operation will be one line at a time (line mode).
  - If you use telnet from an ASCII terminal or ASCII host and connect into a VM or MVS host, operation will be one line at a time (line mode). Some vendor software products allow ASCII terminals to access 3270 full-screen applications, for example the VM software ***SIM3278/TCPIP*** or the MVS software ***SIM3278/VTAM*** offered by ***Simware Inc.*** In the MVS environment, this is also supported by ACF/VTAM for MVS (5665-313) Version 3 Release 1.1 and later versions. Another possibility is to install an ASCII-3270 full screen transformation program on the ASCII system.
- *C and PASCAL macro interface to TCP and UDP*. A C Language interface is provided to allow programs developed using the IBM C Compiler and Library to directly access the TCP and UDP protocol boundaries. A PASCAL language interface is provided to allow programs written in PASCAL to directly access the TCP and UDP protocol boundaries.
- *Administrative Functions*. Statistical information on file transfer rates will be shown when using the *FTP* command. The *NETSTAT* command allows the user to query the TCP/IP virtual machine (VM) or TCP/IP address space (MVS) for information about the network. You can, for example, display information about active TCP connections, active telnet connections and network names/addresses. The *PING* command is implemented as well.
- *SNA Network Link*. Allows connection of VM and/or MVS TCP/IP networks via an SNA Network (ACF/VTAM required). SNA LU0 protocol is used to link the systems.
- Network File System (NFS) feature. VM or MVS systems with the NFS feature may act as a file server for AIX/NFS systems or other systems that have the NFS 3.2 client function installed. The NFS features do not include the NFS client function. The Remote Procedure Call (RPC) function allows the scheduling of processes across the network. The NFS and RPC protocols adhere to the External Data Representation (XDR) specification, which allows the protocols to be machine and language independant.
- X-Windows client function support (Version X.11).

AIX System Affinity

For details about installation and customizing procedures for TCP/IP for VM and TCP/IP for MVS, please refer to the publications listed last in this chapter. For TCP/IP for VM, also see the `I5798FAL MEMO` file that comes with the product.

The following section will concentrate on the functionality of TCP/IP for VM and TCP/IP for MVS as seen from an AIX systems point of view.

X-Windows

The VM/CMS X-Windows and MVS/TSO X-Windows Systems Application Program Interfaces (API) are discussed in "X-Windows Clients" on page 308.

Telnet

The AIX *telnet* command supports an option called ***terminal negotiation***. If the remote host supports this, *telnet* sends the local terminal type to the remote host. If the remote host does not accept the local terminal type, *telnet* attempts to emulate an ***IBM 3270*** terminal and then a DEC ***VT100*** terminal. If you specify a terminal type to emulate by setting the ***EMULATE*** environment variable in AIX, *telnet* does not negotiate terminal type.

If the client and server negotiate to emulate an IBM 3270 terminal, the keyboard mapping is taken from the first of the following files that exist:

- `$HOME/.3270keys`, user specific 3270 keyboard mapping
- `/etc/3270.keys.rt`, standard 3270 keyboard mapping (for console)
- `/etc/3270.keys`, base 3270 keyboard mapping for use with limited function terminals.

In the keyboard mapping files it is also possible to change the color setup for the 3270 emulation.

The *telnet* command supports various screen sizes and different 3270 terminal types; for example, IBM 3278 Model 2, IBM 3278 Model 3 and IBM 3278 Model 4, depending on your display setup. Note the following:

- Make sure that the dimensions of the display or X-Windows window is more than 23 rows and more that 79 columns, or you will not receive the VM or MVS logon screen. The process seems to be hanging but can be terminated with `Ctrl-T`.
- If your AIX terminal or X-Windows window has from 24 to 31 rows, you will get the Model 2 (24x80) 3270 screen.
- If your AIX terminal or X-Windows window has from 32 to 42 rows, you will get the Model 3 (32x80) 3270 screen.
- If your AIX terminal or X-Windows window has 43 rows or more, you will get a 43x80 3270 screen.

When you use *telnet* to connect to TCP/IP for VM or TCP/IP for MVS, you will receive the normal logon screen of the VM or MVS system. You must then log on as a VM/CMS or MVS/TSO user. TCP/IP commands, as well as all CMS and TSO commands, can be used. When you log off the VM/CMS or MVS/TSO system, you also terminate the telnet session and the AIX prompt is displayed.

Using telnet from the VM system to the AIX system works a little different. The main reason is that a VM 3270 terminal behaves different from an AIX system display. Some examples:

- When you have used telnet to logon to an AIX system, you still get ***HOLDING*** and ***MORE*** on the screen, so that every time the screen is filled with data, you must press the `Clear` key to get the next screen.
- The screen is operated in "line-mode".

File Transfer

File transfer using the *ftp* command can be initiated from both the AIX system and from an IBM 3270 terminal connected to the VM system. Example:

A 3279 terminal user running VM/CMS wants to copy the file `/u/username/user.data` from an AIX system with hostname ***sys1*** to a VM/CMS file. The following commands are entered from the 3279 terminal:

```
ftp sys1
get /u/username/user.data  user.data.a
quit
```

These commands will start the FTP connection, transfer the remote file `/u/username/user.data` to the VM/CMS user and name the file `USER DATA A` (filename filetype filemode) and then the FTP connection will be stopped. Both ***binary*** and ***text*** file transfers are supported.

TCP/IP for VM comes with several translation files that you can edit, and a *CONVXLAT EXEC* which converts those files to a binary form for use by the TCP/IP modules. The editable files have a filetype of *TCPXLATE* and the binary files have filetype *TCPXLBIN*. The TCP/IP applications: mail, *SRVRFTP, FTP, TFTP* and *TELNET* first look for `xyz TCPXLBIN`, where "xyz" is the command name (for example `TELNET TCPXLATE`). Then they look for `STANDARD TCPXLBIN`. *FTP, TFTP*, and *TELNET* also have a "TRANSLATE" option on the command line, which can be used to specify an alternate translation table name.

When transferring a text file that was saved with the INed editor, you should watch out for ***tab control characters***. If using ***XEDIT*** to edit an AIX text file with these special control characters, you will see that the tab control character is not interpreted as tabbing but is shown as a displayable character.

File transfer from an AIX system using the *ftp* command is a little more complicated than file transfers between AIX systems. You must:

1. Invoke the *ftp* command, for example: `ftp vmname`. When prompted for a user name, enter the name and then the password of the CMS machine you want to transfer your file to.
2. Use the *cd* subcommand to point to the CMS minidisk you want to transfer the file to. For example: `cd vmuser.198`, where "`vmuser`" is the VM Userid and "`198`" is the minidisk number.
3. Use the *account* command to get access to the minidisk you want to transfer your file to, for example: `account mdiskpw`, where "`mdiskpw`" is the CMS minidisk password. Be sure to enter a "write" password or you will get "read-only" access to the minidisk.

4. Use the *put* subcommand to send the file, for example: `put /localfile filename.filetype`. Do not specify a VM filemode, as you have already (indirectly) chosen a filemode with the *cd* command.
5. Use the *bye* command to leave the *ftp* subcommand mode.

If you are transferring large VM modules from one VM system to another VM system via an AIX system system, some additional steps are required. This is because there are no record boundaries in "binary" or "image" files on an AIX system, but there are on VM. When you transfer a file from VM to the RT in "image" mode, you loose the information about record boundaries and there's no way you can recover that information. There are two ways you can maintain the record information:

1. By using the ***PACK*** option of CMS *COPYFILE*
2. By using the CMS ***DISK DUMP*** command.

In the following sections, we'll assume we want to copy the CMS file `MYPGM MODULE A` from one VM system to another.

Using the PACK option of CMS COPYFILE

1. Copy the file with:

   ```
   copyfile mypgm module a mypgm2 module a (pack
   ```

2. Take a note the logical record length (LRECL) of the packed file.
3. Send the file `mypgm2 module` to the AIX system using any tool you have, for example a 3270 emulator or VM *ftp* (set option "binary" with the subcommand `binary f LRECL`, where "`LRECL`" is the logical record length for the packed file). Put the file in `/tmp/mypgm2.module`.
4. Send the AIX file to VM TCP/IP on the other VM system using the following commands:

   ```
   ftp vmname
   cd vmmuser.191
   account passwd
   binary f LRECL
   put /tmp/mypgm2.module mypgm2.module
   ```

5. Then on the second VM system, issue the following command:

   ```
   copyfile mypgm2 module a mypgm module a (unpack
   ```

Using the CMS DISK DUMP command

1. Transform the module to be transferred to fixed record length of 80 bytes. Use the commands:

   ```
   spool d *
   disk dump mypgm module a
   read mypgm module a
   ```

2. Send the file `mypgm module a` to the AIX system using any tool you have, for example an 3270 emulator or VM *ftp* (set option "binary" with the command `binary f 80`). Put the file in `/tmp/mypgm.module` on the AIX system.
3. Send the AIX file to VM TCP/IP on the other VM system using the following commands:

```
ftp vmname
cd vmmuser.191
account passwd
binary f 80
put /tmp/mypgm.module mypgm.module
```

4. Then, on this second VM system, issue the following commands:

```
spool d *
pun filename filetype (NOH
disk load
```

VM TCP/IP Additional Functions

Additional Network Attachments

- TCP/IP for VM supports attachment to a ProNET network.
- The IBM 9370 Communications Processor supports TCP/IP on the X.25 Communications Subsystem (feature #6031).

Electronic Mail

TCP/IP for VM supports the sending of notes and files using the Simple Mail Transfer Protocol (SMTP) and provides revised *NOTE* and *SENDFILE* commands. By using these commands rather than the standard CMS NOTE and SENDFILE facilities, the users can send mail and files via a TCP/IP network in the same manner as CMS users transmit data over an RSCS network. TCP/IP also supports PROFS sending and receiving notes and files using the Simple Mail Transfer Protocol (SMTP). Notes and files received via SMTP will be put in the user's virtual reader.

Mail transfer between TCP/IP and CMS is handled by a CMS virtual machine ***SMTP***. See also "Mail to VM" on page 289.

TCP/IP for PS/2

TCP/IP for VM includes TCP/IP for PS/2 that will allow IBM PC or IBM PS/2 users on an IBM Token-Ring or Ethernet LAN to communicate with users on the VM host system. See "TCP/IP for PS/2" on page 229 for further information about TCP/IP for PS/2.

Remote Execution Daemon

A remote execution daemon (*REXECD*) allows remote execution of VM EXEC's from AIX TCP/IP and from TCP/IP for PS/2, using the *REXEC* command.

Domain Nameserver and Resolver

TCP/IP for VM has domain nameserver functions that allow a VM system to function as the domain nameserver for a TCP/IP network.

MVS TCP/IP Additional Functions

Additional Network Attachments

- TCP/IP Network connection via X.25 is supported. IBM TCP/IP for MVS provides support for connection to remote TCP/IP networks via X.25 over an IBM 3720/3725/3745 Communications Controller as a network interface processor using VTAM and NPSI. TCP/IP for MVS and the associated interface feature in the IBM 3720/3725/3745 supports connection to X.25, including Defense Data Network (DDN) and Public Data Network (PDN) networks.

- Support (driver) for HYPERchannel. Support is provided for connection of the IBM TCP/IP for MVS program to an NSC HYPERchannel network using an NSC IBM channel adapter. Support conforms to specifications outlined in *DARPA RFC1044* for 16 bit address configuration.

Electronic Mail

Electronic mail to remote hosts uses the TSO/E user interface to the Simple Mail Transfer Protocol (SMTP). TSO commands is used to send and receive mail. All SMTP configuration command options can be permanently specified in a configuration file. SMTP also supports the security and accountability functions:

- Restriction lists (to deny individual users or classes of users access to SMTP services)
- Complete logging (source, destination) of all traffic processed by SMTP.

Specified Operating Environment

- At least 100 Kbytes of Real Main Storage for use by TCP/IP for MVS
- At least 30 Mbyte Direct Access Storage for use by TCP/IP for MVS
- One tape or cartridge drive for installing TCP/IP for MVS.

SNA Network Link Requirements

Table 10. SNA Network Link Requirements

Connected via	VTAM Version	NCP Version	NPSI Version
IBM 3720	V3R2 or V3R1.1 (MVS)	V4R2	V1R4.3
	V3R2 or V3R1.1 (MVS)	V5R1	V3R1
	V3R2 (MVS)	V5R2, 2.1	V3R2
IBM 3725	V3R2 or V3R1.1 (MVS)	V4R2	V1R4.3
	V3R2 (MVS)	V4R3, 3.1	V2R1
IBM 3745	V3R2 or V3R1.1 (MVS)	V5R1	V2R1
	V3R2 (MVS)	V5R2, 2.1	V3R2

TCP/IP on non-IBM Systems

TCP/IP is available on every major and most other non-IBM computers and operating systems. The authors of this publication have successfully connected into a large number of non-IBM systems. A list of some, such systems is given in "AIX/RT NFS Connections to Non-IBM Systems" on page 263, but that list represents only a small fraction of the number of systems you can "talk TCP/IP" to.

Generally, the standard TCP/IP applications are supported by almost any TCP/IP implementation, but there may be several restrictions as to how you can use them. The difference in hardware architecture, for example, may render file transfer of binary files useless (though certainly possible).

When using *telnet* from one host type to a different host type, there may be restrictions like:

1. The connection may time out after a certain time with no activity on the connection.

2. Full-screen mode is not necessarily supported.

Reference Publications for This Chapter

TCP/IP is described in:

IBM RT Interface Program for use with TCP/IP, GC09-1214
AIX PS/2 TCP/IP User's Guide, SC23-2047
TCP/IP for VM, Installation and Maintenance Manual, GC09-1203
TCP/IP for VM, Command reference manual, GC09-1204
TCP/IP for VM, Programmer's Manual, GC09-1206
TCP/IP for VM, Network File System and Remote Procedure Call, SC09-1274
Managing the AIX Operating System (AIX/RT), SC23-2008
Managing the AIX Operating System (AIX PS/2), SC23-2031
Internetworking With TCP/IP, Principles, Protocols and Architectures by Dougles Comer. Prentice Hall, ISBN 0-13-470154-2.

Network File System (NFS)

The ***Network File System*** (NFS) is a de facto standard system for sharing of directories across TCP/IP networks. Developed by SUN Microsystems, NFS is now implemented by all major suppliers of UNIX based systems. In addition, NFS is implemented on several systems with other operating systems such as IBM's VM and MVS. This chapter will describe the implementation of NFS on AIX/RT and discuss other IBM implementations in less detail.

Overview

Network File System is based upon the ***Remote Procedure Call*** (RPC) package which allows communication between processes on different machines and on the ***External Data Representation*** (XDR) standard, which describes protocols that allow dissimilar machines to exchange data via RPC. NFS itself defines the NFS protocol on top of RPC and XDR. The NFS protocol is used by the daemons and commands supplied with the NFS product. RPC- and XDR libraries and the protocol definitions together constitute an application program interface that allow application programs to use RPC (and XDR) at the same level these are used by NFS. We shall not discuss the application program interface in this publication.

AIX/RT Network File System is an implementation of NFS at level 3.2. Other, current IBM implementations of NFS are at the same level. NFS provides transparent, remote access to remote directories in networks of heterogenous machines. Remote directories can be ***mounted*** over local directories or directory "stubs" (empty directories) and accessed by local programs as if they were local directories. Hosts that mount directories from other machines are called ***client*** hosts; hosts that allow other hosts to mount their directories are called ***server*** hosts. A host may be server for one or more clients and may at the same time be a client of one or more other servers.

NFS uses a ***stateless*** protocol. This avoids complex crash recovery; a client just resends requests until a response is received. Actually, a client can not tell the difference between a slow server and a server that is no longer operational. This minimizes (some say eliminates) the chances of data losses due to a server crash.

The stateless design of NFS means that neither server nor client keeps any information about the ***state*** of a mount. Whenever a service is required (say, reading the next bytes from a remote file), the RPC call issued from NFS carries all information required to identify the function requested, the data required, etc. This goes very well in hand with the ***connectionless*** User Datagram Protocol (UDP) which current implementations of NFS are dependent upon for transport services.

Like Distributed Services, NFS uses "virtual inodes", ***vnodes***. When a directory is mounted, the ***inode*** defining that directory is linked to a vnode, and every request for data belonging to the mounted directory is directed over the network to the remote host. Before a given directory on a server can be mounted on a client, the ***file system*** where the directory belongs must be ***exported*** from the server. File systems can be exported to specific client hosts or to ***the world***. Exported file systems are listed in the file /etc/exports on the server.

It's important to note the difference from DS, here: With DS the "exporting" can be done on file-, directory or file system level and any desired granularity can be achieved. NFS, on the other hand, exports only file systems. A client that's on the "export list" may mount the entire file system or any directory within it. Mounting of single files is not supported with NFS; DS does support file mounts. Moreover, NFS does not support the use of remote devices for backup/restore as DS does. Also, NFS does not support ***inherited mounts***.

Another important aspect of NFS and its way of exporting and mounting remote directories is the dependency of standard authentication mechanisms for controlling user access. Whenever a directory is mounted, it appears to the client system with the ***numeric*** user- and group ID it has on the server. This numeric ID may map to different users on different client hosts. There is no mechanism (like the User/Group Table of DS) to map user names and permissions across the network. Consequently, networks using NFS tend to either restrict exports to file systems that everybody may access or to implement some level of ***Single System Image*** through the Yellow Pages product.

The ***Yellow Pages*** (YP) product provides a distributed network lookup service. It, again, is using the server/client model where servers store data bases ***(maps)*** that can be queried from clients. YP servers can answer queries depending on the information they have been asked to store. Generally, YP servers are queried for information about user- and group IDs, network addresses, gateways, etc. In each ***YP domain*** there can exist only one ***YP master server***, but there may be an unlimited number of ***YP slave servers***. Each client has one and only one YP server as its primary server.

The one, final thing we'd like to address at this point is limited availability of a locking mechanism for mounted directories. Of the current and announced implementations of NFS, only NFS on AIX/370 provides the ***lockd*** daemon, which is an optional feature of NFS 3.2 as defined by SUN Microsystems. Without the *lockd* daemon, record or file locking is not possible. You should be aware of this fact when deciding how to use NFS. In fact, if your network has only IBM RTs, you probably should be using Distributed Services to get the benefits of record and file locking across the network.

We shall now give you an overview of the implementations of Network File System that are currently available from or announced by IBM.

IBM Implementations of NFS

NFS is currently available or announced for the following IBM systems:

1. AIX/RT
2. AIX PS/2
3. AIX/370
4. VM
5. MVS.

NFS on AIX

Network File System is currently available on AIX/RT Version 2.2.1. The product is announced, but not yet available, for AIX PS/2 and AIX/370. All AIX implementations support NFS servers and clients as well as Yellow Pages.

Hosts combined into clusters using the TCF ***Transparent Computing Facility*** (as announced for AIX/370 and AIX PS/2) can appear to Network File System as having a single, unified file system.

Network File System is an optional licensed program on all AIX machines. Yellow Pages is part of the licensed program but is a separately installable option. The product numbers are:

AIX/RT	5601-159
AIX PS/2	5713-AFG
AIX/370	5688-046

NFS on VM

Network File System is currently available as a feature of Version 1.2 of TCP/IP for VM. VM/NFS supports only the NFS server function (VM hosts can not run as clients) and Yellow Pages is not provided. The product number for TCP/IP for VM is:

VM/NFS	5789-FAL

Optionally, NFS on VM can use the IBM Information Protection System Cryptographic Programs for VM/CMS (5796-PPK) or a customer supplied encryption procedure.

NFS on MVS

NFS is announced as a feature of TCP/IP for MVS but not available at the time of writing. On MVS, Network File System will support only the NFS server function, and Yellow Pages is not provided. The product number for TCP/IP for MVS is:

MVS/NFS	5685-061

AIX/RT Network File System and Yellow Pages

Now, let's concentrate on the present. Network File System and Yellow Pages have been available on AIX/RT since late 1988 and has shown to be a good and stable implementation of the NFS 3.2 protocol. We shall take you through the following sections:

- Hardware and software prerequisites
- Installing and preparing for NFS and YP
- Customizing an NFS and YP master server
- Customizing an NFS and YP client
- Customizing a YP slave server
- Using NFS and YP
- NFS and YP hints and pitfalls.

Hardware and Software Prerequisites

Network File System and Yellow Pages share the hardware requirements with TCP/IP. On an IBM RT, the following adapters may be used:

- IBM RT Token-Ring Adapter
- IBM RT Baseband Adapter.

Actually, NFS will run across X.25 and probably via an asynchronous (SLIP) TCP/IP connection as well, but if hosts go down or you have network problems, you may run into all sorts of data integrity problems. The reason that NFS and YP can not run across X.25 and SLIP is that TCP/IP does not support broadcasts for these connections. YP depends on broadcasts to get started so it can never be used.

NFS, on the other hand, only uses broadcasts when hosts start. When a NFS is started, it uses broadcasts to tell all hosts that it has forgotten what NFS mounts it had (it doesn't know if it had any, so it does an unconditional broadcast). Any server that the client had mounted directories from when it was rebooted will discard information about these mounts. Both parties can now start from scratch.

However, if the server never gets the broadcast, and the client then issues another mount request, the two will get intermixed and data integrity exposed. There may be ways to overcome this, for example, by explicitly unmounting all potential mounts in /etc/rc.nfs with: *unmount -n host_name* for all hosts on the network after NFS is started. However, doing so is ***entirely your own responsibility***. Officially, neither X.25 nor SLIP is supported.

Apart from the appropriate VRM device drivers and adapter support software, the only additional software requirement is TCP/IP, which is a separately selectable option of the AIX/RT Operating System.

Installing and Preparing for NFS and YP

To install the NFS and YP products on AIX/RT, you must first install the appropriate communications adapters and associated adapter driver software, then install TCP/IP. For details about installing TCP/IP, see "Basic AIX TCP/IP Customizing" on page 184. In short, you need to do the following to get TCP/IP up and running:

1. Mount the adapters you will be using.
2. Use *installp* to install the VRM device drivers for the adapters.
3. Use *installp* to install TCP/IP from the distribution diskettes.
4. Use *updatep* to apply any updates to your system.
5. Use *devices* to add the ***adapter*** and the ***data link***[21] appropriate for the adapter you will be using for TCP/IP.
6. Use *devices* to add ptys for remote login and telnet.
7. Edit the /etc/hosts file to define your local host.
8. Run *chparm* to change your system's node name.
9. Edit /etc/master and insert your node name in the sysparms stanza:

10. Edit /etc/rc.tcpip and insert your system's node name in the command /bin/hostname.

11. Edit /etc/rc.include and remove the comments so that /etc/rc.tcpip will be executed at system start up.

Now see if your system runs TCP/IP okay. Provided you've done everything correctly (and have in fact connected the adapters to the relevant communication media), you should at least be able to contact other systems if you know their ***Internet*** addresses. For now, let's assume you have three systems, as follows:

m135	Internet address = 9.3.1.7
chris	Internet address = 9.3.1.4
sys1	Internet address = 9.3.1.1

When working on host ***chris***, you can check that host ***m135*** responds with the command:

```
ping 9.3.1.7
```

If your TCP/IP installation uses a nameserver, or if you have added the host names and Internet addresses of all machines to the /etc/hosts/ file of every machine, then you should be able to do the same using:

```
ping m135
```

Either way, if you don't get a message saying "***0% packet loss***" or something to that effect, then fix whatever you did wrong before proceeding with the installation of Network File System.

Now you're ready to install Network File System and Yellow Pages. The following steps assume that you want both installed. If you don't, skip what we say about Yellow Pages.

1. Using the *installp* command, install the Network File System and Yellow Pages products from the distribution diskettes.

2. Use the *updatep* command to apply any updates you may have for Network File System and/or Yellow Pages.

That's it! Installing is easy. Now on with the customizing of Network File System and Yellow Pages so you can start using them.

Customizing an NFS and YP Master Server

Let's assume we use the same three systems as we mentioned above. They all run Token-Ring and the host ***m135*** is selected to be the YP master server. It will also be the predominant (!) NFS server. The configuration is shown in Figure 112 on page 252.

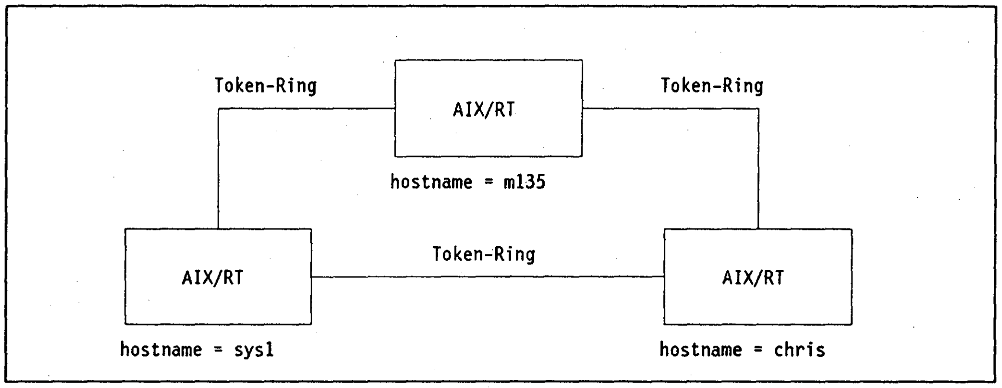


Figure 112. NFS and YP Configuration Sample

Note that the NFS and YP servers do ***not*** have to be on the same host. In small networks like ours though, it makes things simpler for the administrator. We will begin by customizing the host ***m135***, since we need a server before any client can do anything. You'll have to do the following:

1. Log in as superuser.
2. Edit the file `/etc/rc.include` to uncomment the lines that start `/etc/rc.nfs`.
3. Check the `/etc/rc.tcpip` file to see that the *portmap* and *inetd* daemons are started. They should be, or you'd have had trouble running TCP/IP.
4. Edit the `/etc/inetd.conf` file and make sure the ***mountd*** daemon is uncommented. You may also want to uncomment other daemons, but for a server, you definitely need the *mountd* daemon which answers mount requests from clients.

   Notice that the daemon is not started explicitly. By editing `/etc/inetd.conf` you merely tell the *inetd* daemon that the mount daemon should be started on the fly as mount requests are received.
5. Edit the file `/etc/rc.nfs` and do the following:
   a. Since we want this machine to utilize YP, uncomment the lines that define the YP domain name. Also, insert the YP domain name you want to use if the default "ibm" is not to your liking[29].
   b. Make sure that the lines starting the *nfsd* client file system request daemons and the *biod* block I/O (cacheing) daemons are uncommented. Normally, this is already done in the default file, but check it. The *nfsd* daemon only has a mission on an NFS server and will only start if an `/etc/exports` file exists.
   c. Since we'll be acting as a YP server, we need to uncomment lines so that the ***ypserv*** daemon will be started.
   d. As a YP server, we'd probably like to provide clients using the YP server for password information with the ability to change passwords on the server. To enable this, uncomment lines to start the ***yppasswdd*** daemon, which will make it possible to change passwords globally.

[29] The YP domain name should ***not*** be confused with the domain name used by TCP/IP. The YP domain name is used only by Yellow Pages, and each YP master server must have a unique YP domain name, which must be shared by all machines using that YP master server and associated YP slave servers.

e. Finally, because we want to be able to use ourselves as a YP server, we uncomment lines so that the ***ypbind*** daemon will get started.

6. Create the file /etc/exports and insert the lines required for exporting the file systems that clients should be able to mount from. In our example, the file might have the following lines:

```
/u                          # export to the world
/usr    chris               # export to chris only
/tmp    chris sys1          # export to two hosts
```

Now, before we actually start Network File System and Yellow Pages, there's a few things we need to talk about. They are related to the way the *yppasswdd* daemon works and to how we want our system to be used.

YP Password Service

Whenever a user logs into a client host with the *ypbind* daemon active, the standard authentication process is interrupted. The interruption looks for a so-called ***YP escape sequence*** in the files /etc/passwd and /etc/group. If the YP escape sequence is not found, the login processing continues immediately, using the local files and the local /etc/security/passwd and /etc/security/group files.

If the YP escape sequence ***is*** found, a request is sent to the YP server. If the YP server is active, the request is answered with information from the password and group information in the YP maps. If the server is ***not*** active, the login process hangs until the YP server becomes active. The latter is rather unfortunate, but nevertheless the way it's supposed to work according to the NFS definition.

Why would we want to use this service, then? The answer is simple: If we don't, somebody (you?) would have the responsibility to ensure that at all times, every NFS client and server use exact identical user- and group definitions in the local password and group files. It's much easier to keep things updated if there's only one place to update. That place is in the Yellow Pages maps.

Here's what you would want to do:

1. To start things off, telnet into every system that'll be using your NFS server and see what users are defined. Make a list of these users.
2. Add all these users (and groups, if required) to the YP server (which in our case happens to also be the NFS server), using the *adduser* command.
3. When asked whether you want to create directories for the users you add, do add them.

Having done this, you only need to build the YP maps and start NFS and YP. Next, let's look at a simple way to build the maps.

Building the YP Maps

Yellow Pages provides several shell scripts to assist you. The one that we shall discuss is *ypinit*. When you run this script with the option -m, you build the YP master server maps from the files that are already available on the host. Basically, *ypinit -m* will take its input from the local password and group files plus the configuration files related to TCP/IP.

We already discussed the password and group files. The most important of the remaining files is the /etc/hosts file on the server. If you put the names (full domain names as well as short names) and Internet addresses of ***all hosts you want to access*** into this file before you execute *ypinit -m*, your YP server will provide you with Internet nameserver functions ***in addition to*** any nameserver service you may already have available.

This is because of the way YP changes the **gethostbyname** and **gethostbyaddr** system calls: Whenever a TCP/IP request is processed and the exact Internet address and/or domain name of the remote host is unknown or cannot be relied upon, TCP/IP issues one of the above system calls. In systems without YP services, the request is resolved from the local /etc/hosts file if a nameserver is not active, otherwise from the nameserver.

With YP service active (the *ypbind* daemon running), the /etc/hosts file is never used for name- and reverse name resolution. If a domain nameserver is in use, it's asked to resolve the request first. If it cannot, or if no domain nameserver is in use, *ypbind* grabs the request and ask the YP server for its YP domain to resolve the request. So, on client machines that do not use a domain nameserver, YP does it all.

You may want to edit other files before building the YP master server maps. Check the publication *Managing the AIX Operating System* for a list of files. For a first go, however, it should be sufficient to edit /etc/hosts. Let's go through it:

1. Edit the file /etc/hosts to include all hosts you want to talk to. The following is an example of the file on our ***m135*** host:

```
127.0.0.1       # loopback
9.3.1.1         sys1.itsc.austin.ibm.com  sys1
9.3.1.4         chris.itsc.austin.ibm.com  chris
9.3.1.7         m135.itsc.austin.ibm.com  m135
129.35.17.2     bcroom.austin.ibm.com  bcroom
```

   Note the last line defining a host outside our local domain. It could not easily be resolved by a domain nameserver, but with YP it's easy.

2. Before you actually build the maps, issue the *domainname* command to define your local YP domain name. Type:

```
domainname your.domain.name
```

   Do use the domain name you put into /etc/rc.nfs or you will get in trouble.

3. Build the YP master server data base with the command:

```
/etc/yp/ypinit -m
```

   You will be prompted for YP slave servers. For now, ignore the prompt by pressing Ctrl-D. We shall explain later how to define YP slave servers.

Start the Server

Finally, we're ready to go. If you do not have TCP/IP started, start it by typing:

```
sh /etc/rc.tcpip
```

then start NFS and YP by typing:

```
sh /etc/rc.nfs
```

NFS and YP are now running!

Customizing an NFS and YP Client

Your NFS- and YP servers are running. Now you need to customize the clients so they can use the services. Here's what you'd want to do:

1. Log in as superuser.
2. Edit the file /etc/rc.include to uncomment the lines that start /etc/rc.nfs.
3. Check the /etc/rc.tcpip file to see that the *portmap* and *inetd* daemons are started.
4. Edit the /etc/inetd.conf file and make sure the *mountd* daemon is uncommented. You may never use it[30] if this host will only run as a client, but it doesn't hurt.
5. Edit the /etc/rc.nfs file and do the following:
   a. Since we want this machine to utilize YP, uncomment the lines that define the YP domain name. Also, insert the YP domain name you want to use if the default "ibm" is not to your liking[29].
   b. Make sure that the lines starting the *nfsd biod* daemons are uncommented. Again, uncommenting *nfsd* on clien-only hosts doesn't hurt, since the daemon will only start if /etc/exports exists.
   c. Uncomment lines so that the *ypbind* daemon will get started.

YP Client Password Service

As discussed in "YP Password Service" on page 253, you need to edit your local /etc/passwd and /etc/group files to insert the YP escape sequence. You should remove all locally defined users and groups and then insert the YP escape sequence. Of course, if you decided not to use the password services of YP, don't do this!

Here's the steps and a few examples:

1. Edit the /etc/passwd file so it looks like this:

   ```
   root:!:0:0:Sysop:/:bin/sh
   su:!:0:0::/:
   daemon:!:1:1::/etc
   bin:!:2:2::/bin
   sys:!:3:3::/usr/sys
   adm:!:4:4::/usr/adm
   adduser:!:0:0::/usr/adm:/etc/adduser
   +::0:0::
   ```

   You should only remove lines defining users with numeric user IDs of 200 and above.

2. Edit the /etc/group file so it looks like this:

   ```
   system:!:0:root,su
   printq:!:9:root
   +:
   ```

   Your file may have more lines; you just should make sure that any groups defined with the *adduser* command are removed.

[30] Actually, the *showmount* command for the local host uses *mountd* as well. Again, there's no reason to use *showmount* if your host will not act as server, but if you do, at least you'll get a polite answer.

3. Rename the *adduser*, *users* and *passwd* commands or remove them so local users can no longer be added on the client host and no attempts are made to change passwords locally. You'd probably want to keep the *passwd* command under a different name so you can change the root password as required.

Start the Client

That's all. You can now start NFS and YP on the client. You need to have TCP/IP running, so if you don't, type:

```
sh /etc/rc.tcpip
```

Now, start NFS and YP by typing:

```
sh /etc/rc.nfs
```

Of course, now we have to test things. Here's a few ways to ensure that everything is working properly:

1. Execute the following command to check that the YP server is running:

   ```
   ypcat passwd
   ```

   The YP server should respond by sending you a list of the password map in the YP data base.

2. Try to mount a directory. On the server, we have exported the /u file system. Let's mount it over the directory stub /mnt and then check if the directory has actually been mounted:

   ```
   mount -n m135 -v nfs -o soft /u /mnt
   mount
   ```

   The first of the above lines mounts, the second lists all currently mounted directories. You should see a listing like the following:

   ```
   node     mounted      mount over      vfs   date           options
   -------  -----------  --------------  ---   -------------  -------
   -        /dev/hd0     /               aix   18 May 23.48   rw
   -        /dev/hd6     /vrm            aix   18 May 23.48   r0
   -        /dev/hd1     /u              aix   18 May 23.49   rw
   -        /dev/hd2     /usr            aix   18 May 23.48   rw
   -        /dev/hd3     /tmp            aix   18 May 23.48   rw
   -        /dev/hd7     /misc           aix   18 May 23.48   rw
   m135     /u           /mnt            nfs   19 May 02.13   rw,soft
   ```

   The last line tells you that the mount succeeded. Check it by listing the contents of the /mnt directory with the *ls* command.

3. Log out and then log in with one of the user IDs you defined on the server. You should be able to log in with the password you defined when you added the user.

4. Change the password on the YP server by using the *yppasswd* command. You will be prompted for the password just as with the ordinary *passwd* command. Login with this user ID and password is possible from all clients using the host ***m135*** as YP server.

Single System Image with NFS and YP

Before we leave the client, let's discuss mounting design. Basically, if there's one thing you don't want your end users to do, then it's using the mount command with the -n option. It's much easier for them (and for you) if you pre-define the mounts and maybe even do the mounting as the system starts. You can do this by adding stanzas to the /etc/filesystems file on the clients.

Remember the files we exported from the server? They were:

```
/u                       # export to the world
/usr    chris            # export to chris only
/tmp    chris sys1       # export to two hosts
```

In a Single System Image environment, every user should be able to log into any host and get the same service. One very important part is the availability of the user's home directory, no matter where he or she logs in. With Network File System, we can meet this requirement by always mounting the server's /u directory over the local /u directories on the clients.

We need to do two things for this to happen automatically during the start up of the client machines:

1. Edit the file /etc/filesystems and insert a definition like the following:

   ```
   /u:
           dev = /u
           mount = false
           vfs = nfs
           nodename = m135
           options = rw,hard,intr[31]
           type = nfsssi
   ```

2. Add the following line at the end of the file /etc/rc.include[31]:

   ```
   mount -t nfsssi
   ```

   This will cause the remote directory to be mounted over the local one before any user can log into the client.

Actually, the above is just one step towards a Single System Image, but you got the idea. You can add as many mount requests to /etc/filesystems as you want but before you mount every file system from the server, ***think again!*** You should ***not*** mount /etc or you won't be able to continue. Remember: This directory holds the configuration files for almost every customizable part of your host, including TCP/IP and Network File System. The files need to be different on each host for things to work.

[31] The choice of mount options is not obvious. If you specify hard, you should always specify intr as well. However, the hard option may cause the client to hang during processing of the mounts if the server is unavailable. Since the mounts are done from one of the /etc/rc... files, you will end up in maintenance mode if you interrupt the mount.

On the other hand, specifying soft returns an error if the server doesn't respond. The users may not catch that error or don't care, and will start working without required mounts.

The choice is yours, but we suggest you add the actual mount command to /etc/rc.include rather than to /etc/rc.nfs (which is what the publication *Managing the AIX Operating System* says). Doing soft mounts from /etc/rc.nfs has proved unreliable. We never had any problems doing them from /etc/rc.include.

Customizing a YP Slave Server

When your network grows, load on the YP master server can become excessive. You'd then want to split the load between more hosts. One way to do this is by separating the NFS server function and the YP master server functions so they are not both handled by the same host. You should be able to figure out how to do that.

With more load you'd probably want to split the YP server task between more YP servers. Actually, even in rather small networks you may want to have YP slave servers because they will keep your systems operational even when the YP master server is inoperational. To understand this, let's look into the secret life of Yellow Pages.

YP binding

The *ypbind* daemon is responsible for routing requests from a YP client to a YP server. When started, *ypbind* sends out a broadcast to ask for an available YP server for the particular YP domain the client belongs to. If one or more YP servers are operational for that YP domain somewhere on the network, they respond to the broadcast. An algorithm is built into the YP servers, so that they do not all respond at the same time. The idea of this is to balance the load between servers.

When the *ypbind* receives a response from YP servers, it ***binds*** itself to the one it likes best; usually the first one that responds. Hence, if a YP domain has multiple YP servers, the clients will be bound to different servers and the load balanced. At any time, a client can be bound to a particular server with the ***ypset*** command, should you desire this.

But what happens if no server is operational when a user comes in at 4 am and starts his client machine? Quite simple: nothing happens. The *ypbind* daemon doesn't find a server, so it cannot bind to one. The client host starts just fine, and here it sits with the login prompt displayed. Now, the user tries to log into his system. Surprise, surprise! He cannot log in because his local /etc/passwd file points to the *ypbind* daemon, and the daemon hasn't found a YP server.

The user can type in his user ID and is prompted for a password. Then *ypbind* realizes that it has not been bound to a YP server so it gives it another try. It sends broadcasts asking for available servers and retries this with regular intervals until a server responds.[32] So what can we learn from this? Well, maybe you shouldn't get to work at 4 am if your YP servers are not running. More seriously though, with only one YP server, this could happen at any time when that server has a scheduled or unscheduled outage.

So, keep at least one YP slave server around. It's better than having users call you at home at 4 in the morning.

[32] Yes, we know that the *Managing the AIX Operating System* publication says you can log in using user IDs that appear in the /etc/passwd file prior to the YP escape sequence but, unfortunately, this ain't true.

Converting a YP Client to a YP Slave Server

The easy way to make a YP slave server is to convert a YP client. Here's what you need to do:

1. Log in as superuser.
2. Edit the file /etc/rc.nfs and uncomment lines so that the ***ypserv*** daemon will be started.
3. Start the *ypserv* daemon from the command line, typing:

   ```
   /usr/etc/ypserv
   ```

4. Modify your system so that the YP slave server is authorized to copy files from the YP master server. Authorization follows the rules used for the ***rcp*** command. See "Remote Login and Remote Command Execution" on page 195 and the publication *IBM RT Interface Program for use with TCP/IP* for details of this.
5. Issue the following two commands to transfer the YP maps from the YP master server:

   ```
   cd /etc/yp
   ypinit -s m135
   ```

   Of course, ***m135*** should be replaced by the name of your own YP master server.

Congratulations. You now have a YP master server and a YP slave server. But how do you keep the YP slave server updated? There are two things you need to do:

1. Tell the YP master server about the slave.
2. Make the YP slave server request updates periodically.

Register a YP Slave Server

One of the YP maps contains information about YP servers for a domain. You must update that map to include the name of your new YP slave server. In the section, "Adding a New YP Server" in *Managing the AIX Operating System*, you see one method described. Use that method if you like it, or take the easy way and do a complete rebuild of all the YP maps on the YP master server. Then use *yppush* to propagate the maps to all YP slave servers. Here's how:

1. Log in as superuser on the YP master server.
2. Build the data bases, as when you first did so:

   ```
   /etc/yp/ypinit -m
   ```

3. When prompted for YP slave servers, enter the name of all YP slave servers for the YP domain, one at a time, each followed by ENTER. When all are entered, press Ctrl-D.
4. After the YP maps have been built, use *yppush* to propagate the maps to the YP slave servers.

Requesting YP Updates from Master

If a YP slave server is inoperational when you run *yppush* on the YP master server (either from the command line or implicitly because of an automatic update of the YP maps), the new YP maps are not propagated to that YP slave server. You'd want every YP slave server to cope with this by requesting updates periodically.

The section, "Keeping Maps on YP Slave Servers Current" in *Managing the AIX Operating System*, tells you how to build ***crontab*** entries and use the *ypxfr* command for the transfer of maps periodically.

You may also, as yet another precaution, want to put the *yppush* command into one of the start-up files, say, `/etc/rc.include`, on the YP master server. Then, all maps get transferred to all YP slave servers whenever the YP master server starts.

To entirely avoid the use of *crontab* you could use *yppush* from a start-up file on the YP master server and run *rexec* from a start-up file on all YP slave servers. The *rexec* command would then execute *yppush* on the YP master server to propagate all maps. Of course, all prerequisites for running *rexec* must be in place as described in "Remote Login and Remote Command Execution" on page 195.

Using NFS and YP

If you think NFS and YP will take care of themselves, guess again! There's always something to do for you as the administrator of the system: Your network will inevitably have new hosts added; new users will have to be registered, and existing users get new jobs and must be removed from the system. Your file server grows out of space, and new file systems must be added and exported from the file server. Similarly, users may have requirements that you did not foresee, so they may have to mount directories directly.

Maintaining the YP servers.

Whenever you need to make changes to the YP maps, the easy way is to edit the files that were used to build the YP maps on the YP master server and then rebuild the maps. For example, to add a new user, use the following series of commands on the YP master server:

```
adduser
cd /etc/yp
make
```

Maintaining the NFS Servers

Network File System servers are easy. Basically, all you need is to keep the file `/etc/exports` up-to-date. When new file systems are added, add them to `/etc/exports` if they should be available to clients. When new client hosts are added to the network, update the `/etc/exports` file to allow these clients to access the file systems on the NFS server.

Mounting

Occasionally, users may have to mount directories they do not usually have access to. If those directories are parts of exported file systems (or entire, exported file systems), mounting can proceed right away.

A default AIX/RT system assumes remote mounts to be done via Distributed Services. If you are not using DS, or want your systems to use NFS as the default remote mount method, change the file /etc/vfs by uncommenting the line saying:

```
%defaultvfs aix nfs
```

If you don't do this, you must use the option -v nfs whenever you use the *mount*command to mount a remote directory over NFS.

The syntax of the *mount* command when used for remote mounts is:

```
mount -n server [-v vfs] [[[-o option],option],...] remote_dir local_dir
```

where "vfs" must be replaced by "nfs" for an NFS mount, and where "option" may be one of the following:

retry=n	Set the number of times to retry the mount.
rsize=n	Set the read buffer size to ***n*** bytes.
wsize=n	Set the read write size to ***n*** bytes.
timeo=n	Set the timeout value to ***n*** tenths of a second.
retrans=n	Set the number of NFS retransmissions to ***n***.
soft	Return an error if the NFS server doesn't respond.
hard	Continue to try NFS requests until the NFS server responds.
intr	Allow keyboard interrupts on hard mounts.
ro	Mount directory as read-only.
rw	Mount directory in read/write mode.

In our sample configuration, the host **sys1** may want to mount the directory /usr/lpp/X11 over a local directory called /X11. The mount command to do this could be:

```
mount -n m135 -v nfs -o hard,intr,rw /usr/lpp/X11 /X11
```

When the mount is no longer required, it can be unmounted with the *unmount* command, as follows:

```
unmount /X11
```

To remove all mounts from a particular NFS server, you can use:

```
unmount -n m135
```

The "on" Command

Network File System provides the *on* command that allows you to execute commands on other machines with NFS installed. The following applies to the use of the command:

1. Remote host must have same architecture as local host.
2. All environment variables are passed.

3. Current working directory is preserved.
4. Uses /etc/exports on local host.

Especially the last of the above points restricts the use of the command to environments where security is not an issue. The working directory must be in an exported file system and so must other, unspecified files. In practice, the use of the *on* command is only generally useable if you have all file systems exported on the local host.

If you don't have everything exported already, it's probably because you want to protect your local file systems from unauthorized access. Exporting everything just to be able to use *on* might just not be what you want. You'd rather use one of the TCP/IP commands for remote execution.

In installations where everybody can access any host, the *on* command may turn out to be very useful.

NFS and YP Hints and Pitfalls

Root Authority

Our small network (Figure 112 on page 252) may have one administrator per system, so that three different users play the role of the superuser, one for each host. When the superuser on host ***sys1*** works with directories mounted from the NFS server, then he'd presumably be able to exercise full superuser priviledges on any file, since he has a numeric user ID of zero.

No, he is not! NFS maps client root IDs to the kernel variable ***nobody*** on the NFS server. The value of the variable maps to a user ID which, by default, has no priviledges at all. The mapping is done to prevent unauthorized access to server files, of course, but if you don't like the way it works, follow the guidelines in the section, "Removing Superuser Privileges on Mounted File Systems" in *Managing the AIX Operating System*, to allow client root IDs whichever permission you like. The change can be made temporarily or permanently.

Changing Passwords

Users should change their passwords regularly. They should do so using the *yppasswd* command when they work in an environment with YP password servers active. This goes whether the user is logged into a YP server or a YP client.

Forgotten Passwords

Miss Otis regrets that her three-week vacation on Canary Islands erased every bit of her password from her memory. Now she cannot log into the YP client machine she usually works on. The administrator of her local system can not help since you removed the *adduser* command from the system. Even if he does find the copy you saved under another name, he has been told (by you) that he should never use the command.

Miss Otis will have to ask ***you*** to update the YP map so she can get started after the vacation. You can use the same series of commands as when adding a new user:

```
adduser
cd /etc/yp
make
```

You should assign her a temporary password and tell her to use the *yppasswd* command to change it from her client machine.

Reverse Name Resolution

We have already discussed Internet name- and reverse name resolution in "Building the YP Maps" on page 253. At this point, let's emphasize that while YP nameserver functions can come in very handy, they can be quite a nuissance when there are problems with the YP server.

We discussed how the absence of a YP server would block an attempt to log in from a client machine. Basically, the same kind of thing could happen if, for some reason, the YP server does not respond to an Internet name- or reverse name resolution request.

Assume you are working on host ***chris*** and want to telnet into host ***sys1***. If you are not using a domain nameserver, *ypinit* tries to resolve the hostname to an Internet address by asking the YP server. However, if the YP server doesn't respond, what happens? The *telnet* command hangs and does not terminate until the YP server responds.

The request is retried by *ypbind* and messages will be displayed on the console of host ***chris*** saying that the YP server does not respond. What to do about it? Well, terminate the *telnet* command by pressing `Ctrl-Break` and try again.

What, then, if the name resolution request was issued from a daemon? Well, there's not much you can do really. Except avoiding to work from /dev/console where YP messages will be displayed with regular intervals - and fixing the YP server, of course.

Security Issues

If you use Network File System and Yellow Pages, wave good-bye to C2 Security. There's no way the present implementation of the products will run on systems configured for C2 Security. Other than that, NFS and YP provide a reasonable level of security if you:

- Restrict the number of people who can manipulate the critical files (maps, passwords, etc.) on the NFS and YP servers to a minimum.
- Use YP password services.
- Remove the commands *adduser*, *users* and *passwd* from client machines.
- Restrict file system exports to only cover legimate requirements, rather than exporting everything to the world.

AIX/RT NFS Connections to Non-IBM Systems

The AIX/RT implementation of Network File System is well tested and stable. Within a few months after its availability, it has been shown to cooperate with NFS implementations of almost every major vendor of UNIX based systems and several others. Among the vendors whose systems AIX/RT NFS has been tested successfully with are:

- Altos
- Apollo
- Apple

- AT&T
- Data General
- DEC ULTRIX
- DEC VMS (only supports NFS server)
- Hewlett-Packard
- NCR
- NEC
- NeXt
- Sony
- SUN.

Network File System has indeed become a "de facto" industry standard when files can be shared and inter process communication can take place between machines from this many vendors. Actually, the list is much longer. Also, in several cases, tests were carried out with more than one machine type and more than one operating system of a vendor.

All vendors support Ethernet for TCP/IP, and this was the environment tested. Between IBM systems and between SUN and IBM systems, Token-Ring connections have also been tested.

NFS and YP on AIX/370

At the time of writing, neither a final version of the Network File System and Yellow Pages implementations on AIX/370 nor final related publications have been available. The information in this section, therefore, ***may be inaccurate***. Nevertheless, it may be useful for planning purposes. We shall discuss:

1. NFS in a Cluster
2. File- and record locking.

NFS in a Cluster

NFS for AIX/370 is expected to be (but isn't necessarily) distributed by IBM under a cluster-wide licensing agreement. All sites (hosts) in an AIX cluster will normally be running AIX with the TCF option. Thus, if NFS is installed, all sites will normally run NFS.

The basis for the cluster is ***Transparent Computing Facility*** (TCF). TCF allows file systems to be replicated on more than one site in the cluster. Replication can be for the entire file system or only part of it. When file systems are replicated, one copy is designated the primary copy. Secondary copies are automatically kept in sync with the primary copy. All copies of a particular file system have the same ***group file system number*** (GFS). But each copy of a filesystem is uniquely identified with a reference to the physical- or logical disk by a pack number. The key point here is the GFS, which is the principal (and only) way of identifying any file system in an AIX cluster. This is true also for file systems mounted with NFS.

Cluster as NFS Client

When a host in a cluster mounts a remote directory over a directory in the cluster-wide file tree (more precisely: in the replicated root system), that mount becomes available to every host in the cluster. If the mounted-over directory already contains files, the mounted directory replaces the cluster directory. Whether the mounted directory is indeed a file system on a remote host, or it is only part of one, the AIX cluster handles it as a mounted file system.

However, the mounted file system does not have a GFS, so for the cluster sites to be able to use a mounted file system, a GFS has to be assigned to it. As we shall soon see, the assignment of a GFS can be done permanently or on the fly as the file system is mounted.

When all sites in the AIX cluster have access to a mounted file system they may all attempt to apply updates simultaneously. For that reason, each mounted file system is assigned a single site as its ***current synchronization site*** (CSS). All sites can perform basic (read/write) operations on the file system but any operation that changes any of the "statistics information" on the mounted file system must be handled through the CSS site.

If a remote mount is done by a single site in the AIX cluster, all hosts in the cluster can access that file system only as long as the mounting site is operational and does not unmount. All sites in a cluster that need access to a mounted file system should mount that file system. They should not depend on a single site to do the mount.

How, then, do we assign a GFS to the mounted file system? If each site assigned its own GFS, the file system would be known by a different GFS on each site, and there would be no way to ensure that a proper CSS is assigned. Each mounting host would act as CSS for the same remote file system and synchronization would be an illusion. To overcome this problem, cluster-wide NFS mounts must be done in one of two ways:

Administered mounts
: Potential mounts are pre-assigned a GFS by the system administrator. This allows all sites to mount the file system and access it using the same GFS, and a single CSS can be picked. Administered mounts can only happen at one place in the file tree.

Nonadministered mounts
: A GFS is assigned dynamically as the mount happens. Because of the synchronization issue, only one site is allowed to do the mount[33] at a time.

One interesting observation during our tests was that even though the NFS *mountd* daemon was only running on the IBM 9370, all sites could use the mounts we did. In a cluster that depends on a single IBM System/370 host, multiple mounts may not be necessary. If other sites can not continue when one single site is unavailable, all NFS mounts might as well be done from that site. In that case, you wouldn't need to have any NFS daemons running on any other site in the cluster.

[33] Unfortunately, the AIX cluster can only detect multiple mounts if they are done over the same directory. Two sites can mount the same remote directory over two different, cluster-wide directories and fool the system. Even from one single site, the mount can be done twice this way. This should certainly be avoided.

Cluster as NFS Server

Hosts outside the cluster see the AIX cluster as a single host. That (cluster) host (as any other host) has multiple file systems, each identified by a GSF. For remote hosts to mount a file system from the cluster, that file system must be exported by the cluster. This is done the usual way, by listing the exported file systems, along with any restrictions as to who can mount them, in the file `/etc/exports`.

During our tests we were able to access all file systems on a cluster site by exporting and mounting the root file system. This worked like a charm and is identical to the way NFS works on AIX/RT. However, the file systems on other sites in the cluster are also mounted as parts of the root file system. Could we access those?

Yes, we could. But directory listings with *li* and *ls* and the display of current directory with *pwd* produced wrong output. We could list and edit a file if we specified the full path- and file name.

To our surprise, we could even access remotely mounted directories from an NFS client that mounted the root file system of the AIX/370 system. We mounted a directory from our (IBM RT) host ***m135*** onto `/aix370/tmp` on the AIX/370 system. `/aix370` happens to be a separate file system on the IBM 9370. It is mounted onto the replicated root file system of the host. We then mounted the entire root file system of the AIX cluster over `/mnt` on host ***m135***. From the IBM RT system, we could then access our local directory from `/mnt/aix370/tmp`.

In effect, we had succeeded doing an inherited mount. But, beware! This may not be the way it's supposed to work. We did use pre-release code.

File- and Record Locking

Since NFS for AIX/370 is at NFS level 3.2, record- and file locking depend on the two daemons *lockd* and *statd*. For locking to take place, both the NFS server and the NFS client must be running both daemons. The *lockd* daemon grants or denies locks based on tables maintained on the server. The *statd* daemons on the two hosts periodically exchange information about the status of the host, and in this way maintain ***state*** information in the stateless NFS environment.

The exact implementation on NFS for AIX/370 was not known to us at the time of writing. However, there's one significant difference between SUN Microsystem's implementation of the NFS locking mechanism and the implementation on AIX/370. In SUN's implementation all locks are maintained by the *lockd* daemon. On AIX/370, locks on local files are maintained in the kernel; locks on remote files are maintained by the daemon. The *lockd* daemon will check with the kernel before granting a lock. If the kernel is holding a lock, the remote locking client will retry periodically until the kernel grants the lock.

Reference Publications for This Chapter

Network File System and Yellow Pages are described in:

Managing the AIX Operating System, SC23-2008
AIX Operating System Commands Reference, SBOF-1814
Introduction to AIX/370, ZZ81-0203

AIX Mail

In today's business environment, using the computers and computer networks of an enterprise for electronic mail is as natural as using the telephone. When the mail functions of a computer system allows mail to be received from and sent to computer users in other enterprises, the advantages of an electronic mail system become even more obvious.

Overview

AIX, as all UNIX based systems, include facilities for mail. These facilities can be linked with and cooperate with the electronic mail functions of other IBM systems, as well as with many non-IBM systems. Mail can be forwarded through gateways in heterogeneous networks with little or no requirement of the end user to know the details of the network or intermediate nodes.

The AIX mail functions are based upon the */etc/mail* mail program of UNIX 5.2 (known as *bellmail* in AIX) and the Berkeley 4.3 based *mail* program enhanced with *MH (Mail Handling)* package. The purpose of the mail functions are to provide facilities for creating, sending and receiving mail and for storing mail in mail boxes and folders. Mailing can take place to and from:

- Users local to one machine
- Remote users via TCP/IP
- Remote users via *uucp* (BNU).

The AIX mail functions also include facilities for managing distribution- and nickname lists, collecting and presenting statistics, configuring mail options and various other utilities.

The basic mail functions, as well as the Mail Handling Package, are available on all AIX platforms: AIX/RT, AIX PS/2 and AIX/370.

The publication *Managing the AIX Operating System* gives a complete description of how to install and customize the mail functions, while the publication *Using the AIX Operating System* describes how to use mail. Additional information about each of the mail commands is available in *AIX Operating System Commands Reference*.

More than two hundred pages of detailed information about AIX mail is thus readily available to users. Rather than trying to cover all off this information here, we shall concentrate on the following subjects:

1. Mail concepts
2. How you install a minimum mail system
3. How you define aliases and distribution lists
4. What changes you need to configuration files
5. Examples of Using Mail
6. Mail to VM.

If you want to use the more sophisticated features of the AIX mail functions, you'll have to get the above mentioned publications and you're on your own.

Mail Concepts

Provided you install Mail Handler (MH) in addition to the basic mail functions, users have two options to process mail. This section describes these options as seen from an end-user perspective.

Basic Mail System

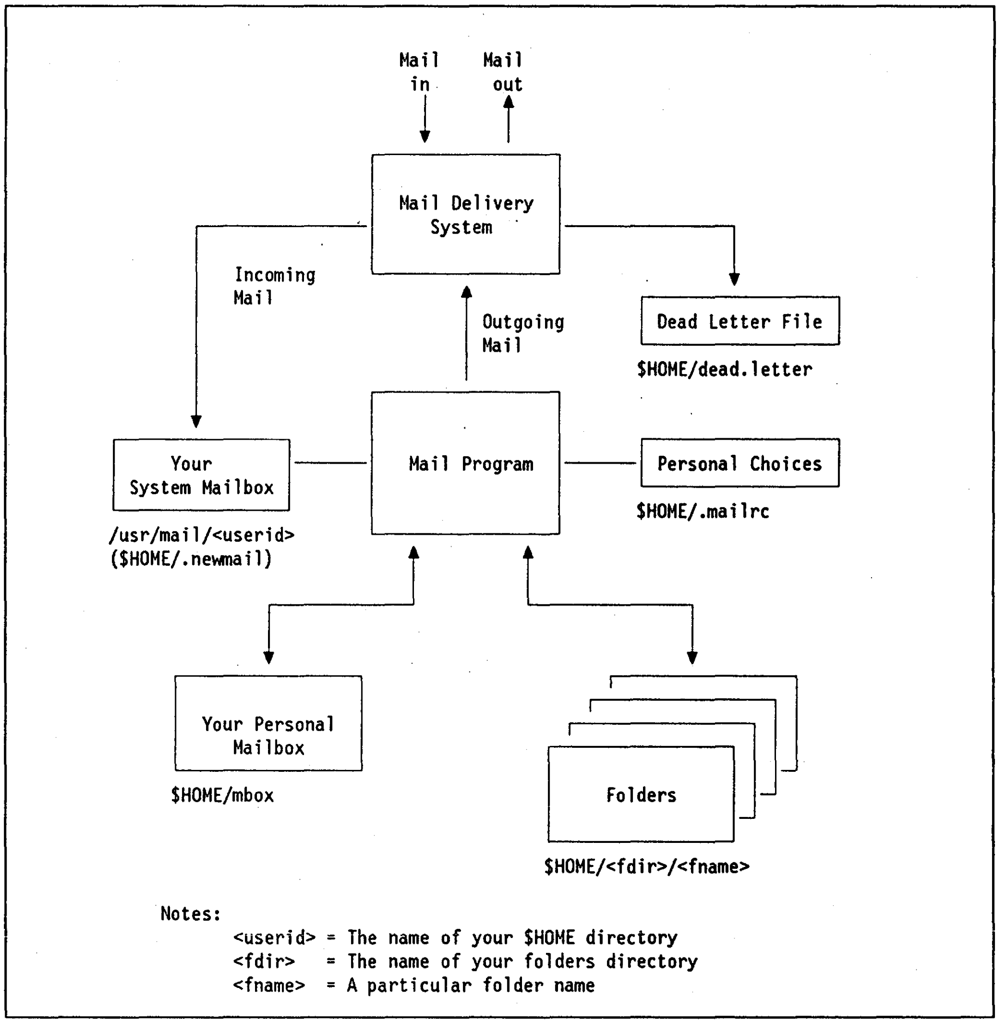


Figure 113. Basic Mail System, Overview

Figure 113 shows a simplified overview of the basic mail system. The important things to notice are the existence of two ***mail boxes*** (the *system mailbox* and the *personal mailbox*) and the *folders*. The following is a brief description of the basic mail system.

Delivery System: The delivery system is a set of programs to route outgoing mail to other users and to deliver incoming mail in the correct system mailbox. Routing of mail is done from the information about receivers provided by the originator of the mail.

The dead.letter File: When the delivery system cannot deliver a message, it places the message in the file `dead.letter` in the `$HOME` directory of the message originator. If the `$HOME/dead.letter` file does not exist, the delivery system creates the file; otherwise, it adds the message to the file. In addition, the delivery system displays a message on the sender's terminal to inform about the failure to deliver mail. If a receiving host can't deliver mail, the mail is returned to the sender.

The system also uses `$HOME/dead.letter` to save partially completed messages when you exit the *mail* editor with the *~q* command. In this case, the previous content of `$HOME/dead.letter` is replaced with that of the partially completed message.

System Mailbox: A ***system mailbox*** exists for each user or is created when the delivery system first receives mail for a user. The system mailbox is a file where messages are stored ready for the user to process, either by moving the messages to his personal mailbox (see below) or a folder or by otherwise removing it from the system mailbox.

A separate system mailbox can exist for each user ID defined in `/etc/passwd`. The mail system keeps all system mailboxes in the directory `/usr/mail` (`$HOME/.newmail` on AIX PS/2). Each system mailbox is named by the user ID associated with it.

If the user's system mailbox has mail, the user is notified when logging in or periodically as the system daemons check the system mailboxes.

Personal Mailbox: The ***personal mailbox*** is similar in concept to an in-basket in an office. You put mail in the in-basket after you have received it, but before you have filed it. The personal mailbox is a working storage place for mail that still requires action.

In the mail system, the personal mailbox is a file assigned to a particular user. The mail system creates the file with the name `$HOME/mbox` when the user receives mail from his or her system mailbox. Normally, the system deletes this file when all messages are removed from the personal mailbox, but you can set a user option to keep the file even when no messages remain.

When you use the *mail* program to view mail in your system mailbox, *mail* automatically puts all messages that you have read, but did not delete, into your personal mailbox. The messages remain in your personal mailbox until you move them to a folder or delete them.

Folders: ***Folders*** provide a way to save messages in an organized fashion. You can create as many folders as you need. Name each folder with a name that pertains to the subject matter of the messages that it contains, similar to file folders in an office. Using the *mail* program, you can put a message into a folder from:

- Your system mailbox
- Your personal mailbox
- The `dead.letter` file
- Another folder.

Like mailboxes, each folder is a text file. The mail system puts each folder in the directory that you specify in your `.mailrc` file. You must create this directory before using folders to store messages. Once the directory exists, *mail* creates the folders in that directory as needed.

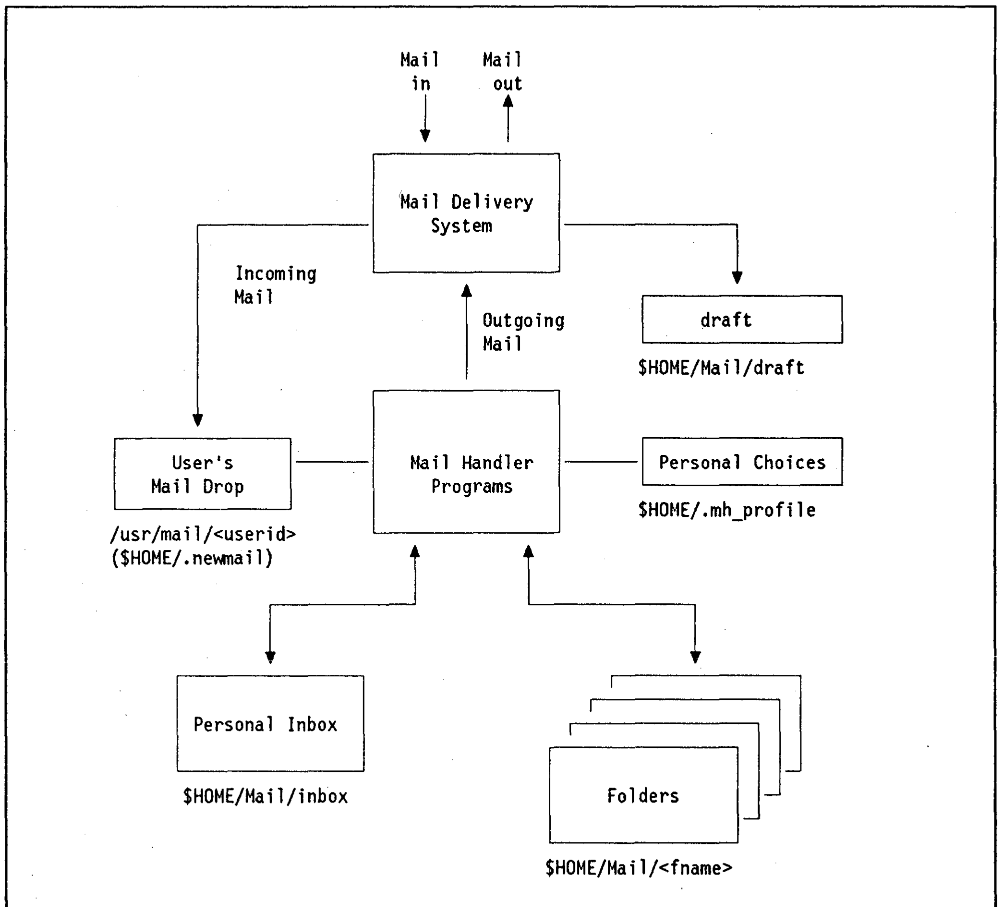
Personal Choices: The mail system allows you to modify the way it operates to suit your needs. These choices include:

- What information to include in message headings
- Whether to forward incoming mail to another user ID
- How you want the messages handled
- Other characteristics pertaining to your terminal.

Refer to *Using the AIX Operating System* for more information about specifying these, and other, personal choices.

Mail Program: The *mail* program allows you to create, send, and receive messages. It includes a line-oriented editor for creating messages and provides a command interface for processing the contents of your system mailbox, your personal mailbox, any folders you may have, and dead.letter.

Mail Handling (MH) Package



```
Notes:
       <userid> = The name of your $HOME directory
       <fdir>   = The name of your folders directory
       <fname>  = A particular folder name
```

Figure 114. Mail Handling System, Overview

Figure 114 shows a simplified overview of the Mail Handling system. It looks very much like the overview of the basic mail system, but there are a few important differences.

Draft: When the delivery system cannot deliver a message, and you quit the *comp* program, the unsent message is saved in `$HOME/Mail/draft`. Next time you compose a message you'll be asked if you want to use the saved message as a starting point or if you prefer to start with a fresh "standard" draft message. This is a very convenient way to resend a message you composed but put an invalid address on. When a message is saved in `$HOME/Mail/draft`, the delivery system displays a message on the sender's terminal to inform about the failure to deliver mail.

User's Mail Drop: You'll see that MH uses the term ***"mail drop"*** rather than "system mailbox" to refer to the place where incoming mail is stored. This is only a matter of terminology. Actually, MH uses the same place to drop mail for MH as for the basic mail system unless you explicitly change this.

A mail drop exists for each user or is created when the delivery system first receives mail for a user. The mail drop is a file where messages are stored ready for the user to process, either by moving the messages to his personal inbox (see below) or a folder, or by otherwise removing it from the mail drop.

A separate system mailbox can exist for each user ID defined in `/etc/passwd`. The mail system keeps all system mailboxes in the directory `/usr/mail` (`$HOME/.newmail` on AIX PS/2). Each system mailbox is named by the user ID associated with it.

If the user's mail drop has mail, the user is notified when logging in or periodically as the system daemons check the mail drops.

Personal Inbox: The ***personal inbox*** is similar in concept to an in-basket in an office. You put mail in the in-basket after you have received it but before you have filed it. The personal mailbox is a working storage place for mail that still requires action.

With MH, the personal mailbox is a directory associated to a particular user. Each message is assigned a number and saved in a file named with that number in the `$HOME/Mail/inbox` directory when the user receives mail from his or her mail drop.

Messages deleted with the *rmm* MH command are saved in the `$HOME/Mail/inbox` directory with the old file name preceded by a comma. You need to take special action to remove such copies of deleted messages or to leave it to a daemon (*crontab*) to clean your inbox out as required.

Folders: Mail Handling ***folders*** are directories, as are the basic mail system folders. However, rather than saving all messages in one big file (as the basic mail system does it), MH saves messages as individual files under each folder directory.

If a folder directory does not exist when you attempt to save messages in it, you are asked if you want to create it. Messages can be saved in folders as they are received and can be moved between folders, including the inbox folder.

Personal Choices: Just as the basic mail system allows each user to define a personalized set of options to use for mail, so does MH. Refer to *AIX Operating System Technical Reference, Files and Extensions* and *Using the AIX Operating System* for details.

Mail Handling Programs: The MH suite of programs offers an alternative to the basic mail functions by providing a series of more user-friendly and fairly simple commands rather than the single *mail* command of the basic mail system. The disadvantage, of course, is that the user must learn many more commands to use MH.

In "Examples of Using Mail" on page 281, you will see examples of basic mail commands, as well as MH commands, to give you an impression of the differences.

Installing a Minimum Mail System

You can customize the mail system to deliver mail to users on your local computer or users on other computers. Setting up the mail system for any of these environments requires similar procedures for each type of configuration.

This section will guide you through the installation of a minimum mail system between three machines as shown in Figure 115. For this discussion it is of no importance what the physical communications link is, only the protocol matters.

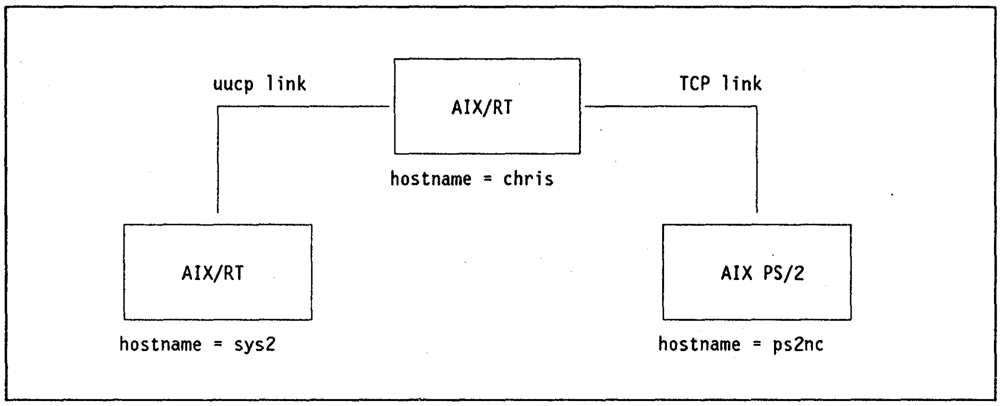


Figure 115. Mail Handling System, Overview

The steps you'll have to go through are the following. All steps apply to each of the three systems unless otherwise noted:

1. Log in as superuser.
2. Install the *sendmail* program which is an option under *Extended Services*.
3. Update the configuration file `/usr/adm/sendmail/sendmail.cf`. This is done by modifying and uncommenting a few lines in the supplied default file. It is suggested that you copy the default lines and make your modifications to the copy. The lines are identified with a two-letter code following the "#" character.

   Note that the lines are scattered throughout the configuration file, which is why we provide you with the default lines so you can locate the lines to change.

a. You will need to change the following lines:

```
#CwYourhostname
#DDgrandchild.child.parent.top
#DElowestLevel
#DFnextLevel
#DGnextLevel
#DHnextLevel
```

b. In the case of the host ***chris*** and assuming the domain name of ***itsc.austin.ibm.com***, the changed lines would look like this:

```
Cwchris
DDitsc.austin.ibm.com
DEitsc
DFaustin
DGibm
DHcom
```

c. If you are using national language characters, you'll probably want them in your mail as well. Then you'll need to modify the definition line for the ***router*** you are using. In our example we use *uucp* and TCP/IP. The default definition for the *uucp* mailer allows for national language characters, but we need to make a change for TCP/IP. Change the line looking like the first line below to look like the second line:

```
Mtcp,  P=[IPC], F=mDFMueXL,  S=14, R=24, A=IPC, $h, E=\r\n
Mtcp,  P=[IPC], F=mDFMueXLN,  S=14, R=24, A=IPC, $h, E=\r\n
```

4. Build a data base version of the configuration file:

```
/usr/lib/sendmail -bz
```

5. Build a data base version of the default alias file:

```
/usr/lib/sendmail -bi
```

6. On systems running TCP/IP, change the file /etc/rc.tcpip so the lines starting the *sendmail* daemon are uncommented. On systems not running TCP/IP, insert the following lines at the end of an appropriate start-up file, for example /etc/rc.include:

```
if [ -f /usr/lib/sendmail ]; then
   /usr/lib/sendmail 1>>/dev/null
fi
```

7. Install and customize TCP/IP. For details about the installation and customizing, see "Transmission Control Protocol/Internet Protocol (TCP/IP)" on page 165 and the reference publications.

8. Install BNU on the systems running *uucp* (BNU) if you have not done so already. According to Figure 115 on page 272, you need to install BNU on machines ***chris*** and ***sys2***. For a minimum BNU installation, you need to:

   a. Install BNU (part of *Extended Services*).

   b. Create a user ID for use by BNU, using the model ID, uucp.

   c. On AIX/RT, remove the line saying restrictions = nologin from the uucp user definition in the file /etc/security/passwd.

   d. Assign a password to the user uucp. In our example we use the password mailtest.

e. Modify the file /usr/adm/uucp/Systems to include a line for each system you want to talk to. If BNU runs across TCP/IP, the line for the ***chris*** host would be:

```
sys2 Any TCP - - in:--in: uucp word: mailtest
```

f. Modify the file /usr/adm/uucp/Devices to include a line for the connection you have. If BNU runs across TCP/IP, the line would be:

```
TCP - - - TCP
```

g. Modify the file /usr/adm/uucp/Permissions to include the following two lines (the example is from the ***chris*** host). Each machine must list all hosts permitted to connect into the local machine:

```
LOGNAME=nuucp
MACHINE=sys2
```

h. Establish the physical connection between the two systems. If TCP/IP is used for BNU, all you need is to make sure the *uucpd* daemon is started from /etc/rc.tcpip.

9. If using AIX/RT, edit the file /usr/spool/cron/crontabs/root and uncomment the line that starts the shell script /usr/adm/sendmail/smdemon.cleanu. If required, you can change the line to run the cleanup at other times than defined in the default entry.
10. If using AIX PS/2, edit the file /usr/spool/cron/crontabs/uucp and uncomment the lines that run the various cleanup scripts. If required, you can change the lines to run the cleanup at other times than defined in the default entry.
11. Shut the system down and reboot or, if TCP/IP and/or BNU is already installed and configured, run the following command to start the mail daemon:

```
/usr/lib/sendmail 1
```

Your mail system is now ready for simple mail operations where you know the full addresses of the recipients of messages.

Local and uucp Mail Delivery

Mail Delivery without TCP/IP Installed: Mail is delivered automatically to users on a local system without running the ***sendmail*** daemon. However, since mail can still enter the sendmail queue, you should process the queue periodically to ensure that the mail is delivered. You can also process the mail queue by entering the following command on the command line:

```
/usr/lib/sendmail -q
```

If your system is connected to others over *uucp* links, the mail system uses *uux -r* to place the mail for those systems in the *uucp* spooling directory. After that, *uucp* controls delivery of the mail to the remote systems.

Defining uucp Routing Information: To deliver mail to another system connected to the local system with a *uucp* link, you must define the addressing and routing information required for that link. The *sendmail* program transfers mail to the *uucp* daemons (through the *uux* command) for delivery across a *uucp* link. The *uusched* program handles the queued routing of mail for *uucp* links.

Sending Mail via uucp: To send a message to user ***dieter*** on host **sys2** from host ***chris***, start the mail prompter with:

```
mail sys2!dieter
```

The above format is the *uucp* format where the host name precedes the user ID, and the two parts are separated by a '!'.

Addressing When the uucp Link is on Another Computer: In a local area or wide area network environment, one of the systems on the network may have a *uucp* connection to a remote *uucp* system. You can use that *uucp* connection to send a message to a user on that remote *uucp* system. Assuming the gateway to the remote *uucp* host is called ***uugate***, use the following command format to send a message:

```
mail @sys1:@uugate.UUCP:user_ID@uuremote
```

This format sends mail first to ***sys1***, then to ***uugate*** which routes it on a *uucp* link to ***uuremote***. The ***.UUCP*** addition to the address for ***uugate*** indicates that the *uucp* mailer at that system should handle the routing of the message to ***uuremote***. Notice that in this format, you are not sending mail to a user at any of the intermediate systems, so no user ID precedes the "@" in the domain address.

Using TCP/IP for Mail

Defining Local Area Network Information Using TCP/IP: To deliver mail in a local area network (where senders and receivers may be on different systems connected by local area network links), the network must be using TCP/IP as one available network protocol. For networks with many systems, you may want to install one or more systems on the network as the nameserver for the network. The nameserver controls the addressing information for the network and for connections to systems beyond the immediate network. Using a nameserver simplifies the task of keeping the address information up-to-date.

Sending Mail via TCP/IP: To send a message to user ***jan*** on host ***ps2nc*** from host ***chris***, start the mail prompter with:

```
mail jan@ps2nc
```

Note the address format for TCP/IP links; the user ID precedes the host name and the two are separated by a "@".

Defining Aliases and Distribution Lists

The mail functions use a hierarchy of alias definitions to transform aliases or distribution lists into addresses of the form required by the mail delivery system. One alias file is shared by the basic mail system and MH: The /usr/adm/sendmail/aliases file, which also holds three aliases required by the mail functions. Please note, that for changes to this file take effect, you must execute ***/usr/lib/sendmail -bi***.

Mailer Alias File

We use the term "mailer" to describe this file because it's shared by all users on a system and has the property that it is used to route incoming mail in addition to being used to find addresses for outgoing mail. This facility is very nice, indeed, because it allows mail to be rerouted without interference from users.

The side effect of this is that mail may be returned or routed on to wrong destinations if you have dubious definitions in the mailer alias file as we shall soon demonstrate. For a first example, using the configuration in Figure 115 on page 272, let's define aliases to make sure all mail to the user ID ***fribert*** is routed to the host ***chris*** where the user (who happens to be a person named ***Johnny***) normally logs on. You need the following definitions in the `/usr/adm/sendmail/aliases` files of the two other hosts:

Host name	Line in mailer alias file
sys2	johnny: chris!fribert
ps2nc	johnny: fribert@chris

Now, whenever a user on hosts ***sys2*** or ***ps2nc*** sends mail to the alias ***johnny***, that mail will be delivered to Johnny's ID on host ***chris***.

Alias Post Office

Now, lets introduce the host ***m135***. This host acts as a ***post office*** in that it holds information about the places where each user in the network wants his mail delivered. Every other host in the network could then have mailer alias files pointing to the "post office" host, which would forward mail to the users' preferred system. Consider the following example:

Host name	Line in mailer alias file
sys2	johnny: m135!johnny
ps2nc	johnny: johnny@m135
chris	johnny: johnny@m135
m135	johnny: fribert@chris

This works just fine. If, say, host ***ps2nc*** sends mail to the alias ***johnny***, then this mail gets sent to ***m135*** which will look up the recipient in its mailer alias file. It finds an entry and follows the directions to forward the mail to user ***fribert*** on host ***chris***.

Of course, the host where Johnny wants his mail delivered does not need to have mail to Johnny directed via the "post office" host, but by making the mailer files identical on all hosts except the "post office" machine, maintenance becomes much simpler. The mailer alias file could easily be shared between all systems in a DS or NFS environment.

Alias Loops

Now let's look at what will happen if we make the mistake of defining the files so the "post office" host forwards mail to the alias name rather than the user ID on the host ***chris***:

Host name	Line in mailer alias file
sys2	johnny: m135!johnny
ps2nc	johnny: johnny@m135
chris	johnny: johnny@m135
m135	johnny: johnny@chris

Sending mail from ***ps2nc*** to alias ***johnny*** will go to ***m135***. The "post office" host finds the alias and forwards the message to the name ***johnny*** at ***chris***. This host also has an alias for ***johnny*** and returns the mail to the host ***m135***, which repeats its looking-up and returns the message to ***chris***.

If it wasn't for a value known as "***hop count***", the mail would continue to circulate between the two last hosts. The hop count defaults to 17, meaning that no mail will do more then 17 "hops" to reach its final destination. When the hop count is exhausted, the message is returned to the sender with a transcript (audit trail) of the message route. Still, doing 17 unnecessary hops consumes lots of resources!

Rules for Mailer Alias Files

As we've seen through the examples above, loops can easily be created by defining and effectively redefining aliases in the mailer alias files of the hosts in a network. However, by following a few simple rules, you can avoid these problems:

1. On "post office" hosts, define all aliases so they forward mail to user IDs - never to another alias.
2. On hosts using a "post office" host to forward mail, only define aliases that point to the "post office" host.
3. As follows from the above points, never make a mailer alias which is identical to a user ID.

Basic Mail System Aliases

In addition to the mailer alias file, the basic mail system accepts aliases from the file /usr/lib/Mailrc, which defines global aliases, and $HOME/.mailrc, which defines personal aliases. Aliases defined in these file are only used for outgoing mail and can not cause loops by themselves. Obviously, each user can have a unique personal alias file. The following is an example of two aliases defined in $HOME/.mailrc:

```
alias: swede   jan@ps2nc
alias: german  sys2!dieter
```

Mail Handler Aliases

The Mail Handler (MH) alias file is /usr/lib/mh/MailAliases and is global for all users on a system. You enter aliases the same way you do in the basic mail personal alias file, but you can also use special notation to create alias lists dynamically as new users are added to a host. The following example shows how to define an alias for all users with a user ID above 200 on a host and reads additional aliases from the file /usr/adm/mh/localAliases:

```
all:  *
< /usr/adm/mh/localAliases
```

The sample MailAliases file shipped with AIX gives examples of the type of entries you can put in the file. In the above example, the alias ***all*** is really a distribution list since the alias is resolved to all user IDs on the local system.

Distribution Lists

A distribution list is very simply an alias that points to a list of users. For example, again using the configuration in Figure 115 on page 272, you could define one alias on the host ***chris*** that resolves to the following users on three different systems:

```
authors: jan@ps2nc, sys2!dieter, fribert@chris, chris
```

Mail sent to the alias ***authors*** would then be sent to:

```
jan       on system:   ps2nc
dieter    on system:   sys2
fribert   on system:   chris
chris     on system:   chris
```

Debugging

There are many ways to check your mail system. We shall discuss two of the commands that can help you test and debug your mail system:

1. The *sendmail -bt* command
2. The *mail -v* command.

Address Test

The command *sendmail* can be invoked in ***Address Test Mode*** that allows you to check that a receiver as given by you is evaluated to the correct user-ID and node ID. The following is a transcript of an address test done with *sendmail -bt*:

```
/usr/lib/sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
> 3,0 chris@ps2nc
rewrite: ruleset  3   input: "chris" "@" "ps2nc"
rewrite: ruleset  3 returns: "chris" "<" "@" "ps2nc" ">"
rewrite: ruleset  3   input: "chris" "<" "@" "ps2nc" ">"
rewrite: ruleset  3 returns: "@" "ps2nc"
rewrite: ruleset  0   input: "@" "ps2nc"
rewrite: ruleset  0 returns: "¬V" "local" "¬X" "@" "ps2nc"
>
```

Figure 116. Address Test Mode of sendmail Command

The text shown in a bold font is what you key in. The example shows how the address is resolved to the local system. If you ask for address test for a user at a remote host and the last line says ***local***, there's something wrong with your configuration files.

Transcript of Mail

To get a transcript of the processing of a mail message, use the *-v* option of *mail*. For example:

```
mail -v niels
```

The transcript will be displayed on your terminal so you can follow how the alias or user ID is transformed into a final user ID and node ID and what mail functions are involved in sending the mail.

```
chris... setsender: uid/gid = 201/0
niels... aliased to          niels@m135
niels@m135... Connecting to m135.itsc.austin.ibm.com.tcp...
220 m135.itsc.austin.ibm.com Sendmail AIX  2.1.2/4.03 ready at Wed, 31 May 89 14:56:06 GDT
>>> HELO chris.itsc.austin.ibm.com
250 m135.itsc.austin.ibm.com Hello chris.itsc.austin.ibm.com, pleased to meet you
>>> MAIL From:<chris@chris.itsc.austin.ibm.com>
250 <chris@chris.itsc.austin.ibm.com>... Sender ok
>>> RCPT To:<niels@m135.itsc.austin.ibm.com>
250 <niels@m135.itsc.austin.ibm.com>... Receipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 Ok
>>> QUIT
221 m135.itsc.austin.ibm.com closing connection
niels@m135... Sent
```

Figure 117. Sample Transcript from the Command: mail -v

Figure 117 shows an example of the output from *mail -v niels* from host ***chris*** when the local alias file defines ***niels*** as shown in the second line of the transcript. In this case, host ***m135*** is a "post office" that will forward the mail to ***chris@chris***.

By studying the transcript, you'll be able to determine if the mail is sent to the correct user on the desired host, but only the first part of the travel is shown by the transcript.

Changing some Configuration Files

When mail is used in an isolated, limited size network, you can customize your system exactly as you like. When sending and receiving mail from other networks, especially networks in other organizations, certain standards should be followed. Those standards are reflected in the default configuration files of the AIX mail system.

You should avoid changing options that influence the way users on foreign networks see mail from your network and the way you expect to receive mail and interpret mail from other networks. Even when you don't connect into other networks this is good practice. Consequently, you should restrict changes to configuration files to ones that affect only the parts of the mail system that's independent of the actual exchange of messages and of message header and trailer formats.

Unless it's required for reasons beyond your control, restrict changes to the `/usr/adm/sendmail/sendmail.cf` file to those required to match your configuration. In the following sections we shall cover some of the changes that you may want to make to other configuration files.

Configuring Basic Mail System

The basic mail system takes its defaults from the configuration file /usr/lib/Mail.rc, then overrides those defaults with those found in the file $HOME/.mailrc if that file exists.

Both files can contain information that:

- ***Set*** or ***unset*** mail options
- Define aliases (see "Basic Mail System Aliases" on page 277)
- Control what data is displayed when a message is viewed.

The Set and Unset Options

The following are examples of the set and unset options. We give no detailed explanation of the options. You must check the reference publication for your AIX system for details. To use the options, prefix them with **set** as in "*set ask*"; to disable an option, prefix it with the ***unset*** option.

append	Causes mail to be stored at the bottom of the mailbox. Default is to add mail at the top of the mailbox.
ask	Prompts for a ***subject*** field.
askcc	Prompts for the ***cc*** field.
autoprint	Causes the next message to be displayed when you delete the current message with the *d* subcommand.
crt=xx	Causes the *pg* command to be used for displaying mail message when the number of lines in the message exceeds xx.
dot	Causes the mail editor to be exited when a dot is entered on a line by itself.
EDITOR=	Selects the editor that will be called up with the *~e* subcommand.
folder=fd	Selects ***fd*** as the default folder directory.
hold	Prevents messages that you have not specified an action for from being moved to the mbox file in your home directory when you exit mail.
keepsave	Prevents a message from being deleted from your mailbox when they are saved with the *save* or *write* subcommands.
metoo	Copies mail to yourself if you send mail to a distribution list (alias) that includes yourself.
noheader	Suppresses display of the mail message list when you first enter the *mail* command.
quiet	Suppresses the display of message number.
record=rf	Logs outgoing mail in the file ***rf***.
screen=xx	Tells that you want the list of mail to be displayed in chunks of xx lines.
toplines=xx	Sets the number of lines displayed from messages when the *top* subcommand is used.

Controlling Displayed Information

The ***ignore*** option controls what parts of messages are displayed by the *mail* subcommands. See your `/usr/lib/Mail.rc` file for examples of fields that can be specified with the *ignore* option.

Configuring the Mail Handling Package (MH)

The Mail Handler Package uses several configuration files. Please refer to the publication *Managing the AIX Operating System* for a list of all files. The one file that each user may want to change is the configuration file `$HOME/.mh_profile`.

When a user first invokes one of the MH commands, the file `.mh_profile` is created in her home directory. The file describes the personal options and initially only defines the directory where the user's message folders are to be stored. This directory name defaults to `$HOME/Mail`. The user may edit the configuration file in her home directory and add personal configuration options.

The publication *Managing the AIX Operating System* lists the options you can set in the `$HOME/.mh_profile` file.

Examples of Using Mail

To illustrate the use of AIX mail, we will show how you:

1. Compose and send a message
2. Send a file
3. Look at incoming mail
4. Answer mail
5. Forward mail
6. Save mail in a folder
7. Handle messages in a folder.

This will be illustrated first through examples for the basic mail system, then for the Mail Handling package. All example output is from AIX PS/2.

Using the Basic Mail System

To send a message to the ***authors*** distribution list defined in "Distribution Lists" on page 278:

Composing and Sending a Message

1. Enter the command: `mail authors` and wait for the system to prompt for the subject of the message.
2. Key the subject text and press ENTER. Then key the text of the message, line by line. The mail editor is used to compose the message.
3. If you want to see the mail editor subcommands, enter ~? and press ENTER.
4. As an example, key: ~v to edit the message using the *vi* editor. Make any desired changes using *vi*, the save the message and exit *vi*.
5. After you exit *vi*, the mail editor takes over. It tells you that it did so by saying: `(continue)`.
6. Type any additional text you may want. When all text is typed, key a dot (full-stop) on a line by itself and press ENTER.
7. The system prompts you for "Cc:" (carbon copy receivers). Key any additional receiver and press ENTER or press ENTER without keying anything if you don't want additional receivers.
8. The message is sent and the AIX prompt displayed.

In the next example, assume we've promised the user ***dieter*** to send him our file `/etc/profile`. Here's how we keep our promise:

Sending a File

1. Key: `mail dieter < /etc/profile` at the command line.
2. The system prefixes the file with header information and sends the resulting message immediately.

User ***dieter*** now logs in to his host. He is greated with the message: "[***YOU HAVE NEW MAIL***]" and responds by looking at the system mail box:

Looking at Incoming Mail

1. The user keys: `mail` at the command line. The system responds by displaying a list of mail in the system mailbox. The last line displays the mail system prompt ("&"). For example:

```
Mail version 5.2+L.  Type ? for help.
"/u/dieter/.newmail": 2 messages 2 unread
>U  1 chris    Thu Jun  1 00:49  15/338 "A sample message"
U  2 chris    Thu Jun  1 00:59  74/1884
&
```

2. As we can see, there are two messages for ***dieter***. If the user doesn't want to do anything at this point, he just enters: *q* to quit.
3. The system exits the mail function with the response:

```
Held  2 messages in /u/dieter/.newmail
```

Sending Reply to Mail Message

1. Key: `mail` at the command line. The system responds by displaying the list of mail in the system mailbox:

```
Mail version 5.2+L.  Type ? for help.
"/u/dieter/.newmail": 2 messages 2 unread
>U  1 chris    Thu Jun  1 00:49  15/338 "A sample message"
U  2 chris    Thu Jun  1 00:59  74/1884
&
```

2. User ***dieter*** wants to look at message-1 and keys: `1` to select that message. The system responds by displaying the message:

```
Message 1 (of 2):
From chris  Thu Jun  1 00:49:18 1989
Date: Thu, 1 Jun 1989 00:49:18 CDT
From: chris
To: authors
Subject: A sample message

There's an authors meeting at 3 pm today.
Please confirm that you'll attend.

(EOF:)
```

3. Our user presses ENTER to stop viewing the message, and the system responds with the mail prompt.

4. The user keys: `r 1` to reply to the message. When you use the ***r*** subcommand, the reply is sent to all authors and receivers of the message you respond to. The ***R*** subcommand sends the reply only to the author(s) of the message you reply to.

5. The system creates the response header and waits for ***dieter*** to key the text of the response. The user keys the response, terminated by a dot on a line by itself and then presses ENTER.

6. The system prompts for "Cc:". The user presses ENTER to send the reply with no additional receivers.

7. The system sends the reply and displays the mail prompt.

8. The user keys: `d 1` to delete the original message. Then keys: `q.` to exit the mail function.

9. The system exits with the response:

```
Held 1 message in /u/dieter/.newmail
```

Forwarding Mail

1. Key: `mail` at the command line. The system responds by displaying the list of mail in the system mailbox:

   ```
   Mail version 5.2+L.  Type ? for help.
   "/u/dieter/.newmail": 1 messages 1 unread
   >U  1 chris    Thu Jun  1 00:59  74/1884
   &
   ```

2. User ***dieter*** wants to forward the file he received from ***chris*** but doesn't remember how to do it, so he keys: `?` to display the help screen. The system responds by displaying a list of subcommands.
3. The user doesn't find any help, so he turns to the publication *AIX Communications Handbook*. He finds help and types: `m johnny` to create a new message for the user ***johnny***.
4. The system responds by prompting for a subject. The user keys a subject text. He then types: `~m 1` to imbed the original message (message-1) from ***chris*** in the message to ***johnny***.
5. To check that the message now looks okay, the user enters `~p` to see the resulting message text. The system displays the text.
6. The user accepts the text and enters a dot on the text line to terminate the editing of the message. The system prompts for "Cc:" and the user presses ENTER because no additional receivers are required.
7. The system sends the message and displays the mail prompt.

Receiving a File

1. The user now wants to save the received message in a file and keys: `cd` to make sure the working directory is his home directory. The system responds with:

   ```
   (wd now /u/dieter)
   ```

2. The user keys: `w profile.chris` to save the received message (without header lines) in the file `/u/dieter/profile.chris`.
3. The system responds with:

   ```
   "profile.chris" [New file] 72/1668
   ```

   to say a new file was created (rather than the message being appended to an existing file of the specified name) and that the new file has 72 lines and is 1668 bytes long.
4. The user deletes the message from the mailbox by keying: `d 1` and keys: `q` to exit the mail function.
5. The AIX prompt is displayed.

When the *mail* command is used without any arguments, it displays a list of mail messages in the system mailbox for your user ID: `$HOME/.newmail`. If you view a message and exit the mail function without doing anything to the message, it's moved to the personal mailbox: `$HOME/mbox`. If you want to save the message in a different mailbox, you will do the following:

Saving Mail in Folder

1. Key: `mail` at the command line. The system responds by displaying the list of mail in the system mailbox.
2. To save, say, the third message in the mailbox `meetings` and delete it from the system mailbox you type: `s 3 meetings` and the system responds with:

   ```
   "meetings" [New file] 12/262
   ```

3. Type: `s 2 meetings` to also move the second message from the system inbox to the same folder. The system responds with:

   ```
   "meetings" [Appended] 15/338
   ```

4. Assuming you had three messages in the system mailbox, now key: `1` to look at the remaining message. The system displays the message.
5. Press `ENTER` to stop the viewing and then press: `q` to exit the mail system.
6. The system exits, saying:

   ```
   Saved 1 message in mbox
   ```

   and returns to the AIX prompt.

We now have two messages in the folder `meetings` and one message in `mbox`. As a last example, let's work with those folders:

Working with Folders

1. Key: `mail -f meetings` at the command line. The system responds by displaying the list of mail in the folder `meetings`. You can now work with the messages as you would do if you were working in the system mailbox.
2. For example, type: `d 1` to delete message-1.
3. If you now exit the mail system, the system responds with:

   ```
   "meetings" complete
   ```

4. Had you deleted all mail from the folder, the response would be :

   ```
   "meetings" removed
   ```

5. Now type: `file mbox` to change to the `mbox` folder (your personal mailbox).
6. The system responds by displaying the list of mail in your personal mailbox. You are now back where you would be if you initially typed: `mail` to work in the system mailbox.

Using the Mail Handler Package (MH)

To send a message to the ***authors*** distribution list defined in "Distribution Lists" on page 278:

Composing and Sending a Message

1. Enter the command: `comp` and wait for the system to prompt for receiver(s) with: `To:`
2. Key `authors` . to send to our distribution list.
3. The system prompts: `cc:`
4. Press `ENTER` to not send a copy to other receivers and wait for the system to prompt: `Subject:`
5. Key the subject text. The system responds by displaying a separator line to indicate that you can start composing the message text.
6. Key the text of the message, line by line. The mail prompter is used to compose the message. When finished, press `Ctrl-D` to terminate the editing of the message.
7. The system responds by displaying a separator line and the prompt: `What now?`.
8. Type: `?` to see what you can reply to that[34]. The system responds by listing the options.
9. As an example, key: `edit e` to edit the message with *INed*. When finished, exit the editor the usual way.
10. The system again responds with the prompt: `What now?`.
11. Type `send` to send the message.
12. The system sends the message and returns to the AIX prompt.

The Mail Handler Package has no command that allows a file to be sent.[35] If you want to send a file to another user, you'd use one of the several other options you have in an AIX system. You could, for example, use BNU or TCP/IP *tftp* to send the file.

If you insist that you want to use MH commands to send a file, prepare to send a message. When composing the message, use an editor to imbed the file you want to send, exit the editor and send the message as usual.

User ***fribert*** now logs in to his host. Mail has arrived so the message: "[***YOU HAVE NEW MAIL***]" is displayed. The user responds by moving the mail from his mail drop to his default personal inbox:

[34] You can display the options of any MH command from the command line by entering the command followed by a question mark.

[35] Actually, if a file contains a "To:" line in the required format, you can send the file using the *send* command.

Looking at Incoming Mail

1. He keys: `inc` and MH responds with:

   ```
   Create folder "/u/fribert/Mail/inbox"? y
   Incorporating new mail into inbox...
      1+ 06/01 chris@pc2nc       A sample message  <<There's an authors mee...
      2  06/01 chris@ps2nc       Sent file  <<# SCCSID(@(#)profile...
   ```

   This user never used MH before, so he's prompted for the creation of his default personal inbox. When he accepts that, the mail is moved to the inbox in /u/fribert/Mail/inbox. At any time, the command *scan* can be used to display the messages in the inbox.

2. The AIX prompt is displayed after *inc* moved the mail to the inbox.

Sending Reply to Mail Message

1. Key: `scan` at the command line to see a list of mail in the inbox. The system lists all mail in the inbox.

2. User ***fribert*** says: `show 1` to see the first message. The system responds with:

   ```
   (Message inbox:1)
   Received: by ps2nc (1.0L/4.03)
             id AA01346; Thu, 1 Jun 89 03:17:17 CDT
   From: chris
   Message-Id: <8906010817.AA01346@nc>
   To: authors@ps2nc
   Subject: A sample message
   Date: Thu, 01 Jun 89 03:17:16 -0600
   Status: O

   There's an authors meeting at 3 pm today.
   Please confirm that you'll attend.
   (EOF)
   ```

3. The user presses ENTER do stop viewing the message and the system returns to the AIX prompt.

4. User ***fribert*** will answer the message. He types: `repl 1` and gets this response from the system:

   ```
   To: chris@ps2nc
   cc: authors@ps2nc, fribert@ps2nc
   Subject: Re: A sample message
   In-reply-to: Your message of Thu, 01 Jun 89 03:17:16 CST.
                <8906010817.AA01346@nc>
   --------
   ```

5. He now keys the response text and presses ENTER. The system then prompts: `What now?`. Johnny keys: `send` to send the reply message.

6. The *repl* command sends the mail and exits. The AIX prompt is displayed.

7. If Johnny wants to remove the message from the inbox, he can type: `rmm 1` to delete it.

Forwarding Mail and Receiving File

1. The user keys: `scan` to see what's in his inbox. The system responds with:

   ```
   2  06/01 chris@ps2nc        Sent file  <<# SCCSID(@(#)profile...
   ```

2. Now Johnny types: `forw 2` to forward the remaining message. The system responds by prompting: `To:`
3. Our user responds by keying the receiver ID: `jan` and the system prompts: `cc:`
4. Johnny presses `ENTER` and the system prompts: `Subject:`
5. The user responds by keying a text and press `ENTER`. The system displays:

   ```
   --------Enter initial text
   ```

6. Johnny keys any text he wants prefixed to the forwarded message and hits `Ctrl-D` to terminate. The system responds by displaying the resulting text of the message and prompts: `What now?`.
7. Johnny keys: `send` and the mail is forwarded to ***jan***. The AIX prompt is displayed.

With MH, folders are directories. If you try to move mail to a non-existent folder, you will be asked if you want to create the folder directory. Let's see an example where the user ***jan*** works with his mail:

Saving Mail in Folder

1. Type: `refile 1 +meetings` to move message-1 from the personal inbox to the folder `meetings`.
2. The system responds by asking you:

   ```
   Create folder: "/u/jan/Mail/meetings"?
   ```

3. Answer: `y` to create the folder. The message is then moved from the inbox to the specified folder.
4. The AIX prompt is redisplayed.
5. Type: `refile 2 +meetings` to move message-2 from the personal inbox to the folder `meetings`.
6. The message is moved and the AIX prompt redisplayed.
7. If the specified message is not in the inbox, you will get the response:

   ```
   refile: message 7 doesn't exist
   ```

If we started out with three messages in the personal inbox before moving two of them, we now have two messages in the folder `meetings` and one message in `inbox`. Now let's work with those folders:

Working with Folders

1. Start by saying: `folder -all` to list all the folders. The system responds with:

```
         Folder      # of messages (  range  ); cur  msg  (other files)
           inbox+ has    1 message  (   3-   3).
       meetings  has    2 messages (   1-   2).

                TOTAL=    3 messages in 2 folders.
```

   The current folder is marked with a "+" after the folder name.

2. Type: `folder +meetings` to change to the `meetings` folder. The system responds with:

```
       meetings+ has    2 messages (   1-   2).
```

   You can now work with the messages in `meetings` as you worked with the personal inbox earlier.

3. For example, type: `rmm 1` to delete the first message. The message is quietly deleted and the AIX prompt displayed.

Mail to VM

An IBM System/370 host with TCP/IP for VM installed may cooperate with the AIX mail functions so that mail can be sent from an AIX system to VM where it can be accessed by VM users, and so that mail sent from VM using *NOTE* and *SENDFILE* can be accessed by users on the AIX system.

To provide these functions, the VM system must have the ***SMTP virtual machine*** active and properly configured. The VM host can act as a gateway between TCP/IP networks on one side and an RSCS network on the other side, forwarding mail between the two networks. The publication *TCP/IP for VM, Installation and Maintenance Manual* describes how to install and configure the SMTP virtual machine.

SMTP NAMES files define:

- Mail aliases
- Forwarding of mail
- Distribution lists.

The file `/usr/adm/sendmail/sendmail.cf` has commented lines that can be uncommented to define a default RSCS gateway. Look for ruleset S24.

Sending Mail from VM

TCP/IP for VM includes enhanced versions of the *NOTE* and *SENDFILE* programs. The enhanced version of the *NOTE* command allows you to specify recipients using the address format used by AIX mail. A single copy of the note is spooled to the SMTP virtual machine which, in turn, delivers one copy of the note to each of the TCP recipients. Mail to recipients on the local host or on a connected RSCS network is delivered in the normal manner.

The enhanced *SENDFILE* command can be used to send ***text*** files to recipients on the TCP network. *SENDFILE* adds a header to the file and spools it to the RSCS virtual machine which, in turn, delivers one copy of the note to each of the TCP recipients. Mail to recipients on the local host or on a connected RSCS network is delivered in the normal manner.

Receiving Mail from TCP/IP Networks

Mail arriving from senders on foreign TCP hosts is delivered directly to the virtual reader of the VM recipient by the SMTP virtual machine. This mail can be read using PROFS or the CMS commands:

- *RDRLIST*
- *PEEK*
- *RECEIVE*.

PROFS Extended Mail

The product *PROFS Extended Mail* (5798-FBJ) provides enhancements to the PROFS mail function to allow for the exchange of mail between PROFS and other mail systems. The product supports addresses and note formats that conform to:

- PROFS
- CMS Note
- RFC822 (ARPA Internet Text Messages)
- CCITT X.400 Recommendations
- PROFS Extended Mail (Internet style)
- PROFS Extended Mail (PROFS style).

For details about PROFS Extended Mail, refer to the publication *PROFS Extended Mail, User's Guide and Installation Manual*.

Reference Publications for This Chapter

Mail is described in:

Using the AIX Operating System (AIX/RT), SC23-2007
Using the AIX Operating System (AIX PS/2), SC23-2024
Managing the AIX Operating System (AIX/RT), SC23-2008
Managing the AIX Operating System (AIX PS/2), SC23-2031
TCP/IP for VM, Installation and Maintenance Manual, GC09-1203
TCP/IP for VM, Command reference manual, GC09-1204
PROFS Extended Mail, User's Guide and Installation Manual, SH21-0044

Other relevant publications are:

IBM RT Interface Program for use with TCP/IP, SC23-2005
AIX Operating System Commands Reference, SBOF-1814
AIX Operating System Technical Reference Files and Extensions, SC23-2126

AIX Access for DOS Users and AIX DOS Server

IBM AIX Access for DOS Users (AADU), program number 5709-030, allows an IBM Personal System/2 or IBM Personal Computer running PC-DOS to communicate with AIX hosts with the DOS Server installed. The principal and most important feature of AIX Access for DOS Users is the capability to access the file system of AIX hosts as logical extensions of the file system on the PS/2 or PC. This access is implemented through the concept of ***virtual drives*** where the file system of each accessed AIX system is assigned a virtual drive number on the PC-DOS system. Access to AIX file systems is totally transparent to almost any PC-DOS application.

Other, important features of AIX Access for DOS Users allow PC-DOS users and programs to use AIX systems as print servers, let PC-DOS users execute commands directly on an AIX system and allow users on the PC-DOS machine to log in to an AIX system using DEC VT100 terminal emulation. All access to AIX systems is subject to the standard authentication procedures for AIX, providing maximum security.

For users that are accustomed to an AIX or UNIX environment, AIX Access for DOS Users also provides a PC-based ***vi editor*** that is largely compatible with *vi* on the AIX systems. This editor stores data in the traditional UNIX-style format. Other utilities are provided, allowing for conversion of files between AIX- and DOS format, etc.

The AIX systems can be accessed through Token-Ring or Ethernet connections or via an RS232-C serial connection.

Hardware Prerequisites

An IBM Personal Computer, PC XT, PC XT Model 286, PC/AT or IBM Personal System/2 with minimum 192 kb. To use the PC-based *vi* editor a minimum of 384 kb is required. Support for virtual files and printers uses a minimum of 70 kb of system memory, in addition to the requirements of IBM PC-DOS Version 3.30.

The following communication adapters are supported:

- For IBM Token-Ring connection:
  - IBM Token-Ring Adapter
  - IBM Token-Ring Adapter/A (for PS/2 Micro Channel system)
- For Ethernet connection:
  - Ungermann-Bass Net/1 PC NIC Model 2274A adapter
  - 3Com Etherlink Adapter
  - Ungermann-Bass NIC Model PS/2 (for PS/2 Micro Channel system)
- For RS232-C connection:
  - IBM Asynchronous Communications Adapter
  - IBM Serial/Parallel Adapter
  - IBM PS/2 Dual Asynchronous Adapter (for PS/2 Micro Channel system).

Software Prerequisites

The following software prerequisites apply:

- The operating system of the PS/2 or PC must be IBM PC-DOS Version 3.30.
- AIX Access requires a connection to at least one AIX system with IBM AIX DOS Server installed. IBM AIX Access for DOS Users is compatible with the following DOS Server programs:
  - AIX/RT DOS Server
  - AIX PS/2 DOS Server
  - AIX/370 DOS Server.
- The required level of TCP/IP support is included with the AIX/RT 2.2.1, AIX PS/2 1.1 and AIX/370 operating systems.

Limitations

For countries that require national language characters other than English or use national keyboard layouts, the most important limitation of AIX Access for DOS Users is ***the total absence of National Language Support*** for the terminal emulation function and the *vi* editor. The terminal emulator makes the PC-DOS system appear as a VT100 terminal, which means that only a 7-bit character set is supported. Similarly, the *vi* editor uses a 7-bit character set because it uses the high-order bit to carry control information for the editor.

Personal computer software that manipulates the device registers of the Asynchronous Communications Adapter cannot execute concurrently with AIX Access when the personal computer is connected to the AIX host via an asynchronous RS232-C connection.

Some DOS commands for disk management are not supported when issued against a virtual disk, but they will function normally when issued against a local physical disk. The commands affected are: *assign, chkdsk, diskcopy, diskcomp, fdisk, format, label, sys* and *tree.* Application programs with similar functions will fail too.

AIX Access cannot coexist with *IBM Lan Support Program* and AIX Access cannot coexist with *TCP/IP for DOS Users.*

Installation

One important thing to be aware of before installing AIX Access, is the copy protection scheme. Although you can make copies for backup reasons, you cannot use two copies at the same time. When the AIX host detects that two copies made from one set of distribution diskettes are active simultaneously, it disables the session of the user who logged in last.

Before you install AIX Access for DOS Users you must install your communications adapter in the personal computer. Study the documentation for your adapter carefully, as you may have to change some adapter settings if you have adapters with confliction assignments. For example, if you have an IBM 3270 Connection Adapter in a PS/2 and you want to install a Token-Ring Adapter in the system as well, you will have to use a different interrupt level (for example interrupt level 3) instead of the default interrupt level 2 for the Token-Ring Adapter. This is also the case if you want to install the Ungermann-

Bass PC NIC Model 2274A adapter in a personal computer that already has the IBM 3278/79 adapter installed.

In a PS/2 micro channel based system, changes to adapter settings are made electronically by using the *Reference* diskette that came with the PS/2. In other PS/2s and PCs, you change the settings by moving jumpers and/or switches on the adapter itself. Appendix A of *AIX Access for DOS Users, User's Guide* gives detailed instructions about the adapters supported and how to install them.

Before you install AIX Access for DOS Users, you should also read the READAADU.DOC file on the first distribution diskette. This file will have information about the latest changes made to the programs.

The installation procedure for AIX Access for DOS Users is as follows:

1. Insert distribution diskette number one in your diskette drive, change your current drive to the diskette drive, for example "A:" and enter the command *install*. You will be asked what network you will be using: Ethernet, Token-Ring, RS-232 or a combination of two of those. When installing on Token-Ring, ***select the IEEE 802 Token-Ring option*** for IBM Token-Ring Adapters.
2. You will be given the choice between three different packages:
   a. Basic package
   b. Standard package
   c. Development package.

   Choose the "Basic package" if you only need host file services, terminal emulation and printer support.

   Choose the "Standard package" if you also want the *vi* editor and some additional AIX Access utilities. The utilities are described in "AIX Access Utilities" on page 299.

   Choose the "Development package" if you also want the *library* and *include* files. These files are only required if you plan to develop your own applications.
3. If you install AIX Access for DOS Users for use on a LAN, you'll be prompted for the Internet address of the PC-DOS system. This address is a unique identification of your machine on the network. You should make sure nobody else is using the Internet address you type in. For details about Internet addresses see "Internet and Internet Routing" on page 166 and "Customizing Host m135" on page 185.

If you change the default interrupt level for an adapter ***after*** you have installed AIX Access, you will have have to edit the file *AUTOEXEC.BAT* and add the "-xn" option to the command *pciinit:*, where "*n*" is the interrupt level plus 8. For example:

```
pciinit -x11 -i9.3.1.15
```

where "11" is the sum of the interrupt level (in this example 3) and 8, and where "9.3.1.15" is the Internet address of your system. This information is inserted in the AUTOEXEC.BAT file when you install AIX Access for DOS Users.

Using AIX Access

AIX Access is initialized using the *pciinit.exe* command. Normally, this command is executed from the AUTOEXEC.BAT file when PC-DOS is started. The *login* and *logout* commands, respectively, establishes and terminates a host connection.

Some programs cannot coexist with AIX Access if they are initiated ***before*** AIX Access is initiated with the *pciinit* command. For example, if you want to use the *IBM PC 3270 Emulation Program Entry Level*, this must be loaded ***after*** the *pciinit* command has been executed. If you have troubles with other programs, experiment with the load sequence.

Establishing Connection to DOS Server

Before you can use any services of an AIX system you must establish connections to the AIX systems you want to use. You must use the ***login*** command of AIX Access for DOS Users once for each such AIX system. If the command is given without arguments, you will see a logo screen. Press ENTER to proceed. AIX Access for DOS Users will then broadcast a message on the LAN trying to identify eligible AIX hosts.

Any AIX systems that have DOS Server running will answer the broadcast supplying your AIX Access machine with their hostname and Internet address. The AIX Access machine waits a certain time to make sure all AIX systems had a chance to respond and then presents you with a menu. This menu lists all LAN connected DOS Servers and (if you have selected RS232C from the installation menu) one line for the asynchronous connection.

Caution

During our testing we found that in a multi-network environment, AIX/RT hosts connected to Token-Ring would only answer broadcasts from AIX Access for DOS Users if the /etc/net file had ***localbroadcast* = *true*** set for the Token-Ring Adapter. This may also be the case in single-network environments.

When you select one of the hosts presented, a connection is established and you will be prompted for a user ID. The ID you type in must exist on the AIX host. You are then prompted to enter a password. If both are correct, AIX Access for DOS Users establishes a link to the AIX system and assigns you the access privileges of the user ID you used for login. The file system of the AIX host is then defined as a virtual drive in your PC-DOS system, using the next, free drive letter. If you have one fixed disk in your PC-DOS system and have a memory VDISK defined, you have already used drive letters C and D. The file system of the first AIX system you access will then be assigned drive letter E.

The *login* command takes command line arguments. When you use arguments, you are only prompted for those arguments you did not supply. To do a login without being prompted for anything, you could type:

```
login /d:f sysl fribert jlpass
```

which accesses the file system of host ***sys3*** as virtual drive ***F*** using the user ID ***fribert*** and the password ***jlpass***.

To stop using an AIX system, use the ***logout*** command. If you use this command without arguments, all connections to AIX hosts will be dropped. If you only want to drop the connection to one host, supply the hostname of that host or the drive letter used to access it followed by a colon.

Use of the AIX File System

It's important to note that although, in principle, the entire file system of the AIX system is defined as a virtual drive on your PC-DOS system, you can only actually access as much of the file system as an AIX user on the same host could access if logged in with the user ID you used. You are also subject to other access restrictions in the same way as a local AIX user, so you can't execute commands without execute permission and you can't delete or change files without write permission.

The file system of the AIX system may have parts that are not local to the AIX system you logged into. If the AIX host has directories mounted remotely with *Distributed Services* or *Network File System*, the file system of the AIX system has been logically extended beyond the capacity of its local file system.

Logically, this shouldn't make much of a difference to you as a PC-DOS user. To you, it all appears like one, big file system. However, AIX Access for DOS Users is not designed to support remotely mounted file systems on the AIX host. Therefore, use it if you like, but don't expect fixes if you have problems. Our experiences with remotely mounted files are:

Distributed Services

If the host you access from AIX Access for DOS Users has directories mounted with *Distributed Services*, you should be able to use such directories as if they were local on the AIX host. No tests have been done with mounted files and remote backup devices. If you use inherited mounts, you will probably run into problems.

Network File System

If the host you access from AIX Access for DOS Users has directories mounted with *Network File System*, you'll have problems with directory lists. When you enter the DOS command *dir* with or without a file specification, you will only see the first matching file (or subdirectory) in each directory block. This is likely to influence any program that uses DOS function calls like *Search Next Entry* (interrupt 21H, function 12H).

Other than that, everything seems to function as if the mounted file system was local on the AIX system. You can access any file in the current directory if you know its name or any other file if you know its full path- and file name. You can change current directory, etc.

DS and NFS

If the host you access from AIX Access for DOS Users has directories mounted with ***both DS and NFS***, you may see all kinds of problems. We strongly recommend that you don't attempt to use AIX Access for DOS Users to connect to such AIX hosts.

Virtual Drive Support

The AIX Access virtual drives present the AIX file system to PC-DOS as if it was locally mounted. The files on the disk are shared by PC-DOS and AIX. DOS application programs and data may reside entirely or in part on these shared volumes. Conversion routines will permit AIX- and IBM PC-DOS utilities and applications to pass text data files between them.

DOS and AIX naming conventions are different. When you access the AIX file system from PC-DOS, names (including directory names) are mapped to PC-DOS naming conventions. Files created from PC-DOS on the AIX file will have file names according to PC-DOS conventions.

AIX Access can support up to 16 virtual drives, but default is 4 drives. The statement "`device=bridge.sys /d:xx`" in the `CONFIG.SYS` file determines how many virtual drives can be used. The "xx" option is the number of drives.

Virtual Printer Support

AIX Access provides access to AIX system printers by allowing the user to reroute output selectively to AIX systems. AIX print requests can be initiated from PC-DOS through use of the print screen (PrtSc) key, by the DOS *copy* command to the remote print device associated with a PC-DOS device name (`LPT1`, `LPT2` or `LPT3`) or from an application program. Only those characters printable by the AIX printer are supported.

The AIX Access program *printer* will associate a PC-DOS device name with an AIX printer. For example:

```
printer sys3 lpt1
```

will route all print to the PC-DOS device ***lpt1*** to the host ***sys3***. Another example:

```
printer f: lpt2 preprog
```

This command will direct all print requests for the DOS device ***lpt2*** to the program named *preprog* on the host associated with the virtual drive F.

The command "`printer lpt2 /r`" will remove the association between the PC-DOS device ***lpt2*** and whichever host it was associated with. If you enter the *printer* command without options you will get a list of the current settings. See *AIX Access for DOS Users, User's Guide* for further information about the available *printer* options.

Terminal Emulation

A terminal emulation program, *EM.COM*, is provided with AIX Access. This program can emulate an ASCII terminal with the keyboard and program interface characteristics of a VT100.

Terminal emulation mode may be started from DOS at any time. If desired, the user can switch back and forth quickly between execution of DOS programs and a terminal emulation session to an AIX system. While a terminal session is inactive on the PC-DOS system, it remains active in AIX, so any background work initiated from the session continues. Output to the PC-DOS system is buffered and displayed on the PC-DOS console when the terminal session is activated. You must press the ***F10*** key to switch from the terminal emulation session to the DOS session. To return to the terminal emulation session again, execute the *em* command. If you want to terminate a terminal emulation session, you must enter the command *logout* and press the ENTER key. Then you must press F7 to get the "Close session menu".

Important

National character support is not available for VT100 emulation. You can neither display nor enter national characters in the emulation session.

Before terminal emulation can be used, it must be initialized with the *nty.exe* command. This is normally taken care of by the installation procedure which will make sure *nty.exe* is started from AUTOEXEC.BAT. If your DOS Server host is an IBM RT, you do not need *nty*, so you can remove the line starting *nty* from the AUTOEXEC.BAT file to save memory.

Using ON to Execute Commands on an AIX Host

The *on* command allows you to execute non-interactive AIX commands from PC-DOS. Commands executed with *on* are executed on the AIX system and output from the command is directed back to the PC-DOS machine. For example, if you want to rename the file file1.fil to file2.fil on the AIX host associated with your virtual drive D, you could issue the command:

```
on d: mv file1.fil file2.fil
```

Of course, since this is a renaming of an AIX file that can be accessed from PC-DOS, you could have used PC-DOS to rename the file, typing:

```
rename d:file1.fil file2.fil
```

but if the file name was, say, longer than file names according to PC-DOS conventions, you couldn't have done it from PC-DOS. You can even rename directories this way (which is not possible with the DOS *rename* command).

The following example will execute the command cal 1989 on the host ***chris*** and return the result to the PC-DOS user's screen, in this case, the calender of 1989.

```
on chris cal 1989
```

You can execute multiple AIX commands if you separate the commands by semicolons, but be aware that redirection of output goes to the your PC-DOS system. For example, the following command will place the output of the AIX command *cal* in the file tempfile in the current directory of the PC-DOS system, so the AIX *cat* command will not find the file.

```
on d: cal 1989 > tempfile; cat tempfile
```

To make the above work, you'd have to type:

```
on d: cal 1989 } tempfile; cat tempfile
```

By default, the *on* command executes AIX commands from the bourne shell (/bin/sh -c). To use the c-shell, set the PC-DOS environment variable ***ONPREFIX*** like this:

```
set ONPREFIX=/bin/csh -c
```

If you set many or large environment variables in PC-DOS, you must expand the default environment space in PC-DOS which is 128 bytes. Place the following line in the CONFIG.SYS file on the PC-DOS system:

```
shell=driveletter:command.com /e:nn /p
```

where “nn” is the size of environment space (from 160 to 32768 bytes).

As mentioned, you can pipe and redirect input and output of the AIX command executed via *on*. The standard symbols used for PC-DOS and AIX piping always signifies redirection between PC-DOS and AIX. Another set of symbols is used for redirection to AIX. The full list if symbols is:

!	pipes output to an AIX command
|	pipes output to a PC-DOS command
{	redirects input to an AIX command
<	redirects input to a PC-DOS command
}	redirects output to an AIX command or file
>	redirects output to a PC-DOS command or file.

Executing AIX Commands Directly from PC-DOS

If you don't want to use the *on* command every time you execute an AIX command from PC-DOS, you can make special copies of the AIX commands in a format that allows you to execute them directly. To save disk space, you should always create such commands using the AIX *ln* command. For example, to create a copy of the AIX command *cat* in a format that will allow you to execute it directly from PC-DOS, type:

```
on - ln on.exe cat.exe
```

Note this form of the *on* command where neither a drive letter nor a hostname is specified. The "-" following the command says that you should execute *on* on the host associated with the current drive.

If you want to convert many AIX commands, first convert the *ln* command, then all the others:

```
on - ln on.exe ln.exe
ln on.exe cat.exe
ln on.exe ps.exe
...
```

As you may have already noticed, this raises the interesting possibility of removing the *on.exe* file from the PC-DOS system after you have converted all the commands you want PC-DOS users to be able to use on the AIX system.

National Language Support

If you installed your PC-DOS system and the AIX systems you want to connect into, with National Language Support you will be able to use national characters and your national keyboard on both systems. Also, files on AIX/RT and AIX PS/2 are stored according to IBM code page 850. You should use the same code page on the PC-DOS system to get full compatibility (actually, you should always use code page 850 and avoid uni-national code pages).

PC-Based vi Editor

PC-DOS users who often need to edit AIX files can use the PC-based *vi* editor of AIX Access for DOS Users. This editor uses the same syntax and commands as the AIX *vi* full-screen editor. By using the PC-based editor, users can perform most editing on the IBM Personal System/2 or PC, thus unloading the AIX system. The files created with the PC-based editor can be used directly by AIX. A file can be saved in PC-DOS format, so the file can be read by DOS applications.

AIX Access Utilities

The *dos2aix* and *aix2dos* commands converts text files between DOS text format and AIX text format. This is necessary because "end-of-line" is indicated by a ***line feed*** character in an AIX text file while "end-of-line" in a PC-DOS file is indicated by both a ***carriage return*** and a ***line feed*** character.

The *doswhat* command displays identifying information on AIX or DOS files, similar to the AIX *what* command. The *jobs* command displays the status of AIX processes initiated by the *on* command.

The *udir* command lists files and directories on the virtual drive in AIX format and mapped DOS format. (as opposed to the DOS *dir* command that only displays the files and directories in mapped DOS format). The *udir* command will also display the AIX permissions for the files and directories. You can use the *on* command to change permissions on files and directories, for example:

```
on e: chmod a=r file1.lst
```

This will make the file `file1.lst` on the virtual drive E a "read-only" file for all users.

The *setdebug* command can be used as a problem determination tool. It can be used by a person knowledgeable in the DOS environment to trace DOS system call activity.

AIX Access Programming Library

PCILIB is a programming library that allows DOS programs written in C Language or Assembly Language to access the extended I/O control functions of the AIX Access program *BRIDGE.SYS*. See *AIX Access for DOS Users, User's Guide* for information about the *PCILIB* library.

AIX DOS Server

The DOS Server program is a part of AIX/RT, AIX PS/2 and AIX/370. It is installed using the *installp* command. When you install the program, one of the AIX startup files (/etc/rc.include for the IBM RT) is modified so it runs the shell script /etc/rc.pci which in turn starts DOS Server.

The script /etc/rc.pci starts two daemons: the connection server *pciconsvr.ip* and the map server *pcimapsvr.ip*. When a connection is established from PC-DOS system running AIX Access for DOS Users, a third daemon *(pcidossrv.ip)* is started. This daemon stays active until the connection is dropped from the PC-DOS system. Should you want to stop DOS Server, first use *logout* on all PC-DOS systems using the AIX host; then go through the following steps:

- Kill all AIX processes with names starting with ***pci*** using the following command:

```
kill -9 `ps -e|grep pci*|cut -c1-6`
```

- Invoke the command: `/usr/pci/bin/sharectl -r`, to remove information about shared memory.

If terminal emulation is required and your DOS Server is a PS/2, you must add a Network terminal device (ttyn) using the AIX *devices* command. If your DOS Server runs on AIX/RT or AIX/370, you must add a pseudo terminal device (pty) using the AIX *devices* command. Set the following parameters:

```
automatic enable: true
terminal type:    vt100-em
parity type:      even
logger:           true
```

DOS Server on AIX/RT provides a command to display the available pseudo terminals in the system. The command is in `/usr/pci/bin/printpty`. Also, the *penable* command, entered without options, lists activated pseudo terminal devices.

Reference Publications for This Chapter

AIX Access for DOS Users is described in:

AIX Access for DOS Users, User's Guide, SC23-2041
AIX Access for DOS Users, Administrator's Guide, SC23-2042

AIX X-Windows in a Network Environment

The IBM AIX X-Windows System (or X-Windows, for short) is based upon X-Windows System Version 11, Release 3 as developed at the Massachusetts Institute of Technology (MIT). The initial release of X-Windows on AIX/RT Version 2.2.1 had an X Toolkit corresponding to X-Windows Version 11, Release 2, but was otherwise at release level 3. An update is available to upgrade to release level 3.

Overview

X-Windows is a network based windowing system that runs on AIX systems with bitmapped displays. The windowing environment supports multiple applications being displayed concurrently on the same screen (more precisely, on the same ***virtual terminal***). An application can show a clock with a sweep second hand in an area of the screen and update it every second while a file is being edited by another application in another area of the screen. Each area is called a ***window***.

X-Windows is not only a graphics system that displays output. It also takes care of user input and provides means for concurrent processes to communicate. In addition to this, applications developed for X-Windows can utilize windows on any monitor in a network in a device independent, network transparent fashion. Programs can be run on a remote computer and the results be presented on a local workstation. Three characteristics of X-Windows should be emphasized:

- X-Windows allows the development of applications that are easily portable to many different hardware platforms.
- X-Windows allows different types of machines to cooperate within a network. It includes a protocol for the communication between separate components of the system.
- X-Windows does not require or imply a particular style of user interface.

The IBM AIX X-Windows System is the presentation interface chosen for the IBM AIX Family Definition.

Clients and Servers

To work on a network, you must have programs running at both ends of the connection to send and receive information and to interpret it. In X-Windows the end that performs two-dimensional drawing on the display and controls input devices is named the ***X server***. Applications programs are called ***X client*** programs. This client/server model provides display and network independency. It is not necessary to rewrite, recompile or even relink an application for a new graphics display because the X server holds all display dependencies; the application does not need to include statements to distinguish between color and monochrome monitors.

On AIX systems, the X server is started automatically when the command *xinit* is invoked. *xclock*, *aixterm*, and all the sample programs provided with IBM AIX X-Windows are examples of X client programs. When used on an IBM AIX

X-Windows System, *xinit* usually starts one copy of *aixterm*, which opens an AIX shell in a window on the X server. Also, the AIX Window Manager *(aixwm)* is initialized. You can customize your X-Windows server on an AIX system by creating the file `$HOME/.Xdefaults` and changing your personal defaults in that file. A sample file is provided in the file `/usr/lpp/X11/Xamples/Xdefaults` on AIX/RT and in `/usr/lpp/X11/samples/Xdefaults` on AIX PS/2.

On AIX/RT, *xinit* is a small shell script in `/usr/bin/xinit` invoking the shell script `/usr/lpp/X11/bin/xinit`. On AIX PS/2, it's a symbolic link to `/usr/lpp/X11/bin/xinit`. The latter shell script defines the default actions taken when *xinit* is executed. You can modify this file to start multiple copies of *aixterm*, to start additional X-Windows programs or to open windows that'll run character based applications like a terminal emulator to a foreign host system (for example *em78* to do IBM 3270 emulation).

An X-Windows application does not need to know where its output goes, nor where its input comes from. As shown in Figure 118, the X-Windows server acts as an intermediary between X clients running on either the local or a remote system and the resources of the X server system; it distributes user input to X clients and accepts requests from various X client programs to display output. The X server maintains complex data structures (windows, cursors, fonts, etc.) as resources that can be shared between X clients and referred to by resource IDs. X client programs may connect to any display in the network if the host they are running on has permission from the X server that controls the display, and X client programs may connect to more than one X server at a time.

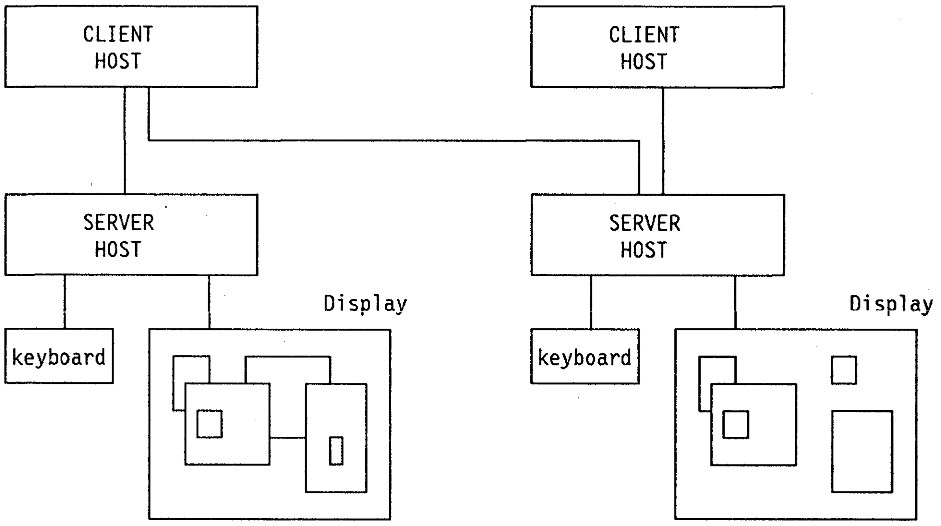
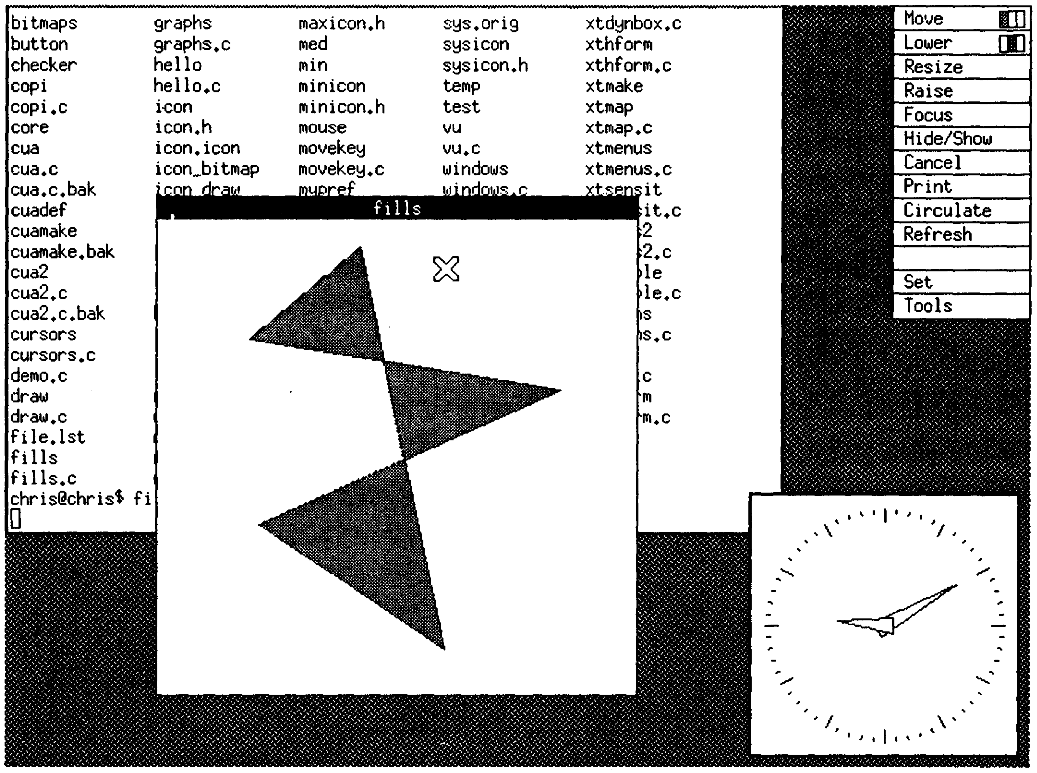


Figure 118. X-Windows Server and Client Programs

The X clients and the X server communicate by means of a protocol which is part of the X-Windows definition (the ***X Protocol***). The X Protocol specifies what makes up each packet of information that is transferred between clients and server. If clients and server are on the same machine, the X Protocol uses a local ***Inter Process Communication*** (IPC) mechanism; if they are on different machines, the X Protocol uses the ***User Datagram Protocol*** (UDP).

Figure 119 on page 303 shows the display of an X-Windows server where the AIX Window Manager, an *aixterm* window, the *xclock* analog clock and an X client application are active.



```
bitmaps        graphs         maxicon.h      sys.orig       xtdynbox.c
button         graphs.c       med            sysicon        xthform
checker        hello          min            sysicon.h      xthform.c
copi           hello.c        minicon        temp           xtmake
copi.c         icon           minicon.h      test           xtmap
core           icon.h         mouse          vu             xtmap.c
cua            icon.icon      movekey        vu.c           xtmenus
cua.c          icon_bitmap    movekey.c      windows        xtmenus.c
cua.c.bak      icon_draw      mypref         windows.c      xtsensit
cuadef                                                           it.c
cuamake                                                          2
cuamake.bak                                                      2.c
cua2                                                             le
cua2.c                                                           le.c
cua2.c.bak                                                       s
cursors                                                          s.c
cursors.c
demo.c                                                           c
draw                                                             m
draw.c                                                           m.c
file.lst
fills
fills.c
chris@chris$ fi
```

Figure 119. Sample X-Windows Server Screen

IBM AIX X-Windows Installation

IBM AIX X-Windows provides X server and X client functions on AIX/RT and AIX PS/2 while the AIX/370 implementation allows X client programs to execute on the System/370, but does not provide an X server itself. The installation procedures described below do not cover installation on AIX/370.

Prerequisites

X-Windows requires a graphics display. Any of the supported graphics displays and corresponding adapters on IBM RT and PS/2 can be used. In addition, you should have a mouse connected to your AIX system.

To run AIX X-Windows over a LAN, a Token-Ring Adapter or Ethernet adapter must be installed and configured in all AIX hosts that will participate. These hosts must also have TCP/IP installed and configured.

Before you can install X-Windows on AIX/RT, you must have the *Advanced Display Graphics Support Library* from Multi-User Services installed. On AIX/RT, you must also have the *Extended Programming Support* package of *Extended Services* installed in order to link the programs (which is done during installation).

Installation

On both AIX/RT and AIX PS/2 you install X-Windows with the *installp* command. The installation process will ask you to select the options you want to install. These vary between AIX/RT and AIX PS/2.

You will be prompted to select what national language keyboard(s) you want to use. Your first choice will be the default keyboard. Other choices will cause the corresponding keyboard mapping files to be installed, but you must run a utility program called *keycomp* to compile these files before they can be used. To activate a non-default keyboard mapping, you move the compiled file to `$HOME/.Xkeymap` before you start the X-Windows server.

Controlling Access to X-Windows Servers

If an X client program knows the hostname of a host running an X-Windows server, that program can open a window on the server. More often, the user supplies command line options to tell the X client program where to open its window(s). However, to prevent anarchy, most X servers require that the X client host is first granted permission to use the X server. This is done with the *xhost* command, issued on the X server.

Since more than one X server may be active at the server host, the *xhost* command must be issued from a shell running under the X server it applies to. On AIX systems this means it must be issued from an *aixterm* window on the X server. To allow the remote host ***chris*** to open windows on the local X server, say:

```
xhost +chris
```

To disallow the host chris to open windows on the local X server, type:

```
xhost -chris
```

If entered without command line options, *xhost* lists all hosts that are currently allowed to use the X server.

If you don't want to issue the *xhost* command by hand, you can create the file(s) `/etc/Xn.hosts` on the server host. The "`n`" stands for the X server number on the host. Each X server is assigned a number when started, beginning with zero. Usually, a host only has one X server running, so you probably only need to create one file: `/etc/X0.hosts`. To allow remote X-Windows client hosts to use your X server, put the hostnames of all authorized hosts in the file, each on a separate line.

Some X servers use other methods to grant access to X client hosts. One such X server is the one provided by X-Windows for DOS Users. Any host that's listed in the `/etc/hosts` file on DOS X Server is allowed to use the X server.

Starting an X Client Program

X client applications should (and usually do) accept a predefined set of command line arguments, in addition to those specific for the application. Examples of such arguments can be listed by entering:

```
aixterm -help
```

The command responds by listing all the valid options. Some of these, which should be accepted by any X-Windows client application, are:

-bg	Sets background color
-display	With argument `host:server` picks server to use
-fg	Sets foreground color
-fn	Tells the default font to use
-geometry	Sets window size and position

If a user on host ***chris*** wants to open an *aixterm* window on the DOS X Server server host ***ps2nc***, the user would type:

```
aixterm -display ps2nc:0
```

In case of an X-Windows for DOS Users server, the number is always zero because only one X server can be started. If the X server was on AIX/RT, for example, multiple X servers could be active and the number would have to correspond to the X server you wanted to use.

Window Manager

The IBM AIX X-Windows Window Manager *(aixwm)* is normally started from *xinit*. It displays a small window with a list of options. The options allow you to manipulate the windows on an X-Windows server. You can move, resize, restack and cancel windows, and you can select options that allow you to open new windows, run applications defined to X-Windows and to change the characteristics of the X server screen.

X-Windows for DOS Users

X-Windows for DOS Users (hereafter abbreviated to DOS X Server) is a PC-DOS program that lets you run X-Windows client programs installed on computers on your network. DOS X Server transforms your PC-DOS system into a graphics terminal.

An X server is a program that knows about a particular type of workstation display, keyboard and mouse. In the case of DOS X Server, it knows about the most common display types on a PC or PS/2. Since the X server handles the specifics of your display, X client programs do not have to be rewritten for every different type of display on which they open windows. An X client program can open a window on any display on the network that has an X server written for it. To summarize:

- An X server handles your hardware: display, keyboard and mouse.
- A X client runs X application programs on a computer on your network.

DOS X Server Installation

Prerequisites

To run the DOS X Server, you must have:

- An IBM Personal Computer XT, AT, or Personal System/2 Model 25, Model 30, Model 50, Model 60, Model 70 or Model 80.
- DOS 3.3 or later version.
- A CGA, EGA or VGA graphics adapter.

- A two-button IBM PS/2 Mouse and associated mouse driver or a three-button mouse with a driver that is compatible with the Microsoft mouse driver.
- At least one of the following communications adapters:

  For Ethernet connection:

  - Ungermann-Bass Net/One PC NIC Model 2274A
  - Ungermann-Bass NIC Model PS/2
  - 3Com Etherlink

  For Token-Ring connection:

  - IBM Token-Ring Adapter
  - IBM Token-Ring Adapter/A
- 640K bytes of memory, with 512K bytes available. Both EMS 3 and EMS 4 expanded memory is supported.
- A fixed disk is recommended but not required. You must have at least 720 kb of space available either on a fixed disk or on a diskette.

Installation

Installation of X-Windows for DOS Users is quite simple. Before you proceed, insert diskette 1 of the distribution diskettes in diskette drive A on your PC-DOS system and type:

```
a:
cd \readme
readme
```

This will bring you to the (only) description of DOS X Server that comes with the product. You should press the F2 key to print the entire document. It's required if you want to do anything non-trivial with DOS X Server.

It also gives you a comprehensive explanation about the installation procedure and explains what directories and files are moved to your PC-DOS system during the installation process.

To start installation, insert the first of the distribution diskettes in diskette drive A, and type:

```
a:
cd \
install
```

You will be prompted for several things, all of which should be pretty obvious:

1. Network type? If you want to use Token-Ring, select one of the options saying IEEE 802.
2. Complete installation? Say yes, if you want all the extra fonts installed. This option should not be selected on systems without a fixed disk.
3. Your systems Internet address and hostname. Be sure you use unique names and that the network address corresponds to that of hosts on the LAN you attach to.
4. Internet address and hostname for the default gateway host. Enter the values for any gateway you may have on your LAN. If you have none such, pick any of the hosts on the local LAN.

5. X client machines' Internet addresses and hostnames. Enter values for all hosts that shall be allowed to use your PC-DOS system as an X server.

After the installation process is completed, you will need to insert a statement in the file AUTOEXEC.BAT on the boot drive to load the mouse driver for the mouse you use.

Using X-Windows for DOS Users

The current version of X-Windows for DOS Users may fail to accept requests from an X client host if the TCP/IP interface has not been initialized. To do so, use the *telnet* command of DOS X Server:

```
telnet chris
Ctrl-]
```

Use the Ctrl-] key combination when you see the login prompt of the X client host. There's no need to log in since the network interface has already been initialized at this point.

The basic problem with starting an X client application on DOS X Server is that after the X server is started, you can not issue any DOS commands. All you can do is wait for a window to open on the X server. Of course, you'd like to control what's running, so you have to prepare properly for what you want to do. The document we asked you to print contains many hints. We have found it convenient to set up the following procedure:

1. Create a small batch file, called X.BAT, which contains:

   ```
   set xstartup=telnet %1
   xondos
   ```

   The first line sets the environment variable ***XSTARTUP*** which is used by the ***XONDOS*** command. The *XONDOS* command starts the DOS X Server but before doing so, it uses *telnet* to connect into the host pointed to from the environment variable.

2. To run an X client program from host ***chris***, type the following at the command line on the PC-DOS system:

   ```
   x chris
   ```

3. When you see the login prompt of ***chris***, log in to the host and issue the following command:

   ```
   nohup aixterm pchostname:0 &
   ```

   where pchostname is the hostname you assigned to your DOS X Server system. Then press Ctrl-D or type exit to log out from the host.

4. Your DOS X Server will now start, and an *aixterm* window will be opened on the X server screen.

5. To start the *aixwm* window manager, from the *aixterm* window, type:

   ```
   aixwm &
   ```

You are now ready to start any other X client application you want to run.

To stop the DOS X Server, press Ctrl-Alt-F10. You will see a small prompt asking you if you really want to stop. If you do, type y.

There's much more to X-Windows for DOS Users than described above. You should consult the documentation we asked you to print to learn about the many options (for example, how you use expanded memory), the limitations, etc. However, there's one limitation which is very important, but is not mentioned:

National Language Support

X-Windows for DOS Users does not have ***any*** national language support, whatsoever. It's made for the US market and supports only US English and the US keyboard. It does not even recognize all the keys of the IBM Enhanced Keyboard. Also, despite stated otherwise in the documentation, the fonts `DOS8X14` and `DOS6X13` do ***not*** provide a full 8-bit code page. Thus, you are not even able to display your national characters.

The final thing we shall point out is, that in addition to the ability to print the graphics screen on a local printer, DOS X Server is compatible with the ***Picture Taker*** (PT.EXE) program of *IBM Storyboard*. This allows you to capture screens for use in *IBM Storyboard* or other programs that can understand the format of the display image as saved with *PT.EXE*.

X-Windows Clients

Several systems allow you to develop application programs that can run as X client applications on an X-Windows server system. Each such system provides an application program interface (API) to develop your application programs. Such APIs are part of the full X-Windows client/server implementations of AIX/RT and AIX PS/2. In addition, APIs are currently available or announced for the following IBM systems:

- AIX/370
- VM/CMS
- MVS/TSO.

In all cases, TCP/IP is required on the systems in order to actually run the X client applications on a X server.

In addition, several non-IBM X client implementations run on the IBM X-Windows servers, just as IBM X client programs run on several non-IBM X servers. This compatibility should (and probably does) exist for all implementations of X-Windows Version 11, Release 3. We have tested several, such implementations but have not found incompatibilities worth mentioning.

Reference Publications for This Chapter

The following publications are referenced in this chapter:

IBM AIX X-Windows User's Guide, SC23-2217
IBM AIX X-Windows Programmer's Reference, SC23-2118
IBM AIX X-Windows Programming Guide, GG24-3382
IBM X-Windows for DOS User's Guide (as available on the distribution diskettes of DOS X Server)

X.25 Communications

This chapter describes the *IBM 6150 X.25 Communications Support*. We shall start by giving you a short, general introduction to X.25.

Introduction to X.25

An X.25 network is similar to a telephone network except that it carries computer files and messages rather than speech. In an X.25 network, computers connect via leased lines to special exchanges called "Packet Switched Exchanges" (PSEs), which are connected together to form the network.

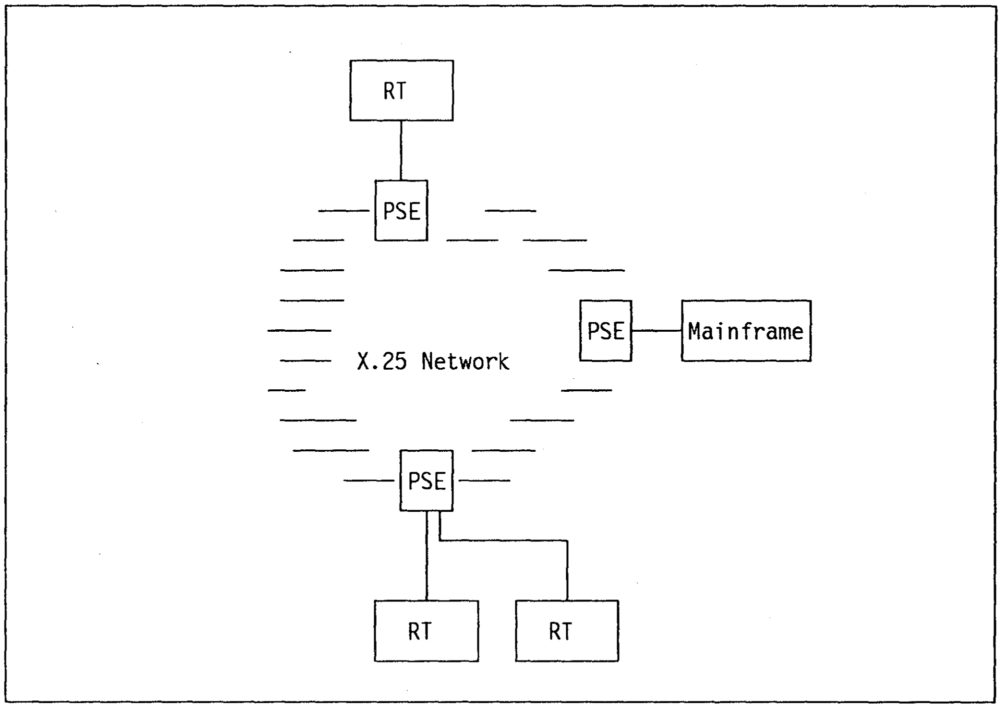


Figure 120. An X.25 Network

Network Addresses

Each X.25 connection or "X.25 line" into an X.25 network is identified by a number called the ***Network User Address*** (NUA). In most countries, a full NUA consists of a 3-digit country code plus a national terminal number ("host number") which uniquely defines every X.25 connection throughout the world. In most countries, the national PTT (Post, Telegraph and Telephone) authority is responsible for establishing the X.25 connections. The PTT will assign a host number to your X.25 connection when it is established.

In the USA, X.25 connections are assigned by several companies with each company managing a NUA prefix of 4 digits which includes the country number. Generally, the number of digits in an NUA depends on the network supplier, but the maximum allowed is 15. If the network uses less than 15 (British Telecom, the PTT in the UK, uses 13 for example), stations on the network may use the remaining digits to subdivide one X.25 connection so it can be used to run several, simultaneous "sessions".

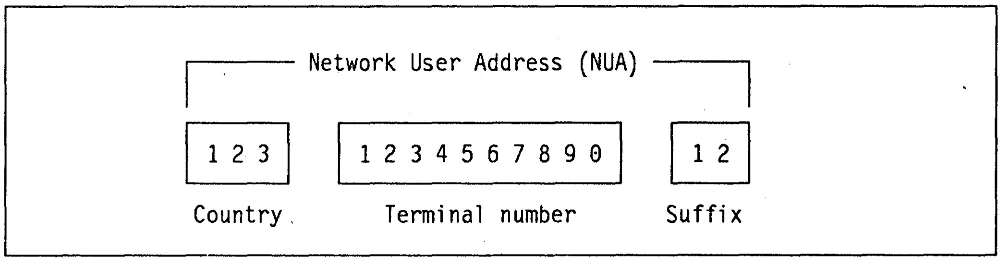


Figure 121. European X.25 Network User Address Structure

Network Protocol

Communication between computers on an X.25 network is also similar to telephone communication between people. When you pick up a phone and dial a number, a path is set up across the telephone lines of the telephone network. When the call is answered the conversation usually lasts until someone hangs up.

For this to work, a protocol, or set of rules, is required. One rule, for example, is that the connection is held until broken by one of the users. Other rules provide for signals to indicate "busy line", "number unobtainable", and so on. Several of these protocols exist, but the one discussed here is the X.25 protocol. "X.25" simply stands for the 25th recommendation of an international standards body known as the CCITT and is designed for a technique known as ***packet switching***.

Packet Switching

In a packet switching network, data is combined with an address (for example the NUA of the station to which data is to be sent) and control information to form a packet. The resulting packet is an independent unit that contains all required information to identify the receiver (and sender) and can be sent through any suitable path in a network.

X.25 packets from and to many destinations can be fed into the network (often referred to as the "X.25 Cloud") where they can be routed to the destination carried in the packets themselves. The actual transport of the packets may involve a large number of computer systems and may cross many, different data links, but this is totally transparent to both sender and receiver of the packets.

On one IBM RT, several calls can be active concurrently, so that many users can establish and use connections to remote systems as the same time.

Switched and Permanent Virtual Circuits

When a call is made over an X.25 network, a connection is established with another station and forms a ***Switched Virtual Circuit*** (SVC) with that station. The circuit acts as if you had made a connection over a switched telephone network, but no switching in traditional sense takes place, hence "Virtual". An SVC lasts only for the duration of the call.

Some Network suppliers provide ***Permanent Virtual Circuits*** (PVCs). These let you establish a permanent (virtual) connection between two stations analogous to a leased (non-switched) telephone line. However, most network providers prefer not to offer PVCs because they tie up network resources. An IBM RT using the IBM PC X.25 Communications Adapter can support a maximum of 20 virtual circuits (calls) concurrently.

SNA over X.25

The X.25 protocol has three layers as shown in Figure 122. These layers are exactly analogous to the bottom three layers of SNA, and they can indeed be used as the bottom three layers of SNA on the IBM RT. It is thus possible to connect an IBM RT to an SNA network over an X.25 connection. The SNA data link type for connections via X.25 is called ***Qualified Logical Link Control (QLLC)***.

LAYERS	DEFINITION
NETWORK	What path is data to follow? What is the NUA of the other station? At this level, you establish a logical connection to the other station.
DATA LINK	Describes how data is wrapped to reach destination safely. Known as LAP-B protocol.
PHYSICAL	The physical, electrical, level that connects a station to the network.

Figure 122. X.25 layers

IBM 6150 X.25 Communications Support Product Overview

IBM 6150 X.25 Communications Support provides X.25 support for the IBM RT family under the AIX/RT Operating System, supporting both SNA over X.25, a native X.25 API for non-SNA applications, and TCP/IP.

Prerequisites

The following components are required to establish X.25 connections from an IBM RT using the IBM PC X.25 Communications Adapter:

Basic Items

- IBM 6150 X.25 Communications Support, Program Number 5601-179, Part Number 07F3233. In order to support TCP/IP over X.25, get the following update from your local support center: PTF U400062 (Version 01 Release 01 Level 0010). When this update is applied, read the file `/usr/lpp/x25x/me.01.01.0010` for update information.
- IBM X.25 Diagnostic Support Pack for IBM 6150, Part Number 07F8892.
- IBM Personal Computer X.25 Communications Adapter, Part Number 2690617.

Prerequisite Items

- Advanced Interactive Executive (AIX) Operating System Version 2.2.1 or later. In order to support the IBM 6150 X.25 Communications Support on AIX/RT 2.2.1, you should request the following updates from you local support center: `IX02690`, `IX02691`, `IX02934`, `IX02966`, `IX02990`, `IX03024`, `IX03102` and `IX03103`.

Optional Items

- IBM X.25 Local Attach Cable, Part Number 2690618, for attachment of an ASCII monitor terminal.
- IBM X.25 Communications Adapter Technical Reference, Part Number 2690545.
- An IBM Personal Computer Communications Adapter Cable, Part Number 1502067, or its equivalent, is needed for connection to the Network Data Communications Equipment (check with your network supplier to see if this is supplied with the service).

IBM PC X.25 Communications Adapter Features

The IBM PC X.25 Communications Adapter with supporting software was originally announced to allow the attachment of an IBM PC, IBM PC XT, IBM PC AT or an IBM PS/2 model 30 to an X.25 packet switched network via an X.21 bis synchronous connection to enable the personal computer to operate as a packet terminal. This announcement was updated to extend the support to include the IBM RT.

The adapter features:

- Onboard Intel 8088 microprocessor, operating at approximately 5 MHz to offload the main processor.
- 128 Kb of parity checked RAM for onboard buffering and 8 Kb of ROM.
- Synchronous X.21 bis communications port for transmission speeds up to 19.2 Kbps in full duplex mode.
- Asynchronous port for secondary ASCII terminal used for line monitoring.
- Interrupt level choice of 3, 4 or 9 is selectable on the adapter. Adapter cannot share interrupt.
- Jumpers on the adapter define it as a primary or secondary device. Use the supplemental diagnostics to test the adapter itself and its jumper settings. The adapter support program supports only one device at a time.

IBM 6150 X.25 Communications Support Implementation

The IBM 6150 X.25 Communications Support program operates in conjunction with the IBM PC X.25 Communications Adapter to support the attachment of an IBM RT to an X.25 packet switched network. It provides an X.25 application programming interface, a set of X.25 applications to support the transfer of messages and files, and enables SNA-to-SNA and TCP/IP communications over the X.25 network.

Components

The X.25 support program is designed to support multiple users in an AIX system. A ***Call Router*** routes individual X.25 packets to the right user. The IBM 6150 X.25 Communications Support provides:

- The X.25 adapter code runs under the control of the Intel 8088 processor on the IBM PC X.25 Communications Adapter. It controls the communications link to an X.25 packet switched network and the asynchronous link to a monitoring terminal or terminal emulator. The IBM PC X.25 Communications Adapter supports X.25 levels 1, 2 and 3. X.25 support is to 1980 standards; X.25 level 2 protocols are HDLC LAP-B; X.25 level 1 protocols are provided by a synchronous X.21 bis communications port, operating at speeds of up to 19.2Kbps full duplex. Up to 20 virtual circuits can be supported concurrently. The maximum packet size is 1024 bytes (128 byte default), and window size can be 1 to 7 (2 is the default). This adapter code is loaded into the adapter by the IBM 6150 X.25 Communications Support VRM Device Driver.
- The IBM 6150 X.25 Communications Support VRM Device Driver provides a queued interface to the adapter and handles interrupts from the adapter. It is a Block I/O Device Driver and uses the services of the VRM Block I/O Device Manager for setting up paths to user processes and for buffer management. All communication with the adapter is via this component. Users of the VRM Device Driver are called Logical Link Control Processes, and may be above or below the VMI, that is, they may be VRM processes or AIX processes.
- The IBM 6150 X.25 Communications Support AIX Device Driver is a multiplexed character device driver which handles system calls for the adapter from multiple application programs and virtual interrupts from the IBM 6150 X.25 Communications Support VRM Device Driver.
- Qualified Logical Link Control (QLLC) VRM process.
- The Qualified Logical Link Control (QLLC) enables SNA-to-SNA connections over an X.25 packet switched network. It uses the IBM 6150 X.25 Communications Support VRM Device Driver to pass SNA PIUs over the X.25 packet switched network. IBM RT SNA Services provides an application programming interface to SNA LU 1, 2, 3 and 6.2 protocols and allows an application program to connect to IBM host applications or to peer hosts (LU 6.2 only) via X.25. Supported applications employing SNA Services are *Network 3270-PLUS (SNA)*, *Network RJE-PLUS (SNA)* and *Distributed Services*.
- An application program interface (API) allows you to write X.25 applications for the IBM RT C Language compiler. Routines are supplied to control the state of the X.25 link, to handle network communications and to control incoming calls. The API consists of a set of C Language functions which can be used by an application program to issue supervisor calls to the IBM 6150 X.25 Communications Support AIX Device Driver. The C function library is located in `/usr/lib/libx25.a`.
- Router daemon (`/etc/x25rtrd`). This daemon can use tables set up by *XROUTE* to route incoming calls to applications. See "Applications" on page 314 for further information about *XROUTE*.
- Receiver daemon (`/etc/x25rcvd`). This daemon handles received packets for API users.

- Supplied applications enable the user to call or accept calls from a remote destination and transfer messages and files over X.25. Multiple concurrent applications are supported.
- An asynchronous port operating at 9600 bps allows attachment of an asynchronous terminal for monitoring of X.25 traffic. The ASCII terminal monitor may be a terminal emulation session on the same IBM RT that is running the X25 product.
- Supports TCP/IP. Two systems can use the TCP/IP protocol set to communicate via the X.25 network. All TCP/IP functions that do not use TCP/IP broadcasts can be used, including *ftp*, *telnet* and *rexec*.
- The IBM 6150 X.25 Communications Support product may only be used to support the attachment of the IBM RT to public X.25 networks in countries where the adapter has been approved (homologated) for this purpose or in countries where approval is not required. No approval is usually necessary for attachment to private X.25 networks. Currently supported public X.25 networks are: TRANSPAC in France, DATEX-P in Germany, IBERPAC in Spain, DCS in Belgium, DATAPAK in the Nordic countries, DATEX-P in Austria and PSS in the UK. (ITAPAC in Italy and DATANET1 in the Netherlands are currently being homologated.)

Limitations

The following implementation limitations exist on the IBM RT and should be accounted for in any software design on X.25:

- *Network 3270-PLUS (SNA)* and *Network RJE-PLUS (SNA)* both implement a 7-bit code page.
- *Network 3270-PLUS (SNA)* and *Network RJE-PLUS* (SNA) do not support multiple PUs (physical units). They emulate only one 3x74 controller; therefore, you can only have one copy of the Network 3270-PLUS and/or one copy of Network RJE-PLUS running at a time.
- When using TCP/IP over the X.25 interface, remember that the X.25 interface does not support server commands that require network broadcast. In AIX/RT, such commands include *gated*, *routed* and *rwhod*.
- When using applications designed to operate in high speed local area networks, for example, TCP/IP and Distributed Services, you should be aware that performance will be degraded.

Applications

The *IBM 6150 X.25 Communications Support* includes application programs which allow an end user to set up nicknames for remote destinations and to send and receive conversational messages and files to and from a remote user. The remote user must be using either another IBM RT with an IBM PC X.25 Communications Adapter and the IBM 6150 X.25 Communications Support or an IBM Personal Computer, IBM Personal Computer XT, IBM Personal Computer AT or an IBM Personal System/2 Model 30 with the IBM PC X.25 Communications Adapter and the IBM PC X.25 Communications Support (Part Number 8553544, Program Number 5604-133). In the latter case, certain restrictions will apply as discussed in the section about *XTALK*.

For the sample programs to work, two AIX daemons must be running. You can start these daemons manually by invoking two programs:

/etc/x25rcvd
/etc/x25rtrd

Lines to start the daemons are included in `/etc/rc.include` but are commented out. Remove the comment if you want the programs to start automatically when the IBM RT is booted.

The applications provided are:

XCOMMS Provides a single access point to the other applications through a menu for selecting the X.25 applications. The *XCOMMS* program is provided for optional use. All applications may also be called directly from AIX.

XTALK Lets you make and receive calls or transfer files and messages. *XTALK* is not limited to users with user IDs on different IBM RTs. It also allows communication between users on the same IBM RT, if the number of digits in your Network User Address (NUA) is less than 15.

XTALK can give you direct access to public services provided via the X.25 network like BTHOSTESS and BTCLOCK in the UK. If you want to try, call

```
23421920100515                 (BTHOSTESS)
23421920100515                 (BTCLOCK).
```

You should be able to run these services from countries outside the UK.

Note: Some restrictions apply when sending messages between *XTALK* of the IBM 6150 X.25 Communications Support and the X.25 Talk program application of the IBM PC X.25 Communications Support. Code points greater than 7F are translated into the following format:

```
\<*> or \<**>
```

where "*" is a similar character from the code point range 20 to 7E. No such restrictions apply when sending messages between the X.25 *XTALK* applications on two IBM RTs.

XNAMES Maintains a personal list of X.25 network user addresses, each with its associated nickname and configuration values for use by the *XTALK* program. *XNAMES* allows you to add, change or delete nicknames, network addresses, and configuration information.

XROUTE As the IBM RT is a multi-user system, it needs some way of routing arriving calls. Calls must be routed to the most appropriate user among the users currently logged on, or rejected if an appropriate user is not available.

This routing is controlled in a ***router file*** that holds one or more entries called ***listen specifications***. A listen specification contains various items of information whose main purpose is to match an outside call with one or more of the IBM RT users.

XMONITOR This program is available only to the superuser. It is a problem-solving tool that allows the superuser to monitor the X.25 line, that is, to actually see and record the data travelling in and out of the X.25 network via the adapter. Data frames are truncated to conserve space.

To monitor the line, the superuser connects a local attach cable from the X.25 adapter to an IBM 3101 asynchronous terminal, an IBM 3161 or IBM 3151 in 3101 emulation mode, an IBM PC with an asynchronous adapter card and 3101 emulator software or to an asynchronous port in an IBM RT. (This IBM RT can be the same one as is running the X.25 applications).

After the cable is connected, the superuser runs *XMONITOR*. The program only allows the superuser to ***display*** data travelling between the adapter and the X.25 network. If a PC is being used, the data can be printed (for example, by using the ***Print Screen*** key). If you also use the IBM RT with the X.25 interface as a monitor, the monitoring can be made to appear in a separate window and may be printed or saved using normal IBM RT procedures.

The data that is displayed consists of a one-character direction identifier (**T** for frames transmitted by the IBM RT to the network and **R** for frames received from the network), followed by a string of hexadecimal values representing the first 8 bytes of each frame.

XMONITOR does not replace the normal IBM RT trace and log procedures. It may be used in addition to these. Also, *XMONITOR* is not designed to be as sophisticated as the "black-box" X.25 monitoring products that are commercially available (datascopes) but may be used as a supplement to such monitor hardware.

XMANAGE Allows the system user or superuser to examine the status of the IBM RT link to the X.25 network. *XMANAGE* also allows the IBM RT to be temporarily disconnected from and reconnected to the network.

If you want to have these applications integrated into Usability Services and/or X-Windows, you can do so by selecting "TOOLS" and from there "TOOLSUPDATE" or "MENU UPDATE" of Usability Services and X-Windows respectively. "TOOLSUPDATE" and "MENU UPDATE" allows you to create subpanels for each of the above commands. However, rather than selecting the active fields with function keys or the mouse cursor you have to work with the `Esc.`, Cursor-up, Cursor-down and `ENTER` keys.

The menus displayed in connection with the commands described above can easily be translated into other languages by modifying the following files in the `/usr/bin` directory:

- xcomms.txt
- xtalk.txt
- xmanage.txt
- xmonitor.txt
- xnames.txt
- xroute.txt.

IBM 6150 X.25 Communications Support Application Program Interface

The IBM 6150 X.25 Communications Support provides an application program interface (API) library consisting of a set of C Language functions that can be used by an application program. The library provides functions to control the state of the link, make and receive X.25 calls and transfer data across the network using permanent and switched virtual circuits (PVCs and SVCs). The API works in conjunction with the "Incoming Call Router" to pass calls to the correct application. The API provides four levels of function as follows:

1. Initialization

   This function initializes the API and must be used before issuing any other API function calls. It neither sends nor receive packets. Note that the application must issue a **setpgrp()** system call before using this function.

   Verb: x25_init

2. Network Services

   Verbs used to establish calls, transmit data, clear calls and make other normal use of the network.

   Verbs:

   x25_query
   x25_call
   x25_listen
   x25_deafen
   x25_receive
   x25_call_accept
   x25_send
   x25_call_clear
   x25_ack
   x25_interrupt
   x25_reset
   x25_pvc_alloc
   x25_pvc_free
   x25_reset_confirm

3. Counters

   As there can be several virtual calls active at the same time and each might have several packets queued, the API provides a counter facility to control the sequence of events.

   An application keeps track of conversations by using a counter. It must ask the API for a counter before opening a switched virtual circuit. Each counter assigned by the API is unique. The API adds one to the counter when an incoming packet is received on the switched virtual circuit. It subtracts one from the counter when the application receives a packet.

   Verbs:

   x25_ctr_get
   x25_ctr_remove
   x25_ctr_test
   x25_ctr_wait

4. System Management

Functions used to control the link between the IBM RT and the X.25 network. They can normally only be used by applications with system user authority.

Verbs:

x25_link_connect
x25_link_disconnect
x25_link_status
x25_link_monitor (must have superuser authority for this)

Running TCP/IP via X.25

TCP/IP in an X.25 environment is announced for VM/CMS, MVS/TSO, AIX/RT and AIX PS/2. To run TCP/IP via X.25 from an IBM RT, you must:

1. Install TCP/IP on the IBM RT and apply the updates mentioned in "Prerequisites" on page 311.
2. Edit the file /etc/net, remove the comment from the X25 entry and change the entry to fit your requirements. For example:

   ```
   x25w0:
           netaddr = 192.48.12.1
           inetlen = 576
           tranfile = /usr/lpp/x25w/tranfile
   ```

   Note: The default X.25 entry in your /etc/net file might be called "x25a0" No matter what the default is, the entry you use must match the name of the device entry. Use the *devices* command to display the names of installed devices.

   The *tranfile* can be thought of as an extension to the normal /etc/hosts file, where you specify the hosts in the network. You still have to put the IP addresses and aliases in the /etc/hosts file. In *tranfile*, you translate the IP address to an X.25 address. It must be used if you want to access the X.25 PDN network. Example of entries in the *tranfile*:

   ```
   192.48.12.1     3106001984
   192.48.12.6     3106008301
   ```

   If the *tranfile* keyword is not used, the X.25 interface is configured for X.25 DDN addresses. For an extended format of *tranfile* and more details about customizing X.25 and TCP/IP together, see the description in "Customizing Host sys1 for X.25" on page 209.
3. Be sure that sh /etc/rc.tcpip is not commented out in the /etc/rc.include file.

As previously mentioned, the X.25 interface does not support server commands like *gated*, *routed* and *rwhod*, so do not use these facilities.

Note: An important thing to know is that your system must be configured to match the capability of doing ***packet size negotiation*** of your X.25 network. See "Customizing Host sys1 for X.25" on page 209 for details.

Running SNA via X.25

The *IBM 6150 X.25 Communications Support* includes support for IBM QLLC (Qualified Logical Link Control) to enable SNA-to-SNA connections via an X.25 network. In this way, support for AIX/RT SNA Services is provided. See *Attaching SNA Nodes to Packet-Switched Data Networks, General Information Manual.*

SNA applications that operate via an X.25 network are:

- Programs written for the IBM RT SNA Services interface
- Distributed Services
- Network 3270-PLUS (SNA) and Network RJE-PLUS (SNA).

When you want to run SNA via X.25, you must install a data link device called ***qllc0*** with the *devices* command. You do not have to modify the device parameters.

Programs written for the IBM RT SNA Services interface

Programs written for LU 1, 2, 3 and 6.2 communications using the IBM RT SNA Services interface will run across X.25 connections. Examples of programs using LU 6.2 communications are the *snaftp* and *RTRT* programs described in "APPC/LU 6.2 Communication" on page 81.

These programs have been tested between two IBM RTs connected via X.25. Sample SNA Services profiles for a link between two IBM RTs over X.25 are provided in "X.25 Connection, RT PC - RT PC (LU 6.2)" on page 398.

Distributed Services over X.25

In order to run distributed services between two IBM RTs over an X.25 link, use the following information to set up a test environment:

1. Install SNA Services and Distributed Services
2. Run *ndtable* and add the following entry:

   ```
   Remote name:  Name of remote system, for example "system2"
   nodeid     :  Nodeid of remote station, for example "20911C9B"
   ```

   Leave the rest of the entries as default. The "Ethernet" entry will be changed later when configuring SNA Services using *snaconfig*.
3. Run *ugtable* and add the following entries:

   ```
   First entry:
           User/Group: User
           Local ID  : 0
           Name      : root
           Net ID    : 0
           Net ID    : 0
           Nickname  : *

   Second entry:
           User/Group: Group
           Local ID  : 0
           Name      : system
           Net ID    : 0
           Net ID    : 0
           Nickname  : *
   ```

4. Edit the file */etc/rc.ds*. Comment out the starting of "EDEFAULT" and add a start statement for X25DS. For example: `start /attachment/X25DS&`.

5. Run the SNA Services command *snaconfig*. Copy the "QDEFAULT" ***Physical Link Profile*** and call it "X25P". In this profile, specify your local X.25 NUA.

6. Using *snaconfig*, modify the ***Attachment Profile*** that was created using the *ndtable* command. The name of the Attachment Profile should match the nodeid name of the remote station, for example 20911C9B. Set the Physical Link Profile to "X25P", select "Call" and "Switched" and enter the X.25 NUA of the remote station.

   If you have other applications using LU 6.2 between the two nodes (for example the *snaftp* program from this publication), remember to let the Connection Profile for the *snaftp* connection point to this Distributed Services Attachment Profile. If this is not done, the *snaftp* program will try to start a second attachment between the same two nodes, and this is not possible. You will get the error message: "***Device already in use***".

7. Take a copy of the "XDEFAULT" Attachment Profile and call it "X25DS". In this profile, set the Physical Link Profile to "X25P".

8. It is recommended that you raise the timers in the Logical Link Profile to its maximum. If not, you can experience problems when, for example, copying files from the remote IBM RT to the local IBM RT. The SNA attachment and connection might even become inactive.

You should now be able to run the script `/etc/rc.ds` that starts SNA Services and Distributed Services. When this is done, issue normal distributed services mounts.

Network 3270-PLUS (SNA) via X.25

To run the Network 3270-PLUS emulation software, you will have to modify the Network 3270-PLUS profile: `/usr/lpp/r/comm/sna/3270/profiles/sc3270.prof`, according to the LU settings you have chosen. To start the emulation:

1. If you have modified Network 3270-PLUS profiles, parse the configuration with the command: `s3270admin parse configuration`.
2. Start SNA.
3. Start Attachment / Connection(s).
4. Start the 3x74 cluster controller emulation with `s3270admin install`.
5. Invoke the terminal emulation with either `s3270 menu` or `s3270 start`.

If the SNA profiles are properly customized, there is no difference in running Network 3270-PLUS, whether you are connected to your host via X.25, SDLC or Token-Ring. Hosts, in this sense, are products implementing SNA type 4 or 5 node capabilities, for example, an IBM System/370 system running VM, MVS or DOS/VSE with corresponding communication controllers. However, systems like IBM S/36, IBM S/38 and IBM AS/400 also support remote 3x74 cluster controllers attached via SDLC lines. The IBM AS/400 also provides SNA support via Token-Ring and X.25 QLLC.

The charging scheme of X.25 packed switched networks is different from that of a switched or leased line: you will be charged under X.25 conditions only if you generate data. So terminal emulation applications like Network 3270-PLUS

(SNA) are quite suitable for use over an X.25 packed switched network, because even if you keep a host session open all day long, you will be charged only when you press the ENTER key or one of the function keys.

See "Customizing Network 3270-PLUS" on page 137 for further information on how to set up *Network 3270-PLUS*.

SNA Services Profiles

The following section describes what must be set up in SNA Services to be able to run the product over X.25. Only the profiles related to X.25 are described. Sample VTAM definitions are provided in "X.25 Connection to SNA System/370 Host using 3725" on page 431.

Use the *snaconfig* program to setup the SNA Services profiles.

Attachment Profile

logical_link_type	QLLC, which is the data link between SNA Services and X.25.
station_type	SECONDARY, when using LU type 1, 2 and 3 emulations.
physical_link_type	X.25
call_type	CALL
autolisten	NO
virtual_circuit_type	"SWITCHED" if you are using SVCs and "PERMANENT" if you are using PVCs.
remote_station_X.25_address	X.25 Network user address (NUA) of the remote station.

Control Point Profile

xid_node_id	Concatenated IDBLK and IDNUM values. If your VTAM listing reads IDBLK=234 and IDNUM=B6789, the XID Node ID is 234B6789.

Physical Data Link Profile

local_X.25_network_address	X.25 Network user address (NUA) of your local system. In the case of a PVC Connection, the first Digit is a "P".

Logical Link Profile

station_type	SECONDARY

Installation hints

X.25 Local Network Address

The X.25 local network address is the Network User Address (NUA) you get from your PTT company, and for which you were prompted in connection with the *installp* procedure. Keep in mind, that an application on top of SNA does not recognize this number unless you have entered the same NUA number in the Physical Data Link Profile of SNA Services.

The NUA is configured into AIX when you install *IBM 6150 X.25 Communications Support* with the *installp* command. If, for some reason, you have to change your NUA, you must edit the /etc/ddi/x25w.ddi file. This file contains the NUA. The NUA is loaded into the X.25 adapter when the IBM RT is booted or when running the commands:

```
vrmconfig -d x25w0
vrmconfig -a x25w0
```

The X.25 adapter description

The following is an example of the device description for the X.25 adapter (***x25w0***):

```
                                     Current  Possible
Name      Description                Choice   Choices

biopa     First I/O port address     AA8      AA8, AAC
Frwnsiz   Frame window size          7        1 - 7
Netopt    Network Dependent Options  0        00-FF
N2count   N2 count (re-tx)           20       1 - 127
Numpvcs   No. of PVC's               0        0 - 20
Noicsvc   No. incoming SVC's         0        0 - 20
Noogsvc   No. outgoing SVC's         0        0 - 20
N2wysvc   No. 2-way SVC's            20       0 - 20
FicSVCn   First incoming SVC channel 1        1 - 4095
FogSVCn   First outgoing SVC channel 1        1 - 4095
F2ySVCn   First 2-way SVC channel no. 1       1 - 4095
T1timer   T1 timeout (frame)         30       20 - 60
SVCwnsz   SVC Window size            2        1 - 7
Iwsdflt   PVC default in window size  2       1 - 7
Owsdflt   PVC default out window size 2       1 - 7
Ipsdflt   PVC default in packet size  128     16,32,64,128,256,512,1024
Opsdflt   PVC default out packet size 128     16,32,64,128,256,512,1024
Resdflt   PVC default reset code     0        00 - FF
```

```
                                     Current  Possible
Name      Description                Choice   Choices

sn        Slot number                6        1 - 8
il1       1st interrupt level number 3        3, 4, 9
lobibp    Length of buffers in pool  1280     320 - 65532 by 4
mnoal     Maximum number of LLCs     3        1 - 5
mnonid    Maximum number of net IDs  4        1 - 255
nidd      Network ID displacement    0        0 - 255
nidl      Network ID length          2        1 - 255
nobibp    Number of buffers in pool  100      1 - 65535
nobodr    No. of buffers on device ring 30    1 - 255
norbosr   No. SLIH ring receive buffers 30    1 - 255
rdto      Receive data transfer offset 0      0 - 255
srbt      SLIH ring buffer threshold 4        1 - 255
```

Note: If you need to change the ***Netopt*** option and this option does not show up using the *devices* command, you should check your /etc/ddi/x25w.ddi file. The last entries in this profile concern the ***Netopt*** options. You should make sure that there is a blank line between the last two entries (the ***F2ySVCn*** and ***Netopt*** options). Example:

```
F2ySVCn:
        display=true
        required=true
        vtype=3
        range=1,4095,1
        type=I
<This must be a blank line>
Netopt:
        display=true
        required=true
        vtype=3
        range=0,FF,1
        type=H
```

Reference Publications for This Chapter

Publications relevant for the use of X.25 are:

IBM RT X.25 Communications Support User's Guide, SC33-0630
IBM RT X.25 Communications Support Programmer's Reference, SC33-0631
IBM RT SNA Services Guide and Reference, SC23-2009
Attaching SNA Nodes to Packed-Switched Data Networks, GA27-3345
Network 3270-PLUS (SNA) User's Guide, SC23-0825
Network RJE-PLUS (SNA) User's Guide, SC23-0828

Appendix A. Obtaining a Software Softcopy

IBM employees can request source for all sample programs presented in this publication from the IBM internal use RTTOOLS repository. Source may be given to customers who are licensed users of IBM's AIX products.

To obtain the package of files (which are stored in *tar* format) follow the following procedure:

1. Logon to your local VM machine.
2. Type **`tools sendto yktvmv tools rttools get itsccoms package`** at the CMS prompt.
3. Press the ENTER key to send the request.
4. The requested package of files will be sent to your CMS reader. Files with a filetype of ***tarbin*** should be downloaded to an AIX system and "un-tar'ed". Files with other filetypes are text files that can be printed from VM or downloaded with ASCII/EBCDIC conversion and printed on an AIX system.

Disclaimer

The programs that are thus downloaded have not been submitted to any formal IBM test and are distributed on an "as is" basis without any warranty either expressed or implied.

Appendix B. LU 6.2 Sample Programs

IBM RT Sample Programs

IBM RT File Transfer Program (snaftp)

The following is a listing of the program to ***initiate*** a file transfer from a local IBM RT. The listing below contains only the main line code; other parts of the program are listed as separate programs in "IBM RT Sub-programs for snaftp and RTRT" on page 343.

```
/*****************************************************************************/
/*                                                                           */
/*   PROGRAM NAME:      snaftp (Source: snaftp.c)                            */
/*                                                                           */
/*   DESCRIPTIVE NAME: APPC Local Transaction Program for File Transfer      */
/*                      between an RT PC running the Advanced Interactive    */
/*                      Executive Operating System (AIX) and a variety of    */
/*                      remote systems.                                      */
/*                                                                           */
/*   COPYRIGHT: XXXXX  (C) COPYRIGHT IBM CORP. 1989                          */
/*              ALL RIGHTS RESERVED                                          */
/*                                                                           */
/*   N-O-T-E:   This program is an AIX program intended to be invoked from  */
/*              the AIX command line to transfer files to and from remote    */
/*              systems using SNA LU 6.2 communication.  For the program     */
/*              to function properly, SNA Profiles must be properly cus-     */
/*              tomized on the local as well as on the remote system.        */
/*                                                                           */
/*   STATUS:    As released with Communications "Cookbook" from the          */
/*              International Technical Support Center, Austin in the        */
/*              spring of 1989.  Programmers:                                */
/*                                                                           */
/*                   Jan Norbäck, IBM Sweden.                                */
/*                   Niels Christiansen, ITSC Austin.                        */
/*                                                                           */
/*   FUNCTION = When invoked, takes command line parameters to specify       */
/*              file transfer function and remote host.  Initiates con-      */
/*              versation with the remote host and request the remote to     */
/*              receive or send a named file.  For cammand line options      */
/*              see the source program 'arg.c' or the "Cookbook".            */
/*                                                                           */
/*              The program invokes the following SNA subroutines:           */
/*                                                                           */
/*                 snaopen      Open connection                              */
/*                 snalloc      Allocate conversation                        */
/*                 snaread      Read from SNA                                */
/*                 snawrite     Write to SNA                                 */
/*                 snadeal      Deallocate conversation                      */
/*                 snaclse      Close connection                             */
/*                                                                           */
/*              The program uses the following external modules:             */
/*                                                                           */
/*                 arg.c        Command line parser                          */
/*                 sna.c        Interface to SNA subroutines                 */
/*                 interpret.c  Interpreter of Control Messages              */
/*                 control.c    Builder of Control Messages                  */
/*                 fileio.c     File input/output routines                   */
/*                 defs.h       Common declarations                          */
/*                                                                           */
/*   REMARKS:   Extensive logging to sysout takes place if all programs      */
/*              are compiled with the "-d DEBUG" option:                     */
/*                                                                           */
/*                 cc -o snaftp snaftp.c -d DEBUG -lsna                      */
/*                                                                           */
```

```
/***************************************************************************/
#include "defs.h"                              /* common declarations         */


char                local_file[NMSZ];          /* local         file name      */
char                remote_file[NMSZ];         /* remote        file name      */
struct data_rec     buffer;                    /* send buffer                  */
struct data_rec     *bp;                       /* pointer to data record       */
struct ctrl_rec     cbuffer;                   /* control record buffer        */
char                *cp;                       /* pointer to character array   */
char                tbuffer[N_BYTES];          /* buffer for text file         */
char                Host[20];                  /* Host type for display        */
unsigned int        *pa;                       /* pointer to int               */
int                 n_read;                    /* bytes read                   */
int                 n_sent;                    /* bytes sent                   */
int                 count;                     /* work                         */
int                 lm;                        /* length of message to send    */
unsigned            file_siz;                  /* file size                    */
unsigned            rec_len;                   /* record length for receive    */
int                 more = YES;                /* loop control variable        */
FILE                *fp = NULL;                /* local file pointer           */
int                 error_found = NO;          /* error found indicator        */
int                 rc;                        /* return code                  */
char                err_msg[64];               /* Error message field          */


/* ---------------------------------------------------------------------- */
/*                               MAIN CODE                                */
/* ---------------------------------------------------------------------- */


main(argc, argv)
int argc;
char *argv[];
{
   if (arg_check(argc,argv) == ERROR)        /* Parse the command line       */
   {
      exit(9);                               /* Terminate if error found     */
   }
   open_connection();                        /* Open connection              */
   allocate_conversation();                  /* Allocate conversation        */


                                             /* **************************** */
   while( more == YES )                      /* Main loop (currently unused  */
                                             /* **************************** */
   {
      more = NO;                             /* Ensure only one time through */
      if( soption == 1) we_do_send();        /* Call SEND or RECEIVE function*/
      else we_do_receive();                  /* as determined by command line*/
      if(error_found == YES) break;
   }


   deallocate();                             /* Deallocate conversation      */
   close_connection();                       /* and let SNA close connection */
   if (fp!=NULL) close_local_file();         /* If local file open, close it */
}  /* end main */


/* ---------------------------------------------------------------------- */
/*              Second level function for SENDING a file.                 */
/*              Check the Communications "Cookbook" for logic.            */
/* ---------------------------------------------------------------------- */


we_do_send()
{
   printf("Sending local file \"%s\" to %s host as file \"%s\"\n",
      local_file,Host,remote_file);
   if (make_send_control_message(&cbuffer) == ERROR) return;
   if (open_local_file() == ERROR)
   {
      fprintf(stderr,"%s",err_msg);
      return;
   }
   if (send_control() == ERROR) return;
   if (receive_message() == ERROR) return;
   if (interpret_send_message(&cbuffer,n_read) == ERROR) return;
   if (send_data() == ERROR) return;
   if (receive_message() == ERROR) return;
   if (interpret_send_message(&cbuffer,n_read) == ERROR) return;
} /* end we_do_send */


/* ---------------------------------------------------------------------- */
/*              Second level function for RECEIVING a file.              */
/*              Check the Communications "Cookbook" for logic.            */
/* ---------------------------------------------------------------------- */


we_do_receive()
{
   printf("Receiving file \"%s\" from %s host as local file \"%s\"\n",
           remote_file,Host,local_file);
   if (make_receive_control_message(&cbuffer) == ERROR) return;
   if (open_local_file() == ERROR)
   {
      make_message(&cbuffer,err_msg,'E');
      fprintf(stderr,"%s",err_msg);
      return;
   }
   if (send_control() == ERROR) return;
   if (receive_message() == ERROR) return;
   file_siz = cbuffer.il;
   if (interpret_receive_message(&cbuffer,n_read) == ERROR) return;
   if (make_message(&cbuffer,"Okay, go ahead\n",'O') == ERROR) return;
   receive_file();
   if(error_found == YES) return;
   if (make_message(&cbuffer,"File received\n",'O') == ERROR) return;
} /* end we_do_receive */


/* ---------------------------------------------------------------------- */
/*              Lower level functions for sending a file                  */
/* ---------------------------------------------------------------------- */


int send_data()                             /* Common function for all sends*/
{
   if((host_type == 'V') || (host_type == 'T') || (host_type == '4'))
      sen_data_ebcdic();                    /* Select appropriate function  */
   else send_data_ascii();                  /* to do the actual sending.    */
   close_local_file();                      /* Close the local file and set */
   fp = NULL;                               /* file pointer to NULL.        */
   if (error_found == YES) return(ERROR);
   else return(GOOD);
}


/* ---------------------------------------------------------------------- */
/*              Send a data file to an EBCDIC host                        */
/* ---------------------------------------------------------------------- */


sen_data_ebcdic()
{
   int     i;
   char    *ptr;


#ifdef DEBUG
   printf("send_data_ebcdic\n");
#endif
                                            /* =========================== */
   if (aoption == 1)                        /* Send TEXT to EBCDIC host    */
                                            /* =========================== */
   {
      error_found=YES;                      /* Nasty trick, eh?            */
      tbuffer[N_BYTES -1] = '\0';           /* terminate by null.          */
```

```
                                          /* ---------------------------- */
      if (host_type=='4')                 /*        Text to AS/400        */
      {                                   /* ---------------------------- */
         memset(tbuffer,' ',N_BYTES);     /* Clear buffer to blanks and   */
         tbuffer[N_BYTES -1] = '\0';      /* terminate by null.           */
         while ((ptr = fgets(tbuffer,lrecl,fp)) != NULL)
         {
            error_found=NO;               /* Second half of nasty trick   */
            tbuffer[strlen(tbuffer)] = ' '; /* Convert any terminating null */
            convert(tbuffer,0,lrecl,NULL); /* Convert text to EBCDIC      */
            n_sent = send_sna(tbuffer, lrecl);  /* Send record to AS/400  */
            if (error_found == YES)
            {
               fprintf(stderr,"%s",err_msg);
               send_error();
               break;
            }
            memset(tbuffer,' ',N_BYTES);  /* Clear buffer to blanks and   */
            tbuffer[N_BYTES -1] = '\0';   /* terminate by null.           */
         }
      }                                   /* ---------------------------- */
      else                                /*        Text to S/370         */
      {                                   /* ---------------------------- */
         memset(bp->dpart,'\0',lrecl);    /* Clear buffer to nulls        */
         bp = (struct data_rec *) tbuffer; /* Load data record pointer    */
         while ((ptr = fgets(bp->dpart,lrecl,fp)) != NULL)
         {
            error_found=NO;               /* Second half of nasty trick   */
            i = strlen(bp->dpart);        /* Get data length              */
            if (bp->dpart[i-1] == '\n')   /* If newline read ...          */
            {
               bp->dpart[i-1] = '\0';     /* ... make it a NULL           */
               i--;                       /* ... and adjust length        */
            }
            convert(bp->dpart,0,lrecl,NULL);    /* Convert text to EBCDIC  */
            if(voption != 1) bp->dlen = lrecl;  /* Insert length depending */
            else bp->dlen = i;                  /* on record format        */
            n_sent = send_sna(bp,bp->dlen+D_FIELD); /* Send record to S/370 */
            if (error_found == YES)
            {
               fprintf(stderr,"%s",err_msg);
               send_error();
               break;
            }
            memset(bp->dpart,'\0',lrecl); /* Clear buffer to nulls        */
         }
      }
   }                                      /* ============================ */
   else                                   /* Send BINARY to EBCDIC host   */
   {                                      /* ============================ */
      error_found=YES;
                                          /* ---------------------------- */
      if (host_type=='4')                 /*       Binary to AS/400       */
      {                                   /* ---------------------------- */
         memset(&buffer,' ',lrecl);       /* Clear buffer to blanks       */
         while ((count = file_read(&buffer,lrecl)) > 0)
         {
#ifdef DEBUG
            fprintf(stdout,"Byte count read %d\n",count);
#endif
            error_found=NO;
            n_sent = send_sna(&buffer, lrecl);
            if (error_found == YES)
            {
               fprintf(stderr,"%s",err_msg);
               send_error();
               break;
            }
            memset(&buffer,' ',N_BYTES);
         }
      }                                   /* ---------------------------- */
      else                                /*       Binary to S/370        */
      {                                   /* ---------------------------- */
         bp = (struct data_rec *) tbuffer; /* Load data record pointer    */
         memset(bp->dpart,'\0',lrecl);    /* Clear buffer to nulls        */
         while ((count = file_read(buffer.dpart,lrecl) ) > 0)
         {
#ifdef DEBUG
            fprintf(stdout,"Byte count read %d\n",count);
#endif
            error_found=NO;
            if(voption != 1) buffer.dlen = lrecl;  /* Set length depending  */
            else buffer.dlen = count;              /* on record format      */
            n_sent = send_sna(&buffer,buffer.dlen + sizeof(buffer.dlen));
            if (error_found == YES)
            {
               fprintf(stderr,"%s",err_msg);
               send_error();
               break;
            }
            memset(bp->dpart,'\0',lrecl);   /* Clear buffer to nulls        */
         }
      }
   }
} /* end sen_data_ebcdic */


/* ---------------------------------------------------------------------- */
/*             Send a data file to an ASCII host                          */
/* ---------------------------------------------------------------------- */


send_data_ascii()
{
   int     i;
   char    *ptr;


   if ((aoption == 1) && ((host_type == 'O') || (host_type == 'D')))
                                          /* =========================== */
                                          /* Send TEXT to Intel based PC  */
   {                                      /* =========================== */
#ifdef DEBUG
      printf("send_ascii to Intel based system\n");
#endif
      error_found=YES;
      while ((ptr = fgets(buffer.dpart,N_BYTES-D_FIELD,fp)) != NULL)
      {                                   /* Read until newline character */
         error_found=NO;
         count = strlen(buffer.dpart) + 1;       /* Increase length by one  */
         buffer.dpart[count] = '\0';             /* to accomodate PC+DOS    */
         buffer.dpart[count-1] = 0xa;            /* and OS/2 standards for  */
         buffer.dpart[count-2] = 0xd;            /* text files.             */
         for (i=0;i<count-2;i++)                 /* If CRs inside string    */
            if (buffer.dpart[i] == '\r')         /* remove them from string */
            {
               memcpy(&buffer.dpart[i],&buffer.dpart[i+1],count-i+1);
               count--;
            }
         buffer.dlen = count;                    /* Move length to block    */
         field_conv(&buffer.dlen);               /* Convert length to Intel */
         n_sent = send_sna(&buffer, count+D_FIELD);
         if (error_found == YES)
         {
            fprintf(stderr,"%s",err_msg);
            send_error();
            break;
         }
      }
   }
   else                                   /* =========================== */
   {                                      /* All but text to Intel PC     */
                                          /* =========================== */
#ifdef DEBUG
      printf("send_ascii to non-Intel based host\n");
#endif
      error_found=YES;
      memset(buffer.dpart,' ',N_BYTES);     /* Reset buffer to blanks       */
      while ((count = file_read(buffer.dpart,N_BYTES - sizeof(buffer.dlen))) > 0)
      {                                 /* Read up to full buffer size  */
#ifdef DEBUG
         fprintf(stdout,"Byte count read %d\n",count);
#endif
         buffer.dlen = count;           /* Insert buffer length in msg  */
         if ((host_type == 'O') || (host_type == 'D'))
            field_conv(&buffer.dlen);   /* Convert length, as required  */
         error_found=NO;
#ifdef DEBUG
         printf("Try to send %d bytes\n",count + sizeof(buffer.dlen));
#endif
         n_sent = send_sna(&buffer, count + sizeof(buffer.dlen));
         if (error_found == YES)
         {
            fprintf(stderr,"%s",err_msg);
            send_error();
            break;
         }
         memset(buffer.dpart,' ',N_BYTES);  /* Reset buffer to blanks       */
      }
   }
} /* end send_data_ascii */


/* ------------------------------------------------------------------------ */
/*             Lower level functions for receiving a file                    */
/* ------------------------------------------------------------------------ */


int receive_file()
{
#ifdef DEBUG
printf("Trying to receive_file of %d bytes\n",file_siz);
#endif
   if ((host_type == '4') && (aoption == 1))
      n_read = sna_receive(tbuffer,rec_len);    /* Initial read from AS/400 */
   else n_read = sna_receive(&buffer,N_BYTES);  /* Initial read from others */
   if (error_found == YES)
   {
      fprintf(stderr,"%s",err_msg);
      send_error();
      return(1);
   }
   if((host_type == 'V') || (host_type == 'T') || (host_type == '4'))
      receive_file_ebcdic();            /* Let appropriate function do  */
   else receive_file_ascii();           /* the remainder of receives    */
   close_local_file();                  /* Close the received local file*/
   fp = NULL;                           /* ..and reset file handle      */
   if (error_found == YES) return(ERROR);
   else return(GOOD);
}


/* ------------------------------------------------------------------------ */
/*              Receiving file from an EBCDIC host                           */
/* ------------------------------------------------------------------------ */


receive_file_ebcdic()
{
   int tmp_count=0;
   int one_write=0;
   int i;


#ifdef DEBUG
   printf("receive_file_ebcdic\n");
#endif
                                        /* ============================ */
   if (aoption == 1)                    /* Get TEXT from an EBCDIC host */
   {                                    /* ============================ */
                                        /* ---------------------------- */
      if (host_type=='4')               /*      Text from AS/400        */
      {                                 /* ---------------------------- */
         while (n_read > 0)
      {
         convert(tbuffer,1,n_read,NULL); /* Convert buffer to ASCII      */
         file_siz = file_siz - n_read;   /* Calculate remainder to read  */
         i = rec_len - 1;                /* Point at last character      */
         tbuffer[i+1] = '\0';            /* and insert null to terminate */
         while ((tbuffer[i]<0x21) && (i>=0)) /* Now scan from end of     */
         {                                   /* record to translate any  */
            tbuffer[i] = '\0';               /* trailing blanks to nulls */
            i--;
         }
         i = strlen(tbuffer);            /* Get new string length        */
         tbuffer[i] = '\n';              /* and insert trailing newline  */
         file_write(tbuffer,i+1);        /* Then write to the local file */
         n_read = 0;
         if (file_siz > 0) n_read = sna_receive(tbuffer,rec_len);
         if (error_found == YES)
         {
            fprintf(stderr,"%s",err_msg);
            send_error;
            break;
         }
      }
   }
                                         /* ---------------------------- */
   else                                  /*      Text from S/370         */
   {                                     /* ---------------------------- */


         /* Routine is not prepared to receive multiple */
         /* blocks returned from one call to snaread.   */


      while (n_read > 0)
      {
         convert(buffer.dpart,1,n_read,NULL); /* Convert buffer to ASCII */
         count = buffer.dlen;            /* Get data length              */
         file_siz = file_siz - count;    /* Calculate remainder to read  */
         i = count - 1;                  /* Point at last character      */
         buffer.dpart[i+1] = '\0';       /* and insert null to terminate */
         while ((buffer.dpart[i]<0x21) && (i>=0))
         {                               /* Now scan from end of data    */
            buffer.dpart[i] = '\0';      /* record to translate any      */
            i--;                         /* trailing blanks to nulls     */
         }
         i = strlen(buffer.dpart);       /* Get new string length        */
         buffer.dpart[i] = '\n';         /* and insert trailing newline  */
         file_write(&buffer,i+1);        /* Then write to the local file */
         n_read = 0;
         if (file_siz > 0) n_read = sna_receive(&buffer,N_BYTES);
         if (error_found == YES)
         {
            fprintf(stderr,"%s",err_msg);
            send_error;
            break;
         }
      }
   }
}
                                         /* ============================ */
else                                     /* Get BINARY from EBCDIC host  */
{                                        /* ============================ */
   if (host_type=='4')
   {                                     /* ---------------------------- */
      while( n_read > 0)                 /*      Binary from AS/400      */
      {                                  /* ---------------------------- */
         file_siz = file_siz - n_read;   /* Adjust remainder to read     */
         file_write(&buffer,n_read);     /* Write to local file          */
         n_read = 0;
         if (file_siz > 0) n_read = sna_receive(&buffer,N_BYTES);
         if (error_found == YES)
         {
            fprintf(stderr,"%s",err_msg);
            send_error;
            break;
         }
         }
      }
      else                                       /* ---------------------------- */
      {                                          /*      Binary from S/370       */
         while( n_read > 0)                      /* ---------------------------- */


               /* Routine is not prepared to receive multiple */
               /* blocks returned from one call to snaread.   */


         {
            bp = &buffer;                 /* Load pointer to data record  */
            cp = (char *) bp;             /* Load character pointer       */
            while( (tmp_count < n_read ) && (error_found != YES) )
            {
               count = file_write(cp+4,bp->dlen);  /* Write to local file   */
               if (count != bp->dlen)
               {
                  error_found = YES;
                  break;
               }
               /* These statements adjust pointers to next logical chunk    */
               tmp_count = tmp_count + bp->dlen + sizeof(buffer.dlen);
               one_write = one_write + bp->dlen;
               cp = cp + bp->dlen + sizeof(buffer.dlen);
               bp = (struct data_rec *) cp;
            }
            n_read = 0;
            file_siz = file_siz - one_write;  /* Adjust remainder to read   */
            tmp_count = 0;
            one_write = 0;
#ifdef DEBUG
            printf("Remains to be received: %d\n",file_siz);
#endif
            if (file_siz > 0) n_read = sna_receive(&buffer,N_BYTES);
            if (error_found == YES)
            {
               fprintf(stderr,"%s",err_msg);
               send_error;
               break;
            }
         }
      }
   }
} /* end receive_file_ebcdic */


/* ---------------------------------------------------------------------- */
/*              Receiving file from an ASCII host                         */
/* ---------------------------------------------------------------------- */


receive_file_ascii()
{
   int sav_count=0;
   int tmp_count=0;
   int one_write=0;
   int intel;
   int n;


#ifdef DEBUG
   printf("Receive ASCII file of %d bytes\n",file_siz);
#endif
   while( n_read > 0)
   {
      bp = &buffer;                        /* Load data record pointer     */
      cp = (char *) bp;                    /* Load character array pointer */
      while( (tmp_count < n_read ) && (error_found != YES) )
      {
#ifdef DEBUG
      printf("bp->dlen = %d \n",bp->dlen);
#endif
         intel = bp->dlen;                 /* Be ready for Intel conversion*/
         sav_count = intel;              /* and save original count      */
         if ((host_type=='O') || (host_type=='D'))
         {
            field_conv(&intel);          /* Convert from Intel if req'd  */
            sav_count = intel;           /* and save as original count   */
            if (aoption == 1)            /* If text file from Intel syst.*/
            {
               for (n=0;n<intel;n++)             /* Scan  received       */
               {                                 /* string for any CR    */
                  if (bp->dpart[n]==0xd)         /* and remove the CRs   */
                  {                              /* from the string      */
                     if (intel-n > 0)
                        memcpy(&bp->dpart[n],&bp->dpart[n+1],intel-n);
                     intel--;
                     n++;
                  }
                  if (bp->dpart[n]==0x1a) intel--;   /* remove EOF mark  */
               }
            }
         }
         count = file_write(cp+4,intel);   /* Write to local file        */
#ifdef DEBUG
      printf("Write routine says %d bytes written\n",count);
#endif
         if (count != intel)
         {
            error_found = YES;
            break;
         }
         /* These statements adjust pointers to next logical chunk */
         tmp_count = tmp_count + sav_count + sizeof(buffer.dlen);
         one_write = one_write + sav_count;
         cp = cp + sav_count + sizeof(buffer.dlen);
         bp = (struct data_rec *) cp;
      }
      n_read = 0;
      file_siz = file_siz - one_write;      /* Adjust remainer to receive   */
      tmp_count = 0;
      one_write = 0;
#ifdef DEBUG
      printf("Remains to be received: %d\n",file_siz);
#endif
      if (file_siz > 0) n_read = sna_receive(&buffer,N_BYTES);
      if (error_found == YES)
      {
         fprintf(stderr,"%s",err_msg);
         send_error;
         break;
      }
   }
} /* end receive_file_ascii */


/* ---------------------------------------------------------------------- */
/*              Lower level functions to send a control message           */
/* ---------------------------------------------------------------------- */


int send_control()
{
#ifdef DEBUG
printf("send_control\n");
#endif
   n_sent = send_sna(&cbuffer,lm);          /* Send control message         */
   if (error_found == YES)
   {
      fprintf(stderr,"Unable to send control message\n");
      fprintf(stderr,"%s",err_msg);
      send_error();
      return(ERROR);
   }
   return(GOOD);
}
```

```
/* ---------------------------------------------------------------------- */
/*             Lower level functions to receive a control message         */
/* ---------------------------------------------------------------------- */


receive_message()
{
#ifdef DEBUG
printf("receive_message\n");
#endif
   memset(&cbuffer,'\0',sizeof(cbuffer));   /* Reset control record buffer  */
   n_read = sna_receive(&cbuffer,N_BYTES);  /* Read whatever comes          */
   if ((error_found == YES) || (n_read == 0))
   {
      fprintf(stderr,"Could not receive expected control message\n");
      fprintf(stderr,"%s",err_msg);
      send_error();
      return(ERROR);
   }
   if (cbuffer.lm < sizeof(cbuffer.msg)) cbuffer.msg[cbuffer.lm] = '\0';
   else cbuffer.msg[strlen(cbuffer.msg)-1] = '\0';   /* Adjust message text */
   return(GOOD);
}
```

IBM RT Remote Transaction Program (RTRT)

The following is a listing of the remote transaction program for file transfer between IBM RTs. The listing below contains only the main line code; other parts of the program are listed as separate programs in "IBM RT Sub-programs for snaftp and RTRT" on page 343.

```
/****************************************************************************/
/*                                                                          */
/*   PROGRAM NAME:     RTRT (Source: rtrt.c)                                */
/*                                                                          */
/*   DESCRIPTIVE NAME: APPC Remote Transaction Program for File Transfer    */
/*                     between two RT PCs running the Advanced Interactive  */
/*                     Executive Operating System (AIX).                    */
/*                                                                          */
/*   COPYRIGHT: XXXXX  (C) COPYRIGHT IBM CORP. 1989                         */
/*              ALL RIGHTS RESERVED                                         */
/*                                                                          */
/*   N-O-T-E:   This program is an AIX program intended to be invoked by    */
/*              an incoming allocate, using SNA LU 6.2 communication.       */
/*              For the program to function properly, SNA Profiles must     */
/*              be customized on the local and the remote system.           */
/*                                                                          */
/*   STATUS:    As released with Communications "Cookbook" from the         */
/*              International Technical Support Center, Austin in the       */
/*              spring of 1989.  Programmers:                               */
/*                                                                          */
/*                   Jan Norbäck, IBM Sweden.                               */
/*                   Niels Christiansen, ITSC Austin.                       */
/*                                                                          */
/*   FUNCTION = When invoked, processes a request message from the snaftp   */
/*              program on a remote RT PC and subsequently performs the     */
/*              requested send or receive of a file.                        */
/*                                                                          */
/*              The program invokes the following SNA subroutines:          */
/*                                                                          */
/*                 snaopen      Open connection                             */
/*                 snalloc      For receive_allocate                        */
/*                 snaread      Read from SNA                               */
/*                 snawrite     Write to SNA                                */
/*                 snadeal      Deallocate conversation                     */
/*                 snaclse      Close connection                            */
/*                                                                          */
/*              The program uses the following external modules:           */
/*                                                                          */
/*                 sna.c        Interface to SNA subroutines                */
/*                 fileio.c     File input/output routines                  */
/*                 defs.h       Common declarations                         */
/*                                                                          */
/*   REMARKS:   Extensive logging takes place if all programs are compiled  */
/*              with the "-d DEBUG" option.  Logging is done to the file    */
/*              or device specified in the SNA Local Transaction Program    */
/*              Profile.  To compile with logging, use:                     */
/*                                                                          */
/*                 cc -o RTRT rtrt.c -d DEBUG -lsna                         */
/*                                                                          */
/****************************************************************************/
#include "defs.h"                          /* common declarations          */

char              connection[NMSZ];        /* connection profile name      */
char              tpn_name[NMSZ];          /* connection profile name      */
char              local_file[NMSZ];        /* local         file name      */
char              remote_file[NMSZ];       /* remote        file name      */
struct ctrl_rec   cbuffer;                 /* control record buffer        */
struct data_rec   buffer;                  /* send buffer                  */
int               connfd;                  /* SNA       file descriptor    */
int               rid;                     /* SNA       resource ID        */
char              *cp;                     /* pointer to character array   */
struct data_rec   *bp;                     /* pointer to data record       */
int               n_read;                  /* bytes read                   */
int               n_sent;                  /* bytes sent                   */
int               count;                   /* error found indicator        */
unsigned          file_siz;                /* file size to receive         */
```

```
struct stat         e_stat;                  /* structure for 'stat' call    */
int                 lm;                      /* length of ctrl message       */
int                 rc;                      /* return code field            */
FILE                *fp;                     /* local stream file handle     */
extern int          errno;                   /* system error return code     */
int                 error_found = NO;        /* initialize error indicator   */
static              int more = YES;          /* switch for main loop         */
char                err_msg[64];             /* Error message field          */
char                strl[120];               /* Work area for editing        */
int aoption;                                 /* option switches              */
int coption;
int foption;
int hoption;
int roption;
int voption;
int soption;
int toption;

/* ------------------------------------------------------------------- */
/*                            MAIN CODE                                */
/* ------------------------------------------------------------------- */

main(argc, argv)
int argc;
char *argv[];
{
   error_found  = NO;                        /* Set error found to NO        */
   connfd = rid = ERROR;                     /* Initialize file descriptors  */
   strcpy( connection, argv[2] );            /* Get connection name          */
   rid = a641(argv[3]);                      /* Get the resource ID          */
   open_connection();                        /* Open connection              */
   sna_allocate_receive();                   /* Receive_allocate             */

                                             /* **************************** */
   while( more == YES )                      /*     M a i n   L o o p        */
   {                                         /* **************************** */

      receive_message();                     /* Get partner's control message*/
      if(error_found == YES) break;          /* Exit if error reading msg    */
#ifdef DEBUG
      strncpy(strl,&cbuffer,4);
      strl[4]='\0';
      printf("First 4 bytes of data = %s\n",strl);
      printf("Length of control message is %d bytes\n",n_read);
      printf("File name/message is: %s\n",buffer.msg);
#endif
      interpret_initial_message();           /* Interpret control message    */
      if(error_found == YES) break;          /* Exit if control msg error    */

      if( soption == 1)                      /* ========================== */
      {                                      /*  We must SEND the file       */
                                             /* ========================== */
      if(((stat(local_file, &e_stat)) == ERROR) || /* Check that file exists*/
         ((fp=fopen(local_file,"r")) == NULL)  ||  /* ..can be opened and    */
         (e_stat.st_size < 1))                     /* ..is not zero length   */
         {
            make_message("File empty or not found\n",'E');
            error_found = YES;
            receive_message();               /* Give partner time to digest  */
            break;
         }
      else
         {
            cbuffer.il = e_stat.st_size;     /* Insert file size in message  */
            sprintf(strl,"Okay, will send %d byte file\n",cbuffer.il);
            make_message(strl,'O');          /* Send "okay" message          */
         }
         if(error_found == YES) break;       /* Exit if error sending msg    */
         receive_message();                  /* Wait for a go-ahead message  */
         if(error_found == YES) break;       /* Exit if error reading msg    */
         interpret_further_message();           /* Check for error message      */
         if(error_found == YES) break;       /* Exit if error message recv'd */
         send_data();                        /* Send the data file           */
         if(error_found == YES) break;       /* Exit if error sending file   */
         receive_message();                  /* Read okay message            */
         if(error_found == YES) break;      /* Exit if receive error        */
         interpret_further_message();          /* Check for error message      */
         if(error_found == YES) break;      /* Exit if error message recv'd */
      }
      else                                  /* =========================== */
      {                                     /*  We must SEND the file       */
                                            /* =========================== */
         /* Check that we can in fact open the file for output     */
         open_local_file();                 /* Try to open local file       */
         if(error_found == YES)             /* Handle result of OPEN oper.  */
         {
            make_message(err_msg,'E');
            receive_message();              /* Give partner time to digest  */
            break;
         }
         else
            make_message("Okay, ready!\n",'O');
         receive_file();                    /* Enter the RECEIVE loop       */
         if(error_found == YES)             /* Check for RECEIVE errors     */
         {
            make_message("File not received properly\n",'E');
            receive_message();              /* Give partner time to digest  */
            break;
         }
         make_message("File received\n",'O');   /* Else send OK message back*/
      }
      close_local_file();                   /* Close local file             */
      if(error_found == YES)                /* Check for CLOSE error on file*/
      {
         make_message(err_msg,'E');         /* Attempt to send error message*/
         send_error();                      /* Terminate conversation       */
         break;
      }
      if (error_found == YES) more = NO;    /* If error, exit main loop     */
   }  /* end main loop */

   close_connection();                      /* Closing the connection       */
}  /* end rtrt.c */


/* ---------------------------------------------------------------------- */
/*    This routine builds all messages to be sent from the RTRT program.  */
/*          After the message is built it is sent to the partner.        */
/* ---------------------------------------------------------------------- */

make_message(txt,code)
char    *txt;
char    code;
{
   sync();                                   /* Do two "sync"s to flush     */
   sync();                                   /* file buffers                */
   cbuffer.cmt = 'M';                        /* Build the message header    */
   cbuffer.ft  = '?';
   cbuffer.rf  = '?';
   cbuffer.stat = code;
   cbuffer.pil = 0;                          /* By convention, zero pad lgth*/
   strncpy(cbuffer.msg,txt,sizeof(cbuffer.msg)-1);   /* Copy message text   */
   cbuffer.msg[sizeof(cbuffer.msg)-2] = '\n';        /* Insert newline      */
   cbuffer.msg[sizeof(cbuffer.msg)-1] = '\0';        /* And terminate string*/
   cbuffer.lm = strlen(cbuffer.msg);                 /* Insert msg length   */
   if ((code != 'O') && (code != 'E')) cbuffer.stat = 'E';
   if (cbuffer.stat == 'E') error_found = YES;
#ifdef DEBUG
   printf("Sending message: %s, type %c\n",cbuffer.msg,code);
#endif
   n_sent = send_sna(&cbuffer,M_FIELD+cbuffer.lm);   /* Send the message    */
   if (error_found == YES)
   {
      fprintf(stderr,"Error sending message:");
      fprintf(stderr," %s",err_msg);
      send_error();
   }
}
```

```
/* ------------------------------------------------------------------- */
/*    This routine reads through the data file to be sent from this node,   */
/*            builds the LU 6.2 messages and sends them.                */
/* ------------------------------------------------------------------- */

send_data()
{
   int count;
   memset(err_msg,'\0',sizeof(err_msg));
   while((count = file_read(buffer.dpart,N_BYTES - D_FIELD)) > 0)
   {                                    /* Read file in max lgth chunks */
      buffer.dlen = count;              /* Set length field in buffer   */
      n_sent = send_sna(&buffer, count + D_FIELD);  /* Send one block    */
      if (error_found == YES)
      {
         fprintf(stderr,"Error sending data: %s",err_msg);
         count = 0;                     /* Reset to terminate loop      */
         send_error();                  /* Notify partner of error      */
         break;
      }
   }
   if (strlen(err_msg)>0)               /* If error other than SNA read */
   {
      make_message(err_msg,'E');        /* Try sending message          */
      send_error();                     /* Notify partner               */
   }
}

/* ------------------------------------------------------------------- */
/* ·    Routine to receive a control message from partner              */
/* ------------------------------------------------------------------- */
receive_message()
{
   memset(&cbuffer,'\0',sizeof(cbuffer));   /* Clear the receive buffer     */
   memset(err_msg,'\0',sizeof(err_msg));    /* and clear error message field*/
   n_read = sna_receive(&cbuffer,sizeof(cbuffer));  /* Then read message    */
   if ((n_read<1) || (error_found==YES))           /* Handle error situat'n*/
   {
      error_found = YES;
      if (strlen(err_msg)==0)
         strcpy(err_msg,"Received length was zero\n");
      fprintf(stderr,"%s",err_msg);
   }
}

/* ------------------------------------------------------------------- */
/*      Routine to receive a file from partner and write it to         */
/*             the local file names as given by partner.               */
/* ------------------------------------------------------------------- */
receive_file()
{
   int count=0;                         /* Initialize four counters     */
   int tmp_count=0;
   int one_write=0;
   int tmp=0;

   n_read = sna_receive(&buffer, N_BYTES);  /* Do initial read from SNA     */
   while(n_read > 0)
   {
      bp = &buffer;                     /* Load data buffer pointer     */
      cp = (char *) bp;                 /* and load character pointer   */
      while( (tmp_count < n_read ) && (error_found != YES) )
      {                                 /* Loop through data blocks     */
         count = file_write(cp+4,bp->dlen); /* Write data block to disk     */
         if (error_found==YES)
         {
            fprintf(stderr,"%s",err_msg);   /* Notify user if disk write err*/
            make_message(err_msg,'E');      /* Try sending message          */
            send_error();                   /* ..and notify partner         */
            break;
         }
         tmp_count = tmp_count + bp->dlen + sizeof(buffer.dlen);/* Adjust   */
         one_write = one_write + bp->dlen;                     /* counters */
         cp = cp + bp->dlen + sizeof(buffer.dlen);   /* Adjust character ptr*/
         bp = (struct data_rec *) cp;                /* and load data in ptr*/
      }
      n_read=0;                              /* Make sure we exit if no more */
      file_siz=file_siz - one_write;         /* Adjust what remains to read  */
      tmp_count = 0;                         /* Reset the two temporary      */
      one_write = 0;                         /* counters                     */
#ifdef DEBUG
      printf("Received: %d bytes, remains: %d\n",buffer.dlen,file_siz);
#endif
      if (file_siz > 0)                      /* If more to read from partner */
         n_read = sna_receive(&buffer, N_BYTES);  /* read max length         */
      if ((n_read < 1) && (file_siz > 0))    /* Check for errors             */
      {
         error_found = YES;
         break;
      }
   }
}

/* --------------------------------------------------------------------- */
/*              Analyze a received INITIAL control message               */
/* --------------------------------------------------------------------- */
interpret_initial_message()
{
   str1[0] = '\0';                           /* Reset message field          */
   switch(cbuffer.cmt)                       /* Process depending on TYPE    */
   {                                         /* ---------------------------- */
      case 'S':                              /* Partner sends, WE RECEIVE    */
               roption = 1;                  /* ---------------------------- */
               soption = 0;
               strncpy(local_file,cbuffer.msg,cbuffer.lm); /* Get file name */
               local_file[cbuffer.lm] = '\0';             /* Terminate str.*/
               file_siz = cbuffer.il;                     /* Get file size */
               sprintf(str1,"File size = %ld\n",file_siz); /* Build log text*/
               if (file_siz == 0) error_found=YES;        /* Check size    */
               break;
                                             /* ---------------------------- */
      case 'R':                              /* Partner receives, WE SEND    */
               soption = 1;                  /* ---------------------------- */
               roption = 0;
               strncpy(local_file,cbuffer.msg,cbuffer.lm); /* Get file name */
               local_file[cbuffer.lm] = '\0';             /* Terminate str.*/
               break;
                                             /* ---------------------------- */
      case 'M':                              /* Unexpected TYPE but valid    */
                                             /* ---------------------------- */
               if (cbuffer.stat == 'O')
               {
                  fprintf(stderr,"Partner returned unexpected OK message\n");
                  error_found = YES;
                  break;
               }
               if (cbuffer.stat == 'E')
               {
                  fprintf(stderr,"Partner returned ERROR");
                  if (cbuffer.lm > 0) fprintf(stderr,": %s\n",cbuffer.msg);
                  fprintf(stderr,"\n");
                  error_found = YES;
                  break;
               }
               /* fall through */
      default:
               fprintf(stderr,"Partner returned invalid message type\n");
               fprintf(stderr,"Type = %c, error = %c, data length = %d\n",
                  cbuffer.cmt,cbuffer.stat,cbuffer.lm);
               break;
   }
#ifdef DEBUG
   printf("Local file = %s\n",local_file);
   if (strlen(str1)>0) printf("%s",str1);
#endif
}

/* --------------------------------------------------------------------- */
/*         Interpret control messages other than initial one             */
/* --------------------------------------------------------------------- */
```

```
interpret_further_message()
{
   int i = 1;
   if(cbuffer.cmt == 'M' )                 /* Only message type M expected */
   {
      if(cbuffer.stat == 'O' )             /* If okay message received     */
      {
#ifdef DEBUG
         printf("OK Message returned\n");
#endif
         if (cbuffer.lm > 0)               /* If message text supplied     */
         {
            printf("Partner accepts");           /* Display that message   */
            printf(" saying: %s\n",cbuffer.msg);  /* ..on stdout            */
         }
      }
      else
      {
         fprintf(stderr,"Partner returned ERROR");
         if (cbuffer.lm > 0) fprintf(stderr,": %s\n",cbuffer.msg);
         else fprintf(stderr,", but sent no explanation\n");
         error_found = YES;
      }
   }
   else
   {
      fprintf(stderr,"Partner sent unexpected or invalid message type\n");
      error_found = YES;
   }
}
```

IBM RT Sub-programs for snaftp and RTRT

The following is a listing of separate C language modules used by the two programs snaftp.c and rtrt.c.

Common Include File (defs.h)

```
#include "luxsna.h"                         /* RT-PC SNA    include        */
#include <stdio.h>                          /* standard IO  include        */
#include <fcntl.h>                          /* file control include        */
#include <sys/stat.h>

#define YES         1                       /* YES                         */
#define NO          0                       /* NO                          */
#define ERROR      -1                       /* ERROR                       */
#define GOOD        0                       /* ERROR                       */
#define NULL        0                       /* NULL                        */
#define pmode       0644                    /* Read/Write for owner        */
                                            /* Read       for group, others*/
#define N_BYTES     4096                    /* size of buffer              */
#define NMSZ        36                      /* size of character strings   */
#define FNSZ        128                     /* size of filenames            */
#define MSG_LENGTH  128                     /* max size of message         */

#define CMT_FIELD       0                    /* start of CMT field         */
#define FT_FIELD        1                    /* start of FT field         */
#define FF_FIELD        2                    /* start of FF field         */
#define S_FIELD         3                    /* start of S field         */
#define PIL_FIELD       4                    /* start of PIL field         */
#define IL_FIELD        8                    /* start of IL field         */
#define MVRL_FIELD      12                   /* start of MVRL field         */
#define BS_FIELD        16                   /* start of BS field         */
#define LM_FIELD        20                   /* start of LM field         */
#define M_FIELD         24                   /* start of M field         */
#define LD_FIELD        0                    /* start of LD field         */
#define D_FIELD         4                    /* start of D field         */


/*#define DEBUG*/


#define V_TPN_DEFAULT "VMRT"                /* default VM transaction name  */
#define V_CONNECTION_DEFAULT "VM62"           /* default connection profile  */

#define T_TPN_DEFAULT "MVSRT"               /* default MVS transaction name  */
#define T_CONNECTION_DEFAULT "MVS62"          /* default connection profile  */

#define AS_TPN_DEFAULT "AS400RT"            /* default AS/400 transaction name  */
#define AS_CONNECTION_DEFAULT "AS40062"      /* default connection profile  */

#define A_TPN_DEFAULT "RTRT"                /* default AIX transaction name  */
#define A_CONNECTION_DEFAULT "RTRT"           /* default connection profile  */

#define O_TPN_DEFAULT "OS2RT"               /* default OS/2 transaction name  */
#define O_CONNECTION_DEFAULT "OS262"          /* default connection profile  */

#define D_TPN_DEFAULT "DOSRT"               /* default DOS transaction name  */
#define D_CONNECTION_DEFAULT "DOS62"          /* default connection profile  */

#define BLKSIZ_DEFAULT 1024                 /* default block size          */
#define LRECL_DEFAULT 128                   /* default logical record len  */
/*#define TPN_DEFAULT    "SYRG208RT.QGPL"*/
/*#define CONNECTION_DEFAULT     "APPCRT"*/

extern  int aoption;            /* 1 = ascii and cr-lf conversion    */
extern  int coption;            /* 1 = connection name profile       */
extern  int foption;            /* 1 = local file name given         */
extern  int hoption;            /* 1 = help requested                */
extern  int roption;            /* 1 = receive requested             */
extern  int soption;            /* 1 = send requested                */
extern  int toption;            /* 1 = transaction profile given     */
extern  int Hoption;            /* 1 = host specified                */
extern  int voption;            /* 1 = variable record length        */
extern  int host_type;          /* Host type: V=VM, T=TSO, 4=AS/400,
                                              A=AIX, O=OS/2 D=DOS */
```

```
extern int error_found;
extern int rc;
extern int lm;
extern int errno;

extern  char connection[];      /* connection profile name          */
extern  char local_file[];      /* local file name                  */
extern  char tpn_name[];        /* transaction profile name         */
extern  char remote_file[];     /* remote file name                 */
extern  unsigned long lrecl;    /* logical record length            */
extern  unsigned long blksiz;   /* block size                       */

/* For transmition status */
#define SEND            1
#define RECEIVE         2
#ifndef CONFIRM
#define CONFIRM         3
#endif
#define DATA            4
#define DATA_COMPLETE   5
#define DATA_INCOMPLETE 6
#define LL_TRUNCATED    7
#define FMH_COMPLETE    8
#define FMH_INCOMPLETE  9
#define NO_CONTROL_RECEIVED     10
#define POLL            11
#define CONFIRM_SEND    12
#define CONFIRM_DEALLOCATE      13
#define NORMAL_DEALLOCATE       14
#define CONFIRM_DEALLOCATE_RETAIN       15
#define NORMAL_DEALLOCATE_RETAIN        16

struct  ctrl_rec        /* ------------ CONTROL RECORD LAYOUT -------------- */
{
        char    cmt;            /* control record type              */
        char    ft;             /* file type                        */
        char    rf;             /* record format                    */
        char    stat;           /* status                           */
        int     pil;            /* padding                          */
        int     il;             /* total file length                */
        int     rl;             /* max/fixed record length          */
        int     bs;             /* block size                       */
        int     lm;             /* message length                   */
        char    msg[128];       /* message/file name                */
} ;


struct data_rec         /* -------------- DATA RECORD LAYOUT -----------------*/
{
        int     dlen;           /* data length                      */
        char    dpart[N_BYTES]; /* data part (4 bytes added as work ar)*/
};
```

Command Line Checking (arg.c)

```
/* ------------------------------ arg.c ------------------------------------

    Function to provide user interface for "snaftp" file transfer program.
          This function can be replaced by interface modules with
          different user interface as long as the external variables
          and argument list are kept unchanged.
   ------------------------------------------------------------------------*/
#include <ctype.h>
#include "defs.h"                      /* Common declarations            */

int aoption = 0;                       /* 1 = ascii and cr-lf conversion  */
int coption = 0;                       /* 1 = connection name profile     */
int foption = 0;                       /* 1 = local file name given       */
int hoption = 0;                       /* 1 = help requested              */
int roption = 0;                       /* 1 = receive requested           */
int soption = 0;                       /* 1 = send requested              */
int toption = 0;                       /* 1 = transaction profile given   */
int Hoption = 0;                       /* 1 = host specified              */
int voption = 0;                       /* 1 = variable record length      */
int host_type;                         /* Host type: V=VM, T=TSO, 4=AS/400,*/
                                       /*             A=AIX, O=OS/2 D=DOS */
char Host[20];                         /* Host type for display           */
char connection[NMSZ];                 /* Connection profile name         */
char local_file[FNSZ];                 /* Local file name                 */
char tpn_name[NMSZ];                   /* Transaction program name        */
char remote_file[FNSZ];                /* Remote file name                */
unsigned long lrecl;                   /* Logical record length           */
unsigned long blksiz;                  /* Block size                      */

void arg_usage();

/* --------------------------------------------------------------------- */
/*      This function analyzes the command line options and either       */
/*       returns an error or sets variables used by main program.        */
/* --------------------------------------------------------------------- */
int arg_check(argc, argv)
int argc;
char *argv[];
{
   int host_cnt=0;                        /* Counter of hosts given       */
   int i;
   int l;
   int c;
   int error = NO;
   extern char *optarg;                   /* For command line parser      */

   blksiz=BLKSIZ_DEFAULT;
   lrecl=LRECL_DEFAULT;
                                          /* ---------------------------- */
                                          /* First we get all arguments   */
                                          /* ---------------------------- */
   while( (c = getopt(argc, argv,"avc:b:l:f:h?rV:T:4:A:O:D:st:") ) != EOF)
   {
      switch(c)
      {
         case 'a':                        /* ASCII conversion             */
                  aoption = 1;
                  break;
         case 'c':                        /* Connection profile name      */
                  if (NMSZ > strlen(optarg))
                     strcpy(connection, optarg);
                  else
                  {
                     fprintf(stderr,"Connection name %s is too long",optarg);
                     error = TRUE;
                  }
                  if( connection[0] == '-')
                  {
                     fprintf(stderr,"Connection name %s is invalid\n",optarg);
                     error = TRUE;
                  }
                  coption = 1;
                  break;
```

```
case 'f':                          /* Remote file path/name        */
         if (FNSZ > strlen(optarg))
            strcpy(local_file, optarg);
         else
         {
            fprintf(stderr,"Local file name %s is too long\n");
            error = TRUE;
         }
         if( local_file[0] == '-')
         {
            fprintf(stderr,"Local file name can\'t begin with a minus ( - ) \n");
            error = TRUE;
         }
         foption = 1;
         break;
case 'l':                          /* Record length                */
         for (l=0;l<strlen(optarg);l++)
         {
            printf("%s\n",optarg[l]);
            if (isdigit(optarg[l])==0)
            {
               fprintf(stderr,"Logical Record length invalid\n");
               error = TRUE;
               break;
            }
         }
         lrecl = atol(optarg);
         break;
case 'b':                          /* Block size                   */
         for (l=0;l<strlen(optarg);l++)
         {
            printf("%s\n",optarg[l]);
            if (isdigit(optarg[l])==0)
            {
               fprintf(stderr,"Block size invalid\n");
               error = TRUE;
               break;
            }
         }
         blksiz = atol(optarg);
         break;
case '?':                          /* The poor guy asked us to     */
case 'h':                          /* help him with the syntax     */
         hoption = 1;
         break;
case 'v':                          /* Variable record length       */
         voption = 1;
         break;
case 'r':                          /* Receive file request         */
         roption = 1;
         break;
case 's':                          /* Send file request            */
         soption = 1;
         break;
case 't':                          /* Transaction program name     */
         if (strlen(optarg) > 8)
            strcpy(tpn_name, optarg);
         else
         {
            fprintf(stderr,"Transact program name %s is longer than 8 bytes\n",optarg);
            error = TRUE;
         }
         if( tpn_name[0] == '-')
         {
            fprintf(stderr,"Transaction program name %s is invalid\n",optarg);
            error = TRUE;
         }
         toption = 1;
         break;
case 'V':                          /* Host type = VM               */
case 'T':                          /* Host type = TSO              */
case '4':                          /* Host type = AS/400           */
case 'A':                          /* Host type = AIX              */
case 'O':                          /* Host type = OS/2             */
case 'D':                          /* Host type = PC-DOS           */
```

```
                    host_type=c;
                    host_cnt++;
                    if (FNSZ > strlen(optarg))
                       strcpy(remote_file, optarg);
                    else
                    {
                       fprintf(stderr,"Remote file name %s is too long\n",optarg);
                       error = TRUE;
                    }
                    Hoption = 1;
                    break;
           default:                        /* Invalid option              */
                    error = YES;
                    break;
        }
    }
                                           /* ---------------------------- */
    if (error == YES)                      /* Exit if errors so far       */
    {                                      /* ---------------------------- */
       fprintf(stderr,"Sorry pal, cannot continue with all the errors you made\n");
       hoption = 1;
    }
    switch(host_type)                      /* ---------------------------- */
    {                                      /* Insert host name for display */
       case 'V':                           /* ---------------------------- */
                 strcpy(Host,"VM");
                 break;
       case 'T':
                 strcpy(Host,"TSO");
                 break;
       case '4':
                 strcpy(Host,"AS/400");
                 break;
       case 'A':
                 strcpy(Host,"AIX");
                 break;
       case 'O':
                 strcpy(Host,"OS/2");
                 break;
       case 'D':
                 strcpy(Host,"DOS");
                 break;
       default:
                 fprintf(stderr,"Host type \"%c\" is invalid\n",host_type);
                 hoption = 1;
                 break;
    }
    if ((coption != 1) && (Hoption == 1))  /* ---------------------------- */
    {                                      /* Insert default connection    */
       switch(host_type)                   /* ---------------------------- */
       {
          case 'V':
                    strcpy(connection, V_CONNECTION_DEFAULT);
                    break;
          case 'T':
                    strcpy(connection, T_CONNECTION_DEFAULT);
                    break;
          case '4':
                    strcpy(connection, AS_CONNECTION_DEFAULT);
                    break;
          case 'A':
                    strcpy(connection, A_CONNECTION_DEFAULT);
                    break;
          case 'O':
                    strcpy(connection, O_CONNECTION_DEFAULT);
                    break;
          case 'D':
                    strcpy(connection, D_CONNECTION_DEFAULT);
                    break;
          default:
                    break;
       }
    }
    if( (toption != 1) && (Hoption == 1) )  /* ---------------------------- */
    {                                      /* Insert default tranact name  */
```

```
   switch(host_type)                    /* ---------------------------- */
   {
      case 'V':
                 strcpy(tpn_name, V_TPN_DEFAULT);
                 break;
      case 'T':
                 strcpy(tpn_name, T_TPN_DEFAULT);
                 break;
      case '4':
                 strcpy(tpn_name, AS_TPN_DEFAULT);
                 break;
      case 'A':
                 strcpy(tpn_name, A_TPN_DEFAULT);
                 break;
      case 'O':
                 strcpy(tpn_name, O_TPN_DEFAULT);
                 break;
      case 'D':
                 strcpy(tpn_name, D_TPN_DEFAULT);
                 break;
      default:
                 break;
   }
}
if ((soption == 1) && (roption == 1))    /* If send AND receive reqested */
{
   fprintf(stderr,"Come on; you cannot send and receive at the same time\n");
   hoption = 1;
}
if ((soption != 1) && (roption != 1) && (hoption != 1)) /* Missing info  /*
{
   fprintf(stderr,"You did not tell if you want to send or to receive\n");
   hoption = 1;
}
if (host_cnt > 1)                        /* More than one host given     */
{
   fprintf(stderr,"Hey!  You can only send to one host at a time...\n");
   hoption = 1;
}
if (Hoption != 1)                        /* If no host given             */
    hoption = 1;
else
   if (strlen(remote_file) == 0)
   {
      fprintf(stderr,"Remote file name is not specified\n");
      hoption = 1;
   }
if (foption != 1)                        /* If no remote file name       */
   hoption = 1;
else
{
   if (strlen(local_file) == 0)
   {
      fprintf(stderr,"Local file name is not specified\n");
      hoption = 1;
   }
}
if (voption==1)                          /* Check variable length option */
{
   if (lrecl>blksiz)
   {
      fprintf(stderr,"Logical record length exceeds block size\n");
      hoption = 1;
   }
   if (host_type=='4')
   {
      fprintf(stderr,"Variable record format not supported for AS-400\n");
      hoption = 1;
   }
}
else
{
   if (blksiz%lrecl > 0)
   {
      fprintf(stderr,"Logical record length is not multiple of block size\n");
```

```
         hoption = 1;
      }
   }

#ifdef DEBUG
   printf("Host type      = %c\n",host_type);
   printf("ASCII convert  = %d\n",aoption);
   printf("Receive option = %d\n",roption);
   printf("Var record len = %d\n",voption);
   printf("Send option    = %d\n",soption);
   printf("Block size     = %d\n",blksiz);
   printf("Record length  = %d\n",lrecl);
   printf("Conn profile   = %s\n",connection);
   printf("Local file     = %s\n",local_file);
   printf("Transact name  = %s\n",tpn_name);
   printf("Remote file    = %s\n",remote_file);
#endif

   if( hoption == 1 )                      /* If help requested or error   */
   {
      arg_usage(argc,argv);                /* Display help text            */
      return(ERROR);
   }
   return(YES);
}

/* --------------------------------------------------------------------- */
/*                   Display usage information.                          */
/* --------------------------------------------------------------------- */
void arg_usage(argc,argv)
int argc;
char *argv[];
{
   fprintf(stderr,
   "\nusage: %s [-ahv] [-c name] -f name -r/s [-t name] -V/T/4/A/D/O \".....\" [-b len] [-l len]\n",
   argv[0]);
   fprintf(stderr,
   "-a : ASCII/EBCDIC and cr-lf conversion must be done \n");
   fprintf(stderr,
   "-h : This help text \n");
   fprintf(stderr,
   "-v : Variable record length file transfer\n");
   fprintf(stderr,
   "-c name : Connection profile name\n");
   fprintf(stderr,
   "-f name : Local file name\n");
   fprintf(stderr,
   "-r : Receive a file from remote host. Can not be used together with -s\n");
   fprintf(stderr,
   "-s : Send a file to remote host. Can not be used together with -r\n");
   fprintf(stderr,
   "-t name : Remote transaction program name \n");
   fprintf(stderr,
   "-V \"remote_file_name\" : Remote system is VM;\n");
   fprintf(stderr,
   "-T \"remote_file_name\" : Remote system is TSO;\n");
   fprintf(stderr,
   "-4 \"remote_file_name\" : Remote system is AS/400;\n");
   fprintf(stderr,
   "-A \"remote_file_name\" : Remote system is AIX;\n");
   fprintf(stderr,
   "-D \"remote_file_name\" : Remote system is DOS;\n");
   fprintf(stderr,
   "-O \"remote_file_name\" : Remote system is OS/2;\n");
   fprintf(stderr,
   "-b len : Physical blocksize at remote host \n");
   fprintf(stderr,
   "-l len : Logical record length (max variable length) at remote host \n");
}
```

SNA Functions (sna.c)

```
/* ------------------------------ sna.c ------------------------------------

                    Functions for SNA Services using
                    the subroutine calls of AIX.
   ----------------------------------------------------------------------*/
#include <sys/errno.h>
#include "defs.h"                             /* Common declarations         */

char           connection[NMSZ];              /* connection profile name     */
int            connection_open = NO;          /* YES if connection open      */
int            conv_allocated = NO;           /* YES if convers. allocated   */
int            connfd = ERROR;                /* SNA      file descriptor    */
int            rid = ERROR;                   /* SNA      resource ID        */
char           err_msg[64];                   /* Error message field         */

struct allo_str      allo_str;                /* allocate     structure      */
struct deal_str      deal_str;                /* deallocate   structure      */
struct read_out      read_out;                /* read         structure      */
struct write_out     write_out;               /* write        structure      */
struct confirm_str   confirm_str;             /* confirm      structure      */
struct prep_str      prep_str;                /* prepare to receive struct   */

/* ---------------------------------------------------------------------- */
/*     Receive from SNA into character array "s" with max length "lgth".   */
/*                  Function returns actual length.                        */
/* ---------------------------------------------------------------------- */
int sna_receive(s,lgth)
char *s;
int lgth;
{
   int i;

   if ((i = snaread(connfd,s,lgth,rid,0,&read_out,"M")) == ERROR)
   {
      if (errno == EBADF) sprintf(err_msg,"Connection was closed\n");
      else sprintf(err_msg,"Error in snaread, errno = %d\n",errno);
      error_found = YES;
      i = 0;
   }
#ifdef DEBUG
   printf("SNA read up to %d bytes, received %d bytes\n",lgth,i);
#endif
   return(i);
}

/* ---------------------------------------------------------------------- */
/*                       Deallocate a conversation.                        */
/* ---------------------------------------------------------------------- */
deallocate()
{
   memset(&deal_str,0,sizeof(struct deal_str));  /* clear dealloc structure */
   deal_str.rid       = rid;                  /* specify resource ID         */
   deal_str.deal_flag = DISCARD;              /* type discard                */
   deal_str.type      = DEAL_FLUSH;           /* deallocate flush            */
   rc = snadeal(connfd,&deal_str,"M" );       /* deallocate conversation     */
   if (rc==ERROR)
   {
      sprintf(err_msg,"snadeal failed, errno = %d\n", errno);
      error_found = YES;
   }
}

/* ---------------------------------------------------------------------- */
/*                          Close a connection.                            */
/* ---------------------------------------------------------------------- */
close_connection()
{
   if (connfd != ERROR)                       /* if connection is open       */
   {
      if ((rc=snaclse(connfd)) == ERROR)
      {
         sprintf(err_msg,"snaclse failed, errno is %4d\n", errno);
         error_found = YES;
      }
   }
}

/* ----------------------------------------------------------------------- */
/*         Send to SNA from character array "s" with length "lgth".        */
/*                    Returns number of bytes actually sent.               */
/* ----------------------------------------------------------------------- */
int send_sna(s,lgth)
char *s;
int lgth;
{
   int i;
   memset(&write_out,0,sizeof(struct write_out));
   i = snawrit(connfd,s,lgth,rid,&write_out,"M");
   if ( i<= 0)                           /* if SNA write failed          */
   {
      error_found == YES;                /* indicate error               */
      sprintf(err_msg,"SNA send failed, errno = %d\n", errno);
   }
#ifdef DEBUG
   printf("SNA write of %d bytes, actually sent %d bytes\n",lgth,i);
#endif
   return(i);
}

/* ----------------------------------------------------------------------- */
/*                        Allocate a conversation.                         */
/* ----------------------------------------------------------------------- */
allocate_conversation()
{
#ifdef DEBUG
   printf("Transaction name = %s\n",tpn_name);
#endif
   memset(&allo_str,0,sizeof(struct allo_str));
   strcpy(allo_str.tpn,tpn_name);    /* insert transaction name     */
   allo_str.type = MAPPED_CONV;      /* conversation type is mapped */

   rid = snalloc(connfd,&allo_str,"M");
   if (rid == ERROR)               /* did allocate fail          */
   {
#ifdef DEBUG
      printf("Allocate failed, error code = %d\n",errno);
#endif
      if (connection_open == YES) close_connection();
      exit(ERROR);
   }
   conv_allocated = YES;
}

/* ----------------------------------------------------------------------- */
/*                          Open a connection.                             */
/* ----------------------------------------------------------------------- */
open_connection()
{
   connfd = snaopen( connection );          /* open the connection          */
   if ( connfd == ERROR )                   /* did open connection work     */
   {
      printf("open to the connection %s failed, errno = %4d\n",
                                    connection, errno);
      exit(ERROR);
   }
   connection_open = YES;
}

/* ----------------------------------------------------------------------- */
/*        Issue SEND_ERROR to inform partner of error condition.           */
/* ----------------------------------------------------------------------- */
send_error()
{
   struct erro_str erro_str;

   erro_str.rid = rid;
   snactl(connfd, SEND_ERROR, erro_str , "M");
}
```

```
/* ----------------------------------------------------------------------- */
/*                       Issue RECEIVE_ALLOCATE.                           */
/* ----------------------------------------------------------------------- */
sna_allocate_receive()
{
   memset( &allo_str, 0, sizeof(struct allo_str) );
   allo_str.rid = rid;                    /* Put resource ID in structure */
   rid = snalloc(connfd,&allo_str,"M" );
   if (rid == ERROR)                      /* Check for allocate error     */
   {
      close_connection();
      exit(ERROR);
   }
   else conv_allocated = YES;
}

/* ----------------------------------------------------------------------- */
/*      Prepare to receive (not used by the sample programs).              */
/* ----------------------------------------------------------------------- */
prepare_to_receive()
{
   prep_str.rid = rid;
   prep_str.type = SYNCL_CONFIRM;
   if( snactl(connfd, PREPARE_TO_RECEIVE, &prep_str, "M") < 0)
   {
      error_found = YES;
      sprintf(err_msg,"Error in prep to receive, errno = %d\n",errno);
   }
}

/* ----------------------------------------------------------------------- */
/*          Request to send (not used by the sample programs).             */
/* ----------------------------------------------------------------------- */
request_to_send()
{
   if (snactl(connfd,REQUEST_TO_SEND,rid,"M") < 0)
   {
      sprintf(err_msg,"Req to send aborted, errno = %d\n",errno);
      error_found = YES;
   }
}

/* ----------------------------------------------------------------------- */
/*        Issue a CONFIRM to partner (not used by the sample programs.     */
/* ----------------------------------------------------------------------- */
confirm()
{
   if ( rid != ERROR )
   {
   memset(&confirm_str,0,sizeof(struct confirm_str));
   confirm_str.rid  = rid;    /* put resource ID in structure*/
   rc = snactl(connfd,CONFIRM,&confirm_str,"M");
   if (rc == ERROR)             /* if confirm failed . . .     */
      {
         sprintf("snactl(confirm) failed, errno = %d\n", errno);
         error_found = YES;
      }
   }
}
```

Local File Access (fileio.c)

```
/* ----------------------------- fileio.c ----------------------------------

      Functions to open, close, read and write from a local file.
   ---------------------------------------------------------------------*/

#include "defs.h"                          /* common declarations         */

FILE             *fp;                      /* local stream file handle    */
char             err_msg[64];              /* Error message field         */
char             strl[120];                /* Work string for logging     */

/* -------------------------------------------------------------------- */
/*   Open a local file with file name in "local_file", return status.      */
/* -------------------------------------------------------------------- */
int open_local_file()
{
   int ec = GOOD;                          /* Assume we succeed           */
   err_msg[0] = '\0';                      /* Reset error message         */
   if( roption == 1 )                      /* If we must RECEIVE          */
   {
      sprintf(strl,"Open local file in write mode\n"); /* Make log message  */
      if( (fp = fopen( local_file, "w")) == NULL )     /* Open in WRITE mode*/
      {
         sprintf(err_msg,"Open of local file %s failed, errno = %4d\n",
            local_file, errno);
         error_found = YES;
         ec = ERROR;
      }
   }
   if( soption == 1 )                      /* If we must SEND             */
      {
      sprintf(strl,"Open local file in read mode\n");  /* Make log message  */
      fp = fopen( local_file, "r");                    /* Open in READ mode */
      if(fp == NULL )
      {
         sprintf(err_msg,"Open to the local file %s failed, errno = %4d\n",
            local_file, errno);
         error_found = YES;
         ec = ERROR;
      }
   }
#ifdef DEBUG
   printf("%s",strl);
   if (strlen(err_msg)>0) printf("%s",err_msg);
#endif
   return(ec);
}

/* -------------------------------------------------------------------- */
/*     Write "number" bytes to file from "s", return bytes written.        */
/* -------------------------------------------------------------------- */
int file_write(s,number)
char *s;
int number;
{
   int ri;
   err_msg[0] = '\0';                      /* Reset error message         */
   if ((ri = fwrite(s, 1, number, fp)) != number) /* Do the write operation*/
      sprintf(err_msg,"Disk write error, errno = %d\n",errno);
#ifdef DEBUG
   printf("File write attempted for %d bytes\n",number);
   if (strlen(err_msg)>0) printf("%s",err_msg);
#endif
   return(ri);
}

/* -------------------------------------------------------------------- */
/*     Read "number" bytes from file into "s", return bytes read.          */
/* -------------------------------------------------------------------- */
int file_read(s,number)
char *s;
int number;
{
   int antal;
   err_msg[0] = '\0';                      /* Reset error message          */
   antal = fread(s, 1, number, fp);        /* Read from the local file     */
   if (antal < 1) sprintf(strl,"Disk read error %d\n",errno);
#ifdef DEBUG
   printf("File read of up-to %d bytes, %d bytes read\n",number,antal);
   if (strlen(err_msg)>0) printf("%s",err_msg);
#endif
   return(antal);
}

/* ---------------------------------------------------------------------- */
/*             Close the local file.                                      */
/* ---------------------------------------------------------------------- */
close_local_file()
{
   if (fp == NULL) return;
   if (fclose(fp) == EOF)
   {
      sprintf(err_msg,"Can\'t close file ");
      fprintf(stderr,"%s",err_msg);
      error_found=YES;
   }
   fp = NULL;
}
```

EBCDIC/ASCII Conversion (convert.c)

```
/* ---------------------------- convert.c -----------------------------------

        Converts an ASCII string to EBCDIC or EBCDIC string to ASCII.

        Arguments: ptr1       Pointer to source string
                   conv       Direction:
                                 0  -  ASCII to EBCDIC
                                 1  -  EBCDIC to ASCII
                   num_bytes  Number of bytes to translate:
                                <= 0  translate until end of string
                                >  0  translate "num_bytes" bytes
                   ptr2       Pointer to destination buffer, if NULL do
                              the translation in place.

        Returns:   Number of bytes actually translated.
                   If error, returns -1.
   --------------------------------------------------------------------- */

#include  <stdio.h>

#define  AS_TO_EB             0        /* Translate ASCII -> EBCDIC */
#define  EB_TO_AS             1        /* Translate EBCDIC -> ASCII */
#define  ERR                  (-1)

static unsigned char     as_to_eb[] = {
        0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a,
        0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15,
        0x16, 0x17, 0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f, 0x40,
        0x5a, 0x7f, 0x7b, 0x5b, 0x6c, 0x50, 0x7d, 0x4d, 0x5d, 0x5c, 0x4e,
        0x6b, 0x60, 0x4b, 0x61, 0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6,
        0xf7, 0xf8, 0xf9, 0x7a, 0x5e, 0x4c, 0x7e, 0x6e, 0x6f, 0x7c, 0xc1,
        0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xd1, 0xd2, 0xd3,
        0xd4, 0xd5, 0xd6, 0xd7, 0xd8, 0xd9, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6,
        0xe7, 0xe8, 0xe9, 0x5b, 0xe0, 0x5d, 0x5f, 0x6d, 0x79, 0x81, 0x82,
        0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x91, 0x92, 0x93, 0x94,
        0x95, 0x96, 0x97, 0x98, 0x99, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7,
        0xa8, 0xa9, 0xc0, 0x4f, 0xd0, 0xa1, 0x7f, 0x80, 0x81, 0x82, 0x83,
        0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8a, 0x8b, 0x8c, 0x8d, 0x8e,
        0x8f, 0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98, 0x99,
        0x9a, 0x9b, 0x9c, 0x9d, 0x9e, 0x9f, 0xa0, 0xa1, 0xa2, 0xa3, 0xa4,
        0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xaa, 0xab, 0xac, 0xad, 0xae, 0xaf,
        0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xba,
        0xbb, 0xbc, 0xbd, 0xbe, 0xbf, 0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5,
        0xc6, 0xc7, 0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf, 0xd0,
        0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6, 0xd7, 0xd8, 0xd9, 0xda, 0xdb,
        0xdc, 0xdd, 0xde, 0xdf, 0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6,
        0xe7, 0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xed, 0xee, 0xef, 0xf0, 0xf1,
        0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf9, 0xfa, 0xfb, 0xfc,
        0xfd, 0xfe, 0xff};

static unsigned char     eb_to_as[] = {
        0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a,
        0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15,
        0x16, 0x17, 0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f, 0x20,
        0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2a, 0x2b,
        0x2c, 0x2d, 0x2e, 0x2f, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36,
        0x37, 0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f, 0x20, 0x41,
        0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4a, 0x2e, 0x3c,
        0x28, 0x2b, 0x7c, 0x26, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57,
        0x58, 0x59, 0x21, 0x24, 0x2a, 0x29, 0x3b, 0x5e, 0x2d, 0x2f, 0x62,
        0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6a, 0x2c, 0x25, 0x5f,
        0x3e, 0x3f, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78,
        0x60, 0x3a, 0x23, 0x40, 0x27, 0x3d, 0x22, 0x80, 0x61, 0x62, 0x63,
        0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x8a, 0x8b, 0x8c, 0x8d, 0x8e,
        0x8f, 0x90, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, 0x70, 0x71, 0x72,
        0x9a, 0x9b, 0x9c, 0x9d, 0x9e, 0x9f, 0xa0, 0x7e, 0x73, 0x74, 0x75,
        0x76, 0x77, 0x78, 0x79, 0x7a, 0xaa, 0xab, 0xac, 0xad, 0xae, 0xaf,
        0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xba,
        0xbb, 0xbc, 0xbd, 0xbe, 0xbf, 0x7b, 0x41, 0x42, 0x43, 0x44, 0x45,
        0x46, 0x47, 0x48, 0x49, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf, 0x7d,
        0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50, 0x51, 0x52, 0xda, 0xdb,
        0xdc, 0xdd, 0xde, 0xdf, 0x5c, 0xe1, 0x53, 0x54, 0x55, 0x56, 0x57,
        0x58, 0x59, 0x5a, 0xea, 0xeb, 0xec, 0xed, 0xee, 0xef, 0x30, 0x31,
    0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0xfa, 0xfb, 0xfc,
    0xfd, 0xfe, 0xff};

int convert(ptrl, conv, num_byt, ptr2)
unsigned char           *ptrl;
int                     conv;
int                     num_byt;
unsigned char           *ptr2;
{
   extern unsigned char     as_to_eb[];
   extern unsigned char     eb_to_as[];
   register unsigned char   *conv_array;
   register unsigned char   *dest;
   register int             num_exam;
   register int             num_trns;
   register unsigned char   *source;

   if ((source = ptrl) == NULL) return(ERR);/* ---------------------------- */
   if (ptr2 != NULL) dest = ptr2;          /* Determine destination addr   */
   else dest = ptrl;                       /* ---------------------------- */

   if (conv == AS_TO_EB) conv_array = as_to_eb;   /* ---------------------- */
   else                                           /* Determine direction    */
      if (conv == EB_TO_AS) conv_array = eb_to_as;/* ---------------------- */
      else return(ERR);

   /* ------------------------------------------------------------ */
   /*                       Infinite loop                          */
   /* ------------------------------------------------------------ */
   for (num_trns = 0,num_exam = 0 ; ; )
   {
      if (num_byt <= 0)                    /* If no length given in args   */
      {
         if (*source == NULL)              /* ..and if end of string       */
         {
            *dest = NULL;                  /* Terminate Destnation string  */
            break;
         }
      }
      else                                 /* If length given explicitly   */
         if (num_exam == num_byt) break;   /* ...break if length reached   */

      if (*source != conv_array[*source]) num_trns++; /* Incr. translate cnt*/
      *dest++ = conv_array[*source++];     /* Do the actual translate      */
      num_exam++;                          /* Increment counter            */
   }
   return(num_trns);                       /* Return chars translated      */
}
```

Make Control Messages (control.c)

```
/* ---------------------------- control.c ----------------------------------

       Functions to build control messages for file transfer program
          'snaftp' which is the only program using this module.
   ----------------------------------------------------------------------*/
#include "defs.h"                          /* Common declarations         */

char             tbuffer[N_BYTES];         /* Read buffer                 */
char             *ptr;                     /* Pointer to character array  */
unsigned         file_siz;                 /* File size to receive        */
struct stat      e_stat;                   /* Structure for 'stat' call   */
FILE             *fp;                      /* Local stream file handle    */
void             field_conv();

/* --------------------------------------------------------------------- */
/*           Build control message for SENDING file from RT              */
/* --------------------------------------------------------------------- */
int make_send_control_message(s)
struct ctrl_rec *s;
{
   int i;

#ifdef DEBUG
   printf("make_send_control_message\n");
#endif
   s->cmt = 'S';                         /* request to send file from RT    */
   s->stat = '?';                        /* leave status fields undefinded  */
   if(aoption == 1)
                                         /* -------------------------------- */
   {                                     /* ********* TEXT FORMAT ********* */
      s->ft = 'T';                       /* -------------------------------- */
      file_siz = 0;
      if ((fp = fopen(local_file,"r")) == NULL)  /* Check if file exists    */
      {
         fprintf(stderr,"Unable to open local file \"%s\"\n",local_file);
         error_found=YES;
         return(ERROR);
      }
      else
      {                                  /* Read through file to get length */
         while ((ptr = fgets(tbuffer,sizeof(tbuffer),fp)) != NULL)
         {
            if ((voption==0) && ((host_type != 'D') && (host_type != 'O')))
            {
               file_siz = file_siz + (strlen(tbuffer) -1) / lrecl; /* Adjust*/
               if ((strlen(tbuffer) % lrecl) > 0)           /* for fixed */
                  file_siz = file_siz + lrecl;              /* length rec*/
            }
            else file_siz = file_siz + (strlen(tbuffer) +1);    /* not fixed */
            if ((host_type == 'D') || (host_type == 'O'))
            {
               for (i=0;i<strlen(tbuffer);i++)        /* Remove any CRs if */
                  if (tbuffer[i] == '\r') file_siz--;  /* OS/2 or DOS host  */
            }
         }
         close_local_file();
#ifdef DEBUG
         printf("Resulting file size is %d\n",file_siz);
#endif
         if (voption==0) s->rf = 'F';               /* Insert rec format   */
         else s->rf = 'V';                          /* in control record   */
         if( (stat(local_file,&e_stat)) == ERROR)   /* Get local file size */
         {
            fprintf(stderr,"Unable to stat local file \"%s\"\n",local_file);
            error_found = YES;
            return(ERROR);
         }
         s->pil = file_siz - e_stat.st_size;        /* Insert pad length   */
         s->il = file_siz;                          /* and total file size */
      }
   }
   else                                  /* -------------------------------- */
   {                                     /* ******** BINARY FORMAT ******** */
```

```
      s->ft = 'B';                          /* ------------------------------- */
      if((stat(local_file,&e_stat)) == ERROR)       /* Get local file size */
      {
         error_found = YES;
         fprintf(stderr,"Unable to stat local file \"%s\"\n",local_file);
         return(ERROR);
      }
      if ((voption==1)||(host_type=='A')||(host_type=='O')||(host_type=='D'))
      {                                     /* Assume variable record length   */
         s->rf = 'V';                       /* and insert in control record    */
         s->pil = 0;                        /* No padding                      */
         s->il = e_stat.st_size;            /* Insert file size                */
      }
      else
      {
         s->rf = 'F';                       /* Else fixed record format        */
         s->pil = lrecl - (e_stat.st_size%lrecl);    /* Insert padding      */
         s->il = s->pil + e_stat.st_size;            /* and file size       */
#ifdef DEBUG
         printf("File size is %ld\n",e_stat.st_size);
#endif
      }
#ifdef DEBUG
      printf("File size is %ld\n",s->il);
#endif
   }
   s->rl = lrecl;                           /* Insert logical record length    */
   s->bs = blksiz;                          /* Insert block size               */
   s->lm = strlen(remote_file);             /* Insert file name length         */
   strcpy(s->msg,remote_file);              /* Insert file name                */
#ifdef DEBUG
   printf("File name is %s, length file name is %d\n",s->msg,s->lm);
#endif
   lm = s->lm + M_FIELD;                    /* Set field for SNA write funct.  */
   if (e_stat.st_size < 1)                  /* Check for valid file size       */
   {
      fprintf(stderr,"Local file \"%s\" has length %d\n",
         local_file,e_stat.st_size);
      error_found = YES;
      return(ERROR);
   }
   do_convert(s);                           /* Convert to EBCDIC if required   */
   intel_conv(s);                           /* Convert to Intel if required    */
   return(GOOD);
}  /* end make_send_control_message */

/* ------------------------------------------------------------------------ */
/*            Build control message for RECEIVING file from partner         */
/* ------------------------------------------------------------------------ */
int make_receive_control_message(s)
struct ctrl_rec *s;
{
#ifdef DEBUG
   printf("make_receive_control_message\n");
#endif
   s->cmt = 'R';                            /* Insert control record type      */
   s->ft  = '?';                            /* Disregard file type             */
   s->rf  = '?';                            /* Disregard record format         */
   s->stat = '?';                           /* Disregard status                */
   s->pil = 0;                              /* Disregard padding length        */
   s->lm = strlen(remote_file);             /* Insert file name length         */
   strcpy(s->msg,remote_file);              /* Insert remote file name         */
   lm = s->lm + M_FIELD;                    /* Load for SNA send function      */
   do_convert(s);                           /* Convert to EBCDIC if required   */
   intel_conv(s);                           /* Convert to Intel if required    */
   return(GOOD);
}

/* ------------------------------------------------------------------------ */
/*            Convert complete control record to EBCDIC                     */
/* ------------------------------------------------------------------------ */
do_convert(s)
struct ctrl_rec *s;
{
   if( (host_type == 'V') || (host_type == 'T') || (host_type == '4') )
   {
      convert(&s->cmt,0,PIL_FIELD,NULL); /* Convert fixed part of ctl rec   */
      if( s->lm > 0  )
      {
         convert(s->msg,0,s->lm,NULL);   /* Convert any file name or message*/
      }
   }
}

/* ------------------------------------------------------------------- */
/*            Make and send a control message of type 'M'              */
/* ------------------------------------------------------------------- */
int make_message(s,txt,code)
struct ctrl_rec *s;
char            *txt;
char            code;
{

   int          i;
   sync(); /* fsync instead? */
   sync();
   s->cmt = 'M';                        /* Insert control record type      */
   s->ft  = '?';                        /* Disregard file type             */
   s->rf  = '?';                        /* Disregard record format         */
   s->stat = code;                      /* Insert status code from caller  */
   s->pil = 0;                          /* Disregard padding length        */
   if (strlen(txt)<sizeof(s->msg)) strcpy(s->msg,txt);
   else
   {
      strncpy(s->msg,txt,sizeof(s->msg)-1);  /* Insert valid message text   */
      s->msg[sizeof(s->msg)-1] = '\n';       /* not overflowing fields      */
   }
   s->lm = strlen(s->msg);              /* Insert message length           */
   i = s->lm;                           /* Save length for SNA send        */
   if ((code != 'O') && (code != 'E')) s->stat = 'E'; /* Make code valid   */
   if (s->stat != 'O') error_found=YES;
#ifdef DEBUG
   printf("Sending message: %s, type %c\n",s->msg,code);
#endif
   do_convert(s);                       /* Convert to EBCDIC if required   */
   intel_conv(s);                       /* Convert to Intel if required    */
   send_sna(s,M_FIELD+i);               /* Send control message to partner */
   if (error_found == YES) return(ERROR);
   else return(GOOD);
}

/* ------------------------------------------------------------------- */
/*            These two functions convert to-from Intel binary format  */
/* ------------------------------------------------------------------- */
int intel_conv(s)                       /* Converts a Control Message      */
struct ctrl_rec *s;
{
   if ((host_type == 'O') || (host_type == 'D'))
   {
      field_conv(&s->pil);
      field_conv(&s->il);
      field_conv(&s->rl);
      field_conv(&s->bs);
      field_conv(&s->lm);
   }
}

void field_conv(ip)                     /* Converts a single 4-byte word   */
int *ip;
{
   char  wk[8];
   char  wp[8];

   memcpy(wk,ip,4);
   wp[0] = wk[3];
   wp[1] = wk[2];
   wp[2] = wk[1];
   wp[3] = wk[0];
   memcpy(ip,wp,4);
}
```

Interpret Control Messages (interpret.c)

```
/* --------------------------- interpret.c ----------------------------------

      Functions to analyze control messages received from remote
      transaction programs by the file transfer program 'snaftp'.
   ------------------------------------------------------------------------*/
#include "defs.h"                            /* Common declarations         */

unsigned            file_siz;                /* File size to receive        */
unsigned            rec_len;                 /* Logical record length       */
char                strl[120];               /* Work area for editing       */

/* ----------------------------------------------------------------------- */
/*         Interpret response to RECEIVE control message.                  */
/* ----------------------------------------------------------------------- */
int interpret_receive_message(s,length)
struct ctrl_rec *s;
int length;
{
   interpret_common(s,length);               /* Do common analysis          */
   if (error_found == YES) return(ERROR);
   if ((s->cmt == 'M') || (s->stat == '0')) /* If Okay message returned    */
   {
      file_siz = s->il;                      /* Get remote file size        */
      rec_len = s->rl;                       /* Get remote record length    */
      if ((host_type == 'O') || (host_type == 'D'))
      {
         field_conv(&file_siz);              /* Convert sizes if partner is */
         field_conv(&rec_len);               /* host with Intel processor   */
      }
   }
   return(GOOD);
}  /* end interpret_receive_message */

/* ----------------------------------------------------------------------- */
/*         Interpret response to SEND control message.                     */
/* ----------------------------------------------------------------------- */
int interpret_send_message(s,length)
struct ctrl_rec *s;
int length;
{
   interpret_common(s,length);               /* Do common analysis          */
   if (error_found == YES) return(ERROR);
   else return(GOOD);
}  /* end interpret_send_message */

/* ----------------------------------------------------------------------- */
/*             Common code for both response types                         */
/* ----------------------------------------------------------------------- */
interpret_common(s,length)
struct ctrl_rec *s;
int length;
{
   int i = 1;

   if( (host_type == 'V') || (host_type == 'T') || (host_type == '4') )
   {
      convert(s,1,PIL_FIELD,NULL);           /* Convert 4 bytes if EBCDIC   */
      if( ( s->lm > 0 ) && ( s->lm <= length - D_FIELD ) )
      {
         convert(s->msg,1,s->lm,NULL);       /* Convert any message/file name*/
      }
   }
   if (s->cmt == 'M')                        /* Only type = M expected      */
   {                                         /* ---------------------------- */
      if (s->stat == '0')                    /* If OKAY message             */
      {                                      /* ---------------------------- */
         if (s->lm > 0)                      /* If partner returned message */
            printf("Partner says: %s\n",s->msg); /* Display that message    */
      }
      else
      {
         fprintf(stderr,"Partner returned ERROR");
         if (s->lm > 0) fprintf(stderr,": %s\n",s->msg);
```

```
            else fprintf(stderr,", but sent no message text\n");
            error_found = YES;
         }
      }                                    /* ---------------------------- */
      else                                 /* If ERROR message             */
      {                                    /* ---------------------------- */
         fprintf(stderr,"Partner returned garbled message, type = %c\n",s->cmt);
         error_found = YES;
      }
   }  /* end interpret_common */
```

IBM AS/400 Remote Transaction Program

All the AS/400 programs were developed by Marcela Adan from the International Technical Support Center in Poughkeepsie.

CRTRTICFF CL Program

```
/* CRTRTICFF: CL Program used to create the ICF file RTICFF        */
/*              Author: Marcela Adan, ITSC Rochester               */
             PGM
             MONMSG     MSGID(CPF0000)
             CHGCURLIB  CURLIB(ASRTPCLIB)
             DLTF       FILE(ASRTPCLIB/RTICFF)
             CRTICFF    FILE(ASRTPCLIB/RTICFF) +
                          SRCFILE(ASRTPCLIB/QDDSSRC) +
                          ACQPGMDEV(RTPGMDEV)
             ADDICFDEVE FILE(ASRTPCLIB/RTICFF) PGMDEV(RTPGMDEV) +
                          RMTLOCNAME(*REQUESTER) CMNTYPE(*APPC) +
                          MODE(RT)
             ENDPGM
```

RTICFF ICF File Data Description Specifications

```
A*----------------------------------------------------------------
A*AS400RT RPG Program  -  Author: Marcela Adan, ITSC Rochester
A*----------------------------------------------------------------
A                                      INDARA
A                                      RCVDETACH(30)
A          R CTLMSG
A            CMT            1A         TEXT('CONTROL MESSAGE TYPE')
A            FT             1A         TEXT('FILE TYPE')
A            FF             1A         TEXT('FILE FORMAT')
A            S              1A         TEXT('STATUS')
A            PIL            5B 0       TEXT('PAD INFO LENGTH')
A            IL             5B 0       TEXT('TOTAL FILE LENGTH')
A            RL             5B 0       TEXT('RECORD LENGTH')
A            BS             5B 0       TEXT('BLOCK SIZE')
A            LM             5B 0       TEXT('MESSAGE LENGTH')
A            M             40A         TEXT('MESSAGE')
A*----------------------------------------------------------------
A*DATA = 4096 Maximum length of data received or sent.
A*LENFLD: Contains the Actual Record Length sent to RT PC
A*----------------------------------------------------------------
A          R RECDATA                   VARLEN(&LENFLD)
A            DATA        4096A
A            LENFLD         5S 0P
```

AS400RT RPG Program

```
F*----------------------------------------------------------------
F*ASRTTX CL Program  -  Author: Marcela Adan, ITSC Rochester
F*RTICFF: ICF File and File Information Data Structure
F*----------------------------------------------------------------
FRTICFF  CF  E                    WORKSTN
F                                              KINFDS FEEDBK
F*----------------------------------------------------------------
F*FILERX: Program File used by the AS/400 program to send back an
F*RT PC file (RT PC Receives).
F*RXFDBK: File Information Data Structure .
F*----------------------------------------------------------------
FFILERX  IF  F    4096            DISK                             UC
F                                              KINFDS RXFDBK
F*----------------------------------------------------------------
F*FILETX: Program File used by the AS/400 program to received a
F* a data base file sent by RT PC.
F*FILERX and FILETX are defined with the maximum record length
F*expected (Record Length 4096).
F*----------------------------------------------------------------
FFILETX  O   F    4096            DISK                             UC
F*----------------------------------------------------------------
F*PRINT: Disk File used to log the CTLMSG sent by the RT PC.
F*Used for problem determination purposes.
```

```
F*-------------------------------------------------------------
FPRINT   O   F      80            DISK
E*-------------------------------------------------------------
E*ARRERR: Array to contain the error messages sent to RT PC in
E*the M field of the CTLMSG when an error occurs.
E*-------------------------------------------------------------
E                   ARRERR  1   2 40
IFILERX  NS
I                                           14096 DATA
I*-------------------------------------------------------------
I*MAJMIN Return Codes: After every input or output operation
I*to the ICF file the communication return codes must be check
I*to determine the result of the operation.
I*-------------------------------------------------------------
IFEEDBK      DS
I                                        401 404 MAJMIN
I                                        401 402 MAJCOD
I                                        403 404 MINCOD
I*-------------------------------------------------------------
I*Before sending back a file to be Received by the RT PC, a CTLMSG
I*record must be sent including the Record Length (RL) and the
I*total length of the file (IL).
I*When the data base file is opened, the record length and the
I*total number of records in the file can be obtained from
I*the File Information Data Structure.
I*-------------------------------------------------------------
IRXFDBK      DS
I                                      B 125 1260RECLEN
I                                      B 156 1590NBRREC
IDATA        DS
C*-------------------------------------------------------------
C*Some fields used in the program:
C*LEN: Command length of the command to be executed by QCMDEXC.
C*RECL, RL1, RL: Length of the record received from the RT PC.
C*RECLEN: Record length of the file to be sent to the RT PC
C*obtained from the File Information Data Structute RXFDBK.
C*N : Number of records in the file sent by the RT PC.
C*NBRREC: Number of records in the file to be sent to the RT PC
C*obtained from the File Information Data Structure RXFDBK.
C*FLDLEN: Actual Record Length of the file sent to the RT PC and
C*used as a parameter of the DDS keyword VARLEN.
C*-------------------------------------------------------------
C*Indicators:
C* *IN30: Received Detach (RCVDETACH)
C* *IN99: End of File FILERX.
C*-------------------------------------------------------------
C* Maximum Command Length to be executed by QCMNEXC = 52
C*-------------------------------------------------------------
C                   Z-ADD52        LEN    155
C          START    TAG
C                   READ CTLMSG                    99
C          *IN30    IFEQ '0'                         *IN30 RCVDETACH
C*-------------------------------------------------------------
C*Log the CTLMSG record for problem determination purposes.
C*-------------------------------------------------------------
C                   EXCPTLOG
C*-------------------------------------------------------------
C*CMT = S : RT PC will send a file (AS/400 receives).
C*-------------------------------------------------------------
C          CMT      IFEQ 'S'
C*-------------------------------------------------------------
C*AS/400 will accept only Fixed Record Length Files.
C*-------------------------------------------------------------
C          FF       IFEQ 'F'
C                   Z-ADDRL        RL1    40
C                   MOVE RL1       RECL    4
C*-------------------------------------------------------------
C*Call the CL program ASRTTX and pass the following parameters:
C*M: Contains Library Name/File Name
C*RECL: Contains the Record Length of the file to be sent by the
C*RT PC. If the file does not exist must be created.
C*ASRTTX performs some checking, creates the file if it does not
C*exist and assembles the OVRDBF command to be executed by
C*QCMDEXC.
C*Upon return, ASRTTX passes the following parameters back to the
```

```
C*calling program:
C*M: Contains either 'OK" or error message depending on the
C*results of the validation.
C*CMDEXC: contains the OVRDBF to be executed.
C*------------------------------------------------------------------
C                     CALL 'ASRTTX'
C                     PARM           M
C                     PARM           CMDEXC 52
C                     PARM           RECL
C           M         IFEQ 'OK'
C                     Z-ADD0         LM
C                     MOVE 'M'       CMT
C                     MOVE 'O'       S
C                     MOVE *BLANK    M
C*------------------------------------------------------------------
C*Send back a CTLMSG indicating the the RT PC sending request
C*was accepted.
C*------------------------------------------------------------------
C                     WRITECTLMSG
C*------------------------------------------------------------------
C*Calculate number of records to be read from the ICF file (N).
C*------------------------------------------------------------------
C           IL        DIV  RL        N       30
C*------------------------------------------------------------------
C*Execute the OVRDBF command before opening the data base file.
C*------------------------------------------------------------------
C                     CALL 'QCMDEXC'
C                     PARM           CMDEXC
C                     PARM           LEN
C                     OPEN FILETX
C*------------------------------------------------------------------
C*Read N records from the ICF File.
C*------------------------------------------------------------------
C                     DO   N
C                     READ RECDATA                 99
C*------------------------------------------------------------------
C*Check for communication errors
C*------------------------------------------------------------------
C           MAJCOD    IFGT '03'                      ERROR
C                     Z-ADD19        LM
C                     MOVE ARRERR,2  M
C                     GOTO ERROR
C                     END                            MAJCOD IFGT '03'
C                     EXCPTWRITE
C                     END                            END DO
C                     ELSE                           M "NOT OK'
C                     Z-ADD25        LM
C                     GOTO ERROR
C                     END                            M IFEQ 'OK'
C                     ELSE                           FF IFNEQ 'F'
C                     Z-ADD33        LM
C                     MOVE ARRERR,1  M
C                     GOTO ERROR
C                     END                            FF IFEQ 'F'
C*------------------------------------------------------------------
C*Send CTLMSG indicating that the file has been received OK.
C*Close the data base file and go back to START to read a new
C*CTLMSG or a RCVDETACH indicator.
C*------------------------------------------------------------------
C                     Z-ADD2         LM
C                     MOVEL'OK'      M
C                     WRITECTLMSG
C                     CLOSEFILETX
C                     GOTO START
C                     END                            CMT IFEQ 'S'
C*------------------------------------------------------------------
C*CMT = R : RT PC Receives a  file sent by AS/400.
C*------------------------------------------------------------------
C           CMT       IFEQ 'R'
C*------------------------------------------------------------------
C*Call the CL program ASRTRX and pass Library Name/File Name (M).
C*ASRTRX performs checking and returns:
C*Error message or "OK" and OVRDBF command to be executed by
C*QCMDEXC.
C*------------------------------------------------------------------
```

```
C                    CALL 'ASRTRX'
C                    PARM           M
C                    PARM           CMDEXC
C           M        IFEQ 'OK'
C                    MOVE 'M'       CMT
C                    MOVE 'O'       S
C*----------------------------------------------------------
C* OVRDBFILE FILE(FILERX) TOFILE(&LIB/&FILE)
C*----------------------------------------------------------
C                    CALL 'QCMDEXC'
C                    PARM           CMDEXC
C                    PARM           LEN
C*----------------------------------------------------------
C                    OPEN FILERX
C*----------------------------------------------------------
C* Get RECLEN and NBRREC from RXFDBK
C*----------------------------------------------------------
C                    Z-ADDRECLEN    LENFLD
C                    Z-ADDRECLEN    RL
C           RECLEN   MULT NBRREC    IL
C*----------------------------------------------------------
C*Send back control message with "OK" plus file information
C*----------------------------------------------------------
C                    WRITECTLMSG
C*----------------------------------------------------------
C*Wait until the RT PC sends back a second CTLMSG.
C*----------------------------------------------------------
C                    READ CTLMSG                  90
C                    EXCPTLOG
C*----------------------------------------------------------
C*Expected answer: CMT = "M" AND S = "O"
C*----------------------------------------------------------
C           CMT      IFNE 'M'
C                    GOTO END
C                    END                            CMT IFNE 'M'
C           S        IFNE 'O'
C                    GOTO END
C                    END                            S IFNE 'O'
C*----------------------------------------------------------
C* Read data base file and write to the ICF file to send to RT PC.
C*----------------------------------------------------------
C           *IN99    DOWEQ'0'                       *IN99 EOF
C                    READ FILERX                  99
C           *IN99    IFEQ '0'
C                    WRITERECDATA                   WRITE TO ICFF
C*----------------------------------------------------------
C*Check for communications errors after every READ or WRITE to
C*the ICF File
C*----------------------------------------------------------
C           MAJCOD   IFGT '03'
C                    Z-ADD19        LM
C                    MOVE ARRERR,2  M
C                    GOTO ERROR
C                    GOTO ERROR
C                    END                            MAJCOD IFGT 03
C                    END                            *IN99 IFEQ '0'
C                    END                            *IN99 DOWEQ '0'
C                    ELSE                           M NOT 'OK'
C                    Z-ADD25        LM
C                    END                            M IFEQ 'OK'
C                    CLOSEFILERX
C                    GOTO START
C                    END                            CMT IFEQ 'R'
C*-----------------------------------------------------------------
C* CMT = M: Log the CTLMSG and end the transaction.
C*-----------------------------------------------------------------
C           CMT      IFEQ 'M'
C                    EXCPTLOG
C                    GOTO END
C                    END                            CMT IFEQ M
C*-----------------------------------------------------------------
C                    END                            *IN30 IFEQ '0'
C*-----------------------------------------------------------------
C           END      TAG
C                    SETON                      LR
```

```
     C                   RETRN
     C*------------------------------------------------------------
     C           ERROR     TAG
     C                     MOVE 'M'       CMT
     C                     MOVE 'E'       S
     C                     WRITECTLMSG
     C                     GOTO START
     OFILETX  E                WRITE
     O                         DATA
     OPRINT   E                LOG
     O                         CMT
     O                         FT
     O                         FF
     O                         PIL
     O                         IL
     O                         RL
     O                         BS
     O                         LM
     O                         M
     O                         MAJMIN
**
VARIABLE REC LENGTH NOT SUPPORTED
COMMUNICATION ERROR
```

ASRTTX CL Program

```
             PGM        PARM(&M &CMDEXC &RECL)
             DCL        VAR(&M) TYPE(*CHAR) LEN(40)
             DCL        VAR(&CMDEXC) TYPE(*CHAR) LEN(52)
             DCL        VAR(&RECL) TYPE(*CHAR) LEN(4)
             DCL        VAR(&LIB) TYPE(*CHAR) LEN(10)
             DCL        VAR(&FILE) TYPE(*CHAR) LEN(10)
             DCL        VAR(&Z) TYPE(*DEC) LEN(2 0) VALUE(1)
             DCL        VAR(&X) TYPE(*DEC) LEN(2 0)
             DCL        VAR(&N) TYPE(*DEC) LEN(2 0) VALUE(1)
             DCL        VAR(&IN) TYPE(*LGL) VALUE('0')
/* Check for Slash(/) separator */
SLASH:
             IF         COND(%SST(&M &Z 1) *EQ '/') THEN(GOTO +
                          CMDLBL(LIBNAM))
             CHGVAR     VAR(&Z) VALUE(&Z + 1)
             IF         COND(&Z *LE 40) THEN(GOTO CMDLBL(SLASH))
             CHGVAR     VAR(&M) VALUE('Forgot Slash (/)??????')
             GOTO       CMDLBL(END)
/* Extract Library Name and verify name length, valid name, object +
   exists, object authority  */
LIBNAM:
             IF         COND(%SST(&M &N 1) *EQ '/') THEN(GOTO +
                          CMDLBL(LIBNAMEND))
             CHGVAR     VAR(&N) VALUE(&n + 1)
             GOTO LIBNAM
             /*                                  */
LIBNAMEND:
             IF         COND(&N *GT 11) THEN(DO)
             CHGVAR     VAR(&M) VALUE('Lib Name too Long')
             GOTO END
             ENDDO
             CHGVAR     VAR(&X) VALUE(&n - 1)
             CHGVAR     VAR(&LIB) VALUE(%SST(&M 1 &X))
             CHKOBJ     OBJ(QSYS/&LIB) OBJTYPE(*LIB)
             MONMSG     MSGID(CPF0001) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('Invalid Library Name')
             GOTO END
             ENDDO
             MONMSG     MSGID(CPF9801) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('Library not found')
             GOTO END
             ENDDO
             MONMSG     MSGID(CPF9820) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('Not Authorized to Library')
             GOTO END
             ENDDO
             CHGVAR     VAR(&N) VALUE(&n + 1)
/* Extract File Name and verify name length, valid name, object +
   existance. */
/* If File doesn't exist, CRTPF  */
/* If File exists, CLRPF         */
FILNAM:
             IF         COND(%SST(&M &N 1) *EQ ' ') THEN(GOTO +
                          CMDLBL(FILNAMEND))
             CHGVAR     VAR(&N) VALUE(&n + 1)
             GOTO FILNAM
FILNAMEND:
             IF         COND(&N *GT 21) THEN(DO)
             CHGVAR     VAR(&M) VALUE('FILE Name too Long')
             GOTO END
             ENDDO
             CHGVAR     VAR(&X) VALUE(&X + 2)
             CHGVAR     VAR(&N) VALUE(&n - 1)
             CHGVAR     VAR(&FILE) VALUE(%SST(&M &X &N))
             CHKOBJ     OBJ(&LIB/&FILE) OBJTYPE(*FILE) MBR(*FIRST)
             MONMSG     MSGID(CPF0001) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('Invalid File Name')
             GOTO END
             ENDDO
             MONMSG     MSGID(CPF9801) EXEC(DO)
             CRTPF      FILE(&LIB/&FILE) RCDLEN(&RECL)
             CHGVAR     VAR(&IN) VALUE('1')
             ENDDO
             IF         COND(&IN *EQ '0') THEN(CLRPFM FILE(&LIB/&FILE))
 /* Assemble OVRDBF command to be passed in &CMDEXC to AS400RT  */
             CHGVAR     VAR(&CMDEXC) VALUE('OVRDBF FILE(FILETX) +
                          TOFILE(')
             CHGVAR     VAR(&CMDEXC) VALUE(&CMDEXC *TCAT &LIB)
             CHGVAR     VAR(&CMDEXC) VALUE(&CMDEXC *TCAT '/')
             CHGVAR     VAR(&CMDEXC) VALUE(&CMDEXC *TCAT &FILE)
             CHGVAR     VAR(&CMDEXC) VALUE(&CMDEXC *TCAT ')')
             CHGVAR     VAR(&M) VALUE('OK')
END:
             RETURN
             ENDPGM
```

ASRTRX CL Program

```
             PGM        PARM(&M &CMDEXC)
             DCL        VAR(&M) TYPE(*CHAR) LEN(40) /* Contains +
                          Library Name/File Name */
             DCL        VAR(&CMDEXC) TYPE(*CHAR) LEN(52) /* To be +
                          returned to AS400RT after assembling  the +
                          OVRDBF Command. */
             DCL        VAR(&LIB) TYPE(*CHAR) LEN(10) /* Library Name +
                          */
             DCL        VAR(&FILE) TYPE(*CHAR) LEN(10) /* File Name */
             DCL        VAR(&Z) TYPE(*DEC) LEN(2 0) VALUE(1) /* Used +
                          as Index */
             DCL        VAR(&X) TYPE(*DEC) LEN(2 0) /* Used as Index */
             DCL        VAR(&N) TYPE(*DEC) LEN(2 0) VALUE(1) /* Used +
                          as Index */
             DCL        VAR(&IN) TYPE(*LGL) VALUE('0') /* Used as +
                          Indicator */
/* Check for Slash(/) separator */
SLASH:
             IF         COND(%SST(&M &Z 1) *EQ '/') THEN(GOTO +
                          CMDLBL(LIBNAM))
             CHGVAR     VAR(&Z) VALUE(&Z + 1)
             IF         COND(&Z *LE 40) THEN(GOTO CMDLBL(SLASH))
             CHGVAR     VAR(&M) VALUE('Forgot Slash (/)??????')
             GOTO       CMDLBL(END)
/* Extract Library Name and verify name length, valid name, object +
   exists, object authority  */
LIBNAM:
             IF         COND(%SST(&M &N 1) *EQ '/') THEN(GOTO +
                          CMDLBL(LIBNAMEND))
             CHGVAR     VAR(&N) VALUE(&n + 1)
             GOTO LIBNAM
             /*                             */
LIBNAMEND:
             IF         COND(&N *GT 11) THEN(DO)
             CHGVAR     VAR(&M) VALUE('Lib Name too Long')
             GOTO END
             ENDDO
             CHGVAR     VAR(&X) VALUE(&n - 1)
             CHGVAR     VAR(&LIB) VALUE(%SST(&M 1 &X))
             CHKOBJ     OBJ(QSYS/&LIB) OBJTYPE(*LIB)
             MONMSG     MSGID(CPF0001) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('Invalid Library Name')
             GOTO END
             ENDDO
             MONMSG     MSGID(CPF9801) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('Library not found')
             GOTO END
             ENDDO
             MONMSG     MSGID(CPF9820) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('Not Authorized to Library')
             GOTO END
             ENDDO
             CHGVAR     VAR(&N) VALUE(&n + 1)
/* Extract File Name and verify name length, valid name, object +
   exists  */
FILNAM:
             IF         COND(%SST(&M &N 1) *EQ ' ') THEN(GOTO +
                          CMDLBL(FILNAMEND))
             CHGVAR     VAR(&N) VALUE(&n + 1)
             GOTO FILNAM
FILNAMEND:
             IF         COND(&N *GT 21) THEN(DO)
             CHGVAR     VAR(&M) VALUE('FILE Name too Long')
             GOTO END
             ENDDO
             CHGVAR     VAR(&X) VALUE(&X + 2)
             CHGVAR     VAR(&N) VALUE(&n - 1)
             CHGVAR     VAR(&FILE) VALUE(%SST(&M &X &N))
             CHKOBJ     OBJ(&LIB/&FILE) OBJTYPE(*FILE) MBR(*FIRST)
             MONMSG     MSGID(CPF0001) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('Invalid File Name')
             GOTO END
             ENDDO
             MONMSG     MSGID(CPF9801) EXEC(DO)
             CHGVAR     VAR(&M) VALUE('File not found')
             GOTO END
             ENDDO
 /* Assemble OVRDBF command to be passed in &CMDEXC to AS400RT  */
             CHGVAR     VAR(&CMDEXC) VALUE('OVRDBF FILE(FILERX) +
                          TOFILE(')
             CHGVAR     VAR(&CMDEXC) VALUE(&CMDEXC *TCAT &LIB)
             CHGVAR     VAR(&CMDEXC) VALUE(&CMDEXC *TCAT '/')
             CHGVAR     VAR(&CMDEXC) VALUE(&CMDEXC *TCAT &FILE)
             CHGVAR     VAR(&CMDEXC) VALUE(&CMDEXC *TCAT ')')
             CHGVAR     VAR(&M) VALUE('OK')
END:
             RETURN
             ENDPGM
```

Operating System/2 Remote Transaction Program

The following program is based upon the sample program that comes with Operation System/2 Extended Edition. The program is capable of doing what it is designed for but has only limited error checking. We apologize for the programming style but changing somebody else's program has its disadvantages.

```
/*****************************************************************************/
/*                                                                           */
/*   PROGRAM NAME:     OS2RT.EXE (Source: OS2RT.C)                           */
/*                                                                           */
/*   DESCRIPTIVE NAME: APPC Remote Transaction Program for File Transfer     */
/*                     between an RT PC running the Advanced Interactive     */
/*                     Executive Operating System (AIX) and a PC or PS/2     */
/*                     running Operating System/2 (OS/2) Extended Edition.   */
/*                                                                           */
/*   COPYRIGHT: XXXXX  (C) COPYRIGHT IBM CORP. 1987,1988,1989                */
/*              LICENSED MATERIAL - PROGRAM PROPERTY OF IBM                  */
/*              ALL RIGHTS RESERVED                                          */
/*                                                                           */
/*   N-O-T-E:   This program is an OS/2 program intended to be scheduled     */
/*              by the OS/2 Attach Manager component of the Communications   */
/*              Manager.  Hence, the profiles for the latter must be con-    */
/*              figured to match the RT configuration and this program.      */
/*                                                                           */
/*   STATUS:    As released with Communications "Cookbook" from the          */
/*              International Technical Support Center, Austin.              */
/*              Modified from original sample program as shipped             */
/*              with Operating System/2 Extended Edition by:                 */
/*                                                                           */
/*                   Niels Christiansen, ITSC Austin, March 1989.            */
/*                                                                           */
/*   FUNCTION = Receives one ore more requests to send or receive named      */
/*              files from an RT PC, requested from the RT PC AIX sample     */
/*              program 'snaftp', also developed for the Communications      */
/*              "Cookbook".  For a full description, see the "Cookbook".     */
/*                                                                           */
/*              Depending on request type and error conditions, this         */
/*              program may:                                                 */
/*                                                                           */
/*              1. Send a file to the RT PC.                                 */
/*              2. Receive a file from the RT PC.                            */
/*              3. Notify the RT PC of errors encountered.                   */
/*                                                                           */
/*              The program uses the following APPC Verbs:                   */
/*                                                                           */
/*                 RECEIVE_ALLOCATE                                          */
/*                 MC_SEND_DATA                                              */
/*                 MC_RECEIVE_AND_WAIT                                       */
/*                 MC_SEND_ERROR                                             */
/*                 MC_DEALLOCATE                                             */
/*                 TP_ENDED                                                  */
/*                                                                           */
/*              The program uses the following General Sevices APPC Verbs:  */
/*                                                                           */
/*                 CONVERT                                                   */
/*                 LOG_MESSAGE                                               */
/*                                                                           */
/*   REMARKS:   Extensive logging to the file C:\CMLIB\OS2RT.LOG is added    */
/*              to provide easy tracing of program progress.  The tracing    */
/*              can be enabled by compiling the program as follows:          */
/*                    CC OS2RT /AL /DTRACE;                                  */
/*                                                                           */
/*              The message error logging of the original sample program     */
/*              is retained but is now only activated if you compile the     */
/*              program as follows:                                          */
/*                    CC OS2RT /AL /DMESSAGE;                                */
/*                                                                           */
/*              To activate both form of logging, use:                       */
/*                    CC OS2RT /AL /DMSGTRACE;                               */
/*                                                                           */
/*   MODULE TYPE = IBM Personal Computer C/2 Compiler Version 1.1            */
/*                 (REQUIRES Large Memory Model)                              */
/*                                                                            */
/*   Requires message file "APX.MSG" at runtime if message error logging      */
/*   is activated.                                                            */
/*                                                                            */
/*   To compile, move the following include files to C2\INCLUDE:              */
/*        C:\CMLIB\APPC_C.H                                                   */
/*        C:\CMLIB\ACSSVCC.H                                                  */
/*        C:\TOOLKT11\DOSCALLS.H                                              */
/*                                                                            */
/******************************************************************************/

#define LINT_ARGS

#include <APPC_C.H>
#include <ACSSVCC.H>
#include <STDIO.H>
#include <STDDEF.H>
#include <STRING.H>
#include <DOS.H>
#include <DOSCALLS.H>

#define TRUE  1
#define FALSE 0
#define N_BYTES 4096
#define M_FIELD 24
#define ERROR -1
#define GOOD 0

/* Macro clear_vcb sets the APPC verb control block to zeros */
#define clear_vcb()     memset(&vcb,(int)'\0',sizeof(vcb))

typedef unsigned int BOOL;

BOOL sys_err      = FALSE;             /* Fatal system error flag      */
BOOL eof_flag     = FALSE;             /* End of File flag             */
BOOL bad_filename = FALSE;             /* Bad file name flag           */

unsigned short dos_rc;                 /* Save area for return codes   */
unsigned short appc_rc_p;              /* APPC retcode - primary       */
unsigned long  appc_rc_s;              /* APPC retcode - secondary     */

struct {                               /* Data to insert in err log    */
    char     error_vrb[5];             /* ASCII Fmt Verb               */
    char     error_pri[5];             /* ASCII Fmt Primary RC         */
    char     error_sec[9];             /* ASCII Fmt Secondary RC       */
} log_ins_text;

char     error_dos_rc[5];              /* ASCII Fmt OS/2 retcode       */

char tp_id[8];                         /* transaction prorgram id      */
unsigned long conv_id;                 /* conversation id              */
char ra_tp_name[64];                   /* tp_name for RCV_ALLOCATE     */

unsigned char far *anbptr;             /* Pointer to shared buffer     */
unsigned short filehandle;             /* Handle returned by DOSOPEN   */
unsigned short bytes_read;             /* Byte count from DOSREAD      */

/*----------------------------------------------------------------------------*/
/*   Verb Control Block - Union of Structures                                 */
/*----------------------------------------------------------------------------*/
union  {
    struct receive_allocate      rcv_a;
    struct mc_receive_and_wait   rcv_wait;
    struct mc_send_data          send;
    struct mc_send_error         senderr;
    struct mc_deallocate         dealloc;
    struct tp_ended              tpend;
    struct convert               cnvt;      /* Genl Svcs CONVERT            */
    struct log_message           lmg;       /* Genl Svcs LOG_MESSAGE        */
} vcb;

unsigned far *vcbptr;                  /* Pointer to the vcb           */
```

```
/*-------------------------------------------------------------------------*/
/*   Definitions added for RT PC AIX File Transfer functions              */
/*-------------------------------------------------------------------------*/
struct ctrl_rec         /* --------- Control Record Layout --------- */
{
    char        cmt;                        /* Control message type S/R/M   */
    char        ft;                         /* File type T=text, B=binary   */
    char        rf;                         /* Record format V/F            */
    char        stat;                       /* Status O=ok, E=error         */
    long int    pil;                        /* Padding length added         */
    long int    il;                         /* File length incl. padding    */
    long int    rl;                         /* Max/variable record length   */
    long int    bs;                         /* Block size                   */
    long int    lm;                         /* Length of message/file name  */
    char        msg[128];                   /* File name or message         */
} ;

struct data_rec         /* ----------- Data Record Layout ---------- */
{
    long int    dlen;                       /* Length of data in record     */
    char        dpart[N_BYTES-4];           /* Data part of record          */
} ;

struct FileStatus Fs;                       /* Structure for DosQFileInfo   */
struct ctrl_rec   *cp;                      /* Pointer to control record    */
struct data_rec   *bp;                      /* Pointer to data record       */
char              *kp;                      /* Character array pointer      */
char              direction;                /* File transfer direction      */
char              local_file[128];          /* Local file name              */
BOOL              more = TRUE;              /* Loop control variable        */
char              strl[128];                /* Work area for messages       */
long              file_siz;                 /* Size of file to send/receive */

#ifdef  MSGTRACE
#define MESSAGE TRUE
#define TRACE   TRUE
#endif

#ifdef   TRACE
FILE              *lf;                      /* Log file handle              */
#endif

/*-------------------------------------------------------------------------*/
/*                          Function Prototypes                            */
/*-------------------------------------------------------------------------*/
#ifdef LINT_ARGS
void main(void);
void INIT_SELF(void);
void SHOW_MSG(unsigned short);
void SHOW_ERR(void);
void SHOW_DOS_ERROR(void);

void DO_RECEIVE_ALLOCATE(void);
int  DO_MC_RCV_AND_WAIT(void);
void DO_MC_SEND_DATA(int);
int  send_message(char[], char,long);
void DO_MC_SEND_ERROR(void);
void DO_MC_DEALLOCATE(void);
void DO_TP_ENDED(void);
int  WAIT_FOR_A_REQUEST(void);
void GET_SEND_CONTROL(void);

void ALLOC_SHARED_BUFFER(void);
void OPEN_DOS_FILE(void);
int  DO_DOSREAD(void);
int  DO_DOSWRITE(int);
void CLOSE_DOS_FILE(void);
void lfclose(void);
#endif

/***************************************************************************/
/*                        Main Program Section                             */
/***************************************************************************/
```

```
void
main()
{
   unsigned int len, tmp_count, one_write, count;

#ifdef TRACE
   lf = fopen("C:\\cmlib\\os2rt.log","w");
   fprintf(lf,"OS2RT started\n");
#endif
   INIT_SELF();                             /* Initialize                 */
   DO_RECEIVE_ALLOCATE();                   /* Process incoming allocate  */
   if (sys_err)                             /* Quit if there was a problem */
   {
      lfclose();
      return;
   }

   while (more)        /* MAIN LOOP */
   {
      len = WAIT_FOR_A_REQUEST();           /* Wait for control message   */
      GET_SEND_CONTROL();                   /* We will be next side to send */
      cp = (struct ctrl_rec *) anbptr;      /* Load pointer to control rec  */
      cp->msg[cp->lm] = '\0';
#ifdef TRACE
      fprintf(lf,"File name is \"%s\"\n",cp->msg);
#endif
      if ((cp->cmt!='S') && (cp->cmt!='R')) /* Only REQUEST records are ok  */
      {
         send_message("I can\'t understand your request\n",'E',(long)0);
         len = WAIT_FOR_A_REQUEST();        /* give partner time to read    */
         lfclose();
         more = FALSE;
         break;
      }
      else
      {
         direction = cp->cmt;               /* Save file transfer direction */
         strcpy(local_file,cp->msg);        /* Save local file name         */
         file_siz = cp->il;                 /* File size (valid for type S) */
         OPEN_DOS_FILE();                   /* Open requested filename      */
         if (bad_filename)
         {
            send_message("Unable to open file\n",'E',(long)0);
            len = WAIT_FOR_A_REQUEST();     /* give partner time to read    */
            lfclose();
            more = FALSE;
            break;
         }
      }
      if (direction=='S') /* ========= WE must RECEIVE the file ======== */
      {
         sprintf(strl,"I\'m ready for your %ld byte file\n",file_siz);
         send_message(strl,'O',(long)0);    /* Send "okay" message to RT PC */
         tmp_count = 0;
         one_write = 0;
         if ((len = WAIT_FOR_A_REQUEST()) > 0)  /* Read a block from SNA    */
         {
            while ((len>0) && (! sys_err))  /* loop while SNA supplies data */
            {
               bp = (struct data_rec *) anbptr; /* Load data record pointer */
               kp = anbptr;                 /* Load character pointer   */
               while ((tmp_count<len) && (! sys_err))  /* While data avail. */
               {
                  count = DO_DOSWRITE(bp->dlen);/* Write the data file      */
                  if (sys_err)
                  {
                     send_message("Error writing file\n",'E',(long)0);
                     lfclose();
                     more = FALSE;
                     len = 0;
                     break;
                  }
                  /* These four statements update pointer to next block      */
                  tmp_count = tmp_count + bp->dlen + sizeof(bp->dlen);
                  one_write = one_write + bp->dlen;
            kp = kp + bp->dlen + sizeof(bp->dlen);
            bp = (struct data_rec *) kp;
         }
         len = 0;                         /* Update for loop control */
         file_siz = file_siz - one_write; /* Calc. rest to receive   */
         one_write = 0;
         tmp_count = 0;
         if (file_siz>0) len = WAIT_FOR_A_REQUEST(); /* If more, read!*/
         if (sys_err==TRUE)
         {
            lfclose();
            more = FALSE;
            len = 0;
            break;
         }
      }
   }
   memset(anbptr,'\0',128);          /* Make sure buffer is cleared  */
   GET_SEND_CONTROL();               /* We will be next side to send */
   if (! sys_err) send_message("I\'m done\n",'O',(long)0);
}
else       /*  =========== WE must SEND the file =========== */
{
   cp = (struct ctrl_rec *) anbptr;  /* Load control record pointer  */
   bp = (struct data_rec *) anbptr;  /* Load data record pointer     */
   dos_rc = DOSQFILEINFO(filehandle,1,(char far *)&Fs,sizeof(Fs));
   if (dos_rc != 0)
   {
      sprintf(str1,"DosQFileInfo error code %04x\n",dos_rc);
      send_message(str1,'E',(long)0);
      lfclose();
      more = FALSE;
      break;
   }
   file_siz = Fs.file_size;          /* Get DosQFileInfo file length */
   if (file_siz<1)
   {
      send_message("File length is zero\n",'E',(long)0);
      lfclose();
      more = FALSE;
      break;
   }
   sprintf(str1,"I\'ll send you a %ld byte file\n",file_siz);
   send_message(str1,'O',file_siz);  /* Inform partner of file size  */
   memset(anbptr,'\0',128);          /* Make sure buffer is cleared  */
   len = WAIT_FOR_A_REQUEST();       /* Wait for control message     */
   if ((cp->cmt != 'M') || (cp->stat != 'O'))
   {
      lfclose();
      more = FALSE;
      break;
   }
   GET_SEND_CONTROL();               /* We will be next side to send */
   if (sys_err==TRUE)
   {
      lfclose();
      more = FALSE;
      break;
   }
   eof_flag = FALSE;
   while ((! eof_flag) && (! sys_err) && (! bad_filename))
   {
      len = DO_DOSREAD();               /* Read the data file        */
      if ((sys_err==TRUE) || (len < 1)) /* If error or no data       */
      {
         send_message("Error reading file\n",'E',(long)0);
         lfclose();
         more = FALSE;
         DO_MC_SEND_ERROR();
         break;
      }
      else
      {
         bp->dlen = len;             /* Insert length field in block */
         DO_MC_SEND_DATA(len+4);     /* ..and send block to partner  */
```

```
            }
         }
      }
     if (! bad_filename) CLOSE_DOS_FILE();  /* Close the input file         */
   }
   DO_MC_DEALLOCATE();                      /* Deallocate conversation      */
   DO_TP_ENDED();                           /* Indicate TP is done          */
   lfclose();                               /* Close log file               */
   dos_rc = DOSFREESEG(FP_SEG(anbptr));     /* Free the shared segment      */
   if (dos_rc != 0) {SHOW_DOS_ERROR();}     /* Display an error msg.        */
}

/****************************************************************************/
/*                                                                          */
/*                       Utility Functions                                  */
/*                                                                          */
/****************************************************************************/

void
INIT_SELF()
/* Handle initialization chores */
{
  vcbptr = (unsigned far *) &vcb;           /* Setup vcb pointer            */
  ALLOC_SHARED_BUFFER();                    /* DOSALLOCSEG APPC data buffer */

  /* Convert names from ASCII to EBCDIC as required by APPC                 */

  memset (ra_tp_name, (int)' ', sizeof(ra_tp_name));
                                            /* all blanks                   */
  memcpy (ra_tp_name,"OS2RT   ",8);         /* Set the tp_name  *itsc*      */
  clear_vcb();                              /* Zero out the control block   */
  vcb.cnvt.opcode = SV_CONVERT;             /* Do a CONVERT general service */
  vcb.cnvt.direction = SV_ASCII_TO_EBCDIC;  /* Cnvt ASCII to EBCDIC         */
  vcb.cnvt.char_set = SV_AE;                /* AE conversion type           */
  vcb.cnvt.len = sizeof(ra_tp_name);        /* Convert 64 bytes             */
  vcb.cnvt.source = vcb.cnvt.target = (unsigned char far *)ra_tp_name;
                                            /* Addr.(convert in place)      */
  ACSSVC_C ((long) vcbptr);                 /* Do the convert to EBCDIC     */
}

void
SHOW_MSG (msgno)
/* This function logs the requested message from message dataset APX.MSG */

unsigned short msgno;
{
   clear_vcb();                             /* zero the vcb, also clears    */
                                            /* last call err data           */
#ifdef MESSAGE
   vcb.lmg.opcode = SV_LOG_MESSAGE;         /* LOG_MSG opcode               */
   vcb.lmg.msg_num = msgno;                 /* Message number in APX.MSG    */
   vcb.lmg.msg_file_name[0] = 'A';          /* Set "APX" as msg filename    */
   vcb.lmg.msg_file_name[1] = 'P';
   vcb.lmg.msg_file_name[2] = 'X';
   vcb.lmg.msg_act = SV_NO_INTRV;           /* No intervention action type  */
   ACSSVC_C ((long) vcbptr);                /* Go log the message           */
   if (vcb.lmg.primary_rc != SV_OK)         /* If bad, try to display err   */
   {
      printf("APPC Sample Program Error - Unable to log error message\n");
      printf("Return code = %04d  Message number = %04d\n",
               vcb.lmg.primary_rc,msgno);
      return;
   }
#endif
}

void
SHOW_ERR ()
/* This function logs the Primary and Secondary APPC return codes. */

{
   sprintf (log_ins_text.error_pri,"%04X",appc_rc_p);
                                            /* Cnvt prim. retcode to ASCII  */
   sprintf (log_ins_text.error_sec,"%08lX",appc_rc_s);
                                            /* Show retcode in hex          */
   sprintf (log_ins_text.error_vrb,"%04X",vcb.send.opcode);
                                         /* Cnvt verb to ASCII          */
   clear_vcb();                          /* zero the vcb, also clears   */
                                         /* last call err data          */
#ifdef MESSAGE
   vcb.lmg.opcode = SV_LOG_MESSAGE;      /* LOG_MSG opcode              */
   vcb.lmg.msg_num = 5;                  /* APPC err msg. # in APX.MSG  */
   vcb.lmg.msg_file_name[0] = 'A';       /* Set "APX" as mes. filename  */
   vcb.lmg.msg_file_name[1] = 'P';
   vcb.lmg.msg_file_name[2] = 'X';
   vcb.lmg.msg_act = SV_NO_INTRV;        /* No intervention action type */
   vcb.lmg.msg_ins_len = 19;             /* Insertion text length       */
   vcb.lmg.msg_ins_ptr = (unsigned char far *) &log_ins_text;
                                         /* Text address to insert      */
                                         /* in logged message           */
   ACSSVC_C ((long) vcbptr);             /* Go log the message          */
   if (vcb.lmg.primary_rc != SV_OK)
   {                                     /* If bad, try to display err  */
      printf("APPC Sample Program Error - Unable to log error message\n");
      printf("Return code = %04d  Message number = %04d \n",
            vcb.lmg.primary_rc,5);
      return;
   }
#endif
}

void
SHOW_DOS_ERROR ()
/* This function logs an OS/2 return code. */

{
   sprintf (error_dos_rc,"%04d",dos_rc);    /* Convert to ASCII            */
#ifdef MESSAGE
   clear_vcb();                          /* zero the vcb, also clears   */
                                         /* last call err data          */
   vcb.lmg.opcode = SV_LOG_MESSAGE;      /* LOG_MSG opcode              */
   vcb.lmg.msg_num = 4;                  /* Msg # for OS/2 err in APX.MSG*/
   vcb.lmg.msg_file_name[0] = 'A';       /* Set message file name "APX" */
   vcb.lmg.msg_file_name[1] = 'P';
   vcb.lmg.msg_file_name[2] = 'X';
   vcb.lmg.msg_act = SV_NO_INTRV;        /* No intervention action type */
   vcb.lmg.msg_ins_len = 5;              /* Length of insertion text    */
   vcb.lmg.msg_ins_ptr = (unsigned char far *) error_dos_rc;
                                         /* Text address to insert      */
                                         /* in logged message           */
   ACSSVC_C ((long) vcbptr);             /* Go log the message          */
   if (vcb.lmg.primary_rc != SV_OK)
   {                                     /* If bad, try to display err  */
      printf("APPC Sample Program Error - Unable to log error message\n");
      printf("Return code = %04d  Message number = %04d\n",
            vcb.lmg.primary_rc,4);
      return;
   }
#endif
}

/****************************************************************************/
/*                                                                          */
/*                APPC RELATED FUNCTIONS                                    */
/*                                                                          */
/****************************************************************************/

void
DO_RECEIVE_ALLOCATE ()
{
   clear_vcb();                          /* Zero out the vcb            */
   vcb.rcv_a.opcode = AP_RECEIVE_ALLOCATE;  /* Set the APPC opcode         */
   memcpy (vcb.rcv_a.tp_name, ra_tp_name, sizeof(ra_tp_name));
                                         /* Get tp_name (in EBCDIC)     */
   vcb.rcv_a.sync_level = AP_CONFIRM_SYNC_LEVEL;  /* Confirm sync level    */
   APPC_C((long) vcbptr);                /* Call APPC                   */

   appc_rc_p = vcb.rcv_a.primary_rc;
   appc_rc_s = vcb.rcv_a.secondary_rc;   /* Save APPC return codes      */
   conv_id = vcb.rcv_a.conv_id;          /* Save the conversation_id    */
   memcpy (tp_id, vcb.rcv_a.tp_id, sizeof(tp_id));   /* Save the tp_id      */
#ifdef TRACE
   fprintf(lf,"RECEIVE_ALLOCATE: ");
   fprintf(lf,"CID = %08x, TPID = %016x, ",conv_id,tp_id);
   fprintf(lf,"Rc_1 = %04x, Rc_2 = %08lx \n",appc_rc_p,appc_rc_s);
#endif
   if (appc_rc_p != AP_OK)
   {
      SHOW_ERR();
      sys_err = TRUE;
   }
}

int WAIT_FOR_A_REQUEST ()
{
   int i;
   i = DO_MC_RCV_AND_WAIT ();              /* MC_RECEIVE_AND_WAIT         */
   if (sys_err == TRUE) return(i);         /* Quit if there was an error   */

/* Verify we received DATA_COMPLETE, display an error otherwise */

   if ( (*(unsigned *)&vcb.rcv_wait.what_rcvd) != AP_DATA_COMPLETE)
   {
      sys_err = TRUE;
      SHOW_MSG (16);
   }
   return(i);
}


int DO_MC_RCV_AND_WAIT ()
{
   clear_vcb();                            /* Zero the vcb                */
   vcb.rcv_wait.opcode = AP_M_RECEIVE_AND_WAIT;  /* = MC_RECEIVE_AND_WAIT */
   vcb.rcv_wait.opext = AP_MAPPED_CONVERSATION;  /* Set MC ext. type      */
   vcb.rcv_wait.conv_id = conv_id;         /* Set Conversation_id         */
   memcpy (vcb.rcv_wait.tp_id, tp_id, sizeof(tp_id));  /* Set TP_id       */
   vcb.rcv_wait.max_len = N_BYTES;         /* Max length of to read       */
   vcb.rcv_wait.dptr = anbptr;             /* Set data buffer pointer     */

   APPC_C((long) vcbptr);                  /* Call APPC                   */

   appc_rc_p = vcb.rcv_wait.primary_rc;
   appc_rc_s = vcb.rcv_wait.secondary_rc;  /* Save APPC return codes      */
#ifdef TRACE
   fprintf(lf,"RCV_AND_WAIT: ");
   fprintf(lf,"Rc_1 = %04x, Rc_2 = %08lx, ",appc_rc_p,appc_rc_s);
   fprintf(lf,"data length received = %d\n",vcb.rcv_wait.dlen);
#endif
   if (appc_rc_p != AP_OK)
   {
      SHOW_ERR();
      sys_err = TRUE;
      return(0);
   }
   return(vcb.rcv_wait.dlen);
}

void
DO_MC_DEALLOCATE ()
{
   clear_vcb();                            /* Zero the vcb                */
   vcb.dealloc.opcode = AP_M_DEALLOCATE;   /* Verb-MC_DEALLOCATE          */
   vcb.dealloc.opext  = AP_MAPPED_CONVERSATION;
                                           /* Set MC ext. type            */
   vcb.dealloc.conv_id = conv_id;          /* Set conversation_id         */
   memcpy (vcb.dealloc.tp_id, tp_id, sizeof(tp_id));   /* Set tp_id       */
   if ((bad_filename == TRUE) || (sys_err == TRUE))/* Determine DEALLOC type*/
      vcb.dealloc.dealloc_type = AP_ABEND;  /* ABEND ? or..  Normal       */
   else vcb.dealloc.dealloc_type = AP_SYNC_LEVEL;

   APPC_C((long) vcbptr);                  /* Call APPC                   */

   appc_rc_p = vcb.dealloc.primary_rc;
   appc_rc_s = vcb.dealloc.secondary_rc;   /* Save APPC return codes      */
   if (appc_rc_p != AP_OK)
   {
      SHOW_ERR();
      sys_err = TRUE;
   }
}

void
DO_TP_ENDED ()
{
   clear_vcb();                           /* Zero out the vcb            */
   vcb.tpend.opcode = AP_TP_ENDED;        /* TP_ENDED opcode             */
   memcpy (vcb.tpend.tp_id, tp_id, sizeof(tp_id));     /* Set the tp_id     */

   APPC_C((long) vcbptr);                 /* Call APPC                   */

   appc_rc_p = vcb.tpend.primary_rc;
   appc_rc_s = vcb.tpend.secondary_rc;    /* Save APPC return codes      */
   if (appc_rc_p != AP_OK)
   {
      SHOW_ERR();
      sys_err = TRUE;
   }
}

void
GET_SEND_CONTROL ()
{
   DO_MC_RCV_AND_WAIT();                  /* Wait for permission to send  */
   if (sys_err == TRUE) return;           /* Quit if an error occured     */
   if ((*(unsigned *)&vcb.rcv_wait.what_rcvd) != AP_SEND)
   {
                                          /* Verify WHAT_RCVD code = SEND */
      sys_err = TRUE;                     /* Else, we had an error...     */
      SHOW_MSG (17);                      /* Permit to send not rcvd msg. */
      return;
   }
}

void
DO_MC_SEND_DATA (lgth)
int  lgth;
{
   clear_vcb();                           /* Zero the vcb fields          */
   vcb.send.opcode = AP_M_SEND_DATA;      /* MC_SEND_DATA opcode          */
   vcb.send.opext = AP_MAPPED_CONVERSATION; /* MC type extension          */
   vcb.send.conv_id = conv_id;            /* Set conversation_id          */
   memcpy (vcb.send.tp_id, tp_id, sizeof(tp_id));     /* Set tp_id          */
   vcb.send.dlen = lgth;                   /* Send amount read from file   */
   vcb.send.dptr = anbptr;                 /* Set pointer to data          */
                                           /* in shared segment            */

  APPC_C((long) vcbptr);                   /* Call APPC                    */

   appc_rc_p = vcb.send.primary_rc;
   appc_rc_s = vcb.send.secondary_rc;      /* Save APPC return codes       */
#ifdef TRACE
   fprintf(lf,"SNA send %d bytes of data: ",lgth);
   fprintf(lf,"Rc_1 = %04x, Rc_2 = %08lx \n",appc_rc_p,appc_rc_s);
#endif
   if (appc_rc_p != AP_OK) {               /* Abnormal retcode ?           */
      if (appc_rc_p == AP_DEALLOC_ABEND)   /* Requester abended            */
          return;                          /* But SERVER is still ok       */
      sys_err = TRUE;                      /* Otherwise, show error status */
      SHOW_ERR();
      return;
   }
}

int send_message (txt,code,flen)
char    *txt;
char    code;
long    flen;
{
   struct  ctrl_rec *s;
   s = (struct ctrl_rec *) anbptr;          /* Load control rec pointer     */
   s->cmt  = 'M';                           /* Message type                 */
   s->ft   = '?';                           /* Disregard file type          */
   s->rf   = '?';                           /* Disregard record format      */
   s->stat = code;                          /* Insert status code           */
   s->pil  = 0;                             /* Reset padding length         */
   s->il  = flen;                           /* Insert file size (or zero)   */
   if (strlen(txt)<sizeof(s->msg))          /* Depending on message length..*/
      strcpy(s->msg,txt);                   /* ..take full message...       */
   else
   {
      strncpy(s->msg,txt,sizeof(s->msg)-1); /* ..or maximul message length  */
      s->msg[sizeof(s->msg)-1] = '\n';
   }
   s->lm = strlen(s->msg);                  /* Insert message length and..  */
   if ((code != 'O') && (code != 'E')) s->stat = 'E'; /* ..valid type field */

   clear_vcb();                             /* Zero the vcb fields          */
   vcb.send.opcode = AP_M_SEND_DATA;        /* MC_SEND_DATA opcode          */
   vcb.send.opext = AP_MAPPED_CONVERSATION; /* MC type extension            */
   vcb.send.conv_id = conv_id;              /* Set conversation_id          */
   memcpy (vcb.send.tp_id, tp_id, sizeof(tp_id));    /* Set tp_id          */
   vcb.send.dlen = M_FIELD + s->lm;         /* Send amount read from file   */
   vcb.send.dptr = anbptr;                  /* Set pointer to data          */
                                            /* in shared segment            */

   APPC_C((long) vcbptr);                   /* Call APPC                    */

   appc_rc_p = vcb.send.primary_rc;
   appc_rc_s = vcb.send.secondary_rc;       /* Save APPC return codes       */
#ifdef TRACE
   fprintf(lf,"Send message: %s",txt);
   fprintf(lf,"Rc_1 = %04x, Rc_2 = %08lx \n",appc_rc_p,appc_rc_s);
#endif
   if (appc_rc_p != AP_OK) {                /* Abnormal retcode ?           */
      if (appc_rc_p == AP_DEALLOC_ABEND)    /* Requester abended            */
         return;                            /* But SERVER is still ok       */
      sys_err = TRUE;                       /* Otherwise, show error status */
      SHOW_ERR();
      return(ERROR);
   }
   return(GOOD);
}

void
DO_MC_SEND_ERROR ()
{
#ifdef TRACE
   fprintf(lf,"SEND_ERROR\n");
#endif
   clear_vcb();                             /* Zero out the vcb             */
   vcb.senderr.opcode = AP_M_SEND_ERROR;    /* Verb-MC_SEND_ERROR           */
   vcb.senderr.opext = AP_MAPPED_CONVERSATION;          /* Set MC type      */
   vcb.senderr.conv_id = conv_id;           /* Set Conversation_id          */
   memcpy (vcb.senderr.tp_id, tp_id, sizeof(tp_id));    /* Set TP_id        */

   APPC_C ((long) vcbptr);                  /* Call APPC                    */

   appc_rc_p = vcb.senderr.primary_rc;
   appc_rc_s = vcb.senderr.secondary_rc;    /* Save APPC return codes       */
   if (appc_rc_p != AP_OK)
   {
      SHOW_ERR();
      sys_err = TRUE;
   }
}

/**************************************************************************/
/*                                                                        */
/*                       OS/2 RELATED FUNCTIONS                           */
/*                                                                        */
/**************************************************************************/
```

```
void
ALLOC_SHARED_BUFFER()
/* APPC requires a data buffer in a shared unnamed segment */
{
   unsigned short selector;               /* Selector from DOSALLOCSEG    */

   dos_rc = DOSALLOCSEG (N_BYTES*2,(unsigned far *)&selector, 1);

   if (dos_rc == 0) {                     /* If there is no error         */
      FP_OFF(anbptr) = 0;                 /* set the offset to zero       */
      FP_SEG(anbptr) = selector;          /* address = Selector:0         */
   }
   else                                   /* Else show dos error          */
   {
      SHOW_DOS_ERROR();                   /* set system to true,          */
      sys_err = TRUE;                     /* zero out the pointer         */
      anbptr = (unsigned char far *)0;
   }
}

void
OPEN_DOS_FILE()
  /* Open the filename sent by the Requester. The filename is an ASCIIZ    */
  /* string pointed to by anbptr+1. (+1 to skip Pascal type LL length byte)*/
{
   unsigned short action;

   eof_flag = FALSE;
   if (direction=='R')        /* partner wants to receive, so we send       */
      dos_rc = DOSOPEN (local_file,       /* addr. of file & path name    */
                  (unsigned far *)&filehandle,
                                          /* addr. of filehandle -        */
                                          /* returned by dos              */
                  (unsigned far *)&action,
                                          /* addr. of action taken by dos */
                  (long)0,                /* file primary allocation      */
                  0,                      /* file attribute               */
                  0x0001,                 /* type of function to be done  */
                  0x0020,                 /* Open mode of the file        */
                  (long)0);               /* Reserved double word         */
   else                       /* partner wants to send, so we receive       */
      dos_rc = DOSOPEN (local_file, (unsigned far *)&filehandle,
              (unsigned far *)&action,(long)cp->il,0,0x0012,0x00a2,(long)0);
#ifdef TRACE
   fprintf(lf,"OPEN_DOS_FILE, ");
   fprintf(lf,"return code = %04x\n",dos_rc);
#endif
   if (dos_rc == 0) bad_filename = FALSE;
   else {                                 /* Show error on bad open       */
      if (dos_rc != 110) SHOW_DOS_ERROR();  /* RC=110 = File not found    */
      bad_filename = TRUE;                /* Flag bad file                */
   }
}

void
CLOSE_DOS_FILE ()
{
   dos_rc = DOSCLOSE(filehandle);         /* Close the file               */
   if (dos_rc != 0) {SHOW_DOS_ERROR(); sys_err = TRUE; return;}
}

int DO_DOSREAD ()
   /* Read the requested file into the buffer in 4092 byte blocks */

{
   int i;
   dos_rc = DOSREAD (filehandle, (char far *) &anbptr[4], N_BYTES-4,
                  (unsigned far *)&bytes_read);
   if (dos_rc != 0) {SHOW_DOS_ERROR(); sys_err = TRUE; return(0);}
   if (bytes_read < N_BYTES-4) eof_flag = TRUE;  /* Are we at EOF ?   */
#ifdef TRACE
   fprintf(lf,"DOS read, %d bytes read\n",bytes_read);
#endif
   return(bytes_read);
}
```

```
int DO_DOSWRITE (len)
   /* Write the received file from the buffer in 4092 byte blocks */
int len;
{
   dos_rc = DOSWRITE(filehandle, (char far *) &anbptr[4], len,
                    (unsigned far *)&bytes_read);
#ifdef TRACE
   fprintf(lf,"DOS WRITE, return code = %04x\n",dos_rc);
#endif
   if (dos_rc != 0)
   {
      SHOW_DOS_ERROR(),
      sys_err = TRUE;
      return(0);
   }
   return(bytes_read);
}

void lfclose()                          /* Subroutine to close log file */
{
#ifdef TRACE
   fclose(lf);
#endif
   return;
}

/* EOF OS2RT.C */
```

Appendix C. LU 6.2 Communication Profiles

AS/400 Network and System Descriptions

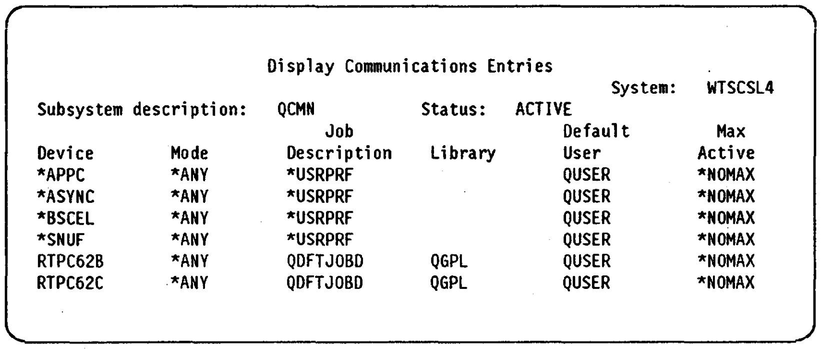
```
                        Display Network Attributes
                                                         System:   WTSCSL4
Current system name . . . . . . . . . . . . . . :   WTSCSL4
  Pending system name . . . . . . . . . . . . . :
Local network ID . . . . . . . . . . . . . . . . :   APPN
Local control point name . . . . . . . . . . . . :   WTSCSL4
Default local location . . . . . . . . . . . . . :   WTSCSL4
Default mode . . . . . . . . . . . . . . . . . . :   BLANK
Maximum number of conversations for a remote
  location . . . . . . . . . . . . . . . . . . . :   64
APPN node type . . . . . . . . . . . . . . . . . :   *NETNODE
Maximum number of intermediate sessions . . . . :   200
Route addition resistance . . . . . . . . . . . :   128
Server network ID/control point name . . . . . . :
```

```
                        Display Network Attributes
                                                         System:   WTSCSL4
Alert status . . . . . . . . . . . . . . . . . . :   *OFF
Alert primary focal point . . . . . . . . . . . :   *NO
Alert default focal point . . . . . . . . . . . :   *NO
Alert logging status . . . . . . . . . . . . . . :   *NONE
Alert controller description . . . . . . . . . . :   *NONE
Message queue . . . . . . . . . . . . . . . . . :   QSYSOPR
  Library . . . . . . . . . . . . . . . . . . . :     QSYS
Output queue . . . . . . . . . . . . . . . . . . :   QPRINT
  Library . . . . . . . . . . . . . . . . . . . :     QGPL
Job action . . . . . . . . . . . . . . . . . . . :   *FILE
Maximum hop count . . . . . . . . . . . . . . . :   16
DDM request access . . . . . . . . . . . . . . . :   *OBJAUT
PC Support request access . . . . . . . . . . . :   *OBJAUT
```

Figure 123. AS/400 Network Attributes

```
                        Display Mode Description

Mode description name . . . . . . :   MODD         RT
Class-of-service . . . . . . . . . :   COS          #CONNECT
Maximum number of sessions . . . . :   MAXSSN       4
Maximum conversations . . . . . . :   MAXCNV       8
Locally controlled sessions . . . :   LCLCTLSSN    2
Pre-established sessions . . . . . :   PREESTSSN    1
Inbound pacing value . . . . . . . :   INPACING     3
Outbound pacing value . . . . . . :   OUTPACING    3
Max length of request unit . . . . :   MAXLENRU     *CALC
Text . . . . . . . . . . . . . . . :   TEXT         Mode Description for RT PC
```

Figure 124. AS/400 Mode Description



```
                        Display Communications Entries
                                                              System:   WTSCSL4
Subsystem description:    QCMN          Status:   ACTIVE
                               Job                          Default          Max
Device         Mode          Description    Library         User            Active
*APPC          *ANY          *USRPRF                        QUSER           *NOMAX
*ASYNC         *ANY          *USRPRF                        QUSER           *NOMAX
*BSCEL         *ANY          *USRPRF                        QUSER           *NOMAX
*SNUF          *ANY          *USRPRF                        QUSER           *NOMAX
RTPC62B        *ANY          QDFTJOBD       QGPL            QUSER           *NOMAX
RTPC62C        *ANY          QDFTJOBD       QGPL            QUSER           *NOMAX
```

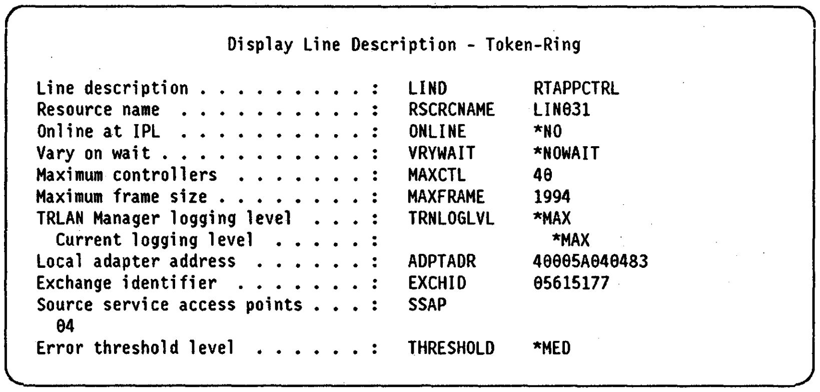
Figure 125. AS/400 Subsystem Description

Token-Ring Connection, RT PC - AS/400 (LU 6.2)

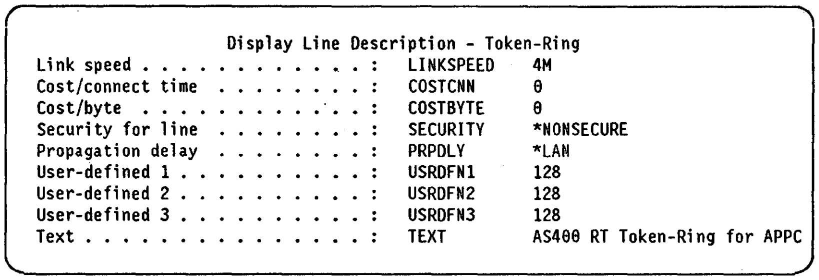
AS/400 Descriptions

```
5728PW1 R01M02  881028                  SEU SOURCE LISTING
SOURCE FILE . . . . . . .  STELLA/QCLSRC
MEMBER  . . . . . . . . .  RTAPPCTR
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7.
PGM
/**********************************************************************/
/* THIS PROGRAM CREATES THE LINE AND CONTROLLER DESCRIPTIONS          */
/* WHICH CAN BE USED TO CONNECT THE AS/400 VIA A Token-Ring           */
/* TO AN RT FOR APPC. DEVICE DESCRIPTIONS ARE AUTOMATICALLY           */
/* CREATED BY THE AS/400.DEVICE DESCRIPTIONS ARE NAMED BY THE AS/400 */
/* GENERICALLY USING THE RMTCPNAME (SEE CONTROLLER DESCRIPTION).      */
/* ANY CHANGES THAT NEED TO BE MADE TO THE Token-Ring OR CONTROLLER  */
/* DESCRIPTIONS CAN BE MADE IN THIS PROGRAM. WHEN RUN, THE            */
/* EXISTING TR, WITH ITS ATTACHED CONTROLLER AND DEVICES, WILL BE     */
/* VARIED OFF AND DELETED (IF THEY EXIST) THEN CREATED.               */
/*                                                                    */
/**********************************************************************/
             MONMSG     MSGID(CPF0000)
             VRYCFG     CFGOBJ(RTAPPCTRC) CFGTYPE(*CTL) STATUS(*OFF)
             VRYCFG     CFGOBJ(RTAPPCTRL) CFGTYPE(*LIN) STATUS(*OFF)
             DLTDEVD    DEVD(RTPC62C*)
             DLTCTLD    CTLD(RTAPPCTRC)
             DLTLIND    LIND(RTAPPCTRL)

             CRTLINTRN  LIND(RTAPPCTRL) RSRCNAME(LIN031) ONLINE(*NO) +
                          ADPTADR(40005A040483) SSAP((04)) +
                          TEXT('AS400 RT Token-Ring for APPC') +
                          TRNLOGLVL(*MAX) THRESHOLD(*MED)
             CRTCTLAPPC CTLD(RTAPPCTRC) LINKTYPE(*TRLAN) ONLINE(*NO) +
                          SWTLINLST(RTAPPCTRL) RMTCPNAME(RTPC62C) +
                          ADPTADR(50005A1A0AE9) CPSSN(*NO) +
                          NODETYPE(*LENNODE) TEXT('AS400 RT Token +
                          Ring for APPC')
ENDPGM
```

Figure 126. AS/400 CL Program for Creating Token-Ring Descriptions

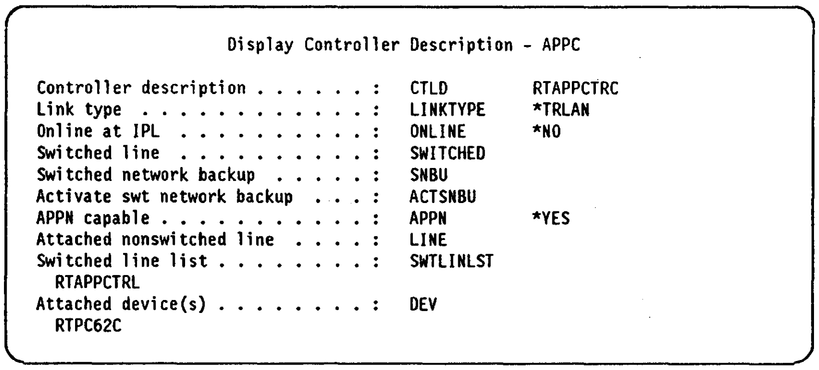


```
                     Display Line Description - Token-Ring

 Line description . . . . . . . . . :   LIND         RTAPPCTRL
 Resource name  . . . . . . . . . . :   RSCRCNAME    LIN031
 Online at IPL  . . . . . . . . . . :   ONLINE       *NO
 Vary on wait . . . . . . . . . . . :   VRYWAIT      *NOWAIT
 Maximum controllers  . . . . . . . :   MAXCTL       40
 Maximum frame size . . . . . . . . :   MAXFRAME     1994
 TRLAN Manager logging level  . . . :   TRNLOGLVL    *MAX
   Current logging level  . . . . . :                  *MAX
 Local adapter address  . . . . . . :   ADPTADR      40005A040483
 Exchange identifier  . . . . . . . :   EXCHID       05615177
 Source service access points . . . :   SSAP
   04
 Error threshold level  . . . . . . :   THRESHOLD    *MED
```

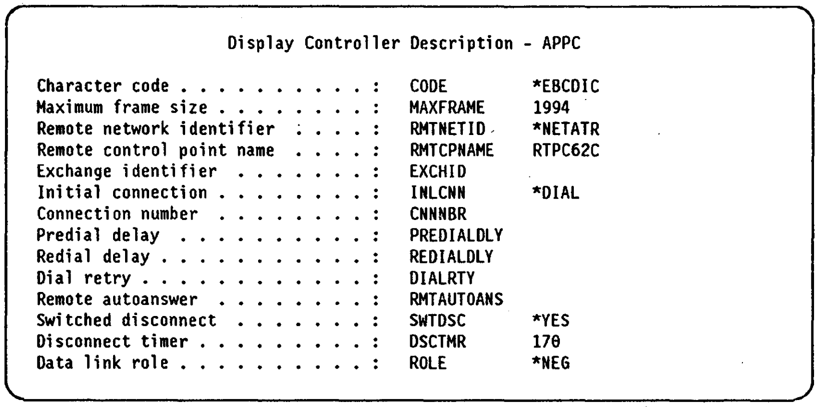


```
                     Display Line Description - Token-Ring
 Link speed . . . . . . . . . . . . :   LINKSPEED    4M
 Cost/connect time  . . . . . . . . :   COSTCNN      0
 Cost/byte  . . . . . . . . . . . . :   COSTBYTE     0
 Security for line  . . . . . . . . :   SECURITY     *NONSECURE
 Propagation delay  . . . . . . . . :   PRPDLY       *LAN
 User-defined 1 . . . . . . . . . . :   USRDFN1      128
 User-defined 2 . . . . . . . . . . :   USRDFN2      128
 User-defined 3 . . . . . . . . . . :   USRDFN3      128
 Text . . . . . . . . . . . . . . . :   TEXT         AS400 RT Token-Ring for APPC
```

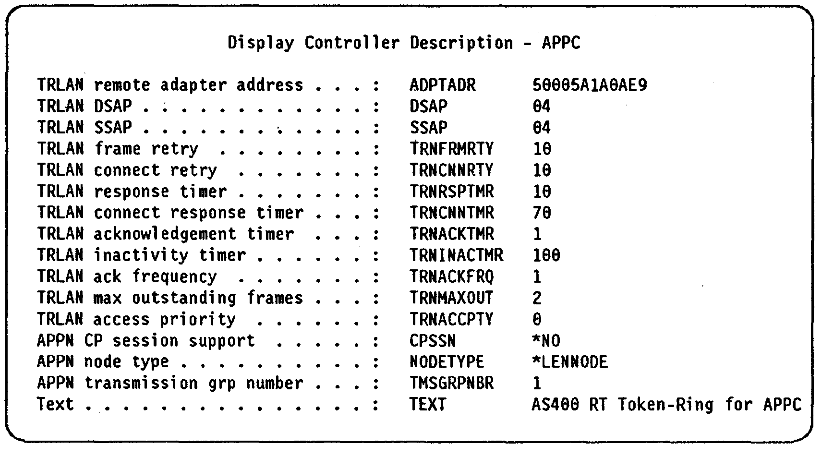
Figure 127. AS/400 Token-Ring Controller Description for APPC



```
                    Display Controller Description - APPC

Controller description . . . . . . :   CTLD         RTAPPCTRC
Link type  . . . . . . . . . . . . :   LINKTYPE     *TRLAN
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Switched line  . . . . . . . . . . :   SWITCHED
Switched network backup  . . . . . :   SNBU
Activate swt network backup  . . . :   ACTSNBU
APPN capable . . . . . . . . . . . :   APPN         *YES
Attached nonswitched line  . . . . :   LINE
Switched line list . . . . . . . . :   SWTLINLST
  RTAPPCTRL
Attached device(s) . . . . . . . . :   DEV
  RTPC62C
```



```
                    Display Controller Description - APPC

Character code . . . . . . . . . . :   CODE         *EBCDIC
Maximum frame size . . . . . . . . :   MAXFRAME     1994
Remote network identifier  . . . . :   RMTNETID     *NETATR
Remote control point name  . . . . :   RMTCPNAME    RTPC62C
Exchange identifier  . . . . . . . :   EXCHID
Initial connection . . . . . . . . :   INLCNN       *DIAL
Connection number  . . . . . . . . :   CNNNBR
Predial delay  . . . . . . . . . . :   PREDIALDLY
Redial delay . . . . . . . . . . . :   REDIALDLY
Dial retry . . . . . . . . . . . . :   DIALRTY
Remote autoanswer  . . . . . . . . :   RMTAUTOANS
Switched disconnect  . . . . . . . :   SWTDSC       *YES
Disconnect timer . . . . . . . . . :   DSCTMR       170
Data link role . . . . . . . . . . :   ROLE         *NEG
```



```
                    Display Controller Description - APPC

TRLAN remote adapter address . . . :   ADPTADR      50005A1A0AE9
TRLAN DSAP . . . . . . . . . . . . :   DSAP         04
TRLAN SSAP . . . . . . . . . . . . :   SSAP         04
TRLAN frame retry  . . . . . . . . :   TRNFRMRTY    10
TRLAN connect retry  . . . . . . . :   TRNCNNRTY    10
TRLAN response timer . . . . . . . :   TRNRSPTMR    10
TRLAN connect response timer . . . :   TRNCNNTMR    70
TRLAN acknowledgement timer  . . . :   TRNACKTMR    1
TRLAN inactivity timer . . . . . . :   TRNINACTMR   100
TRLAN ack frequency  . . . . . . . :   TRNACKFRQ    1
TRLAN max outstanding frames . . . :   TRNMAXOUT    2
TRLAN access priority  . . . . . . :   TRNACCPTY    0
APPN CP session support  . . . . . :   CPSSN        *NO
APPN node type . . . . . . . . . . :   NODETYPE     *LENNODE
APPN transmission grp number . . . :   TMSGRPNBR    1
Text . . . . . . . . . . . . . . . :   TEXT         AS400 RT Token-Ring for APPC
```

Figure 128. AS/400 Token-Ring Line Description for APPC/LU 6.2

IBM RT SNA Services Profiles

```
AS40062_CONNECTION:
type = CONNECTION
profile_name = AS40062
attachment_name = TRAS4A
local_lu_name = RTPC62
network_name = APPN
remote_lu_name = WTSCSL4
stop_connection_on_inactivity = NO
lu_type = LU6.2
interface_type = EXTENDED
remote_tpn_list_name = AS400RTLIST
mode_list_name = RTPCM
node_verification = NO
inactivity_timeout_value = 0
notify = NO
cp_sessions = NO
parallel_sessions = PARALLEL
negotiate_session_limits = YES
```

Figure 129. SNA Profile: RT - AS/400 Connection, LU 6.2, Token-Ring

```
RTPC62_LOCALLU:
type = LOCALLU
profile_name = RTPC62
local_lu_name = RTPC62C
network_name = APPN
lu_type = LU6.2
independent_lu = YES
cp_sessions = NO
tpn_list_name = RTLOCALLIST
local_lu_address = 1
sscp_id = 000000000000
number_of_rows = 1
number_of_columns = 1
destination_address = 0
```

Figure 130. SNA Profile: RT - AS/400 Local LU, LU 6.2, Token-Ring

```
RTPCM_MODELIST:
type = MODELIST
Listname = RTPCM
list_members = RTM


RTM_MODE:
type = MODE
profile_name = RTM
mode_name = RT
maximum_number_of_sessions = 4
minimum_contention_winners = 0
minimum_contention_losers = 0
receive_pacing = 3
send_pacing = 3
maximum_ru_size = 2816
recovery_level = NO_RECONNECT
```

Figure 131. SNA Profile: RT - AS/400 Mode, LU 6.2, Token-Ring

```
AS400RTLIST_REMOTETPNLIST:
type = REMOTETPNLIST
Listname = AS400RTLIST
list_members = AS400RT


AS400RT_REMOTETPN:
type = REMOTETPN
profile_name = AS400RT
tpn_name = AS400RT
tpn_name_hex = C1E2F4F0F0D9E3
pip_data = NO
conversation_type = MAPPED
recovery_level = NO_RECONNECT
sync_level = NONE
```

Figure 132. SNA Profile: RT - AS/400 RTPN, LU 6.2, Token-Ring

```
TRAS4A_ATTACHMENT:
type = ATTACHMENT
profilename = TRAS4A
control_point_profile_name = RTAS400
logical_link_profile_name = TDEFAULT
physical_link_profile_name = TDEFAULT
logical_link_type = TOKEN_RING
stop_attachment_on_inactivity = NO
station_type = NEGOTIABLE
physical_link_type = TOKEN_RING
remote_secondary_station_address = 1
smart_modem_command_sequence =
length_of_command_sequence = 0
call_type = CALL
autolisten = NO
timeout_value = 0
remote_link_name_ethernet =
remote_link_name_token_ring =
remote_link_address_token_ring = 40005A040483
selection_sequence =
length_of_selection_sequence = 0
network_type = SWITCHED
access_routing = LINK_ADDRESS
remote_sap_address = 04
remote_sap_address_range_lower = 04
remote_sap_address_range_upper = EC
virtual_circuit_type = SWITCHED
remote_station_X.25_address =
optional_X.25_facilities = NO
logical_channel_number_of_PVC = 1
reverse_charging = NO
rpoa = NO
default_packet_size = YES
default_window_size = YES
default_throughput_class = YES
closed_user_group = NO
data_network_identification_code =
packet_size_for_received_data = 128
packet_size_for_transmit_data = 128
window_size_for_received_data = 2
window_size_for_transmit_data = 2
throughput_class_for_received_data = 9600
throughput_class_for_transmit_data = 9600
index_to_selected_closed_user_group =
lu_address_registration = NO
local_lu_addresses = 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Figure 133. SNA Profile: RT - AS/400 Attachment, LU 6.2, Token-Ring

```
RTAS400_CONTROLPOINT:
type = CONTROLPOINT
profile_name = RTAS400
xid_node_id = 05C00000
network_name = APPN
cp_name = RTAS400
end_node_status = NO_BIND
locate_gds_support = NO
directory_services_support = NO
resource_registration_support = NO
registration_of_characteristics = NO
topology_database_update_support = NO
request_reply_cp_msus_support = NO
unsolicited_CP_MSUs_supported = NO
parallel_CP_CP_sessions_supported = NO
flow_reduction_sequence_number = 0
```

Figure 134. SNA Profile: RT - AS/400 Control Point, LU 6.2, Token-Ring

SNA/SDLC Connection, RT PC - AS/400 (LU 6.2)

```
5728PW1 R01M02  881028                   SEU SOURCE LISTING
SOURCE FILE . . . . . . .  STELLA/QCLSRC
MEMBER  . . . . . . . . .  RTAPPCNS2
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
              PGM       /* Role NEG XID in the CTL = Blank */
/*********************************************************************/
/* THIS PROGRAM CREATES THE LINE AND CONTROLLER DESCRIPTIONS         */
/* WHICH CAN BE USED TO CONNECT THE AS/400 VIA A NON-SWITCHED SDLC   */
/* LINE TO AN RT FOR APPC. DEVICE DESCRIPTIONS ARE AUTOMATICALLY     */
/* CREATED BY THE AS/400.DEVICE DESCRIPTIONS ARE NAMED BY THE AS/400 */
/* GENERICALLY USING THE RMTCPNAME (SEE CONTROLLER DESCRIPTION).     */
/* ANY CHANGES THAT NEED TO BE MADE TO THE LINE OR CONTROLLER        */
/* DESCRIPTIONS CAN BE MADE IN THIS PROGRAM. WHEN RUN, THE           */
/* EXISTING LINE, WITH ITS ATTACHED CONTROLLER AND DEVICES, WILL BE  */
/* VARIED OFF AND DELETED (IF THEY EXIST) THEN CREATED.              */
/*                                                                   */
/* NOTE : ROLE = NEGOTIABLE                                          */
/*********************************************************************/

             VRYCFG     CFGOBJ(RTAPPCNSL2) CFGTYPE(*LIN) STATUS(*OFF)
             MONMSG     MSGID(CPF9999)

             DLTDEVD    DEVD(RTPC62B*)
             MONMSG     MSGID(CPF9999)

             DLTCTLD    CTLD(RTAPPCNSC2)
             MONMSG     MSGID(CPF9999)

             DLTLIND    LIND(RTAPPCNSL2)
             MONMSG     MSGID(CPF9999)

             CRTLINSDLC LIND(RTAPPCNSL2) RSRCNAME(LIN012) ONLINE(*NO) +
                          ROLE(*NEG) EXCHID(05615177) TEXT('AS400 RT +
                          SDLC for APPC (negotiable)')
             CRTCTLAPPC CTLD(RTAPPCNSC2) LINKTYPE(*SDLC) ONLINE(*NO) +
                          LINE(RTAPPCNSL2) RMTNETID(APPN) +
                          RMTCPNAME(RTPC62B) ROLE(*NEG) STNADR(C1) +
                          CPSSN(*NO) NODETYPE(*LENNODE) TEXT('AS400 +
                          RT SDLC for APPC (negotiable)')

ENDPGM
```

Figure 135. AS/400 CL Program for Creating SNA/SDLC Descriptions

```
                          Display Line Description - SDLC

 Line description . . . . . . . . . :   LIND         RTAPPCNSL2
 Resource name  . . . . . . . . . . :   RSRCNAME     LIN012
 Online at IPL  . . . . . . . . . . :   ONLINE       *NO
 Data link role . . . . . . . . . . :   ROLE         *NEG
 Physical interface . . . . . . . . :   INTERFACE    *RS232V24
 Connection type  . . . . . . . . . :   CNN          *NONSWTPP
 Switched network backup  . . . . . :   SNBU         *NO
 Activate swt network backup  . . . :   ACTSNBU
 Vary on wait . . . . . . . . . . . :   VRYWAIT
 Autocall unit  . . . . . . . . . . :   AUTOCALL
 Attached nonswitched ctl(s)  . . . :   CTL
   RTAPPCNSC2
 Exchange identifier  . . . . . . . :   EXCHID       05615177
 NRZI data encoding . . . . . . . . :   NRZI         *YES
 Maximum controllers  . . . . . . . :   MAXCTL       1
 Line speed . . . . . . . . . . . . :   LINESPEED    9600
 Modem type supported . . . . . . . :   MODEM        *NORMAL
 Modem data rate select . . . . . . :   MODEMRATE    *FULL
```

```
                          Display Line Description - SDLC
 Maximum frame size . . . . . . . . :   MAXFRAME     521
 Error threshold level  . . . . . . :   THRESHOLD    *OFF
 Duplex . . . . . . . . . . . . . . :   DUPLEX       *HALF
 Modulus  . . . . . . . . . . . . . :   MODULUS      8
 Maximum outstanding frames . . . . :   MAXOUT       7
 Inactivity timer . . . . . . . . . :   INACTTMR     300
 Poll response delay  . . . . . . . :   POLLRSPDLY   0
 Nonproductive receive timer  . . . :   NPRDRCVTMR   320
 Idle timer . . . . . . . . . . . . :   IDLTMR       30
 Connect poll timer . . . . . . . . :   CNNPOLLTMR   30
 Poll cycle pause . . . . . . . . . :   POLLPAUSE    0
 Frame retry  . . . . . . . . . . . :   FRAMERTY     7
```

```
                          Display Line Description - SDLC
 Link speed . . . . . . . . . . . . :   LINKSPEED    9600
 Cost/connect time  . . . . . . . . :   COSTCNN      0
 Cost/byte  . . . . . . . . . . . . :   COSTBYTE     0
 Security for line  . . . . . . . . :   SECURITY     *NONSECURE
 Propagation delay  . . . . . . . . :   PRPDLY       *TELEPHONE
 User-defined 1 . . . . . . . . . . :   USRDFN1      128
 User-defined 2 . . . . . . . . . . :   USRDFN2      128
 User-defined 3 . . . . . . . . . . :   USRDFN3      128
 Text . . . . . . . . . . . . . . . :   TEXT         AS400 RT SDLC for APPC
                                                     (negotiable)
```

Figure 136. AS/400 SNA/SDLC Line Description

```
                    Display Controller Description - APPC

Controller description . . . . . . :   CTLD          RTAPPCNSC2
Link type  . . . . . . . . . . . . :   LINKTYPE      *SDLC
Online at IPL  . . . . . . . . . . :   ONLINE        *NO
Switched line  . . . . . . . . . . :   SWITCHED      *NO
Switched network backup  . . . . . :   SNBU          *NO
Activate swt network backup  . . . :   ACTSNBU
APPN capable . . . . . . . . . . . :   APPN          *YES
Attached nonswitched line  . . . . :   LINE          RTAPPCNSL2
Switched line list . . . . . . . . :   SWTLINLST
Attached device(s) . . . . . . . . :   DEV
```

```
                    Display Controller Description - APPC
Character code . . . . . . . . . . :   CODE          *EBCDIC
Maximum frame size . . . . . . . . :   MAXFRAME      521
Remote network identifier  . . . . :   RMTNETID      APPN
Remote control point name  . . . . :   RMTCPNAME     RTPC62B
Exchange identifier  . . . . . . . :   EXCHID        05C00000
Initial connection . . . . . . . . :   INLCNN
Connection number  . . . . . . . . :   CNNNBR
Predial delay  . . . . . . . . . . :   PREDIALDLY
Redial delay . . . . . . . . . . . :   REDIALDLY
Dial retry . . . . . . . . . . . . :   DIALRTY
Remote autoanswer  . . . . . . . . :   RMTAUTOANS
Switched disconnect  . . . . . . . :   SWTDSC
Disconnect timer . . . . . . . . . :   DSCTMR
Data link role . . . . . . . . . . :   ROLE          *NEG
```

```
                    Display Controller Description - APPC
Station address  . . . . . . . . . :   STNADR        C1
SDLC poll priority . . . . . . . . :   POLLPTY       *NO
SDLC poll limit  . . . . . . . . . :   POLLLMT       0
SDLC connect poll retry  . . . . . :   CNNPOLLRTY    *NOMAX
SDLC NRM poll timer  . . . . . . . :   NRMPOLLTMR    0
SDLC NDM poll timer  . . . . . . . :   NDMPOLLTMR    *CALC
APPN CP session support  . . . . . :   CPSSN         *NO
APPN node type . . . . . . . . . . :   NODETYPE      *LENNODE
APPN transmission grp number . . . :   TMSGRPNBR     1
Text . . . . . . . . . . . . . . . :   TEXT          AS400 RT SDLC for APPC
                                                     (negotiable)
```

Figure 137. AS/400 SNA/SDLC Controller Description

```
AS40062_CONNECTION:
type = CONNECTION
profile_name = AS40062
attachment_name = SDLCAS4N
local_lu_name = RTPC62
network_name = APPN
remote_lu_name = WTSCSL4
top_connection_on_inactivity = NO
lu_type = LU6.2
interface_type = EXTENDED
remote_tpn_list_name = AS400RTLIST
mode_list_name = RTPCM
node_verification = NO
inactivity_timeout_value = 0
notify = NO
cp_sessions = NO
parallel_sessions = PARALLEL
negotiate_session_limits = YES
```

Figure 138. SNA Profile: RT - AS/400 Connection, LU 6.2, SNA/SDLC

```
RTPC62_LOCALLU:
type = LOCALLU
profile_name = RTPC62
local_lu_name = RTPC62B
network_name = APPN
lu_type = LU6.2
independent_lu = YES
cp_sessions = NO
tpn_list_name = RTLOCALLIST
local_lu_address = 10
sscp_id = 050000000000
number_of_rows = 1
number_of_columns = 1
destination_address = 0
```

Figure 139. SNA Profile: RT - AS/400 Local LU, LU 6.2, SNA/SDLC

```
RTPCM_MODELIST:
type = MODELIST
Listname = RTPCM
list_members = RTM


RTM_MODE:
type = MODE
profile_name = RTM
mode_name = RT
maximum_number_of_sessions = 4
minimum_contention_winners = 0
minimum_contention_losers = 0
receive_pacing = 3
send_pacing = 3
maximum_ru_size = 2816
recovery_level = NO_RECONNECT
```

Figure 140. SNA Profile: RT - AS/400 Mode, LU 6.2, SNA/SDLC

```
AS400RTLIST_REMOTETPNLIST:
type = REMOTETPNLIST
Listname = AS400RTLIST
list_members = AS400RT


AS400RT_REMOTETPN:
type = REMOTETPN
profile_name = AS400RT
tpn_name = AS400RT
tpn_name_hex = C1E2F4F0F0D9E3
pip_data = NO
conversation_type = MAPPED
recovery_level = NO_RECONNECT
sync_level = NONE
```

Figure 141. SNA Profile: RT - AS/400 RTPN, LU 6.2, SNA/SDLC

```
SDLCAS4N_ATTACHMENT:
type = ATTACHMENT
profilename = SDLCAS4N
control_point_profile_name = RTAS400
logical_link_profile_name = SDLCAS4L
physical_link_profile_name = SDLCAS4P
logical_link_type = SDLC
stop_attachment_on_inactivity = NO
station_type = NEGOTIABLE
physical_link_type = RS232
remote_secondary_station_address = 193
smart_modem_command_sequence =
length_of_command_sequence = 0
call_type = LISTEN
autolisten = NO
timeout_value = 0
remote_link_name_ethernet =
remote_link_name_token_ring =
remote_link_address_token_ring = 000000000000
selection_sequence =
length_of_selection_sequence = 0
network_type = SWITCHED
access_routing = LINK_NAME
remote_sap_address = 04
remote_sap_address_range_lower = 04
remote_sap_address_range_upper = EC
virtual_circuit_type = SWITCHED
remote_station_X.25_address =
optional_X.25_facilities = NO
logical_channel_number_of_PVC = 1
reverse_charging = NO
rpoa = NO
default_packet_size = YES
default_window_size = YES
default_throughput_class = YES
closed_user_group = NO
data_network_identification_code =
packet_size_for_received_data = 128
packet_size_for_transmit_data = 128
window_size_for_received_data = 2
window_size_for_transmit_data = 2
throughput_class_for_received_data = 9600
throughput_class_for_transmit_data = 9600
index_to_selected_closed_user_group =
lu_address_registration = NO
local_lu_addresses = 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Figure 142. SNA Profile: RT - AS/400 Attachment, LU 6.2, SNA/SDLC

```
RTAS400_CONTROLPOINT:
type = CONTROLPOINT
profile_name = RTAS400
xid_node_id = 05615177
network_name = APPN
cp_name = RTAS400
end_node_status = NO_BIND
locate_gds_support = NO
directory_services_support = NO
resource_registration_support = NO
registration_of_characteristics = NO
topology_database_update_support = NO
request_reply_cp_msus_support = NO
unsolicited_CP_MSUs_supported = NO
parallel_CP_CP_sessions_supported = NO
flow_reduction_sequence_number = 0
```

Figure 143. SNA Profile: RT - AS/400 Control Point, LU 6.2, SNA/SDLC

```
SDLCAS4L_SDLC:
type = SDLC
profile_name = SDLCAS4L
station_type = NEGOTIABLE
seriel_encoding = NRZI
transmit_window_count = 7
retransmit_count = 10
retransmit_threshold = 10
drop_link_on_inactivity = YES
force_disconnect_timeout = 120
link_trace = TRACE_OFF
trace_entry_size = SHORT
link_type = POINT_TO_POINT
local_secondary_station_address = 1
secondary_inactivity_timeout = 30
primary_repoll_timeout = 30
primary_repoll_count = 15
primary_repoll_threshold = 10
primary_idle_list_timeout = 10
primary_slow_list_timeout = 1
maximum_i_field = SYSTEM_DEFINED
maximum_i_field_size = 265
```

Figure 144. SNA Profile: RT - AS/400 Logical Link, LU 6.2, SNA/SDLC

```
SDLCAS4P_RS232:
type = RS232
profile_name = SDLCAS4P
device_name = sdlcllc0
request_to_send = CONTINUOUS
bit_clock = EXTERNAL
switched_network_backup = BACKUP_OFF
network_type = NONSWITCHED
transmission_rate = 1200
data_rate_select = FULL
call_type = LISTEN
autocall_listen = NO
call_override = NO
answer_mode = MANUAL
dtr_control = DTR
```

Figure 145. SNA Profile: RT - AS/400 Physical Link, LU 6.2, SNA/SDLC

IBM RT to IBM RT Profiles

Token-Ring Connection, RT PC - RT PC (LU 6.2)

IBM RT SNA Services Profiles

```
RTRT_CONNECTION:
type = CONNECTION
profile_name = RTRT
attachment_name = TRRTA
local_lu_name = RTPC62
network_name = APPN
remote_lu_name = RTPC62B
stop_connection_on_inactivity = NO
lu_type = LU6.2
interface_type = EXTENDED
remote_tpn_list_name = RTRTREMOTELIST
mode_list_name = RTPCM
node_verification = NO
inactivity_timeout_value = 0
notify = NO
cp_sessions = NO
parallel_sessions = PARALLEL
negotiate_session_limits = YES
```

Figure 146. SNA Profile: RT - RT Connection, LU 6.2, Token-Ring

```
RTPC62_LOCALLU:
type = LOCALLU
profile_name = RTPC62
local_lu_name = RTPC62C
network_name = APPN
lu_type = LU6.2
independent_lu = YES
cp_sessions = NO
tpn_list_name = RTLOCALLIST
local_lu_address = 1
sscp_id = 000000000000
number_of_rows = 1
number_of_columns = 1
destination_address = 0
```

Figure 147. SNA Profile: RT - RT Local LU, LU 6.2, Token-Ring

```
RTPCM_MODELIST:
type = MODELIST
Listname = RTPCM
list_members = RTM


RTM_MODE:
type = MODE
profile_name = RTM
mode_name = RT
maximum_number_of_sessions = 4
minimum_contention_winners = 0
minimum_contention_losers = 0
receive_pacing = 3
send_pacing = 3
maximum_ru_size = 2816
recovery_level = NO_RECONNECT
```

Figure 148. SNA Profile: RT - RT Mode, LU 6.2, Token-Ring

```
RTLOCALLIST_TPNLIST:
type = TPNLIST
Listname = RTLOCALLIST
list_members = RTLOCAL


RTLOCAL_TPN:
type = TPN
profile_name = RTLOCAL
tpn_name = RTRT
tpn_name_hex = D9E3D9E3
conversation_type = MAPPED
pip_data = NO
sync_level = EITHER
recovery_level = NO_RECONNECT
path_to_server_program = /bin/RTRT
program_to_execute = RTRT
home_directory = /bin
multiple_instances = NO
user_id = 0
group_id = 0
umask = 000
maximum_file_size = 8192
server_synonym_name = svr
subserver_types = NO
unique_server_profile_name = svr
external_notify = NO
restart_action = ONCE
communication_type = SIGNALS
stdin = /dev/console
stdout = /dev/console
stderr = /dev/console
subfields = 0
notify_ipc_queue_key = 0
communication_ipc_queue_key = 0
```

Figure 149. SNA Profile: RT - RT Local TPN, LU 6.2, Token-Ring

```
RTRTREMOTELIST_REMOTETPNLIST:
type = REMOTETPNLIST
Listname = RTRTREMOTELIST
list_members = RTRTREMOTE


RTRTREMOTE_REMOTETPN:
type = REMOTETPN
profile_name = RTRTREMOTE
tpn_name = RTRT
tpn_name_hex = D9E3D9E3
pip_data = NO
conversation_type = MAPPED
recovery_level = NO_RECONNECT
sync_level = CONFIRM
```

Figure 150. SNA Profile: RT - RT Remote TPN, LU 6.2, Token-Ring

```
TRRTA_ATTACHMENT:
type = ATTACHMENT
profilename = TRRTA
control_point_profile_name = RTRTCP
logical_link_profile_name = TDEFAULT
physical_link_profile_name = TDEFAULT
logical_link_type = TOKEN_RING
stop_attachment_on_inactivity = NO
station_type = SECONDARY
physical_link_type = TOKEN_RING
remote_secondary_station_address = 1
smart_modem_command_sequence =
length_of_command_sequence = 0
call_type = LISTEN
autolisten = YES
timeout_value = 0
remote_link_name_ethernet =
remote_link_name_token_ring =
remote_link_address_token_ring = 50005A1AFAD9
selection_sequence =
length_of_selection_sequence = 0
network_type = SWITCHED
access_routing = LINK_ADDRESS
remote_sap_address = 04
remote_sap_address_range_lower = 04
remote_sap_address_range_upper = EC
virtual_circuit_type = SWITCHED
remote_station_X.25_address =
optional_X.25_facilities = NO
logical_channel_number_of_PVC = 1
reverse_charging = NO
rpoa = NO
default_packet_size = YES
default_window_size = YES
default_throughput_class = YES
closed_user_group = NO
data_network_identification_code =
packet_size_for_received_data = 128
packet_size_for_transmit_data = 128
window_size_for_received_data = 2
window_size_for_transmit_data = 2
throughput_class_for_received_data = 9600
throughput_class_for_transmit_data = 9600
index_to_selected_closed_user_group =
lu_address_registration = NO
local_lu_addresses = 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Figure 151. SNA Profile: RT - RT Attachment, LU 6.2, Token-Ring

```
RTRTCP_CONTROLPOINT:
type = CONTROLPOINT
profile_name = RTRTCP
xid_node_id = 05C00000
network_name = APPN
cp_name = RTRTCP
end_node_status = NO_BIND
locate_gds_support = NO
directory_services_support = NO
resource_registration_support = NO
registration_of_characteristics = NO
topology_database_update_support = NO
request_reply_cp_msus_support = NO
unsolicited_CP_MSUs_supported = NO
parallel_CP_CP_sessions_supported = NO
flow_reduction_sequence_number = 0
```

Figure 152. SNA Profile: RT - RT Control Point, LU 6.2, Token-Ring

X.25 Connection, RT PC - RT PC (LU 6.2)

Only the SNA Services profiles that are changed from default are listet here. Remember that these are the profiles for one of the two IBM RTs communicating over X.25. The profiles on the other machine are almost identical, except for the X.25 addresses and LU names specified. If you compare these profile with the profiles for LU 6.2 over Token-Ring and over SNA/SDLC link, you will see, that many of the profiles are identical, for example the local and remote transaction program profiles. These are independent of the physical network. The only profiles that are really different are the Attachment Profile and the Data Link Profiles (logical link control and physical link control). In this example, the machine with the profiles below was configured as a ***Primary*** station and the other station was configured as a ***Secondary*** station.

```
RTRT_CONNECTION:
type = CONNECTION
profile_name = RTRT
attachment_name = X25A
local_lu_name = X25LOCAL
network_name = APPN
remote_lu_name = RTPC2
stop_connection_on_inactivity = NO
lu_type = LU6.2
interface_type = EXTENDED
remote_tpn_list_name = RTRTREMOTELIST
mode_list_name = RTPCM
node_verification = NO
inactivity_timeout_value = 0
notify = NO
cp_sessions = NO
parallel_sessions = PARALLEL
negotiate_session_limits = YES
```

Figure 153. X.25 Connection Profile, RT PC - RT PC (LU 6.2)

```
X25LOCAL_LOCALLU:
type = LOCALLU
profile_name = X25LOCAL
local_lu_name = RTPC1
network_name = APPN
lu_type = LU6.2
independent_lu = YES
cp_sessions = NO
tpn_list_name = RTRTLOCALLIST
local_lu_address = 1
sscp_id = 000000000000
number_of_rows = 1
number_of_columns = 1
destination_address = 0
```

Figure 154. X.25 Local LU Profile, RT PC - RT PC (LU 6.2)

```
RTM_MODE:
type = MODE
profile_name = RTM
mode_name = RT
maximum_number_of_sessions = 20
minimum_contention_winners = 0
minimum_contention_losers = 0
receive_pacing = 3
send_pacing = 3
maximum_ru_size = 1024
recovery_level = NO_RECONNECT
```

Figure 155 (Part 1 of 2). X.25 Mode Profile and Mode List, RT PC - RT PC (LU 6.2)

```
RTPCM_MODELIST:
type = MODELIST
Listname = RTPCM
list_members = RTM
```

Figure 155 (Part 2 of 2). X.25 Mode Profile and Mode List, RT PC - RT PC (LU 6.2)

```
RTRTLOCAL_TPN:
type = TPN
profile_name = RTRTLOCAL
tpn_name = RTRT
tpn_name_hex = D9E3D9E3
conversation_type = MAPPED
pip_data = NO
sync_level = NONE
recovery_level = NO_RECONNECT
path_to_server_program = /bin/RTRT
program_to_execute = svr
home_directory = /bin
multiple_instances = NO
user_id = 0
group_id = 0
umask = 000
maximum_file_size = 8192
server_synonym_name = svr
subserver_types = NO
unique_server_profile_name = svr
external_notify = NO
restart_action = ONCE
communication_type = SIGNALS
stdin = /dev/null
stdout = /dev/null
stderr = /dev/null
subfields = 0
notify_ipc_queue_key = 0
communication_ipc_queue_key = 0
```

Figure 156 (Part 1 of 2). X.25 Local TPN profile and List, RT PC - RT PC (LU 6.2)

```
RTRTLOCALLIST_TPNLIST:
type = TPNLIST
Listname = RTRTLOCALLIST
list_members = RTRTLOCAL
```

Figure 156 (Part 2 of 2). X.25 Local TPN profile and List, RT PC - RT PC (LU 6.2)

```
RTRTREMOTE_REMOTETPN:
type = REMOTETPN
profile_name = RTRTREMOTE
tpn_name = RTRT
tpn_name_hex = D9E3D9E3
pip_data = NO
conversation_type = MAPPED
recovery_level = NO_RECONNECT
sync_level = NONE
```

Figure 157 (Part 1 of 2). X.25 Remote TPN profile and List, RT PC - RT PC (LU 6.2)

```
RTRTREMOTELIST_REMOTETPNLIST:
type = REMOTETPNLIST
Listname = RTRTREMOTELIST
list_members = RTRTREMOTE
```

Figure 157 (Part 2 of 2). X.25 Remote TPN profile and List, RT PC - RT PC (LU 6.2)

```
X25CONT_CONTROLPOINT:
type = CONTROLPOINT
profile_name = X25CONT
xid_node_id = 05C00000
network_name = APPN
cp_name = RTPC1
end_node_status = NO_BIND
locate_gds_support = NO
directory_services_support = NO
resource_registration_support = NO
registration_of_characteristics = NO
topology_database_update_support = NO
request_reply_cp_msus_support = NO
unsolicited_CP_MSUs_supported = NO
parallel_CP_CP_sessions_supported = NO
flow_reduction_sequence_number = 0
```

Figure 158. X.25 Control Point Profile, RT PC - RT PC (LU 6.2)

```
X25A_ATTACHMENT:
type = ATTACHMENT
profilename = X25A
control_point_profile_name = X25CONT
logical_link_profile_name = X25L
physical_link_profile_name = X25P
logical_link_type = QLLC
stop_attachment_on_inactivity = NO
station_type = SECONDARY
physical_link_type = X.25
remote_secondary_station_address = 1
smart_modem_command_sequence =
length_of_command_sequence = 0
call_type = CALL
autolisten = NO
timeout_value = 0
remote_link_name_ethernet =
remote_link_name_token_ring =
remote_link_address_token_ring = 000000000000
selection_sequence =
length_of_selection_sequence = 0
network_type = SWITCHED
access_routing = LINK_NAME
remote_sap_address = 04
remote_sap_address_range_lower = 04
remote_sap_address_range_upper = EC
virtual_circuit_type = SWITCHED
remote_station_X.25_address = 3106008381
optional_X.25_facilities = NO
logical_channel_number_of_PVC = 1
reverse_charging = NO
rpoa = NO
default_packet_size = YES
default_window_size = YES
default_throughput_class = YES
closed_user_group = NO
data_network_identification_code =
packet_size_for_received_data = 128
packet_size_for_transmit_data = 128
window_size_for_received_data = 2
window_size_for_transmit_data = 2
throughput_class_for_received_data = 9600
throughput_class_for_transmit_data = 9600
index_to_selected_closed_user_group =
lu_address_registration = NO
local_lu_addresses = 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Figure 159. X.25 Attachment Profile, RT PC - RT PC (LU 6.2)

```
X25L_QLLCLOGICAL:
type = QLLCLOGICAL
profile_name = X25L
station_type = PRIMARY
negotiable = NO
drop_link_on_inactivity = NO
force_disconnect_timeout = 100
link_trace = TRACE_OFF
trace_entry_size = SHORT
secondary_inactivity_timeout = 60
primary_repoll_timeout = 10
primary_repoll_count = 10
maximum_i_field = SYSTEM_DEFINED
maximum_i_field_size = 1417
```

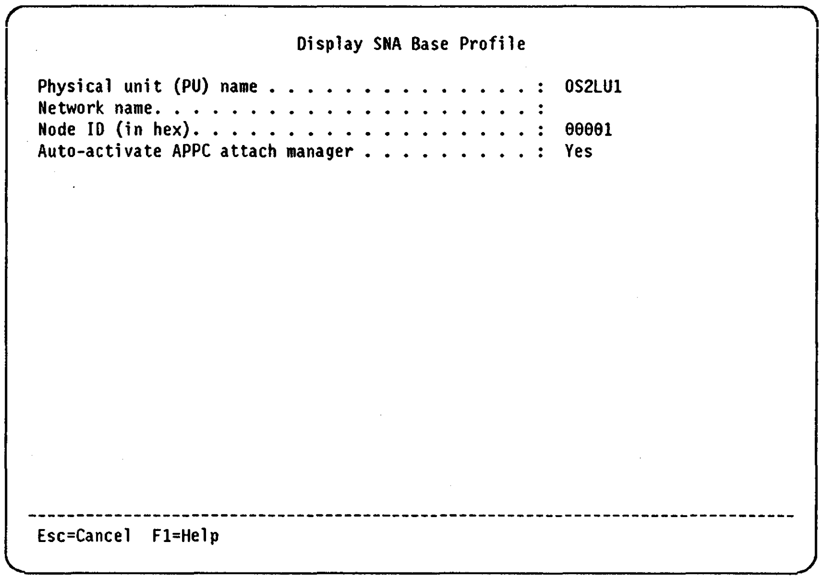
Figure 160. X.25 Logical Link Profile, RT PC - RT PC (LU 6.2)

```
X25P_X.25PHYSICAL:
type = X.25PHYSICAL
profile_name = X25P
device_name = qllc0
maximum_link_stations = 20
local_x25_network_address = 3106001904
```

Figure 161. X.25 Physical Link Profile, RT PC - RT PC (LU 6.2)

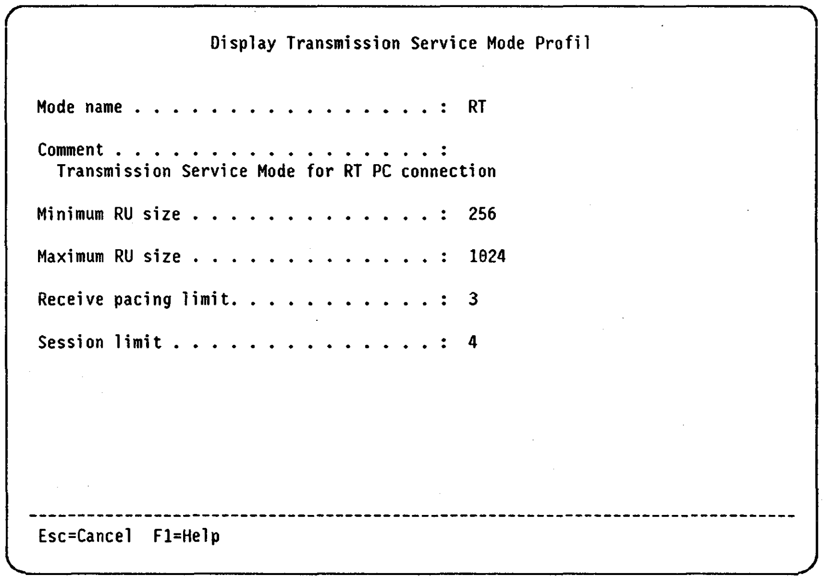
Operating System/2 Profiles

Operating System/2 Common Profiles



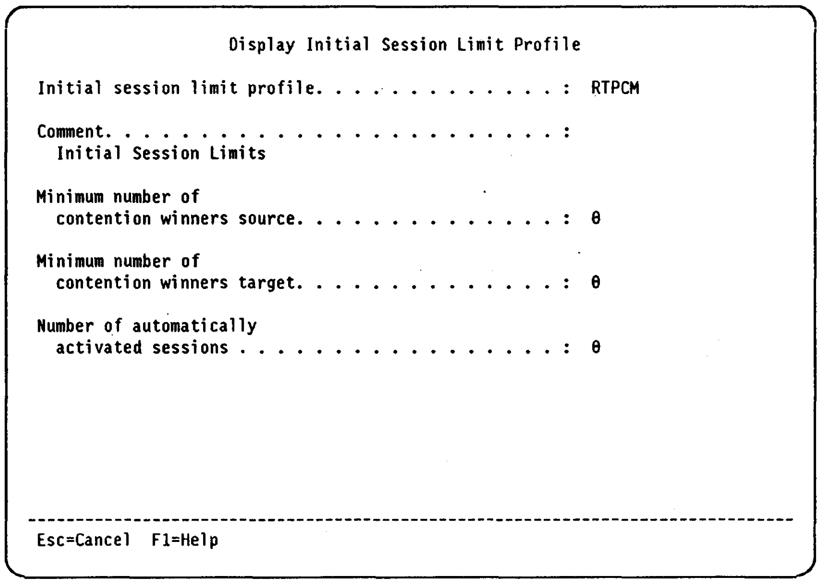
```
                         Display SNA Base Profile

Physical unit (PU) name . . . . . . . . . . . . . . . :  OS2LU1
Network name. . . . . . . . . . . . . . . . . . . . . :
Node ID (in hex). . . . . . . . . . . . . . . . . . . :  00001
Auto-activate APPC attach manager . . . . . . . . . . :  Yes

--------------------------------------------------------------------------------
Esc=Cancel  F1=Help
```

Figure 162. Operating System/2 SNA Base Profile



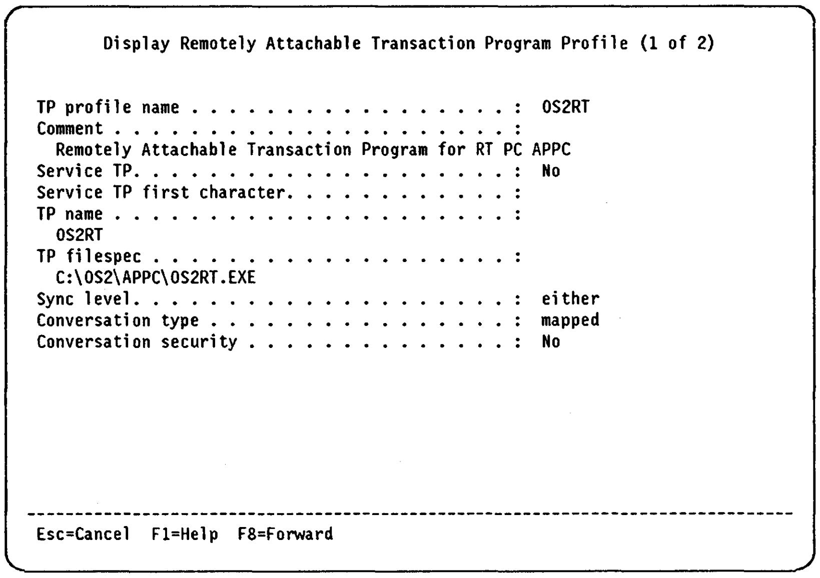
```
                 Display Transmission Service Mode Profil

Mode name . . . . . . . . . . . . . . . . . :  RT

Comment . . . . . . . . . . . . . . . . . . :
  Transmission Service Mode for RT PC connection

Minimum RU size . . . . . . . . . . . . . . :  256

Maximum RU size . . . . . . . . . . . . . . :  1024

Receive pacing limit. . . . . . . . . . . . :  3

Session limit . . . . . . . . . . . . . . . :  4

--------------------------------------------------------------------------------
Esc=Cancel  F1=Help
```

Figure 163. Operating System/2 Transmission Service Mode Profil. The same profile is used for SDLC and Token-Ring connections.



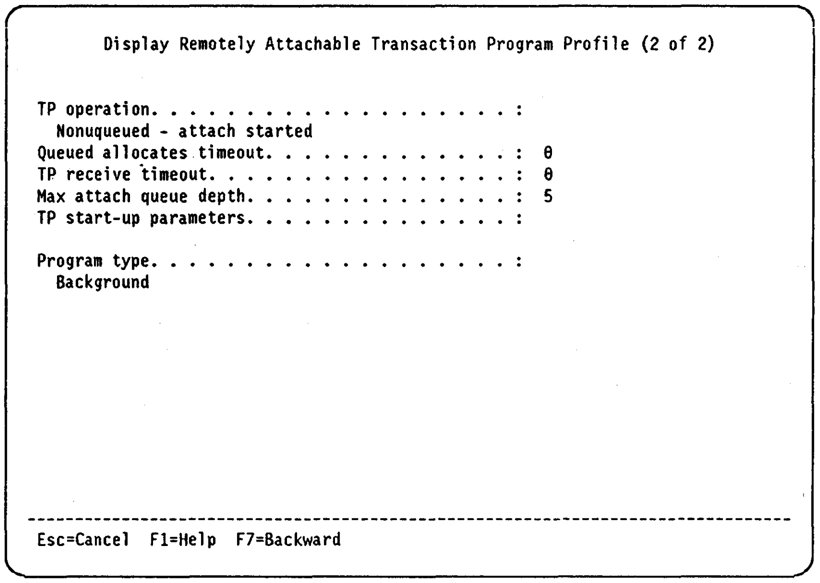
```
                 Display Initial Session Limit Profile

Initial session limit profile. . . . . . . . . . . . . :  RTPCM

Comment. . . . . . . . . . . . . . . . . . . . . . . . :
  Initial Session Limits

Minimum number of
  contention winners source. . . . . . . . . . . . . . :  0

Minimum number of
  contention winners target. . . . . . . . . . . . . . :  0

Number of automatically
  activated sessions . . . . . . . . . . . . . . . . . :  0

--------------------------------------------------------------------------------
Esc=Cancel  F1=Help
```

Figure 164. Operating System/2 Initial Session Limit Profile. The same profile is used for SDLC and Token-Ring connections.



```
       Display Remotely Attachable Transaction Program Profile (1 of 2)


TP profile name . . . . . . . . . . . . . . . . . . :  OS2RT
Comment . . . . . . . . . . . . . . . . . . . . . . :
  Remotely Attachable Transaction Program for RT PC APPC
Service TP. . . . . . . . . . . . . . . . . . . . . :  No
Service TP first character. . . . . . . . . . . . . :
TP name . . . . . . . . . . . . . . . . . . . . . . :
  OS2RT
TP filespec . . . . . . . . . . . . . . . . . . . . :
  C:\OS2\APPC\OS2RT.EXE
Sync level. . . . . . . . . . . . . . . . . . . . . :  either
Conversation type . . . . . . . . . . . . . . . . . :  mapped
Conversation security . . . . . . . . . . . . . . . :  No

--------------------------------------------------------------------------------
Esc=Cancel  F1=Help  F8=Forward
```

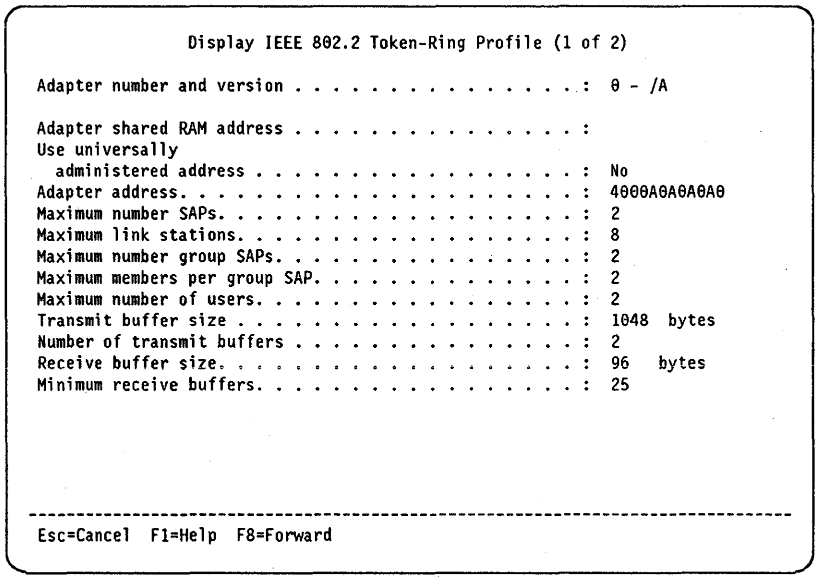
Figure 165. Operating System/2 Remotely Attachable Transaction Program Profile (1 of 2). The same profile is used for SDLC and Token-Ring connections.



```
        Display Remotely Attachable Transaction Program Profile (2 of 2)


TP operation. . . . . . . . . . . . . . . . . . . . :
  Nonuqueued - attach started
Queued allocates timeout. . . . . . . . . . . . . . :  0
TP receive timeout. . . . . . . . . . . . . . . . . :  0
Max attach queue depth. . . . . . . . . . . . . . . :  5
TP start-up parameters. . . . . . . . . . . . . . . :

Program type. . . . . . . . . . . . . . . . . . . . :
  Background

--------------------------------------------------------------------------------
Esc=Cancel  F1=Help  F7=Backward
```

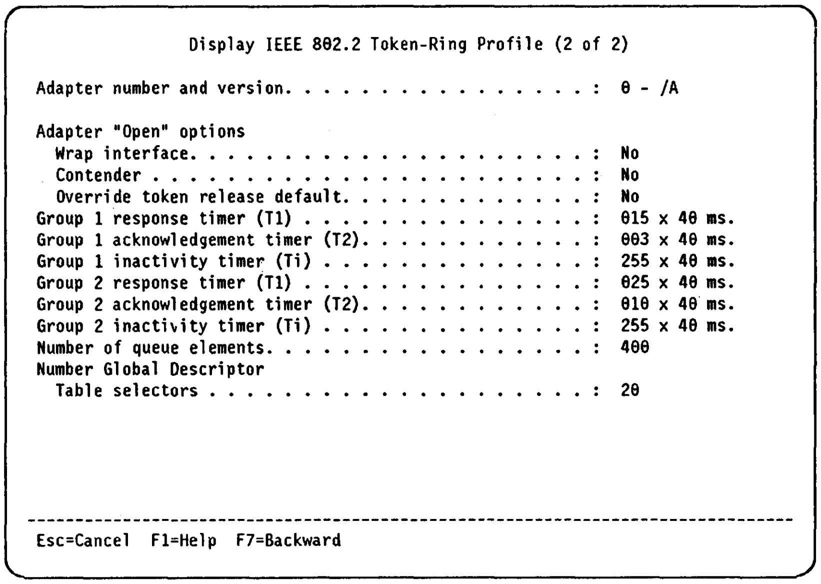
Figure 166. Operating System/2 Remotely Attachable Transaction Program Profile (2 of 2). The same profile is used for SDLC and Token-Ring connections.

Operating System/2 Token-Ring Profiles



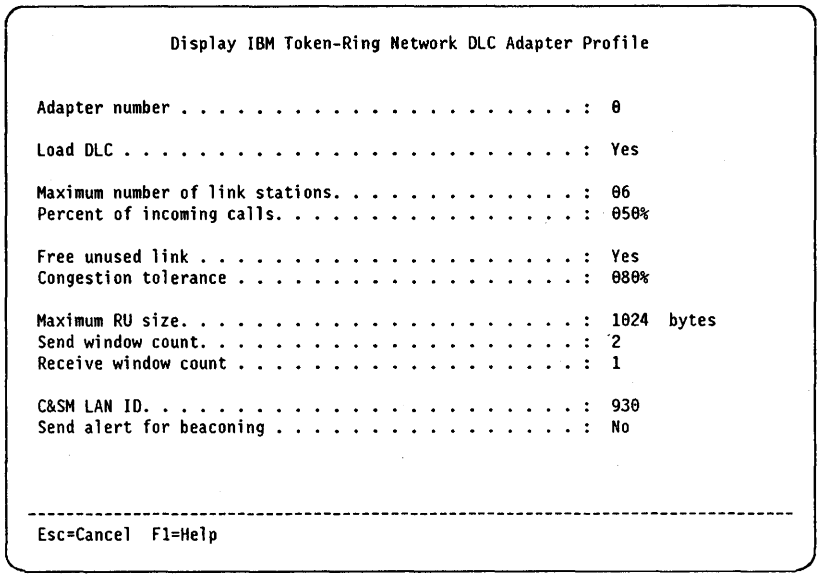
```
                  Display IEEE 802.2 Token-Ring Profile (1 of 2)

Adapter number and version . . . . . . . . . . . . . . . :  0 - /A

Adapter shared RAM address . . . . . . . . . . . . . . . :
Use universally
  administered address . . . . . . . . . . . . . . . . . :  No
Adapter address. . . . . . . . . . . . . . . . . . . . . :  4000A0A0A0A0
Maximum number SAPs. . . . . . . . . . . . . . . . . . . :  2
Maximum link stations. . . . . . . . . . . . . . . . . . :  8
Maximum number group SAPs. . . . . . . . . . . . . . . . :  2
Maximum members per group SAP. . . . . . . . . . . . . . :  2
Maximum number of users. . . . . . . . . . . . . . . . . :  2
Transmit buffer size . . . . . . . . . . . . . . . . . . :  1048  bytes
Number of transmit buffers . . . . . . . . . . . . . . . :  2
Receive buffer size. . . . . . . . . . . . . . . . . . . :  96   bytes
Minimum receive buffers. . . . . . . . . . . . . . . . . :  25

--------------------------------------------------------------------------------
Esc=Cancel  F1=Help  F8=Forward
```

Figure 167. Operating System/2 IEEE 802.2 Token-Ring Profile (1 of 2)



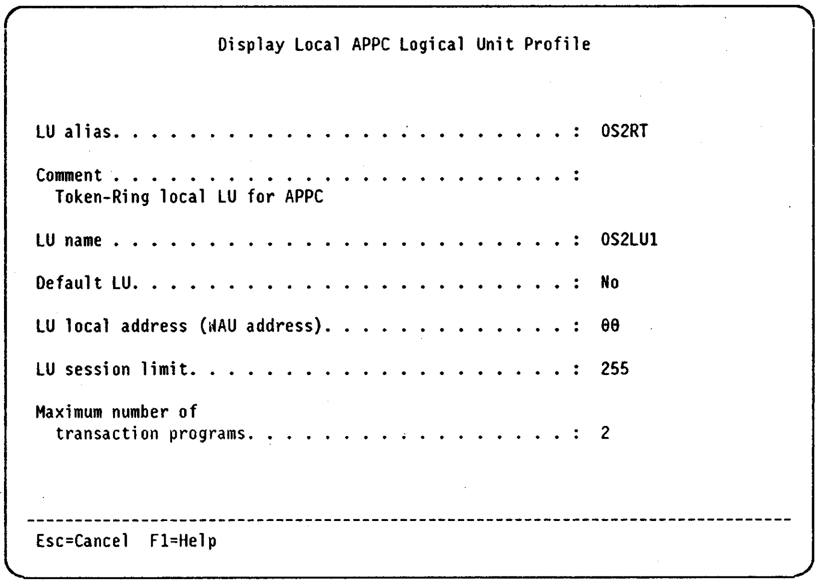
```
                Display IEEE 802.2 Token-Ring Profile (2 of 2)

Adapter number and version. . . . . . . . . . . . . . . . :  0 - /A

Adapter "Open" options
  Wrap interface. . . . . . . . . . . . . . . . . . . . . :  No
  Contender . . . . . . . . . . . . . . . . . . . . . . . :  No
  Override token release default. . . . . . . . . . . . . :  No
Group 1 response timer (T1) . . . . . . . . . . . . . . . :  015 x 40 ms.
Group 1 acknowledgement timer (T2). . . . . . . . . . . . :  003 x 40 ms.
Group 1 inactivity timer (Ti) . . . . . . . . . . . . . . :  255 x 40 ms.
Group 2 response timer (T1) . . . . . . . . . . . . . . . :  025 x 40 ms.
Group 2 acknowledgement timer (T2). . . . . . . . . . . . :  010 x 40 ms.
Group 2 inactivity timer (Ti) . . . . . . . . . . . . . . :  255 x 40 ms.
Number of queue elements. . . . . . . . . . . . . . . . . :  400
Number Global Descriptor
  Table selectors . . . . . . . . . . . . . . . . . . . . :  20





-------------------------------------------------------------------------------
Esc=Cancel  F1=Help  F7=Backward
```

Figure 168. Operating System/2 IEEE 802.2 Token-Ring Profile (2 of 2)



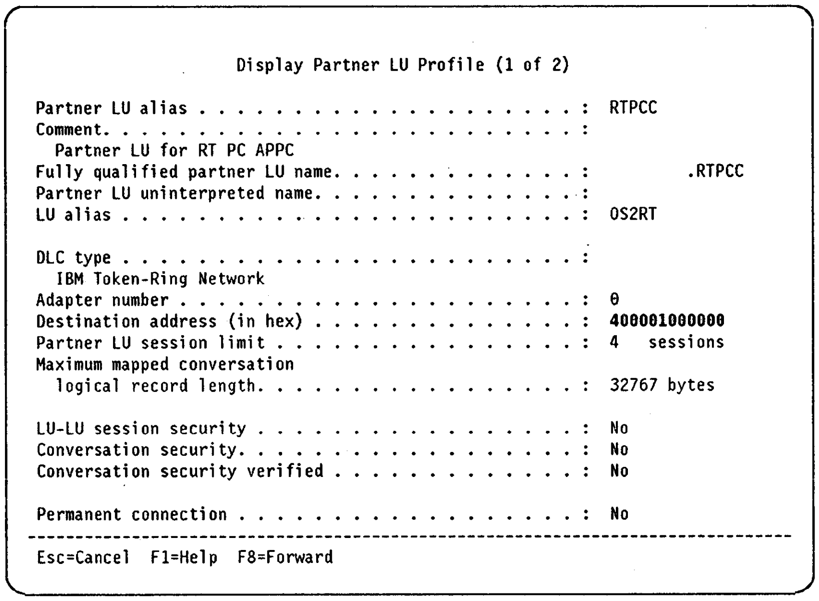
```
              Display IBM Token-Ring Network DLC Adapter Profile


Adapter number . . . . . . . . . . . . . . . . . . . . . :  0

Load DLC . . . . . . . . . . . . . . . . . . . . . . . . :  Yes

Maximum number of link stations. . . . . . . . . . . . . :  06
Percent of incoming calls. . . . . . . . . . . . . . . . :  050%

Free unused link . . . . . . . . . . . . . . . . . . . . :  Yes
Congestion tolerance . . . . . . . . . . . . . . . . . . :  080%

Maximum RU size. . . . . . . . . . . . . . . . . . . . . :  1024  bytes
Send window count. . . . . . . . . . . . . . . . . . . . :  2
Receive window count . . . . . . . . . . . . . . . . . . :  1

C&SM LAN ID. . . . . . . . . . . . . . . . . . . . . . . :  930
Send alert for beaconing . . . . . . . . . . . . . . . . :  No



-------------------------------------------------------------------------------
Esc=Cancel  F1=Help
```

Figure 169. Operating System/2 Token-Ring Network DLC Adapter Profile



```
                     Display Local APPC Logical Unit Profile


LU alias. . . . . . . . . . . . . . . . . . . . . . . . :  OS2RT

Comment . . . . . . . . . . . . . . . . . . . . . . . . :
  Token-Ring local LU for APPC

LU name . . . . . . . . . . . . . . . . . . . . . . . . :  OS2LU1

Default LU. . . . . . . . . . . . . . . . . . . . . . . :  No

LU local address (NAU address). . . . . . . . . . . . . :  00

LU session limit. . . . . . . . . . . . . . . . . . . . :  255

Maximum number of
  transaction programs. . . . . . . . . . . . . . . . . :  2



--------------------------------------------------------------------------------
Esc=Cancel  F1=Help
```

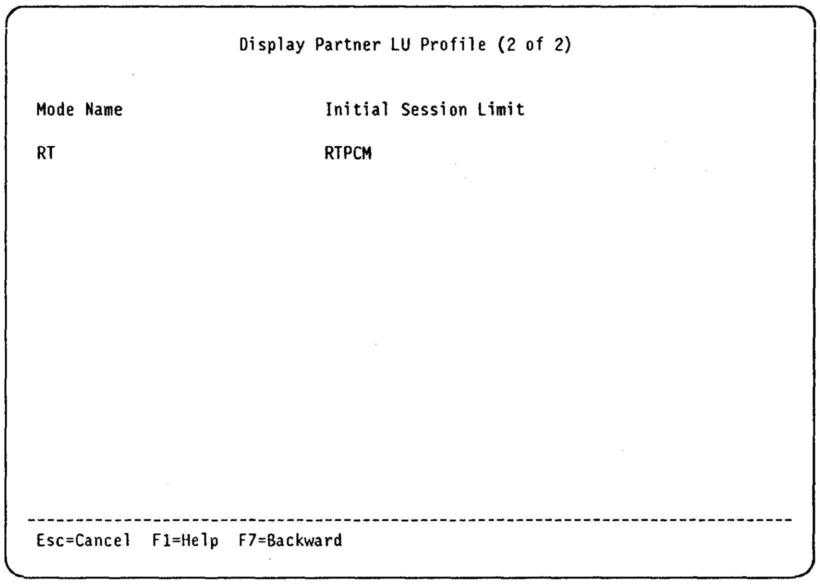
Figure 170. Operating System/2 Token-Ring Local APPC Logical Unit Profile



```
                       Display Partner LU Profile (1 of 2)

Partner LU alias . . . . . . . . . . . . . . . . . . . . :  RTPCC
Comment. . . . . . . . . . . . . . . . . . . . . . . . . :
  Partner LU for RT PC APPC
Fully qualified partner LU name. . . . . . . . . . . . . :          .RTPCC
Partner LU uninterpreted name. . . . . . . . . . . . . . :
LU alias . . . . . . . . . . . . . . . . . . . . . . . . :  OS2RT

DLC type . . . . . . . . . . . . . . . . . . . . . . . . :
  IBM Token-Ring Network
Adapter number . . . . . . . . . . . . . . . . . . . . . :  0
Destination address (in hex) . . . . . . . . . . . . . . :  400001000000
Partner LU session limit . . . . . . . . . . . . . . . . :  4   sessions
Maximum mapped conversation
  logical record length. . . . . . . . . . . . . . . . . :  32767 bytes

LU-LU session security . . . . . . . . . . . . . . . . . :  No
Conversation security. . . . . . . . . . . . . . . . . . :  No
Conversation security verified . . . . . . . . . . . . . :  No

Permanent connection . . . . . . . . . . . . . . . . . . :  No
--------------------------------------------------------------------------------
Esc=Cancel  F1=Help  F8=Forward
```

The destination address has been highlighted to emphasize that this value must correspond to the LAA or built-in adapter address of the Token-Ring adapter on the IBM RT.

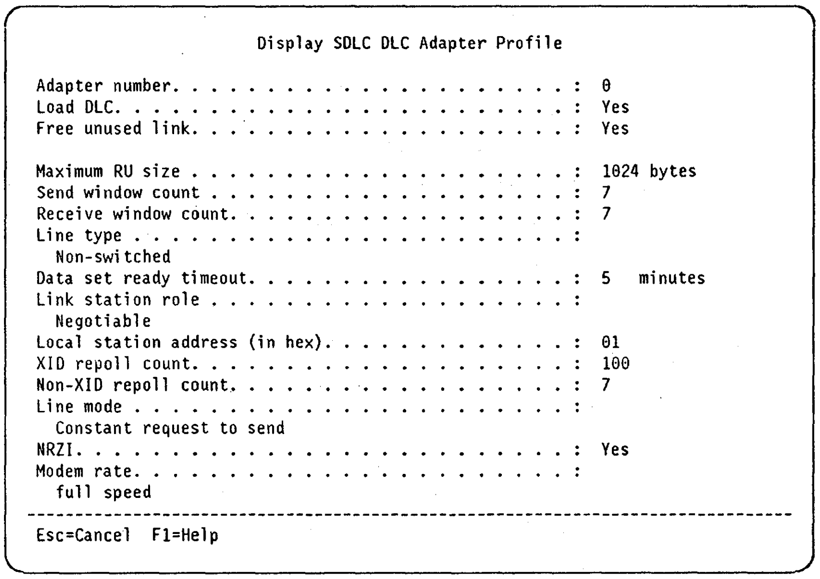
Figure 171. Operating System/2 Token-Ring Partner LU Profile (1 of 2)



```
                    Display Partner LU Profile (2 of 2)

Mode Name                       Initial Session Limit

RT                              RTPCM















--------------------------------------------------------------------------------
Esc=Cancel  F1=Help  F7=Backward
```

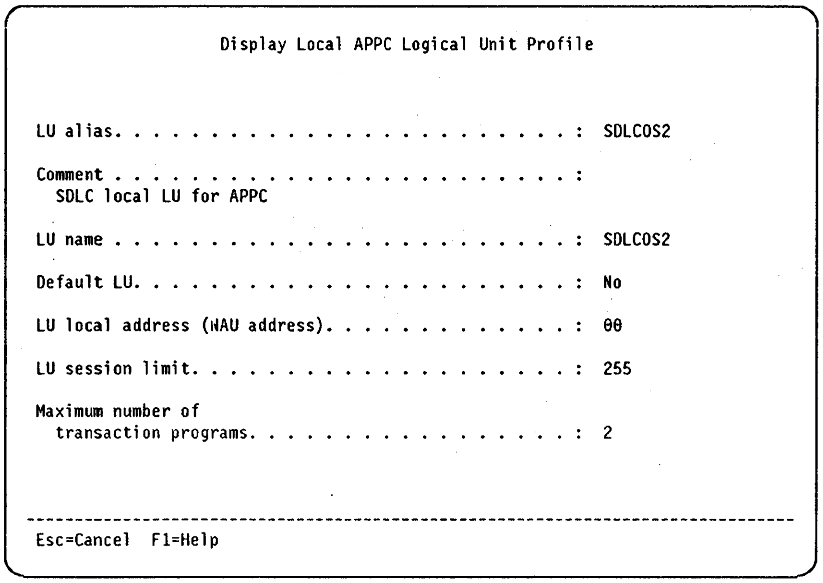
Figure 172. Operating System/2 Token-Ring Partner LU Profile (2 of 2)

Operating System/2 SDLC Profiles



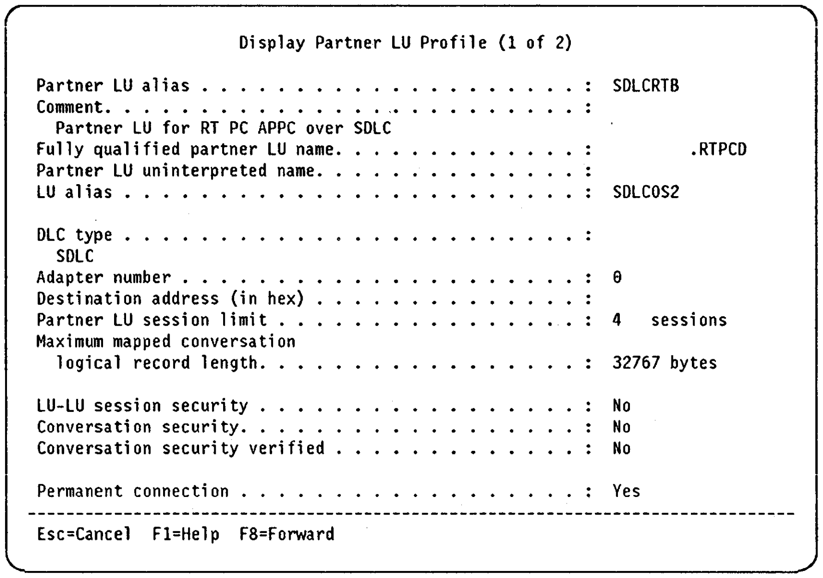
```
                      Display SDLC DLC Adapter Profile

Adapter number. . . . . . . . . . . . . . . . . . . . . . :  0
Load DLC. . . . . . . . . . . . . . . . . . . . . . . . . :  Yes
Free unused link. . . . . . . . . . . . . . . . . . . . . :  Yes

Maximum RU size . . . . . . . . . . . . . . . . . . . . . :  1024 bytes
Send window count . . . . . . . . . . . . . . . . . . . . :  7
Receive window count. . . . . . . . . . . . . . . . . . . :  7
Line type . . . . . . . . . . . . . . . . . . . . . . . . :
  Non-switched
Data set ready timeout. . . . . . . . . . . . . . . . . . :  5   minutes
Link station role . . . . . . . . . . . . . . . . . . . . :
  Negotiable
Local station address (in hex). . . . . . . . . . . . . . :  01
XID repoll count. . . . . . . . . . . . . . . . . . . . . :  100
Non-XID repoll count. . . . . . . . . . . . . . . . . . . :  7
Line mode . . . . . . . . . . . . . . . . . . . . . . . . :
  Constant request to send
NRZI. . . . . . . . . . . . . . . . . . . . . . . . . . . :  Yes
Modem rate. . . . . . . . . . . . . . . . . . . . . . . . :
  full speed
--------------------------------------------------------------------------------
Esc=Cancel  F1=Help
```

Figure 173. Operating System/2 SDLC DLC Adapter Profile



```
                    Display Local APPC Logical Unit Profile


LU alias. . . . . . . . . . . . . . . . . . . . . . . . :  SDLCOS2

Comment . . . . . . . . . . . . . . . . . . . . . . . . :
  SDLC local LU for APPC

LU name . . . . . . . . . . . . . . . . . . . . . . . . :  SDLCOS2

Default LU. . . . . . . . . . . . . . . . . . . . . . . :  No

LU local address (NAU address). . . . . . . . . . . . . :  00

LU session limit. . . . . . . . . . . . . . . . . . . . :  255

Maximum number of
  transaction programs. . . . . . . . . . . . . . . . . :  2



--------------------------------------------------------------------------------
Esc=Cancel  F1=Help
```

Figure 174. Operating System/2 SDLC Local APPC Logical Unit Profile



```
                      Display Partner LU Profile (1 of 2)

Partner LU alias . . . . . . . . . . . . . . . . . . . . :  SDLCRTB
Comment. . . . . . . . . . . . . . . . . . . . . . . . . :
  Partner LU for RT PC APPC over SDLC
Fully qualified partner LU name. . . . . . . . . . . . . :          .RTPCD
Partner LU uninterpreted name. . . . . . . . . . . . . . :
LU alias . . . . . . . . . . . . . . . . . . . . . . . . :  SDLCOS2

DLC type . . . . . . . . . . . . . . . . . . . . . . . . :
  SDLC
Adapter number . . . . . . . . . . . . . . . . . . . . . :  0
Destination address (in hex) . . . . . . . . . . . . . . :
Partner LU session limit . . . . . . . . . . . . . . . . :  4   sessions
Maximum mapped conversation
  logical record length. . . . . . . . . . . . . . . . . :  32767 bytes

LU-LU session security . . . . . . . . . . . . . . . . . :  No
Conversation security. . . . . . . . . . . . . . . . . . :  No
Conversation security verified . . . . . . . . . . . . . :  No

Permanent connection . . . . . . . . . . . . . . . . . . :  Yes
--------------------------------------------------------------------------------
Esc=Cancel  F1=Help  F8=Forward
```

Figure 175. Operating System/2 SDLC Partner LU Profile (page 2 not shown)

IBM RT Profiles for OS/2 Token-Ring

```
OS262_CONNECTION:
        type = CONNECTION
        profile_name = OS262
        attachment_name = RTOS2A
        local_lu_name = RTPC62
        network_name =
        remote_lu_name = OS2LU1
        stop_connection_on_inactivity = NO
        lu_type = LU6.2
        interface_type = EXTENDED
        remote_tpn_list_name = OS2REMOTELIST
        mode_list_name = RTPCM
        node_verification = NO
        inactivity_timeout_value = 0
        notify = NO
        cp_sessions = NO
        parallel_sessions = PARALLEL
        negotiate_session_limits = YES
```

Figure 176. SNA Connection Profile, Token-Ring LU 6.2 to OS/2

```
RTPC62_LOCAL LU:
        type = LOCALLU
        profile_name = RTPC62
        local_lu_name = RTPCC
        network_name =
        lu_type = LU6.2
        independent_lu = YES
        cp_sessions = NO
        tpn_list_name = RTLOCALLIST
        local_lu_address = 1
        sscp_id = 000000000000
        number_of_rows = 1
        number_of_columns = 1
        destination_address = 0
```

Figure 177. SNA Local LU Profile, Token-Ring LU 6.2 to OS/2

```
RTLOCALLIST_TPN LIST:
        type = TPNLIST
        Listname = RTLOCALLIST
        list_members = RTRT


RTRT_TPN:
        type = TPN
        profile_name = RTRT
        tpn_name = RTRT
        tpn_name_hex = D9E3D9E3
        conversation_type = MAPPED
        pip_data = NO
        sync_level = EITHER
        recovery_level = NO_RECONNECT
        path_to_server_program = /bin/RTRT
        program_to_execute = RTRT
        home_directory = /bin
        multiple_instances = NO
        user_id = 0
        group_id = 0
        umask = 000
        maximum_file_size = 9999
        server_synonym_name = svr
        subserver_types = NO
        unique_server_profile_name = svr
        external_notify = NO
        restart_action = ONCE
        communication_type = SIGNALS
        stdin = /dev/console
        stdout = /dev/console
        stderr = /dev/console
        subfields = 0
        notify_ipc_queue_key = 0
        communication_ipc_queue_key = 0
```

Figure 178. SNA Local TP Profiles, Token-Ring LU 6.2 to OS/2

```
RTOS2A_ATTACHMENT:
        type = ATTACHMENT
        profilename = RTOS2A
        control_point_profile_name = CDEFAULT
        logical_link_profile_name = TDEFAULT
        physical_link_profile_name = TDEFAULT
        logical_link_type = TOKEN_RING
        stop_attachment_on_inactivity = NO
        station_type = NEGOTIABLE
        physical_link_type = TOKEN_RING
        remote_secondary_station_address = 1
        smart_modem_command_sequence =
        length_of_command_sequence = 0
        call_type = CALL
        autolisten = NO
        timeout_value = 0
        remote_link_name_ethernet =
        remote_link_name_token_ring = C0B00C69
        remote_link_address_token_ring = 4000A0A0A0A0
        selection_sequence =
        length_of_selection_sequence = 0
        network_type = SWITCHED
        access_routing = LINK_ADDRESS
        remote_sap_address = 04
        remote_sap_address_range_lower = 04
        remote_sap_address_range_upper = EC
        virtual_circuit_type = PERMANENT
        remote_station_X.25_address =
        optional_X.25_facilities = NO
        logical_channel_number_of_PVC = 1
        reverse_charging = NO
        rpoa = NO
        default_packet_size = YES
        default_window_size = YES
        default_throughput_class = YES
        closed_user_group = NO
        data_network_identification_code =
        packet_size_for_received_data = 128
        packet_size_for_transmit_data = 128
        window_size_for_received_data = 2
        window_size_for_transmit_data = 2
        throughput_class_for_received_data = 1200
        throughput_class_for_transmit_data =
        index_to_selected_closed_user_group =
        lu_address_registration = NO
        local_lu_addresses = 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Figure 179. SNA Attachment Profile, Token-Ring LU 6.2 to OS/2. Notice that the profile uses the predefined profiles for Control Point, Logical- and Physical Link. Those are not listed here.

```
OS2REMOTELIST_REMOTE TPN LIST:
        type = REMOTETPNLIST
        Listname = OS2REMOTELIST
        list_members = OS2REMOTE

OS2REMOTE_REMOTE TPN:
        type = REMOTETPN
        profile_name = OS2REMOTE
        tpn_name = OS2RT
        tpn_name_hex = D6E2F2D9E3
        pip_data = NO
        conversation_type = MAPPED
        recovery_level = NO_RECONNECT
        sync_level = CONFIRM

RTPCM_MODE LIST:
        type = MODELIST
        Listname = RTPCM
        list_members = RTM
```

Figure 180. SNA Remote TP Profiles, Token-Ring LU 6.2 to OS/2

```
RTM_MODE:
        type = MODE
        profile_name = RTM
        mode_name = RT
        maximum_number_of_sessions = 5
        minimum_contention_winners = 0
        minimum_contention_losers = 0
        receive_pacing = 8
        send_pacing = 8
        maximum_ru_size = 2816
        recovery_level = NO_RECONNECT
```

Figure 181. SNA Mode Profiles, Token-Ring LU 6.2 to OS/2

IBM RT Profiles for OS/2 SDLC

```
SDLCOS2N_ATTACHMENT:
        type = ATTACHMENT
        profilename = SDLCOS2N
        control_point_profile_name = SDLCOS2C
        logical_link_profile_name = ZDEFAULT
        physical_link_profile_name = SDLCOS2P
        logical_link_type = SDLC
        stop_attachment_on_inactivity = NO
        station_type = NEGOTIABLE
        physical_link_type = RS232
        remote_secondary_station_address = 1
        smart_modem_command_sequence =
        length_of_command_sequence = 0
        call_type = LISTEN
        autolisten = YES
        timeout_value = 0
        remote_link_name_ethernet =
        remote_link_name_token_ring =
        remote_link_address_token_ring = 000000000000
        selection_sequence =
        length_of_selection_sequence = 0
        network_type = SWITCHED
        access_routing = LINK_NAME
        remote_sap_address = 04
        remote_sap_address_range_lower = 04
        remote_sap_address_range_upper = EC
        virtual_circuit_type = SWITCHED
        remote_station_X.25_address =
        optional_X.25_facilities = NO
        logical_channel_number_of_PVC = 1
        reverse_charging = NO
        rpoa = NO
        default_packet_size = YES
        default_window_size = YES
        default_throughput_class = YES
        closed_user_group = NO
        data_network_identification_code =
        packet_size_for_received_data = 128
        packet_size_for_transmit_data = 1024
        window_size_for_received_data = 2
        window_size_for_transmit_data = 2
        throughput_class_for_received_data = 9600
        throughput_class_for_transmit_data =
        index_to_selected_closed_user_group =
        lu_address_registration = NO
        local_lu_addresses = 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Figure 182. SNA Attachment Profile, SDLC LU 6.2 to OS/2. The profile uses the predefined profile for Logical Link (XDEFAULT). That profile is not listed here.

```
SDLCOS262_CONNECTION:
        type = CONNECTION
        profile_name = SDLCOS262
        attachment_name = SDLCOS2N
        local_lu_name = RTPC62
        network_name =
        remote_lu_name = SDLCOS2
        stop_connection_on_inactivity = NO
        lu_type = LU6.2
        interface_type = EXTENDED
        remote_tpn_list_name = OS2REMOTELIST
        mode_list_name = RTPCM
        node_verification = NO
        inactivity_timeout_value = 0
        notify = NO
        cp_sessions = NO
        parallel_sessions = PARALLEL
        negotiate_session_limits = YES
```

Figure 183. SNA Connection Profile, SDLC LU 6.2 to OS/2. Local LU, local/remote transaction program lists, local/remote transaction program and mode profiles are identical to those for Token-Ring. They are not repeated here.

```
SDLCOS2C_CONTROL POINT:
        type = CONTROLPOINT
        profile_name = SDLCOS2C
        xid_node_id = 05D00001
        network_name =
        cp_name =
        end_node_status = NO_BIND
        locate_gds_support = NO
        directory_services_support = NO
        resource_registration_support = NO
        registration_of_characteristics = NO
        topology_database_update_support = NO
        request_reply_cp_msus_support = NO
        unsolicited_CP_MSUs_supported = NO
        parallel_CP_CP_sessions_supported = NO
        flow_reduction_sequence_number = 0
```

Figure 184. SNA Control Point Profile, SDLC LU 6.2 to OS/2

```
SDLCOS2P_RS232:
        type = RS232
        profile_name = SDLCOS2P
        device_name = sdlcllc0
        request_to_send = CONTINUOUS
        bit_clock = EXTERNAL
        switched_network_backup = BACKUP_OFF
        network_type = NONSWITCHED
        transmission_rate = 1200
        data_rate_select = FULL
        call_type = LISTEN
        autocall_listen = NO
        call_override = NO
        answer_mode = MANUAL
        dtr_control = DTR
```

Figure 185. SNA Physical Link Profile, SDLC LU 6.2 to OS/2

Appendix D. Network PLUS Configurations

Notice

Not all the configurations described in this appendix have actually been tested at the ITSC. Those we did test are the connections to IBM AS/400 and the Token-Ring connection to IBM 3725.

SDLC Dial-up Line to SNA System/370 Host

You have been asked to configure an IBM RT for 3270 communications to an IBM SNA host. You have been asked to configure two 3278 model 2 terminals and one 3287 printer. A dial-up line supporting Hayes Smartmodems has been configured for you and its telephone number is 0256-56144. The network administrator has given you the VTAM listing for the line, which reads as in Figure 186.

```
LYNDA1    PU   ADDR= C1,                                   x
               DISCNT=NO,                                  x
               IDBLK=017,                                  x
               IDNUM=BD100,                                x
               MAXDATA=265,                                x
               MAXOUT=7,                                   x
               PACING=0,                                   x
               PUTYPE=2,                                   x
               VPACING=0,                                  x
               LOGAPPL=UIWZPSVM,                           x
               USSTAB=UIWUSTAB
*
LYNSA100 LU    LOCADDR=2,                                  x
               MODETAB=S3278M2
*
LYNSA101 LU    LOCADDR=3,                                  x
               MODETAB=S3278M2
*
LYNSA102 LU    LOCADDR=4,                                  x
               MODETAB=SPRNTM
```

Figure 186. Network 3270-PLUS: VTAM Listing for SDLC Dial-up Line

The network Administrator tells you that if the ***MODETAB*** parameter is set to ***S3278M2***, then the LU is defined as a 3278 model 2 terminal; if set to ***SPRNTM***, the LU is defined as a 3287 model 2 printer.

You also find out from the responsible system programmer that the ***SSCP ID*** of the machine you are connecting to is decimal 050000000101. The hexadecimal equivalent that you must use in the Local LU Profiles is X'050000000065'.

SNA Services Profiles

Valid SNA Services profiles to connect to this host are shown in Table 11 through Table 16.

Table 11. Network 3270-PLUS: Connection Profiles for SDLC Dial-up Line

Connection Profile Name	LU2	LU3	LU4
Attachment Profile Name	SDLC370A	SDLC370A	SDLC370A
Local LU Profile Name	LLU2	LLU3	LLU4
Network Name			
Remote LU Name			
Stop Connection on Inactivity	NO	NO	NO
LU Type	LU2	LU2	LU3
Notify	YES	YES	YES

Table 12. Network 3270-PLUS: Local LU Profiles for SDLC Dial-up Line

Local LU Profile Name	LLU2	LLU3	LLU4
LU Type	LU2	LU2	LU3
Network Name			
Local LU Name	LYNSA100	LYNSA101	LYNSA102
Number of Rows	24	24	24
Number of Columns	80	80	80
Local LU Address	2	3	4
SSCP ID	050000000065	050000000065	050000000065

Table 13. Network 3270-PLUS: Attachment Profile for SDLC Dial-up Line

Attachment Profile Name	SDLC370A
Control Point Profile Name	SDLC370C
Logical Link Profile Name	SDLC370L
Physical Link Profile Name	SDLC370P
Logical Link Type	SDLC
Station Type	SECONDARY
Physical Link Type	Smartmodem
Modem Command Sequence	ATDP025656144
Stop Attachment on Inactivity	NO

Table 14. Network 3270-PLUS: Control Point Profile for SDLC Dial-up Line

Control Point Profile Name	SDLC370C
XID Node Id	017BD100

Table 15. Network 3270-PLUS: Logical Link Profile for SDLC Dial-up Line

Logical Link Profile Name	SDLC370L
Station Type	SECONDARY
Local Secondary Station Address	193
Serial Encoding	NRZI
Transmit Window Count	7
Retransmit Count	10
Retransmit Threshold	10
Drop Link on Inactivity	NO
Force Disconnect Timeout	3
Maximum I-Field Size	SYSTEM_DEFIND
Link Trace	TRACE_OFF

Table 16. Network 3270-PLUS: SmartModem Physical Link Profile

Physical Link Profile Name	**SDLC370L**
Data Link Device Name	sdlcllc0
Request To Send	CONTINUOUS
Bit Clocking	EXTERNAL
Data Rate Select	FULL
Switched Network Backup	BACKUP_OFF
Network Type	SWITCHED
Call Type	CALL
Autocall	YES
DTR Control	CDSTLM

Note:

1. LLU4 must be set up as an LU type 3 as it is a printer.
2. LU addresses start at number 2 as that is the first address that can be used for 3270 devices.
3. The XID Node ID parameter in the Control Point Profile (`XID Node Id=017BD100`) is a concatenation of the IDBLK (`IDBLK=017`) and IDNUM (`IDNUM=BD100`) parameters in the VTAM listing.
4. The ADDR value in the VTAM listing (`ADDR=C1`) matches the Local Secondary Station Address value (`Local Secondary Station Address=193`) set in the Logical Link Profile. The ADDR value in the VTAM listing is ***hexadecimal***, whilst the Local Secondary Station Address in SNA Services is specified as a ***decimal*** value. 193 is the decimal equivalent of X'C1'.
5. The MAXOUT value in the VTAM listing (`MAXOUT=7`) matches the value set in the Transmit Window Count parameter (`Transmit Window Count=7`) in the Logical Link Profile.
6. The Modem Command Sequence parameter in the Attachment Profile (`Modem Command Sequence=ATDP025656144`) includes the telephone number of the host (0256-56144).
7. The SSCP ID value in the VTAM listing (`SSCPID=101`) is coded ***decimal***; the SSCP ID in SNA Services (`SSCP ID=050000000065`) is ***hexadecimal***.
8. For host file transfer the *IND$FILE* PC/Host File Transfer and Emulator Program (program number 5665-311 for MVS/TSO, 5664-281 for VM/SP, 5798-DQH for CICS and 5666-316 for VSE/SP 2.1) must be installed at the host.

Network 3270-PLUS Profile

There is only one profile that needs to be changed in this case; the main configuration profile: /usr/lpp/r/comm/sna/3270/profiles/sc3270.prof. This profile is shown in Figure 187.

```
#
# @(#)sc3270.prof   1.3
#

       System Configuration Profile for Rabbit SNA 3270-Plus

:Protocol(SNA)
:ControllerType(3274)

:PH Parms("/usr/lpp/r/comm/sna/profiles/SNAParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = OFF, PH Data = OFF, DH Errors = OFF,
         PSErrors = OFF, PSData = OFF, PSTiming = OFF,
         PEErrors = OFF, PEData = OFF, PETiming = OFF )

     T E R M I N A L     D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
          "/usr/lpp/r/comm/sna/3270/profiles/TermOpts.prof",
          "/usr/lpp/r/comm/sna/3270/profiles/KbdMap"",
          Validation Mode = 1)

:Terminal(2, On, 3278)

:Terminal(3, On, 3278)

     P R I N T E R     D E F I N I T I O N S

:PrinterParms(TIMEOUT = 15, SPOOLSIZE = 3)

:Printer(4, ON, 3287, 2, 1920,
         "/usr/lpp/r/comm/sna/3270/profiles/PrtOpts.prof")

:End;
```

Figure 187. Network 3270-PLUS: Configuration Profile for two 3278 Terminals and one 3287 Printer

The terminals are set up as IBM 3278 terminals and the printer is set to IBM 3287 to match the VTAM definitions in Figure 186 on page 415.

SDLC leased Line to SNA System/370 Host

You have been asked to configure an IBM RT for 3270 and RJE communications to an SNA host. You have been asked to configure one RJE workstation and two 3278 terminals. A leased line supporting IBM 3265 modems has been configured for you. The VTAM listing for this line is as shown in Figure 188.

```
**************************************************************************
***********    LINE 11 --- SDLC 9600 BPS HALF DUPLEX LINE     **********
**************************************************************************
*
LINE11    LINE  ADDRESS=(011,HALF),                                     X
                CLOCKNG=EXT,                                            X
                MAXDATA=265,                                            X
                MAXLU=200,                                              X
                MAXPU=6,                                                X
                NEWSYNC=(NO),                                           X
                RETRIES=(7,4,5),                                        X
                ISTATUS=ACTIVE,                                         X
                VPACING=(1)
*               STATOPT=('LINE 11')
        SERVICE ORDER=(RTPU1),
                MAXLIST=6
*
**************    IBM RT LINE DEFINITION - 3270 and RJE    ****************
*
RTPU1     PU    ADDR=C1,                                                X
                PASSLIM=7,                                              X
                PUTYPE=2,                                               X
                DISCNT=NO,                                              X
                IRETRY=NO,                                              X
                ANS=CONT,                                               X
                MAXDATA=265,                                            X
                MAXOUT=7
RTLU1     LU    LOCADDR=1,LUDR=YES
RTLU2     LU    LOCADDR=2,LUDR=YES
RTLU3     LU    LOCADDR=3,LUDR=YES
```

Figure 188. Network PLUS: VTAM Listing for SDLC Leased Line

Notice that, in this case, there is no MODETAB entry for each terminal. This is because the LU's are dynamic (LUDR=YES). This means that each LU can be defined by the application (SNA Services) to whichever type is required. The first address that you have (LOCADDR=1) cannot be used as a terminal session (limitation of SNA). It must be configured as an RJE (3770) device. The other two sessions, (LOCADDR=2) and (LOCADDR=3) should be configured as terminals. If you want to use more than one RJE workstation, you can also use the addresses in the range from 2 to 129 for defining RJE workstations. For the RJE Workstation, you have a logon defined in JES2 as shown in Figure 189.

```
LOGON1   APPLID=JES2
LINE2    UNIT=SNA
RMT2     LUTYPE1,BUFSIZE=256,COMP,NOCMPCT,NUMPR=1,NUMRD=1,SETUPMSG,
         NUMPU=2
R2.PR1   PRWIDTH=132,FCBLOAD
R2.PU1   CLASS=B,SELECT=PUNCH1,NOSEP
R2.PU2   CLASS=B,SELECT=EXCH2,NOSEP
R2.RD1   PUDEST=1,PULCL
```

Figure 189. Network RJE-PLUS: JES2 Definitions

This means that your logon ID is RMT2 and you have 1 printer, 1 card punch, 1 exchange, and 1 reader defined. JES3 uses other definitions for RJE workstations. An example is shown in Figure 190.

```
RJPWS,N=RTEST,RD=1,PR=1,PU=1,C=R
CONSOLE,JNAME=RTEST,TYPE=RJP,LEVEL=10,DEST=NONE
DEVICE,DTYPE=RMTPRINT,JNAME=RTESTPR1,CHNSIZE=(4,0),
       FORMS=(YES,STANDARD),XLATE=NO,HEADER=NO,BURST=NO
DEVICE,DTYPE=RMTPUNCH,JNAME=RTESTPU1,CHNSIZE=(4,0),
       FORMS=(YES,STANDARD),XLATE=NO,HEADER=NO,BURST=NO
```

Figure 190. Network RJE-PLUS: JES3 Definitions for an SNA RJE Workstation

This means that your logon ID is RJPWS and you have 1 console, 1 reader, 1 printer, and 1 card punch defined.

The JES3 parameters have the following meaning:

RJPWS	SNA workstation description
N	5-character name of workstation
RD	Maximum number of reader units (0-F)
PR	Maximum number of printer units (0-F)
PU	Maximum number of punch units (0-F)
C	Console support, R=console and printer are separate devices
CONSOLE	Console definition for RJP workstation
JNAME	Must match the name on RJPWS statement
TYPE	Console is RJP console
LEVEL	Console authority (JES3 commands allowed? Installation dependent!)
DEST	Messages sent from host to remote console (messages classes)
DEVICE	JES3 device
DTYPE	Device type (RMTPRINT=SNA-RJP printer)
JNAME	Name of JES3 device (for RJP must include the workstation name followed by PRn or PUn or RDn).
CHNSIZE	Size of RU chain transmitted to SNA workstation
DTYPE	RMTPUNCH=SNA-RJP puncher

For more details see *JES3 Initialization and Tuning Guide.*

SNA Services Profiles

Valid SNA Services profiles to connect to this host are shown in Table 17 through Table 22.

Table 17. Network PLUS: Connection Profiles for SDLC Leased Line

Connection Profile Name	LU1	LU2	LU3
Attachment Profile Name	SDLC370A	SDLC370A	SDLC370A
Local LU Profile Name	LLU1	LLU2	LLU3
Network Name			
Remote LU Name			
Stop Connection on Inactivity	NO	NO	NO
LU Type	LU1	LU2	LU2
Notify	NO	YES	YES

Table 18. Network PLUS: Local LU Profiles for SDLC Leased Line

Local LU Profile Name	LLU2	LLU3	LLU4
LU Type	LU1	LU2	LU2
Network Name			
Local LU Name	RTLU1	RTLU2	RTLU3
Number of Rows	24	24	24
Number of Columns	80	80	80
Local LU Address	1	2	3
SSCP ID	05000000000A	05000000000A	05000000000A

Table 19. Network PLUS: Attachment Profile for SDLC Leased Line

Attachment Profile Name	SDLC370A
Control Point Profile Name	SDLC370C
Logical Link Profile Name	SDLC370L
Physical Link Profile Name	SDLC370P
Logical Link Type	SDLC
Station Type	SECONDARY
Physical Link Type	RS232
Stop Attachment on Inactivity	NO

Table 20. Network PLUS: Control Point Profile for SDLC Leased Line

Control Point Profile Name	SDLC370C
XID Node Id	00000000

Table 21. Network PLUS: Logical Link Profile for SDLC Leased Line

Logical Link Profile Name	**SDLC370L**
Station Type	SECONDARY
Local Secondary Station Address	193
Serial Encoding	NRZI
Transmit Window Count	7
Retransmit Count	10
Retransmit Threshold	10
Drop Link on Inactivity	NO
Force Disconnect Timeout	3
Maximum I-Field Size	SYSTEM_DEFIND
Link Trace	TRACE_OFF

Table 22. Network PLUS: RS232C Physical Link Profile for SDLC Leased Line

Physical Link Profile Name	**SDLC370P**
Data Link Device Name	sdlcllc0
Request To Send	CONTINUOUS
Bit Clocking	EXTERNAL
Data Rate Select	FULL
Switched Network Backup	BACKUP_OFF
Network Type	NONSWITCHED
DTR Control	DTR

Note:

1. Address number one (RTLU1) must be configured as an LU type 1 (RJE workstation) as it cannot be used as an LU type 2 (3270 terminal).
2. Notice that LLU1 is set up as an LU type 1 as it will be used as an RJE session, while the two other LUs are defined as LU type 2 for 3270 sessions.
3. In this case, the XID can be left blank as we are using a leased line.
4. The ADDR value in the VTAM listing (`ADDR=C1`) matches the Local Secondary Station Address value (`Local Secondary Station Address=193`) set in the Logical Link Profile. The ***hexadecimal*** ADDR value (`ADDR=C1`) in the VTAM listing matches the ***decimal*** value 193 in the SNA Services profile.
5. The MAXOUT value in the VTAM listing (`MAXOUT=7`) matches the value set in the Transmit Window Count parameter (`Transmit Window Count=7`) in the Logical Link Profile.
6. The ***decimal*** SSCP ID value in the VTAM listing (`SSCPID=10`) matches the ***hexadecimal*** value for SSCP ID in SNA Services: (`SSCP ID=05000000000A`).
7. For host file transfer, the *IND$FILE* PC/Host File Transfer and Emulator Program (program number 5665-311 for MVS/TSO, 5664-281 for VM/SP, 5798-DQH for CICS and 5666-316 for VSE/SP 2.1) must be installed at the host.

Network RJE-PLUS Profile

There is only one Network RJE-PLUS profile that needs to be changed; the configuration profile: /usr/lpp/r/comm/sna/rje/profiles/scRJE.prof. Only address number 1 (LOCADDR=1) needs to be configured as that is the RJE session. The configuration profile in Figure 192 provides automatic logon to JES2. If you want to logon to JES3, you must change the :LU Spec and the :Logon statement as shown in Figure 191.

```
:LU SPEC (Address=1, AutoLogon=On, LogonID=jes3)

:Logon(LogonID=jes3), LogonString="LOGON APPLID=JES3) DATA(RTEST)")
```

Figure 191. Network RJE-PLUS: Logon Statements for JES3

```
@(#)scRJE.prof 1.1

                       SNA RJE-PLUS Profile

:Input Parms ( Compression = On,
              TabChar = 9,
              TabLength = 8,
              EndOfRecord = 10,
              Max Input Sessions = 1,
              Max Elements Per Chain = 5 )

:Console ( FileSize = 10240 )

:Log Parms (Error Log = Off, Main Log = Off, SNAMgr Log = Off,
           INMgr Log  = Off, OUTMgr Log = Off, CONMgr Log = Off,
           JOBMgr Log = Off, LUMPH0 Log = Off, LUMPH1 Log = Off,
           PHLUM0 Log = Off, PHLUM1 Log = Off)

:LU Spec (Address=1, AutoLogon=On, LogonID=jes2)

:Logon (LogonID=jes2, LogonString="LOGON APPLID(JES2) DATA(RMT2)")

:Output Device (Medium = CARD,
               Device Number = 1,
               Destination ID = PUNCH)

:Destination Control (Destination ID = PUNCH,
                     FCB Name = 8S,
                     Output Routing = FILE,
                     Output Path = "/u/bern/rje/")

:Output Device (Medium = EXCHANGE,
               Device Number = 2,
               Destination ID = PUNCH)

:Output Device (Medium = PRINTER,
               Device Number = 1,
               Destination ID = PRTR)

:Destination Control (Destination ID = PRTR,
                     FCB Name = 8S,
                     Output Routing = Spooler,
                     Output Path = "/u/bern/rje/",
                     Queue Name = lpq)

:Forms Control Block  (FCBName = 8S,
                      PageLength = 66,
                      LineLength = 132,
                      topMargin = 1,
                      bottomMargin = 66,
                      leftMargin = 1,
                      rightMargin = 132,
                      vertTabs = "5,10,15,20",
                      horizTabs = "5,10,15,20" )
:end;
```

Figure 192. Network RJE-PLUS: Configuration Profile for one RJE Station

Network 3270-PLUS Profile

Only the main configuration profile, /usr/lpp/r/comm/sna/3270/profiles/sc3270.prof. need to be edited. In this, addresses 2 and 3 (LOCADDR=2 and LOCADDR=3) need to be configured as 3270 sessions. For the complete configuration profile see Figure 193.

```
#
# @(#)sc3270.prof   1.3
#

      System Configuration Profile for Rabbit SNA 3270-Plus

:Protocol(SNA)
:ControllerType(3274)

:PH Parms("/usr/lpp/r/comm/sna/profiles/SNAParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = OFF, PH Data = OFF, DH Errors = OFF,
        PSErrors = OFF, PSData = OFF, PSTiming = OFF,
        PEErrors = OFF, PEData = OFF, PETiming = OFF )


     T E R M I N A L     D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
        "/usr/lpp/r/comm/sna/3270/profiles/TermOpts.prof",
        "/usr/lpp/r/comm/sna/3270/profiles/KbdMap",
        Validation Mode = 1)

:Terminal(2, On, 3278)

:Terminal(3, On, 3278)

:End;
```

Figure 193. Network 3270-PLUS: Configuration Profile for two 3278 Terminals

This example shows how you can use Network 3270-PLUS and Network RJE-PLUS simultaneously on one host connection. The RJE workstation is configured as LOCADDR=1 through the :LU Spec(Address=1 ...) keyword and the terminals are configured as LOCADDR=2 and LOCADDR=3 through the :Terminal(2, On, 3278) and the :Terminal(3, On, 3278) statements to match up with the VTAM definitions in Figure 188 on page 419.

DFT Connection to System/370 Host using BSC configured 3274

You are asked to configure an IBM RT for a five session DFT connection to an IBM 3274 which is connected to the host over a ***BSC*** line. This would not normally require you to get a VTAM listing from a network administrator. All you need to know is the number of sessions defined on the IBM 3274 and the type of terminals defined. In this case, there are five sessions and they are defined as 3278 model 2 terminals.

Network 3270-PLUS Profiles

In this case, only the Network 3270-PLUS profiles need to be changed, as we are ***not*** in an SNA environment. First set the BSC Parameters profile: `/usr/lpp/r/comm/bsc/profiles/BSCParms.prof` as shown in Figure 194.

```
BSCParms.prof
 @(#)BSCParms.pr    1.5

:Line Parms(Connection = POINT-TO-POINT,
            Line = LEASED,
            RTS = SEND,
            Configuration = Secondary)

:Control Unit(Mode = 3274,
              Poll Address = 64,
              MinorDevice = dft0)



:Logging(BufferErrors = OFF,Transmit Messages = OFF,Control = OFF,
         Prog Errors = OFF,Data Link Errors = OFF,FromDH = OFF,
         ToDH = OFF,Data Link States = ON)

:End;
```

Figure 194. Network 3270-PLUS: BSC Parameter Configuration Profile for DFT Connection

Note: The parameter `MinorDevice = dft0` in the `:Control Unit` stanza must correspond to the name of the device that you added with the *devices* command. The other profile that needs to be changed is the main configuration profile: `/usr/lpp/r/comm/bsc/3270/profiles/sc3270.prof`. It should be set as shown in Figure 195.

```
#
# @(#)sc3270.prof   1.3
#

        System Configuration Profile for Rabbit BSC 3270-Plus

:Protocol(BSC)
:ControllerType(3274)

:PH Parms("/etc/ddi/BSCParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = OFF, PH Data = OFF, DH Errors = OFF,
         DH Timing = OFF, PSErrors = OFF, PSData = OFF, PSTiming = OFF,
         PEErrors = OFF, PEData = OFF, PETiming = OFF )


      T E R M I N A L     D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
          "/usr/lpp/r/comm/bsc/3270/profiles/TermOpts.prof",
          "/usr/lpp/r/comm/bsc/3270/profiles/KbdMap",
          Validation Mode = 1)


:Terminal(0, On, 3278)
:Terminal(1, On, 3278)
:Terminal(2, On, 3278)
:Terminal(3, On, 3278)
:Terminal(4, On, 3278)

:End;
```

Figure 195. Network 3270-PLUS: BSC System Configuration Profile for DFT Connection

Note:

1. For a DFT configuration, the first address should always be defined as zero: `:Terminal(0, On, 3278)`.
2. When using DFT, host to printer operations are not supported.

These profile definitions should work for most DFT connections.

BSC leased Line to non-SNA 370 Host

You have been asked to configure an IBM RT to connect remotely to an IBM host over a leased BSC line. You have been asked to configure three 3278 model 2 terminals and one 3278 model 2 printer. The host is running Binary Telecommunication Access Method (BTAM) and the communication controller is running Emulation Program (EP) as the network control program. The EP listing for the line is as shown in Figure 196.

```
*****************   R T P C   ********************
D032       CLUSTER CUTYPE=3274,GPOLL=407F,LINE=061
           TERMINAL TERM=3278,MODEL=2,SELECT=6040
           TERMINAL TERM=3278,MODEL=2,SELECT=60C1
           TERMINAL TERM=3278,MODEL=2,SELECT=60C2
           TERMINAL TERM=3287,SELECT=60C3
           SPACE
```

Figure 196. Network 3270-PLUS: EP Listing for Leased Line BSC Connection

The EP listing is slightly different from a VTAM listing but should be fairly easy to understand. The figure shows that the line is configured for an IBM 3274 controller with the first three devices being 3278 terminals, and the last a 3287 printer.

Network 3270-PLUS Profiles

Again, as with example 3, there are only two profiles to change. First set the BSC Parameters profile as shown in Figure 197.

```
 @(#)BSCParms.prof  1.5


:Line Parms(Connection = MULTI-POINT,
            Line = LEASED,
            AutoDisconnect = OFF,
            Timeout = 22,
            TerminalID = "",
            RTS = PERMANENT,
            Inquiry = Off,
            Configuration = SECONDARY,
            Extended Enquiry = 22,
            Clocking = EXTERNAL,
            Autocall = NONE,
            SelectStandby = OFF,
            AlterBaudRate = OFF,
            RaiseDTR = IMMEDIATE)

:Control Unit(Mode = 3274,
              PollAddress = 64,
              MinorDevice = bsc0)


:Logging(BufferErrors = OFF, Transmit Messages = OFF, Control = ON,
         Prog Errors = ON, Data Link Errors = ON, FromDH = ON,
         ToDH = ON, Data Link States = ON)

:end;
```

Figure 197. Network 3270-PLUS: BSC Parameter Profile for a Leased Line

Note:

1. The Poll Address can be deduced from the EP listing as it is defined as the first two digits of the GPOLL parameter (`GPOLL=407F`). This gives a poll address of X'40'. Hence, `PollAddress=64` is the decimal equivalent of X'40'.
2. In this case, the MinorDevice parameter is set to `bsc0` which would have been configured into the system when the Network 3270-PLUS (BSC) software was installed (`MinorDevice=bsc0`).

The other profile that needs to be changed is the main configuration profile. In this case, it should be set as shown in Figure 198.

```
#
# @(#)sc3270.prof  1.3
#

     System Configuration Profile for Rabbit BSC 3270-Plus

:Protocol(BSC)
:ControllerType(3274)

:PH Parms("/etc/ddi/BSCParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = ON, PH Data = ON, DH Errors = ON, DH Timing = ON,
         PSErrors = ON, PSData = ON, PSTiming = ON,
         PEErrors = ON, PEData = ON, PETiming = ON )


     T E R M I N A L     D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
      "/usr/lpp/r/comm/bsc/3270/profiles/TermOpts.prof",
      "/usr/lpp/r/comm/bsc/3270/profiles/KbdMap",
      Validation Mode = 1)

:Terminal(0, On)

:Terminal(1, On)

:Terminal(2, On)

     P R I N T E R     D E F I N I T I O N S

:PrinterParms(TIMEOUT = 15, SPOOLSIZE = 3)

:Printer(3, ON, 3287, 2, 1920,
         "/usr/lpp/r/comm/sna/3270/profiles/PrtOpts.prof")

:End;
```

Figure 198. Network 3270-PLUS: BSC Configuration Profile for a Leased Line

X.25 Connection to SNA System/370 Host using 3725

You have been asked to configure an IBM RT to connect to an IBM SNA host via X.25 for 3270 emulation. You have been asked to configure three 3279 model 2 terminals. A line has been configured for you at the host and the VTAM listing for it is as shown in Figure 199.

```
RTX25PU  PU    ADDR=C1,                    ** STATION ADDRESS       **
               IDBLK=017,                  ** ID FOR IBM RT          **
               IDNUM=E6150,
               DISCNT=YES,                 ** HANG-UP ON LU LOGOFF  **
               MAXDATA=265,
               MAXOUT=7,
               MAXPATH=1,                  ** NO OF DIAL-OUT PATHS  **
               PUTYPE=2,ISTATUS=ACTIVE
*
*
RTX25LU2 LU    LOCADDR=2,ISTATUS=ACTIVE,
               MODETAB=C32792,             ** MODE TABLE FOR 3279M2 **
               USSTAB=UT2X25,
               SSCPFM=USSSCS
*
*
RTX25LU3 LU    LOCADDR=3,ISTATUS=ACTIVE,
               MODETAB=C32792,             ** MODE TABLE FOR 3279M2 **
               USSTAB=UT2X25,
               SSCPFM=USSSCS
*
*
RTX25LU4 LU    LOCADDR=4,ISTATUS=ACTIVE,
               MODETAB=C32792,             ** MODE TABLE FOR 3279M2 **
               USSTAB=UT2X25,
               SSCPFM=USSSCS
*
```

Figure 199. Network 3270-PLUS: VTAM Listing for X.25 Connection

The network administrator tells you that the X.25 address (NUA) for the host is 234-24490023413 and the NUA for the IBM RT is 234-24490023214. He also tells you that the SSCP ID of the host is decimal 128. The hexadecimal equivalent that you must use in the Local LU Profiles is X'050000000080'.

SNA Services Profiles

Valid profiles to connect to this host are shown in Table 23 through Table 28.

Table 23. Network 3270-PLUS: Connection Profiles for X.25 Connection

Connection Profile Name	**LU2**	**LU3**	**LU4**
Attachment Profile Name	X25370A	X25370A	X25370A
Local LU Profile Name	LLU2	LLU3	LLU4
Network Name			
Remote LU Name			
Stop Connection on Inactivity	NO	NO	NO
LU Type	LU2	LU2	LU2
Notify	YES	YES	YES

Table 24. Network 3270-PLUS: Local LU Profiles for X.25 Connection

Local LU Profile Name	**LLU2**	**LLU3**	**LLU4**
LU Type	LU2	LU2	LU2
Network Name			
Local LU Name	RTX25LU2	RTX25LU3	RTX25LU4
Number of Rows	24	24	24
Number of Columns	80	80	80
Local LU Address	2	3	4
SSCP ID	050000000080	050000000080	050000000080

Table 25. Network 3270-PLUS: Attachment Profile for X.25 Connection

Attachment Profile Name	**X25370A**
Control Point Profile Name	X25370C
Logical Link Profile Name	X25370L
Physical Link Profile Name	X25370P
Logical Link Type	QLLC
Call Type	CALL
Virtual Circuit Type	SWITCHED
Remote Station X.25 Address	23424490023413
Optional X.25 Facilities	NO
Stop Attachment on Inactivity	NO

Table 26. Network 3270-PLUS: Control Point Profile for X.25 Connection

Control Point Profile Name	**X25370C**
XID Node Id	017E6150

Table 27. Network 3270-PLUS: QLLC Logical Link Profile for X.25 Connection

Logical Link Profile Name	**X25370L**
Station Type	SECONDARY
Secondary Inactivity Timeout	120
Maximum I-Field Size	SYSTEM_DEFINED
Drop Link on Inactivity	NO
Link Trace	TRACE_OFF
Force Disconnect Timeout	3

Table 28. Network 3270-PLUS: Physical Link Profile for X.25 Connection

Physical Link Profile Name	**X25370P**
Data Link Device Name	qllc0
Local X.25 Network Address	23424490023214
Maximum Link Stations	20

Note:

1. LU addresses start at number 2 which is the first address that can be used for 3270 devices.
2. The XID Node ID parameter in the Control Point Profile (`XID Node Id=017E6150`) is a concatenation of the IDBLK (`IDBLK=017`) and IDNUM (`IDNUM=E6150`) parameters in the VTAM listing.
3. The ADDR value in the VTAM listing (`ADDR=C1`) must be defined, but is not used.
4. The SSCP ID value in the VTAM listing (`SSCPID=128`) is coded ***decimal***, whilst the SSCP ID in SNA Services (`SSCP ID=050000000080`) is coded ***hexadecimal***.
5. For host file transfer the *IND$FILE* PC/Host File Transfer and Emulator Program (program number 5665-311 for MVS/TSO, 5664-281 for VM/SP, 5798-DQH for CICS and 5666-316 for VSE/SP 2.1) must be installed at the host.

Network 3270-PLUS Profile

The main configuration profile, /usr/lpp/r/comm/sna/3270/profiles/sc3270.prof need to be changed, only. In this case, addresses 2, 3 and 4 (LOCADDR=2, LOCADDR=3) and (LOCADDR=4) need to be configured as shown in Figure 200.

```
#
# @(#)sc3270.prof   1.3
#

       System Configuration Profile for Rabbit SNA 3270-Plus

:Protocol(SNA)
:ControllerType(3274)

:PH Parms("/usr/lpp/r/comm/sna/profiles/SNAParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = OFF, PH Data = OFF, DH Errors = OFF,
         PSErrors = OFF, PSData = OFF, PSTiming = OFF,
         PEErrors = OFF, PEData = OFF, PETiming = OFF )


     T E R M I N A L    D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
          "/usr/lpp/r/comm/sna/3270/profiles/TermOpts.prof",
          "/usr/lpp/r/comm/sna/3270/profiles/KbdMap",
          Validation Mode = 1)

:Terminal(2, On, 3279)

:Terminal(3, On, 3279)

:Terminal(4, On, 3279)

:End;
```

Figure 200. Network 3270-PLUS: Configuration Profile for three 3279 Terminals

All terminals are defined as 3279 terminals to match the VTAM definitions in Figure 199 on page 431.

Token-Ring Connection to SNA System/370 Host using 3725

You have been asked to configure an IBM RT for connection to an IBM 3725 host network controller via Token-Ring. You must configure three 3279 model 2 terminals. The Token-Ring address of the adapter on the IBM 3725 is 400000100101, and the network administrator has created entries as shown in the VTAM listing in Figure 201.

```
T05P004  PU     ADDR=C1,                   ** CONTROLLER ADDRESS          **
                IDBLK=020,                 ** FIRST PART OF XID           **
                IDNUM=E6150,               ** SECOND PART OF XID          **
                PUTYPE=2,
                DLOGMOD=SNX32792,          ** TERMINAL TYPE (3279 MODEL 2)**
                DISCNT=NO,
                MAXOUT=7,
                MAXPATH=2,
                PASSLIM=7,
                ISTATUS=ACTIVE,
                VPACING=0,
                SSCPFM=USSSCS
*
T05L0401 LU     LOCADDR=2
T05L0402 LU     LOCADDR=3
T05L0403 LU     LOCADDR=4
```

Figure 201. Network 3270-PLUS: VTAM Listing for Token-Ring Connection via IBM 3725

The network Administrator also tells you that the SSCP ID of the host is decimal 10. You must use the hexadecimal equivalent: X'05000000000A' in the SNA Services Local LU Profiles.

SNA Services Profiles

Valid profiles to connect to this host are shown in Table 29 through Table 34.

Table 29. Network 3270-PLUS: Connection Profiles, Token-Ring via IBM 3725

Connection Profile Name	**LU2**	**LU3**	**LU4**
Attachment Profile Name	TR370A	TR370A	TR370A
Local LU Profile Name	LLU2	LLU3	LLU4
Network Name			
Remote LU Name			
Stop Connection on Inactivity	NO	NO	NO
LU Type	LU2	LU2	LU2
Notify	YES	YES	YES

Table 30. Network 3270-PLUS: Local LU Profiles, Token-Ring via IBM 3725

Local LU Profile Name	**LLU2**	**LLU3**	**LLU4**
LU Type	LU2	LU2	LU2
Network Name			
Local LU Name	T05L0401	T05L0402	T05L0403
Number of Rows	24	24	24
Number of Columns	80	80	80
Local LU Address	2	3	4
SSCP ID	05000000000A	05000000000A	05000000000A

Table 31. Network 3270-PLUS: Attachment Profile, Token-Ring via IBM 3725

Attachment Profile Name	**TR370A**
Control Point Profile Name	TR370C
Logical Link Profile Name	TR370L
Physical Link Profile Name	TR370P
Logical Link Type	TOKEN_RING
Call Type	CALL
Access Routing	LINK_ADDRESS
Remote Link Address	400000100101
Remote SAP Address	04
Stop Attachment on Inactivity	NO

Table 32. Network 3270-PLUS: Control Point Profile, Token-Ring via IBM 3725

Control Point Profile Name	**TR370C**
XID Node Id	020E6150

Table 33. Network 3270-PLUS: Logical Link Profile, Token-Ring via IBM 3725

Logical Link Profile Name	**TR370L**
Transmit Window Count	127
Dynamic Window Increment	1
Retransmit Count	8
Receive Window Count	127
Ring Access Priority	0
Inactivity Timeout	48
Drop link on Inactivity	YES
Response Timeout	2
Acknowledgement Timeout	1
Force Disconnect Timeout	3
Maximum I-Field Size	SYSTEM_DEFINED
Link Trace	TRACE_OFF

Table 34. Network 3270-PLUS: Physical Link Profile, Token-Ring via IBM 3725

Physical Link Profile Name	**TR370P**
Data Link Device Name	trllc0
Local Link Name	
Local SAP Address	04
Maximum Number of Logical Links	2

Note:

1. LU addresses start at number 2 which is the first address that can be used for 3270 devices.
2. The XID Node ID parameter in the Control Point Profile (`XID Node Id=020E6150`) is a concatenation of the IDBLK (`IDBLK=020`) and IDNUM (`IDNUM=E6150`) parameters in the VTAM listing.
3. The ADDR value in the VTAM listing (`ADDR=C1`) must be defined, but is not used.
4. The SSCP ID value in the VTAM listing (`SSCPID=10`) is coded ***decimal***, whilst the SSCP ID in SNA Services (`SSCP ID=05000000000A`) is coded ***hexadecimal***.
5. For host file transfer the *IND$FILE* PC/Host File Transfer and Emulator Program (program number 5665-311 for MVS/TSO, 5664-281 for VM/SP, 5798-DQH for CICS and 5666-316 for VSE/SP 2.1) must be installed at the host.

Network 3270-PLUS Profile

You only need to change the main configuration profile: /usr/lpp/r/comm/sna/3270/profiles/sc3270.prof. In this case, addresses 2, 3 and 4 (LOCADDR=2,LOCADDR=3 and LOCADDR=4) need to be configured as shown in Figure 202.

```
#
# @(#)sc3270.prof    1.3
#

       System Configuration Profile for Rabbit SNA 3270-Plus

:Protocol(SNA)
:ControllerType(3274)

:PH Parms("/usr/lpp/r/comm/sna/profiles/SNAParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = OFF, PH Data = OFF, DH Errors = OFF,
         PSErrors = OFF, PSData = OFF, PSTiming = OFF,
         PEErrors = OFF, PEData = OFF, PETiming = OFF )

     T E R M I N A L     D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
          "/usr/lpp/r/comm/sna/3270/profiles/TermOpts.prof",
          "/usr/lpp/r/comm/sna/3270/profiles/KbdMap",
          Validation Mode = 1)

:Terminal(2, On, 3279)

:Terminal(3, On, 3279)

:Terminal(4, On, 3279)

:End;
```

Figure 202. Network 3270-PLUS: Configuration Profile for three 3279 Terminals

All the terminals are defined as 3279 terminals to match the VTAM definitions in Figure 201 on page 435. As you can see, the Network 3270-PLUS profiles are very much the same, independently of the physical link type.

Token-Ring Connection to SNA System/370 Host using 3174

You have been asked to configure an IBM RT to connect via Token-Ring to an IBM 3174 Cluster Controller. You must configure one 3278 model 2 and two 3279 model 2 terminals. The 3174 is connected to an SNA host via a leased telephone line. The network administrator tells you that the Token-Ring Adapter address of the IBM 3174 is 40000020A013. He has given you a VTAM listing for a multi-dropped line for the IBM 3174 and the IBM RT (Figure 203).

```
**********     LINE11 --- SDLC 9600 BPS HALF DUPLEX LINE     **********
LINE11   LINE  ADDRESS=(011,HALF),
               CLOCKNG=EXT,
               MAXDATA=265,
               MAXLU=200,
               MAXPU=6,
               NEWSYNC=(NO),
               RETRIES=(7,4,5),
               ISTATUS=ACTIVE,
               VPACING=(1)
*              STATOPT=('LINE 11')
        SERVICE ORDER=(LSCA20, RTPU1),
               MAXLIST=6
*
**********     DEFINITION FOR 3174 WITH 2 3278 TERMINALS    ************
LSCA20   PU    ADDR=C1,               ** CONTROLLER ADDRESS          **
               PASSLIM=7,
               PUTYPE=2,
               DISCNT=NO,
               IRETRY=NO,
               ANS=CONT,
               DLOGMOD=D4A32782,      ** TERMINAL TYPE (3278 MODEL 2)**
               MAXDATA=265,
               MAXOUT=7
LSSA2000 LU    LOCADDR=2              ** 1ST TERMINAL ON 3174        **
LSSA2001 LU    LOCADDR=3              ** 2ND TERMINAL ON 3174        **
*
***********  DEFINE IBM RT CONNECTED TO 3174 VIA TOKEN RING  ***********
RTPU1    PU    ADDR=C2,               ** ADDRESS OF IBM RT           **
               PASSLIM=7,
               PUTYPE=2,
               DISCNT=NO,
               IRETRY=NO,
               ANS=CONT,
               MAXDATA=265,
               MAXOUT=7
RTLU2    LU    LOCADDR=2              ** 1ST TERMINAL ON IBM RT      **
               DLOGMOD=D4A32782,      ** TERMINAL TYPE (3278 MODEL 2)**
RTLU3    LU    LOCADDR=3              ** 2ND TERMINAL ON IBM RT      **
               DLOGMOD=D4A32792,      ** TERMINAL TYPE (3279 MODEL 2)**
RTLU4    LU    LOCADDR=4              ** 3RD TERMINAL ON IBM RT      **
               DLOGMOD=D4A32792,      ** TERMINAL TYPE (3279 MODEL 2)**
```

Figure 203. Network 3270-PLUS: VTAM Listing for Token-Ring Connection via IBM 3174

As can be seen, the IBM RT is defined to the host as if it were multi-dropped on the same line as the IBM 3174 and with address X'C2'. The IBM 3174 is customized so that it maps the line address X′C2′ to the Token-Ring SAP address of X′04′. The adapter of the 3174 has a network address of X′40000020A013′. Finally, you find from a host programmer that the SSCP ID is decimal 10. The hexidacimal equivalent: X′05000000000A′ and must be used in the SNA Services Local LU Profiles.

SNA Services Profiles

Valid profiles to connect to this host are shown in Table 35 through Table 40.

Table 35. Network 3270-PLUS: Connection Profiles for Token-Ring Connection via IBM 3174

Connection Profile Name	LU2	LU3	LU4
Attachment Profile Name	TR370A	TR370A	TR370A
Local LU Profile Name	LLU2	LLU3	LLU4
Network Name			
Remote LU Name			
Stop Connection on Inactivity	NO	NO	NO
LU Type	LU2	LU2	LU2
Notify	YES	YES	YES

Table 36. Network 3270-PLUS: Local LU Profiles for Token-Ring Connection via IBM 3174

Local LU Profile Name	LLU2	LLU3	LLU4
LU Type	LU2	LU2	LU2
Network Name			
Local LU Name	RTLU2	RTLU3	RTLU4
Number of Rows	24	24	24
Number of Columns	80	80	80
Local LU Address	2	3	4
SSCP ID	05000000000A	05000000000A	05000000000A

Table 37. Network 3270-PLUS: Attachment Profile for Token-Ring Connection via IBM 3174

Attachment Profile Name	TR370A
Control Point Profile Name	TR370C
Logical Link Profile Name	TR370L
Physical Link Profile Name	TR370P
Logical Link Type	TOKEN_RING
Call Type	CALL
Access Routing	LINK_ADDRESS
Remote Link Address	40000020A013
Remote SAP Address	04
Stop Attachment on Inactivity	NO

Table 38. Network 3270-PLUS: Control Point Profile for Token-Ring Connection via IBM 3174

Control Point Profile Name	TR370C
XID Node Id	00000000

Table 39. Network 3270-PLUS: Logical Link Profile for Token-Ring Connection via IBM 3174

Logical Link Profile Name	TR370L
Transmit Window Count	127
Dynamic Window Increment	1
Retransmit Count	8
Receive Window Count	127
Ring Access Priority	0
Inactivity Timeout	48
Drop link on Inactivity	YES
Response Timeout	2
Acknowledgement Timeout	1
Force Disconnect Timeout	3
Maximum I-Field Size	SYSTEM_DEFINED
Link Trace	TRACE_OFF

Table 40. Network 3270-PLUS: Physical Link Profile for Token-Ring Connection via IBM 3174

Physical Link Profile Name	TR370P
Data Link Device Name	trllc0
Local Link Name	
Local SAP Address	04
Maximum Number of Logical Links	2

Note:

1. LU addresses start at number 2 which is the first address that can be used for 3270 devices.
2. There is no need to setup an XID on the IBM RT, as the IBM RT is effectively connected to the host via a leased line.
3. The Remote Link Address in the attachment profile (`Remote Link Address=40000020A013`) is the address of the Token-Ring adapter of the 3174.
4. The Remote SAP Address in the attachment profile (`Remote SAP Address=04`) is not the address setup in the VTAM listing for the IBM RT, but the SAP address which the 3174 maps to X'04'.
5. The SSCP ID value in the VTAM listing (`SSCPID=10`) is coded ***decimal***, whilst the SSCP ID in SNA Services (`SSCP ID=05000000000A`) is coded ***hexadecimal***.
6. For host file transfer the *IND$FILE* PC/Host File Transfer and Emulator Program (program number 5665-311 for MVS/TSO, 5664-281 for VM/SP, 5798-DQH for CICS and 5666-316 for VSE/SP 2.1) must be installed at the host.

Network 3270-PLUS Profile

Only one profile needs to be changed: the main configuration profile /usr/lpp/r/comm/sna/3270/profiles/sc3270.prof. In this case, addresses 2, 3 and 4 (LOCADDR=2, LOCADDR=3 and LOCADDR=4) need to be configured as shown in Figure 204.

```
#
# @(#)sc3270.prof   1.3
#

      System Configuration Profile for Rabbit SNA 3270-Plus

:Protocol(SNA)
:ControllerType(3274)

:PH Parms("/usr/lpp/r/comm/sna/profiles/SNAParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = OFF, PH Data = OFF, DH Errors = OFF,
        PSErrors = OFF, PSData = OFF, PSTiming = OFF,
        PEErrors = OFF, PEData = OFF, PETiming = OFF )

     T E R M I N A L     D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
         "/usr/lpp/r/comm/sna/3270/profiles/TermOpts.prof",
         "/usr/lpp/r/comm/sna/3270/profiles/KbdMap",
         Validation Mode = 1)

:Terminal(2, On, 3278)

:Terminal(3, On, 3279)

:Terminal(4, On, 3279)

:End;
```

Figure 204. Network 3270-PLUS: Configuration Profile for one 3278 and two 3279 Terminals

The first terminal is set up as a 3278 terminal, the other two terminals are set up as 3279 terminals in accordance with the VTAM definitions in Figure 203 on page 439.

Using IBM RT Console and ASCII Keyboards with IBM System/370 Host

When you are logged on from the IBM RT to an IBM System/370 host using Network 3270-PLUS, you have to use the keyboard of the IBM RT console or an ASCII terminal to work with the IBM System/370 host. Network 3270-PLUS provides the `ibm5151.s3270` VTS profile for an IBM RT console and the `ibm3161.s3270` VTS profile for ASCII terminals to make them look like a 3270 terminal. Both profiles can be found in the `/usr/lpp/r/comm/sna/3270/profiles` directory. We provide you with Table 41 that shows how the functions of an IBM 3270 keyboard is mapped to the keyboards of the IBM RT console and ASCII terminals.

Several 3270 functions require that you use a combination of keys. In the table, we use the + (plus) character to describe that you have to use the two keys of a key combination together. For example: `Alt+F1` means:

Press the `Alt` key and while holding it down press the `F1` key; then release both keys.

We are using the > (greater than) character to illustrate when you have to use the two keys of a key combination one after the other. For example: `Esc>c` means:

Press the `Esc` key, release it, and then press the `c` key.

Table 41 (Page 1 of 2). Mapping of the 3270 Functions for use on the IBM RT

3270 keys	3270 keys emulated on the Console Keyboard	3270 keys emulated on the ASCII Terminal Keyboard
PF1	F1	Esc>1
PF2	F2	Esc>2
PF3	F3	Esc>3
PF4	F4	Esc>4
PF5	F5	Esc>5
PF6	F6	Esc>6
PF7	F7	Esc>7
PF8	F8	Esc>8
PF9	F9	Esc>9
PF10	F10	Esc>0
PF11	F11	Esc>-
PF12	F12	Esc>=
PF13	Shift+F1	Esc>Shift+1
PF14	Shift+F2	Esc>Shift+2
PF15	Shift+F3	Esc>Shift+3
PF16	Shift+F4	Esc>Shift+4
PF17	Shift+F5	Esc>Shift+5
PF18	Shift+F6	Esc>Shift+6
PF19	Shift+F7	Esc>Shift+7
PF20	Shift+F8	Esc>Shift+8
PF21	Shift+F9	Esc>Shift+9
PF22	Shift+F10	Esc>Shift+0
PF23	Shift+F11	Esc>Shift+-
PF24	Shift+F12	Esc>Shift+=

Table 41 (Page 2 of 2). Mapping of the 3270 Functions for use on the IBM RT

3270 keys	3270 keys emulated on the Console Keyboard	3270 keys emulated on the ASCII Terminal Keyboard
PA1 PA2 PA3	Alt+PF1 Alt+PF2 Alt+PF3	Esc>, Esc>. Esc>/
Attention	Esc>a Esc>A	Esc>a
Backtab	Esc>Tab	Esc>Tab
Clear	Esc>c Esc>C	Esc>c
Cursor Select Tab	Esc>g Esc>G	Esc>g Esc>G
Dup	Esc>d Esc>D	Esc>d
Enter	Enter	Return
Erase EOF	Esc>e Esc>E	Esc>e Esc>E
Erase Input	Esc>w Esc>W	Esc>w Esc>W
Field Mark	Esc>m Esc>M	Esc>m Esc>M
Home	Esc>q Esc>Q	Home Esc>q
Reset	Esc>r Esc>R	Esc>r Esc>R
System Req	Esc>s Esc>S	Esc>s Esc>S
Vertical bar	Esc>\	Esc>\

You can also use all of the function keys performing a local function of the Network 3270-PLUS emulation. See Table 42 for the mapping of the function keys.

Table 42. Mapping of the special Functions of Network 3270-PLUS for use on the Console and the ASCII Terminals on the IBM RT

Network 3270-PLUS special Function Keys	**Network 3270-PLUS special Function Keys on the Console**	**Network 3270-PLUS special Function Keys on the ASCII Terminals**
Banner	Alt+F11	Esc>b Esc>B
Cancel	Pause	Esc>Delete Ctrl+Home (Del)
Cursor Sel (toggle)	Alt+F8	Esc>z Esc>Z
Exit	Alt+F10	Esc>x Esc>X
Finish/Suspend	End	Esc>f Esc>F
File Transfer	Alt+t	Esc>t Esc>T
Help	Alt+h	Esc>h
Next Page	Page Down	Esc>n Esc>N
Prev Page	Page Up	Esc>p
Redraw	Alt+F9	Esc>l Esc>L
Screen-to-File	Alt+f	Esc>Shift+]
Screen-to-Printer	PrintScreen	Esc>]
On-line profile change	Alt+F12	Esc>o Esc>O

Using an IBM RT as Gateway Between DEC VAX and System/370.

You are asked to configure a connection from a DEC VAX system to an IBM System/370 host using the IBM RT as a gateway between TCP/IP and SNA. You can do this by using TCP/IP to log in to the IBM RT from the VAX and then start a 3270 emulation to the IBM System/370 host using Network 3270-PLUS. The VMS operating system does not provide TCP/IP support so you must use a non-DEC TCP/IP package as (for example) the Wollongong TCP/IP Software for VMS. This software must be installed and properly customized on the VAX. The IBM RT must have TCP/IP installed an customized as well, and the two hosts must be connected via Ethernet. For the customization of the IBM RT for TCP/IP, see "Basic AIX TCP/IP Customizing" on page 184.

For the 3270 connection from the IBM RT to the IBM System/370 host, you can use any of the connections described in:

- "SDLC Dial-up Line to SNA System/370 Host" on page 415
- "SDLC leased Line to SNA System/370 Host" on page 419
- "BSC leased Line to non-SNA 370 Host" on page 428
- "X.25 Connection to SNA System/370 Host using 3725" on page 431
- "Token-Ring Connection to SNA System/370 Host using 3725" on page 435
- "Token-Ring Connection to SNA System/370 Host using 3174" on page 439.

Logging on from a VAX to an IBM System/370 Host

You can use a VT100, VT220 or VT320 terminal connected to the VAX to log in to the IBM RT using the *telnet* command. You should be defined to the IBM RT as a VT100 terminal because the VTS profile of Network 3270-PLUS that maps the keyboard of the VT terminal into a 3270 terminal keyboard uses the `Esc` key in combination with other keys to emulate many of the 3270 keys.

The `Esc` key is only available on a VT100 terminal, not on the more advanced terminals like VT220 and VT320. You can change the appearance of these terminals by switching to "general setup mode", then change the terminal characteristic to VT100 mode. This defines the `Esc` key on the `F11` key of the VT220 or VT320 keyboard.

A step-by-step description of how to log on to the IBM System/370 host:

1. Logon to the VAX.
2. Set the terminal mode of the VT220 or VT320 terminal to VT100 mode using "general setup".
3. Establish a *telnet* connection to the IBM RT.
4. Log in to the IBM RT and set the `TERM` environment variable to `vt100`, when you are working with a VT100 terminal, by typing: `TERM=vt100`. Set the `TERM` environment variable to `vt220` when you use a VT220 or VT320 terminal by keying: `TERM=vt220`.
5. Type: `export TERM` to export the new value of `TERM` to the shell.
6. Start a 3270 session on the IBM RT by keying: `s3270 s n` - where "n" is a valid terminal number. Network 3270-PLUS must already have been started.

Now you should receive the logon screen of the IBM System/370 host on your terminal.

Using VT Terminal Keyboard with IBM System/370 Host

When you are logged on from the VAX to the IBM System/370 host through Network 3270-PLUS, you have to use the keyboard of the VT terminal to work with the IBM System/370 host. Network 3270-PLUS provides the `vt100.s3270` VTS profile for a VT100 terminal and the `vt220.s3270` VTS profile for the VT220 and VT320 terminals to make them appear to the IBM System/370 host as a 3270 terminal. Both profiles are in the `/usr/lpp/r/comm/sna/3270/profiles` directory. Table 43 shows the mapping between an IBM 3270 keyboard and the keyboards of the VT terminals.

Some 3270 functions are invoked by a combination of keys on the VT terminal. We are using the > (greater than) character to illustrate that you must use the two keys of a key combination one after the other. For example: `Esc>1` means:

Press the `Esc` key and release it; then press the `1` key.

We are using the + (plus) character to illustrate that you must use the two keys of a key combination simultaneously. For example: `Esc>Shift+1` means:

Press the `Esc` key and release it; then press the `Shift` key and hold it down while pressing the `1` key; then release both keys.

Table 43 (Page 1 of 2). Mapping of the 3270 Functions for Use on a VT terminal

3270 keys	3270 keys emulated on the VT Keyboard
PF1	Esc>1
PF2	Esc>2
PF3	Esc>3
PF4	Esc>4
PF5	Esc>5
PF6	Esc>6
PF7	Esc>7
PF8	Esc>8
PF9	Esc>9
PF10	Esc>0
PF11	Esc>-
PF12	Esc>=
PF13	Esc>Shift+1
PF14	Esc>Shift+2
PF15	Esc>Shift+3
PF16	Esc>Shift+4
PF17	Esc>Shift+5
PF18	Esc>Shift+6
PF19	Esc>Shift+7
PF20	Esc>Shift+8
PF21	Esc>Shift+9
PF22	Esc>Shift+0
PF23	Esc>Shift+-
PF24	Esc>Shift+=

Table 43 (Page 2 of 2). Mapping of the 3270 Functions for Use on a VT terminal

3270 keys	3270 keys emulated on the VT Keyboard
PA1 PA2 PA3	PF1 PF2 PF3
Attention	Esc>a
BACKTAB	Esc>Tab
Clear	Esc>c
Cursor Select Tab	Esc>g Esc>G
DUP	Esc>d Esc>D
Enter	Return
Erase EOF	Esc>e Esc>E
Erase Input	Esc>w Esc>W
Field Mark	Esc>m Esc>M
Home	Esc>q Esc>Q
Reset	Esc>r Esc>R
System Req	Esc>s Esc>S
Vertical bar	Esc>\

You can also use all of the function keys performing a local Network 3270-PLUS function. See Table 44 for the mapping of the function keys.

Table 44. Mapping of the special Functions of Network 3270-PLUS for Use on a VT Keyboard

Network 3270-PLUS special Function Keys	Network 3270-PLUS special Function Keys on a VT Keyboard
Banner	Esc>b Esc>B
Cancel	Esc>Delete
Cursor Sel (toggle)	Esc>z Esc>Z
Exit	Esc>x Esc>X
Finish/Suspend	Esc>f Esc>F
File Transfer	Esc>t Esc>T
Help	Esc>h Esc>H
Next Page	Esc>n Esc>N
Prev Page	Esc>p Esc>P
Redraw	Esc>l Esc>L
Screen-to-File	Esc>Shift+]
Screen-to-Printer	Esc>]
On-line profile change	Esc>o

SDLC SNA Connection to IBM AS/400 Host

You are asked to configure an IBM RT for 3270 communications to an IBM AS/400 host using an SDLC leased line. You are asked to configure two 3277 terminals, one SNA printer and one non-SNA printer. An SDLC leased line on the IBM AS/400 has been configured for you. The IBM AS/400 description for the line is shown in Figure 206, for the controller in Figure 205, for the terminals in Figure 207 and for the printers in Figure 208. A control language program that automatically configures the IBM AS/400 descriptions is shown in Figure 209.

IBM AS/400 Descriptions

```
                  Display Controller Description - Remote WS

Controller description . . . . . . :   CTLD         RT3270NSC
Controller type  . . . . . . . . . :   TYPE         3274
Controller model . . . . . . . . . :   MODEL        0
Link type  . . . . . . . . . . . . :   LINKTYPE     *SDLC
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Switched line  . . . . . . . . . . :   SWITCHED     *NO
Switched network backup  . . . . . :   SNBU         *NO
Activate swt network backup  . . . :   ACTSNBU
Attached nonswitched line  . . . . :   LINE         RT3270NSL
Switched line list . . . . . . . . :   SWTLINLST
```

```
                  Display Controller Description - Remote WS

Attached device(s) . . . . . . . . :   DEV
  RT3270NSD1    RT3270NSD2    RT3270NSD3    RT3270NSD4
Character code . . . . . . . . . . :   CODE         *EBCDIC
Device wait timer  . . . . . . . . :   DEVWAITTMR   120
Maximum frame size . . . . . . . . :   MAXFRAME
Exchange identifier  . . . . . . . :   EXCHID
SSCP identifier  . . . . . . . . . :   SSCPID       050000000015
Initial connection . . . . . . . . :   INLCNN
Connection number  . . . . . . . . :   CNNNBR
Predial delay  . . . . . . . . . . :   PREDIALDLY
Redial delay . . . . . . . . . . . :   REDIALDLY
Dial retry . . . . . . . . . . . . :   DIALRTY
Remote autoanswer  . . . . . . . . :   RMTAUTOANS
```

```
                  Display Controller Description - Remote WS

Station address  . . . . . . . . . :   STNADR       C1
SDLC poll priority . . . . . . . . :   POLLPTY      *NO
SDLC poll limit  . . . . . . . . . :   POLLLMT      0
SDLC connect poll retry  . . . . . :   CNNPOLLRTY   *NOMAX
SDLC NRM poll timer  . . . . . . . :   NRMPOLLTMR   0
SDLC NDM poll timer  . . . . . . . :   NDMPOLLTMR   *CALC
Text . . . . . . . . . . . . . . . :   TEXT         AS400 to IBM RT
                                                    SDLC for
                                                    3270 Emulation
```

Figure 205. Network 3270-PLUS: IBM AS/400 SNA/SDLC Controller Description

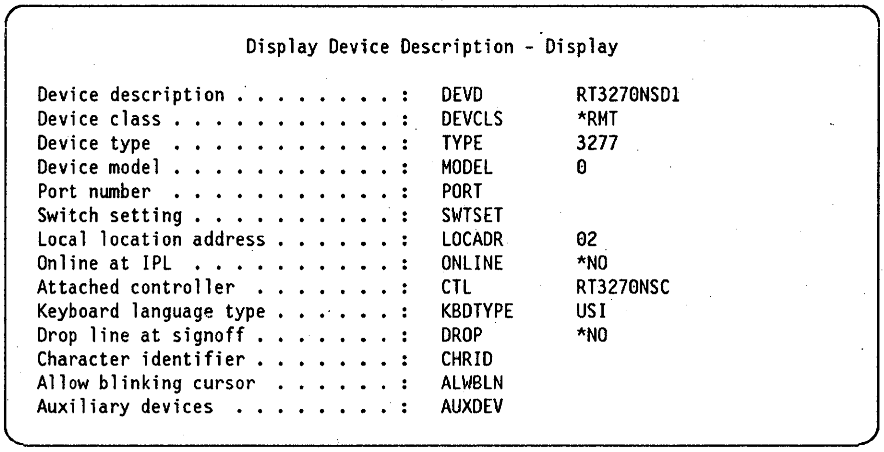
```
                    Display Line Description - SDLC

Line description . . . . . . . . . :   LIND          RT3270NSL
Resource name  . . . . . . . . . . :   RSRCNAME      LIN012
Online at IPL  . . . . . . . . . . :   ONLINE        *NO
Data link role . . . . . . . . . . :   ROLE          *PRI
Physical interface . . . . . . . . :   INTERFACE     *RS232V24
Connection type  . . . . . . . . . :   CNN           *NONSWTPP
Switched network backup  . . . . . :   SNBU          *NO
Activate swt network backup  . . . :   ACTSNBU
Vary on wait . . . . . . . . . . . :   VRYWAIT
Autocall unit  . . . . . . . . . . :   AUTOCALL
Attached nonswitched ctl(s)  . . . :   CTL
  RT3270NSC
```

```
                    Display Line Description - SDLC

NRZI data encoding . . . . . . . . :   NRZI          *YES
Maximum controllers  . . . . . . . :   MAXCTL        1
Line speed . . . . . . . . . . . . :   LINESPEED     9600
Modem type supported . . . . . . . :   MODEM         *NORMAL
Modem data rate select . . . . . . :   MODEMRATE     *FULL
```

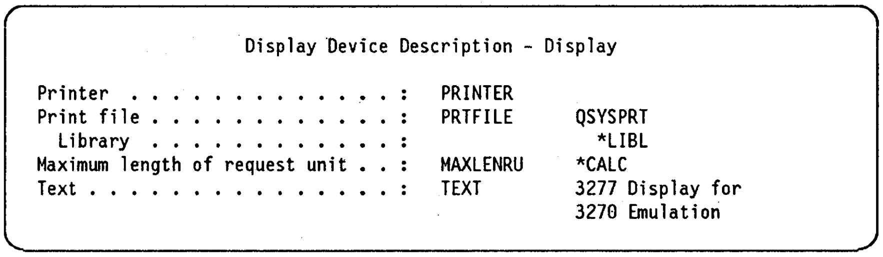
```
                    Display Line Description - SDLC

Maximum frame size . . . . . . . . :   MAXFRAME      521
Error threshold level  . . . . . . :   THRESHOLD     *OFF
Duplex . . . . . . . . . . . . . . :   DUPLEX        *HALF
Modulus  . . . . . . . . . . . . . :   MODULUS       8
Maximum outstanding frames . . . . :   MAXOUT        7
Inactivity timer . . . . . . . . . :   INACTTMR
Poll response delay  . . . . . . . :   POLLRSPDLY
Nonproductive receive timer  . . . :   NPRDRCVTMR    320
Idle timer . . . . . . . . . . . . :   IDLTMR        30
Connect poll timer . . . . . . . . :   CNNPOLLTMR    30
Poll cycle pause . . . . . . . . . :   POLLPAUSE     0
Frame retry  . . . . . . . . . . . :   FRAMERTY      7
```

```
                    Display Line Description - SDLC

Link speed . . . . . . . . . . . . :   LINKSPEED     9600
Cost/connect time  . . . . . . . . :   COSTCNN       0
Cost/byte  . . . . . . . . . . . . :   COSTBYTE      0
Security for line  . . . . . . . . :   SECURITY      *NONSECURE
Propagation delay  . . . . . . . . :   PRPDLY        *TELEPHONE
User-defined 1 . . . . . . . . . . :   USRDFN1       128
User-defined 2 . . . . . . . . . . :   USRDFN2       128
User-defined 3 . . . . . . . . . . :   USRDFN3       128
Text . . . . . . . . . . . . . . . :   TEXT          AS400 to IBM RT
                                                     SDLC for
                                                     3270 Emulation
```

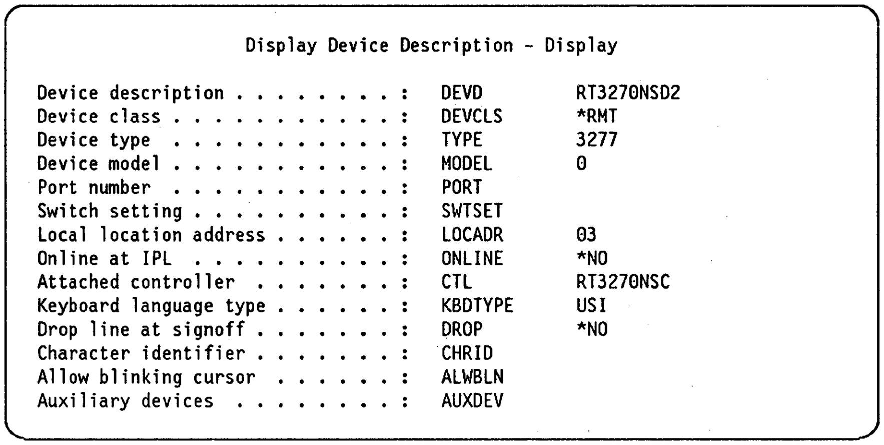
Figure 206. Network 3270-PLUS: IBM AS/400 SNA/SDLC Line Description



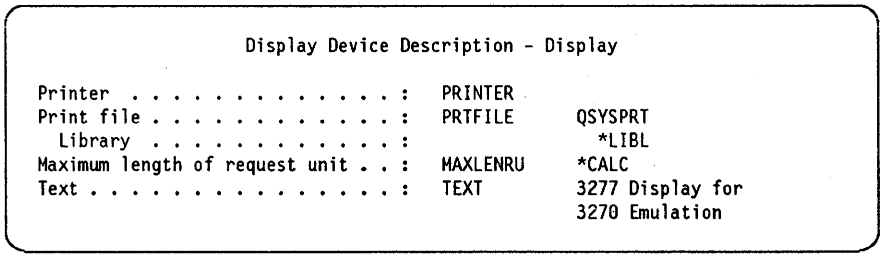
```
                   Display Device Description - Display

Device description . . . . . . . . :   DEVD         RT3270NSD1
Device class . . . . . . . . . . . :   DEVCLS       *RMT
Device type  . . . . . . . . . . . :   TYPE         3277
Device model . . . . . . . . . . . :   MODEL        0
Port number  . . . . . . . . . . . :   PORT
Switch setting . . . . . . . . . . :   SWTSET
Local location address . . . . . . :   LOCADR       02
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Attached controller  . . . . . . . :   CTL          RT3270NSC
Keyboard language type . . . . . . :   KBDTYPE      USI
Drop line at signoff . . . . . . . :   DROP         *NO
Character identifier . . . . . . . :   CHRID
Allow blinking cursor  . . . . . . :   ALWBLN
Auxiliary devices  . . . . . . . . :   AUXDEV
```



```
                   Display Device Description - Display

Printer  . . . . . . . . . . . . . :   PRINTER
Print file . . . . . . . . . . . . :   PRTFILE      QSYSPRT
  Library  . . . . . . . . . . . . :                  *LIBL
Maximum length of request unit . . :   MAXLENRU     *CALC
Text . . . . . . . . . . . . . . . :   TEXT         3277 Display for
                                                    3270 Emulation
```

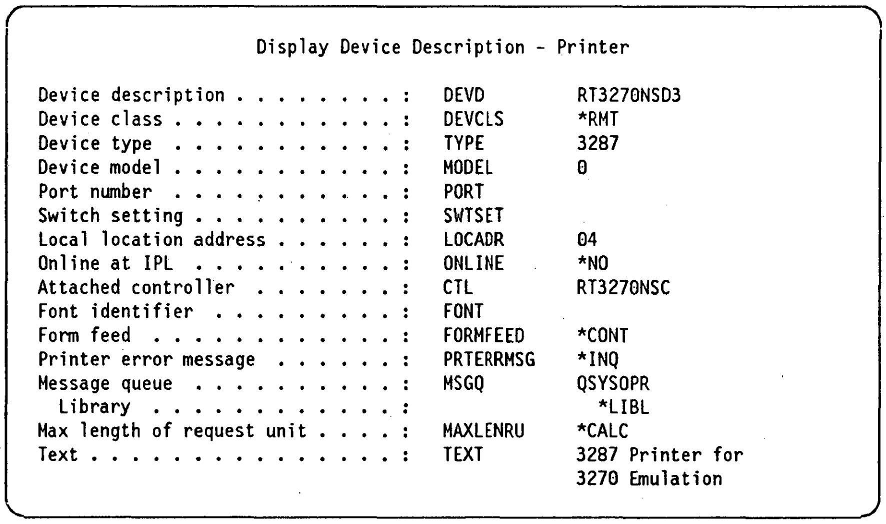


```
                   Display Device Description - Display

Device description . . . . . . . . :   DEVD         RT3270NSD2
Device class . . . . . . . . . . . :   DEVCLS       *RMT
Device type  . . . . . . . . . . . :   TYPE         3277
Device model . . . . . . . . . . . :   MODEL        0
Port number  . . . . . . . . . . . :   PORT
Switch setting . . . . . . . . . . :   SWTSET
Local location address . . . . . . :   LOCADR       03
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Attached controller  . . . . . . . :   CTL          RT3270NSC
Keyboard language type . . . . . . :   KBDTYPE      USI
Drop line at signoff . . . . . . . :   DROP         *NO
Character identifier . . . . . . . :   CHRID
Allow blinking cursor  . . . . . . :   ALWBLN
Auxiliary devices  . . . . . . . . :   AUXDEV
```

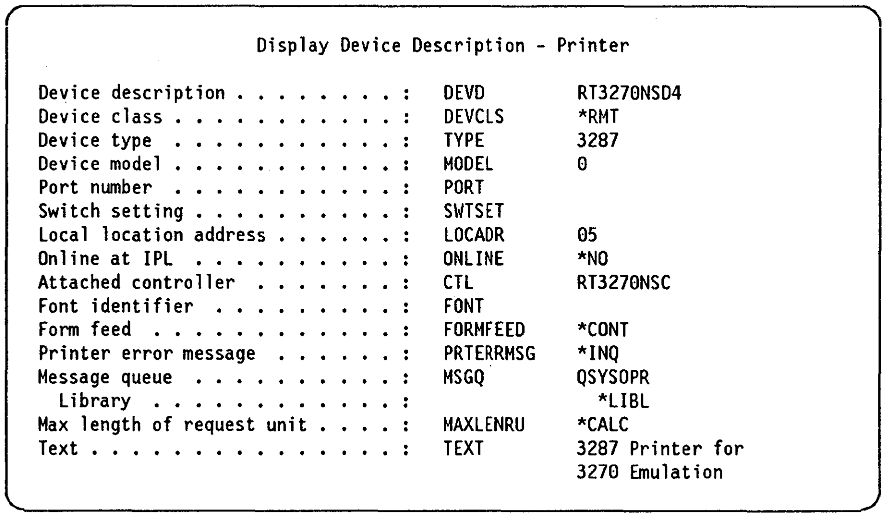


```
                   Display Device Description - Display

Printer  . . . . . . . . . . . . . :   PRINTER
Print file . . . . . . . . . . . . :   PRTFILE      QSYSPRT
  Library  . . . . . . . . . . . . :                  *LIBL
Maximum length of request unit . . :   MAXLENRU     *CALC
Text . . . . . . . . . . . . . . . :   TEXT         3277 Display for
                                                    3270 Emulation
```

Figure 207. Network 3270-PLUS: IBM AS/400 SNA/SDLC Display Device Description



```
                     Display Device Description - Printer

Device description . . . . . . . . :   DEVD         RT3270NSD3
Device class . . . . . . . . . . . :   DEVCLS       *RMT
Device type  . . . . . . . . . . . :   TYPE         3287
Device model . . . . . . . . . . . :   MODEL        0
Port number  . . . . . . . . . . . :   PORT
Switch setting . . . . . . . . . . :   SWTSET
Local location address . . . . . . :   LOCADR       04
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Attached controller  . . . . . . . :   CTL          RT3270NSC
Font identifier  . . . . . . . . . :   FONT
Form feed  . . . . . . . . . . . . :   FORMFEED     *CONT
Printer error message  . . . . . . :   PRTERRMSG    *INQ
Message queue  . . . . . . . . . . :   MSGQ         QSYSOPR
  Library  . . . . . . . . . . . . :                  *LIBL
Max length of request unit . . . . :   MAXLENRU     *CALC
Text . . . . . . . . . . . . . . . :   TEXT         3287 Printer for
                                                    3270 Emulation
```



```
                     Display Device Description - Printer

Device description . . . . . . . . :   DEVD         RT3270NSD4
Device class . . . . . . . . . . . :   DEVCLS       *RMT
Device type  . . . . . . . . . . . :   TYPE         3287
Device model . . . . . . . . . . . :   MODEL        0
Port number  . . . . . . . . . . . :   PORT
Switch setting . . . . . . . . . . :   SWTSET
Local location address . . . . . . :   LOCADR       05
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Attached controller  . . . . . . . :   CTL          RT3270NSC
Font identifier  . . . . . . . . . :   FONT
Form feed  . . . . . . . . . . . . :   FORMFEED     *CONT
Printer error message  . . . . . . :   PRTERRMSG    *INQ
Message queue  . . . . . . . . . . :   MSGQ         QSYSOPR
  Library  . . . . . . . . . . . . :                  *LIBL
Max length of request unit . . . . :   MAXLENRU     *CALC
Text . . . . . . . . . . . . . . . :   TEXT         3287 Printer for
                                                    3270 Emulation
```

Figure 208. Network 3270-PLUS: IBM AS/400 SNA/SDLC Printer Device Description

```
5728PW1 R01M02  881028                 SEU SOURCE LISTING
SOURCE FILE . . . . . . .  STELLA/QCLSRC
MEMBER  . . . . . . . . .  RT3270NS2
PGM
/**********************************************************************/
/* THIS PROGRAM CREATES THE LINE, CONTROLLER AND DEVICE DESCRIPTIONS */
/* WHICH CAN BE USED TO CONNECT THE AS/400 VIA A NON-SWITCHED SDLC   */
/* LINE TO AN RT RUNNING 3270 EMULATION. THE DEVICES THAT ARE CREATED*/
/* ARE : 2X3277 AND 2X3287.                                          */
/* ANY CHANGES THAT NEED TO BE MADE TO THE LINE, CONTROLLER OR       */
/* DEVICE DESCRIPTIONS CAN BE MADE IN THIS PROGRAM. WHEN RUN, THE    */
/* EXISTING LINE, WITH ITS ATTACHED CONTROLLER AND DEVICES, WILL BE  */
/* VARIED OFF AND DELETED (IF THEY EXIST) THEN CREATED.              */
/*                                                                   */
/**********************************************************************/
             VRYCFG     CFGOBJ(RT3270NSL) CFGTYPE(*LIN) STATUS(*OFF)
             MONMSG     MSGID(CPF9999)

             DLTDEVD    DEVD(RT3270NSD*)
             MONMSG     MSGID(CPF9999)

             DLTCTLD    CTLD(RT3270NSC)
             MONMSG     MSGID(CPF9999)

             DLTLIND    LIND(RT3270NSL)
             MONMSG     MSGID(CPF9999)

             CRTLINSDLC LIND(RT3270NSL) RSRCNAME(LIN012) ONLINE(*N
                          ROLE(*PRI) TEXT('AS400 to IBM RT +
                          SDLC for 3270 Emulation')
             CRTCTLRWS  CTLD(RT3270NSC) TYPE(3274) MODEL(0) +
                          LINKTYPE(*SDLC) ONLINE(*NO) LINE(RT3270NSL) +
                          STNADR(C1) TEXT('AS400 to IBM RT SDLC
                          for 3270 Emulation')
             CRTDEVDSP  DEVD(RT3270NSD1) DEVCLS(*RMT) TYPE(3277) +
                          MODEL(0) LOCADR(02) ONLINE(*NO) +
                          CTL(RT3270NSC) KBDTYPE(USI) DROP(*NO) +
                          TEXT('3277 Display for 3270 Emulation')
             CRTDEVDSP  DEVD(RT3270NSD2) DEVCLS(*RMT) TYPE(3277) +
                          MODEL(0) LOCADR(03) ONLINE(*NO) +
                          CTL(RT3270NSC) KBDTYPE(USI) DROP(*NO) +
                          TEXT('3277 Display for 3270 Emulation')
             CRTDEVPRT  DEVD(RT3270NSD3) DEVCLS(*RMT) TYPE(3287) +
                          MODEL(0) LOCADR(04) ONLINE(*NO) +
                          CTL(RT3270NSC) TEXT('3287 Printer for 3270 +
                          Emulation')
             CRTDEVPRT  DEVD(RT3270NSD4) DEVCLS(*RMT) TYPE(3287) +
                          MODEL(0) LOCADR(05) ONLINE(*NO) +
                          CTL(RT3270NSC) TEXT('3287 Printer for 3270 +
                          Emulation')
ENDPGM
```

Figure 209. Network 3270-PLUS: IBM AS/400 CL Program for Creating SNA/SDLC Descriptions

SNA Services Profiles

Valid profiles to connect to this host are shown in Table 45 through Table 50.

Table 45. Network 3270-PLUS: Connection Profiles for SDLC Leased Line to IBM AS/400

Connection Profile Name	LU2	LU3	LU4	LU5
Attachment Profile Name	SDLCAS4A	SDLCAS4A	SDLCAS4A	SDLCAS4A
Local LU Profile Name	LLU2	LLU3	LLU4	LLU5
Network Name				
Remote LU Name				
Stop Connection on Inactivity	NO	NO	NO	NO
LU Type	LU2	LU2	LU1	LU3
Notify	YES	YES	YES	YES

Table 46. Network 3270-PLUS: Local LU Profiles for SDLC Leased Line to IBM AS/400

Local LU Profile Name	LLU2	LLU3	LLU4	LLU5
LU Type	LU2	LU2	LU1	LU3
Network Name				
Local LU Name	ASLU2	ASLU3	ASLU4	ASLU5
Number of Rows	24	24	24	24
Number of Columns	80	80	80	132
Local LU Address	2	3	4	5
SSCP ID	05000000000F	05000000000F	05000000000F	05000000000F

Table 47. Network 3270-PLUS: Attachment Profile for SDLC Leased Line to IBM AS/400

Attachment Profile Name	**SDLCAS4A**
Control Point Profile Name	SDLCAS4C
Logical Link Profile Name	SDLCAS4L
Physical Link Profile Name	SDLCAS4P
Logical Link Type	SDLC
Station Type	SECONDARY
Physical Link Type	RS232
Stop Attachment on Inactivity	NO

Table 48. Network 3270-PLUS: Control Point Profile for SDLC Leased Line to IBM AS/400

Control Point Profile Name	**SDLCAS4C**
XID Node Id	00000000

Table 49. Network 3270-PLUS: Logical Link Profile for SDLC Leased Line to IBM AS/400

Logical Link Profile Name	**SDLCAS4L**
Station Type	SECONDARY
Local Secondary Station Address	193
Serial Encoding	NRZI
Transmit Window Count	7
Retransmit Window Count	10
Retransmit Threshold	10
Drop Link on Inactivity	NO
Force Disconnect Timeout	120
Maximum I-Field Size	SYSTEM_DEFINED
Link Trace	TRACE_OFF

Table 50. Network 3270-PLUS: Physical Link Profile for SDLC Leased Line to IBM AS/400

Physical Link Profile Name	**SDLCAS4P**
Data Link Device Name	sdlcllc0
Request to Send	CONTINUOUS
Bit Clocking	EXTERNAL
Data Rate Select	FULL
Switched Network Backup	BACKUP_OFF
Network Type	NONSWITCHED
DTR Control	DTR

Note: Several parameters of IBM AS/400 and IBM RT profiles ***must*** match. Otherwise, you will ***not*** be able to establish a connection:

1. The IBM AS/400 is acting as a primary station (`ROLE=*PRI`), so the IBM RT must be defined as secondary (`Station Type=SECONDARY`). A primary station always starts a connection (polls) and a secondary station always waits for polling before it can answer.

2. The line protocol is set to SDLC in the IBM AS/400 controller description. (`LINKTYPE=*SDLC`). This protocol must also be used in the IBM RT attachment profile (`Logical Link Type=SDLC`).

3. The station address is set to X'C1' in the IBM AS/400 controller description (`STNADR=C1`). The corresponding decimal value in the SNA Services Logical Link Profile station address is set to 193 to match this (`Local Secondary Station Address=193`).

4. The data encoding is set to NRZI in the IBM AS/400 line description (`NRZI=*YES`). This parameter value must also be set in the IBM RT Logical Link Profile (`Serial Encoding=NRZI`).

5. The SSCP identification is set to 050000000015 in the IBM AS/400 controller description (`SSCPID=050000000015`). In the SNA Services Local LU Profiles you must enter the hexadecimal equivalent (`SSCPID=05000000000F`).

6. The XID parameter is not used.

7. The local station addresses for the terminals and printers in the IBM AS/400 device descriptions have the numbers 02, 03, 04, and 05 (`LOCADR=02, ..., LOCADR=05`). These parameter values must match those in the IBM RT SNA Services Local LU Profiles (`Local LU Address=2, ..., Local LU Address=5`) and those in the Network 3270-PLUS configuration profile (`:Terminal(2, On, 3277), :Terminal(3, On, 3277), :Printer(4, On, 3287 ...)` and `:Printer(5, On, 3287 ...)`).

8. The type of terminal and printer you use in the IBM AS/400 device description (`TYPE=3277, ..., TYPE=3287`) must match the LU type in the SNA Services Connection Profile and the Local LU Profile on the IBM RT (`LU Type=LU2, ..., LU Type=LU1` and `LU Type=LU3`), and must also match the entries in the Network 3270-PLUS Configuration Profile (`:Terminal(2, On, 3277), ..., :Printer(5, On, 3287, ...)`). Do ***not*** confuse the Connection Profile names (LU2, LU3, LU4, and LU5) with the LU type (LU1 is the LU type for a non SNA printer, LU2 is the LU type for a terminal, and LU3 is the LU type for an SNA printer).

9. For the connection to the IBM AS/400, you can only specify device type 3277 for a terminal and 3287 for a printer in the IBM AS/400 device descriptions. This means that the IBM AS/400 sends only a generic 3270 data stream to the device. Such a data stream does ***not*** include extended attributes such as underlining, reverse image, or color. The device types 3278 with extended attributes and 3279 with extended attributes, including color, could be specified, but will not work with the Network 3270-PLUS emulation. A 3287 printer device specified in the IBM AS/400 device description can support SNA character string (SCS) commands and orders. The LU Type 3 specified in the IBM RT Connection- and Local LU Profiles is also capable of SCS. For a complete description of the 3270 devices emulated by the IBM AS/400, see *IBM AS/400 3270 Device Emulation Users Guide*.

10. There is no file transfer program supported on the IBM AS/400, so you can't use the file transfer commands with the IBM AS/400. The 3270-PLUS Programmer Interface Modules (PIM) enable application programs on the host to directly interface with the Network 3270-PLUS program. You can use the functions of PIM to do file transfers. An example of how to do this with the IBM AS/400 in IBM S/36 emulation mode is shown in Appendix E, "Network 3270-PLUS File Transfer Programs" on page 483.

Network 3270-PLUS Profile

Only the Network 3270-PLUS main configuration profile /usr/lpp/r/comm/sna/profiles/sc3270.prof needs to be changed. Addresses 2, 3, 4, and 5 must be configured as shown in Figure 210.

```
#
# @(#)sc3270.prof 1.3
#

System Configuration Profile for Rabbit SNA 3270-Plus

:Protocol(SNA)
:ControllerType(3274)

:PH Parms("/usr/lpp/r/comm/sna/profiles/SNAParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = OFF, PH Data = OFF, DH Errors = OFF, DH Timing = OFF,
      PSErrors = OFF, PSData = OFF, PSTiming = OFF,
         PEErrors = OFF, PEData = OFF, PETiming = OFF )

     T E R M I N A L     D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
"/usr/lpp/r/comm/sna/3270/profiles/TermOpts.prof",
"/usr/lpp/r/comm/sna/3270/profiles/KbdMap",
Validation Mode = 1)

:Terminal(2, On, 3277)

:Terminal(3, On, 3277)

     P R I N T E R     D E F I N I T I O N S

:PrinterParms(TIMEOUT = 15, SPOOLSIZE = 3)

:Printer(4, ON, 3287, 1, 1920,
"/usr/lpp/r/comm/sna/3270/profiles/PrtOpts.prof")

:Printer(5, ON, 3287, 2, 1920,
"/usr/lpp/r/comm/sna/3270/profiles/PrtOpts.prof")

:End;
```

Figure 210. Network 3270-PLUS: SNA Configuration Profile for two 3277 and two 3287

The terminals are defined as IBM 3277 terminals to match the IBM AS/400 display device descriptions in Figure 207 on page 452. The printers are defined as 3287 printers to match the IBM AS/400 printer device descriptions in Figure 208 on page 453.

Token-Ring Connection to SNA IBM AS/400 Host

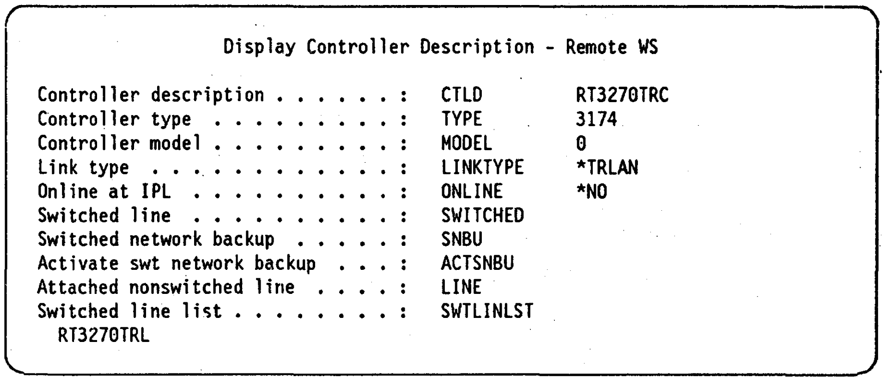
You are asked to configure an IBM RT for 3270 communications to an IBM AS/400 host via Token-Ring. You are asked to configure two 3277 terminals, one SNA printer and one non SNA printer. A Token-Ring connection on the IBM AS/400 has been configured for you. The Token-Ring adapter in the IBM AS/400 has the address 4005A04083. The IBM AS/400 description for the Token-Ring line is shown in Figure 211, for the controller in Figure 212, for the terminals in Figure 213 and for the printers in Figure 214. A control language program to automatically configure the IBM AS/400 descriptions is shown in Figure 215.

IBM AS/400 Descriptions

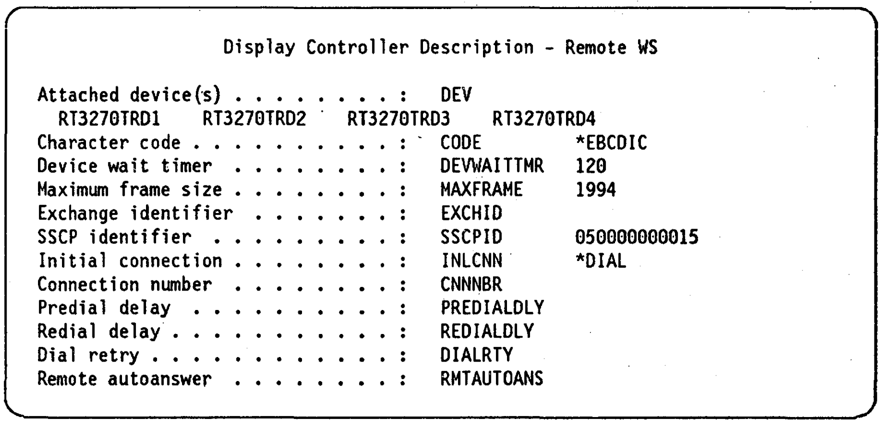
```
                    Display Line Description - Token-Ring

Line description . . . . . . . . . :   LIND         RT3270TRL
Resource name  . . . . . . . . . . :   RSCRCNAME    LIN031
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Vary on wait . . . . . . . . . . . :   VRYWAIT      *NOWAIT
Maximum controllers  . . . . . . . :   MAXCTL       40
Maximum frame size . . . . . . . . :   MAXFRAME     1994
TRLAN Manager logging level  . . . :   TRNLOGLVL    *OFF
Current logging level  . . . . . . :                *OFF
Local adapter address  . . . . . . :   ADPTADR      *ADPT
Exchange identifier  . . . . . . . :   EXCHID
Source service access points . . . :   SSAP
  04
Error threshold level  . . . . . . :   THRESHOLD    *OFF
```

```
                    Display Line Description - Token-Ring

Link speed . . . . . . . . . . . . :   LINKSPEED    4M
Cost/connect time  . . . . . . . . :   COSTCNN      0
Cost/byte  . . . . . . . . . . . . :   COSTBYTE     0
Security for line  . . . . . . . . :   SECURITY     *NONSECURE
Propagation delay  . . . . . . . . :   PRPDLY       *LAN
User-defined 1 . . . . . . . . . . :   USRDFN1      128
User-defined 2 . . . . . . . . . . :   USRDFN2      128
User-defined 3 . . . . . . . . . . :   USRDFN3      128
Text . . . . . . . . . . . . . . . :   TEXT         AS400 to IBM RT
                                                    Token-Ring
                                                    3270 Emulation
```

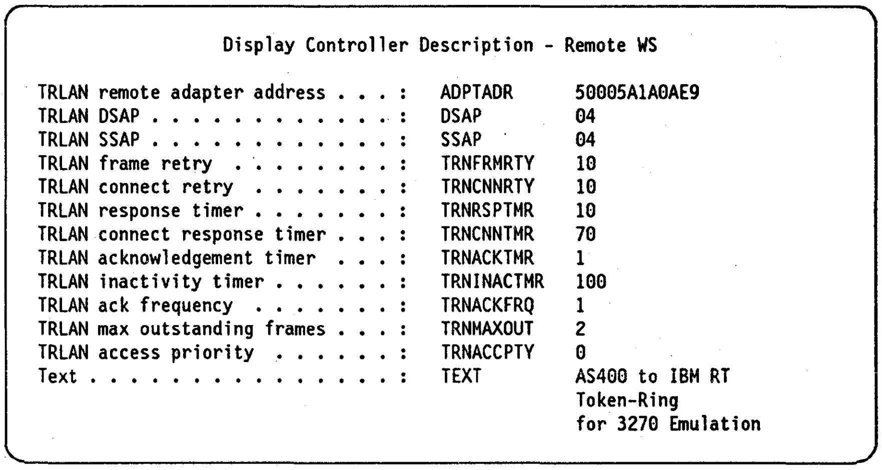
Figure 211. Network 3270-PLUS: IBM AS/400 Token-Ring Line Description



```
                  Display Controller Description - Remote WS

Controller description . . . . . . :   CTLD          RT3270TRC
Controller type  . . . . . . . . . :   TYPE          3174
Controller model . . . . . . . . . :   MODEL         0
Link type  . . . . . . . . . . . . :   LINKTYPE      *TRLAN
Online at IPL  . . . . . . . . . . :   ONLINE        *NO
Switched line  . . . . . . . . . . :   SWITCHED
Switched network backup  . . . . . :   SNBU
Activate swt network backup  . . . :   ACTSNBU
Attached nonswitched line  . . . . :   LINE
Switched line list . . . . . . . . :   SWTLINLST
  RT3270TRL
```



```
                  Display Controller Description - Remote WS

Attached device(s) . . . . . . . . :   DEV
  RT3270TRD1    RT3270TRD2    RT3270TRD3    RT3270TRD4
Character code . . . . . . . . . . :   CODE          *EBCDIC
Device wait timer  . . . . . . . . :   DEVWAITTMR    120
Maximum frame size . . . . . . . . :   MAXFRAME      1994
Exchange identifier  . . . . . . . :   EXCHID
SSCP identifier  . . . . . . . . . :   SSCPID        050000000015
Initial connection . . . . . . . . :   INLCNN        *DIAL
Connection number  . . . . . . . . :   CNNNBR
Predial delay  . . . . . . . . . . :   PREDIALDLY
Redial delay . . . . . . . . . . . :   REDIALDLY
Dial retry . . . . . . . . . . . . :   DIALRTY
Remote autoanswer  . . . . . . . . :   RMTAUTOANS
```



```
                  Display Controller Description - Remote WS

TRLAN remote adapter address . . . :   ADPTADR       50005A1A0AE9
TRLAN DSAP . . . . . . . . . . . . :   DSAP          04
TRLAN SSAP . . . . . . . . . . . . :   SSAP          04
TRLAN frame retry  . . . . . . . . :   TRNFRMRTY     10
TRLAN connect retry  . . . . . . . :   TRNCNNRTY     10
TRLAN response timer . . . . . . . :   TRNRSPTMR     10
TRLAN connect response timer . . . :   TRNCNNTMR     70
TRLAN acknowledgement timer  . . . :   TRNACKTMR     1
TRLAN inactivity timer . . . . . . :   TRNINACTMR    100
TRLAN ack frequency  . . . . . . . :   TRNACKFRQ     1
TRLAN max outstanding frames . . . :   TRNMAXOUT     2
TRLAN access priority  . . . . . . :   TRNACCPTY     0
Text . . . . . . . . . . . . . . . :   TEXT          AS400 to IBM RT
                                                     Token-Ring
                                                     for 3270 Emulation
```

Figure 212. Network 3270-PLUS: IBM AS/400 Token-Ring Controller Description

```
                    Display Device Description - Display

Device description . . . . . . . . :   DEVD         RT3270TRD1
Device class . . . . . . . . . . . :   DEVCLS       *RMT
Device type  . . . . . . . . . . . :   TYPE         3277
Device model . . . . . . . . . . . :   MODEL        0
Port number  . . . . . . . . . . . :   PORT
Switch setting . . . . . . . . . . :   SWTSET
Local location address . . . . . . :   LOCADR       02
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Attached controller  . . . . . . . :   CTL          RT3270TRC
Keyboard language type . . . . . . :   KBDTYPE      USI
Drop line at signoff . . . . . . . :   DROP         *NO
Character identifier . . . . . . . :   CHRID
Allow blinking cursor  . . . . . . :   ALWBLN
Auxiliary devices  . . . . . . . . :   AUXDEV
```

```
                    Display Device Description - Display

Printer  . . . . . . . . . . . . . :   PRINTER
Print file . . . . . . . . . . . . :   PRTFILE      QSYSPRT
  Library  . . . . . . . . . . . . :                  *LIBL
Maximum length of request unit . . :   MAXLENRU     *CALC
Text . . . . . . . . . . . . . . . :   TEXT         3277 Display for
                                                    3270 Emulation
```

```
                    Display Device Description - Display

Device description . . . . . . . . :   DEVD         RT3270TRD2
Device class . . . . . . . . . . . :   DEVCLS       *RMT
Device type  . . . . . . . . . . . :   TYPE         3277
Device model . . . . . . . . . . . :   MODEL        0
Port number  . . . . . . . . . . . :   PORT
Switch setting . . . . . . . . . . :   SWTSET
Local location address . . . . . . :   LOCADR       03
Online at IPL  . . . . . . . . . . :   ONLINE       *NO
Attached controller  . . . . . . . :   CTL          RT3270TRC
Keyboard language type . . . . . . :   KBDTYPE      USI
Drop line at signoff . . . . . . . :   DROP         *NO
Character identifier . . . . . . . :   CHRID
Allow blinking cursor  . . . . . . :   ALWBLN
Auxiliary devices  . . . . . . . . :   AUXDEV
```

```
                    Display Device Description - Display

Printer  . . . . . . . . . . . . . :   PRINTER
Print file . . . . . . . . . . . . :   PRTFILE      QSYSPRT
  Library  . . . . . . . . . . . . :                  *LIBL
Maximum length of request unit . . :   MAXLENRU     *CALC
Text . . . . . . . . . . . . . . . :   TEXT         3277 Display for
                                                    3270 Emulation
```

Figure 213. Network 3270-PLUS: IBM AS/400 Token-Ring Display Device Description

```
                    Display Device Description - Printer

Device description . . . . . . . . :   DEVD          RT3270TRD3
Device class . . . . . . . . . . . :   DEVCLS        *RMT
Device type  . . . . . . . . . . . :   TYPE          3287
Device model . . . . . . . . . . . :   MODEL         0
Port number  . . . . . . . . . . . :   PORT
Switch setting . . . . . . . . . . :   SWTSET
Local location address . . . . . . :   LOCADR        04
Online at IPL  . . . . . . . . . . :   ONLINE        *NO
Attached controller  . . . . . . . :   CTL           RT3270TRC
Font identifier  . . . . . . . . . :   FONT
Form feed  . . . . . . . . . . . . :   FORMFEED      *CONT
Printer error message  . . . . . . :   PRTERRMSG     *INQ
Message queue  . . . . . . . . . . :   MSGQ          QSYSOPR
  Library  . . . . . . . . . . . . :                   *LIBL
Max length of request unit . . . . :   MAXLENRU      *CALC
Text . . . . . . . . . . . . . . . :   TEXT          3287 Printer for
                                                     3270 Emulation
```

```
                    Display Device Description - Printer

Device description . . . . . . . . :   DEVD          RT3270TRD4
Device class . . . . . . . . . . . :   DEVCLS        *RMT
Device type  . . . . . . . . . . . :   TYPE          3287
Device model . . . . . . . . . . . :   MODEL         0
Port number  . . . . . . . . . . . :   PORT
Switch setting . . . . . . . . . . :   SWTSET
Local location address . . . . . . :   LOCADR        05
Online at IPL  . . . . . . . . . . :   ONLINE        *NO
Attached controller  . . . . . . . :   CTL           RT3270TRC
Font identifier  . . . . . . . . . :   FONT
Form feed  . . . . . . . . . . . . :   FORMFEED      *CONT
Printer error message  . . . . . . :   PRTERRMSG     *INQ
Message queue  . . . . . . . . . . :   MSGQ          QSYSOPR
  Library  . . . . . . . . . . . . :                   *LIBL
Max length of request unit . . . . :   MAXLENRU      *CALC
Text . . . . . . . . . . . . . . . :   TEXT          3287 Printer for
                                                     3270 Emulation
```

Figure 214. Network 3270-PLUS: IBM AS/400 Token-Ring Printer Device Description

```
5728PW1 R01M02  881028                SEU SOURCE LISTING
SOURCE FILE . . . . . . .  STELLA/QCLSRC
MEMBER  . . . . . . . . .  RT3270TR
PGM

/**********************************************************************/
/* THIS PROGRAM CREATES THE LINE, CONTROLLER AND DEVICE DESCRIPTIONS */
/* WHICH CAN BE USED TO CONNECT THE AS/400 VIA A TOKEN RING TO AN    */
/* RT EMULATING 3270 DEVICES.                                         */
/* ANY CHANGES THAT NEED TO BE MADE TO THE TOKEN RING, CONTROLLER OR */
/* DEVICE DESCRIPTIONS CAN BE MADE IN THIS PROGRAM. WHEN RUN, THE    */
/* EXISTING TR, WITH ITS ATTACHED CONTROLLER AND DEVICES, WILL BE    */
/* VARIED OFF AND DELETED (IF THEY EXIST) THEN CREATED.              */
/*                                                                    */
/* DEVICES CREATED - 2X3277, 2X3287                      */
/**********************************************************************/
             VRYCFG     CFGOBJ(RT3270TRD1 RT3270TRD2 RT3270TRD3 +
                          RT3270TRD4
                          CFGTYPE(*DEV) STATUS(*OFF)
             MONMSG     MSGID(CPF9999)

             VRYCFG     CFGOBJ(RT3270TRC) CFGTYPE(*CTL) STATUS(*OFF)
             MONMSG     MSGID(CPF9999)

             VRYCFG     CFGOBJ(RT3270TRL) CFGTYPE(*LIN) STATUS(*OFF)
             MONMSG     MSGID(CPF9999)

             DLTDEVD    DEVD(RT3270TRD*)
             MONMSG     MSGID(CPF9999)

             DLTCTLD    CTLD(RT3270TRC)
             MONMSG     MSGID(CPF9999)

             DLTLIND    LIND(RT3270TRL)
             MONMSG     MSGID(CPF9999)

             CRTLINTRN  LIND(RT3270TRL) RSRCNAME(LIN031) ONLINE(*NO) +
                          SSAP((04)) TEXT('AS400 to RT Token Ring for +
                          3270 Emulation')

             CRTCTLRWS  CTLD(RT3270TRC) TYPE(3174) MODEL(0) +
                          LINKTYPE(*TRLAN) ONLINE(*NO) +
                          SWTLINLST(RT3270TRL) ADPTADR(50005A1A0AE9) +
                          TEXT('AS400 to RT Token Ring for 3270 +
                          Emulation')
             CRTDEVDSP  DEVD(RT3270TRD1) DEVCLS(*RMT) TYPE(3277) +
                          MODEL(0) LOCADR(02) ONLINE(*NO) +
                          CTL(RT3270TRC) KBDTYPE(USI) DROP(*NO) +
                          TEXT('3277 Display for 3270 Emulation')
             CRTDEVDSP  DEVD(RT3270TRD2) DEVCLS(*RMT) TYPE(3277) +
                          MODEL(0) LOCADR(03) ONLINE(*NO) +
                          CTL(RT3270TRC) KBDTYPE(USI) DROP(*NO) +
                          TEXT('3277 Display for 3270 Emulation')
             CRTDEVPRT  DEVD(RT3270TRD3) DEVCLS(*RMT) TYPE(3287) +
                          MODEL(0) LOCADR(04) ONLINE(*NO) +
                          CTL(RT3270TRC) TEXT('3287 Printer for 3270 +
                          Emulation')
             CRTDEVPRT  DEVD(RT3270TRD4) DEVCLS(*RMT) TYPE(3287) +
                          MODEL(0) LOCADR(05) ONLINE(*NO) +
                          CTL(RT3270TRC) TEXT('3287 Printer for 3270 +
                          Emulation')
ENDPGM
```

Figure 215. Network 3270-PLUS: IBM AS/400 CL Program to Create Token-Ring Descriptions

SNA Services Profiles

Valid profiles to connect to this host are shown in Table 51 through Table 56.

Table 51. Network 3270-PLUS: Connection Profiles for Token-Ring to IBM AS/400

Connection Profile Name	LU2	LU3	LU4	LU5
Attachment Profile Name	TRAS4A	TRAS4A	TRAS4A	TRAS4A
Local LU Profile Name	LLU2	LLU3	LLU4	LLU5
Network Name				
Remote LU Name				
Stop Connection on Inactivity	NO	NO	NO	NO
LU Type	LU2	LU2	LU1	LU3
Notify	YES	YES	YES	YES

Table 52. Network 3270-PLUS: Local LU Profiles for Token-Ring to IBM AS/400

Local LU Profile Name	LLU2	LLU3	LLU4	LLU5
LU Type	LU2	LU2	LU1	LU3
Network Name				
Local LU Name	ASLU2	ASLU3	ASLU4	ASLU5
Number of Rows	24	24	24	24
Number of Columns	80	80	80	132
Local LU Address	2	3	4	5
SSCP ID	05000000000F	05000000000F	05000000000F	05000000000F

Table 53. Network 3270-PLUS: Attachment Profile for Token-Ring to IBM AS/400

Attachment Profile Name	TRAS4A
Control Point Profile Name	TRAS4C
Logical Link Profile Name	TRAS4L
Physical Link Profile Name	TRAS4P
Logical Link Type	TOKEN_RING
Call Type	CALL
Access Routing	LINK_ADDRESS
Remote Link Address	40005A040483
Remote SAP Address	04
Stop Attachment on Inactivity	NO

Table 54. Network 3270-PLUS: Control Point Profile for Token-Ring to IBM AS/400

Control Point Profile Name	TRAS4C
XID Node Id	00000000

Table 55. Network 3270-PLUS: Logical Link Profile for Token-Ring to IBM AS/400

Logical Link Profile Name	**TRAS4L**
Transmit Window Count	10
Dynamic Window Increment	1
Retransmit Count	8
Receive Window Count	127
Ring Access Priority	0
Inactivity Timeout	120
Drop Link on Inactivity	YES
Response Timeout	2
Acknowledgement Timeout	1
Force Disconnect Timeout	120
Maximum I-Field Size	SYSTEM_DEFINED
Link Trace	TRACE_OFF

Table 56. Network 3270-PLUS: Physical Link Profile for Token-Ring to IBM AS/400

Physical Link Profile Name	**TRAS4P**
Data Link Device Name	trllc0
Local Link Name	
Local SAP Address	04
Maximum Number of Logical Links	32

Note: The IBM RT SNA Services profiles and the IBM AS/400 profiles ***must*** match for a connection to be established:

1. The Remote Link Address in the Attachment Profile (`Remote Link Station Address=40005A040483`) is the address of the Token-Ring adapter of the IBM AS/400.

2. The SSCP identification is set to 050000000015 in the IBM AS/400 controller description (`SSCPID=050000000015`). The IBM RT SNA Services Local LU Profiles must have the corresponding hexadecimal value (`SSCPID=05000000000F`).

3. The XID parameter is not used.

4. The local station addresses for the terminals and printers in the IBM AS/400 device descriptions have the numbers 02 through 05 (`LOCADR=02, ..., LOCADR=05`). These parameter values must match those in the IBM RT Local LU Profiles (`Local LU Address=2, ..., Local LU Address=5`) and those in the Network 3270-PLUS Configuration Profile (`:Terminal(2, On, 3277), :Terminal(3, On, 3277), :Printer(4, On, 3287 ...)` and `:Printer(5, On, 3287 ...)`).

5. The type of terminal and printer defined in the IBM AS/400 device description (`TYPE=3277, ..., TYPE=3287`) must match the LU type in the SNA Services Connection Profile and Local LU Profile on the IBM RT (`LU Type=LU2, ..., LU Type=LU1` and `LU Type=LU3`) and must also match the entries in the Network 3270-PLUS Configuration Profile (`:Terminal(2, On, 3277), ..., :Printer(5, On, 3287, ...)`). Do ***not*** confuse the Connection Profile names (LU2, LU3, LU4, and LU5) with the LU type (LU1 is the LU type for a non SNA printer, LU2 is the LU type for a terminal, and LU3 is the LU type for an SNA printer).

6. In the IBM AS/400 device descriptions, you can only specify a device type 3277 for a terminal and 3287 for a printer. The IBM AS/400 never sends a data stream that includes extended attributes such as underlining, reverse image and color. The device types 3278 with extended attributes and 3279 with extended attributes, including color, could be specified, but will not work with the Network 3270-PLUS emulation. A 3287 printer device specified in the IBM AS/400 device description can support SNA character string (SCS) commands and orders. The LU Type 3 specified in the IBM RT Connection- and Local LU Profiles is also capable of SCS. For a complete description of the 3270 devices emulated by the IBM AS/400, see *IBM AS/400 3270 Device Emulation Users Guide*.

7. There is no file transfer program like the System/370 *IND$FILE* supported on the IBM AS/400 so you can't use file transfer commands with the IBM AS/400. The 3270-PLUS Programmer Interface Modules (PIM) allow application programs on the host to directly interface with the Network 3270-PLUS program. You can use the functions of PIM for a file transfer. An example of how to do this with the IBM AS/400 in IBM S/36 mode is shown in Appendix E, "Network 3270-PLUS File Transfer Programs" on page 483.

Network 3270-PLUS Profile

You need only change the Network 3270-PLUS main configuration profile in /usr/lpp/r/comm/sna/profiles/sc3270.prof for the addresses 2, 3, 4, and 5 to be configured as shown in Figure 216.

```
#
# @(#)sc3270.prof 1.3
#

System Configuration Profile for Rabbit SNA 3270-Plus

:Protocol(SNA)
:ControllerType(3274)

:PH Parms("/usr/lpp/r/comm/sna/profiles/SNAParms")

:DLC Parms("")

:DH Parms(32, 32, ON, 100)

:Logging(Dev Data = OFF, PH Data = OFF, DH Errors = OFF, DH Timing = OFF,
      PSErrors = OFF, PSData = OFF, PSTiming = OFF,
         PEErrors = OFF, PEData = OFF, PETiming = OFF )

     T E R M I N A L     D E F I N I T I O N S

:Terminal(DEFAULT, On, 3278, , 1920, 24, 80, 400,
"/usr/lpp/r/comm/sna/3270/profiles/TermOpts.prof",
"/usr/lpp/r/comm/sna/3270/profiles/KbdMap",
Validation Mode = 1)

:Terminal(2, On, 3277)

:Terminal(3, On, 3277)

     P R I N T E R     D E F I N I T I O N S

:PrinterParms(TIMEOUT = 15, SPOOLSIZE = 3)

:Printer(4, ON, 3287, 1, 1920,
"/usr/lpp/r/comm/sna/3270/profiles/PrtOpts.prof")

:Printer(5, ON, 3287, 2, 1920,
"/usr/lpp/r/comm/sna/3270/profiles/PrtOpts.prof")

:End;
```

Figure 216. Network 3270-PLUS: SNA Configuration Profile for two 3277 and two 3287

The terminals are defined as IBM 3277 terminals to match the IBM AS/400 display device descriptions in Figure 213 on page 461. The printers are defined as IBM 3287 printers to match the IBM AS/400 printer device descriptions in Figure 214 on page 462.

Using the IBM RT Console and ASCII Keyboards with the IBM AS/400

When you are logged on from the IBM RT to the IBM AS/400 using Network 3270-PLUS, you have to use the keyboard of the IBM RT console or an ASCII terminal to work with the IBM AS/400. To do this, you go through two terminal emulations: First through the 3270 emulation provided by Network 3270-PLUS, then through the 5250 emulation provided by the IBM AS/400. Network 3270-PLUS provides the `ibm5151.s3270` VTS profile for the IBM RT console and the `ibm3161.s3270` VTS profile for ASCII terminals to make them look like an IBM 3270 terminal. Both profiles can be found in the `/usr/lpp/r/comm/sna/3270/profiles` directory. Table 57 on page 469 shows how the functions of an IBM 5250 keyboard are mapped to the keyboards of an IBM RT console and an ASCII terminal.

Some 5250 functions are implemented as a key combination. We are using the + (plus) character to show when you must use the two keys of a key combination simultaneously. For example: `Alt+F1` means:

Press the `Alt` key and hold it down while pressing the `F1` key; then release both keys.

We are using the > (greater than) character to show when you have to use the two keys of a key combination one after the other. For example: `Alt+F1>F1` means:

Press the `Alt` key and hold it down while pressing the `F1` key; then release both keys and press the `F1` key.

Table 57. Network 3270-PLUS: Mapping of 5250 Functions for use on IBM RT

5250 keys	**5250 keys emulated on the 3270 keyboard**	**5250 keys emulated on the Console Keyboard**	**5250 keys emulated on the ASCII Terminal Keyboard**
F1	PA1 > PF1	Alt+F1 > F1	Ctrl > Pad1 > F1
F2	PA1 > PF2	Alt+F1 > F2	Ctrl > Pad1 > F2
F3	PA1 > PF3	Alt+F1 > F3	Ctrl > Pad1 > F3
F4	PA1 > PF4	Alt+F1 > F4	Ctrl > Pad1 > F4
F5	PA1 > PF5	Alt+F1 > F5	Ctrl > Pad1 > F5
F6	PA1 > PF6	Alt+F1 > F6	Ctrl > Pad1 > F6
F7	PA1 > PF7	Alt+F1 > F7	Ctrl > Pad1 > F7
F8	PA1 > PF8	Alt+F1 > F8	Ctrl > Pad1 > F8
F9	PA1 > PF9	Alt+F1 > F9	Ctrl > Pad1 > F9
F10	PA1 > PF10	Alt+F1 > F10	Ctrl > Pad1 > F10
F11	PA1 > PF11	Alt+F1 > F11	Ctrl > Pad1 > F11
F12	PA1 > PF12	Alt+F1 > F12	Ctrl > Pad1 > F12
F13	PA2 > PF1	Alt+F2 > F1	Ctrl > Pad2 > F1
F14	PA2 > PF2	Alt+F2 > F2	Ctrl > Pad2 > F2
F15	PA2 > PF3	Alt+F2 > F3	Ctrl > Pad2 > F3
F16	PA2 > PF4	Alt+F2 > F4	Ctrl > Pad2 > F4
F17	PA2 > PF5	Alt+F2 > F5	Ctrl > Pad2 > F5
F18	PA2 > PF6	Alt+F2 > F6	Ctrl > Pad2 > F6
F19	PA2 > PF7	Alt+F2 > F7	Ctrl > Pad2 > F7
F20	PA2 > PF8	Alt+F2 > F8	Ctrl > Pad2 > F8
F21	PA2 > PF9	Alt+F2 > F9	Ctrl > Pad2 > F9
F22	PA2 > PF10	Alt+F2 > F10	Ctrl > Pad2 > F10
F23	PA2 > PF11	Alt+F2 > F11	Ctrl > Pad2 > F11
F24	PA2 > PF12	Alt+F2 > F12	Ctrl > Pad2 > F12
Enter	Enter	Return	Return
Help	PF1	F1	F1
3270 Help	PF2	F2	F2
Clear	PF3	F3	F3
Print	PF4	F4	F4
Display embedded attributes	PF5	F5	F5
Test Request	PF6	F6	F6
Roll Down	PF7	F7	F7
Roll Up	PF8	F8	F8
Error Reset	PF10	F10	F10
Sys Req	PF11	F11	F11
Record Backspace	PF12	F12	F12

You can also use key combinations to invoke the local functions of the Network 3270-PLUS emulation. Table 58 shows the mapping.

Table 58. Network 3270-PLUS: Mapping of Special Functions for use on IBM RT

Network 3270-PLUS special Function Keys	**Network 3270-PLUS special Function Keys on the Console**	**Network 3270-PLUS special Function Keys on the ASCII Terminals**
Banner	Alt+F11	Esc>b Esc>B
Cancel	Pause	Esc>Delete Ctrl+Home (Del)
Cursor Sel (toggle)	Alt+F8	Esc>z Esc>Z
Exit	Alt+F10	Esc>x Esc>X
Finish/Suspend	End	Esc>f Esc>F
File Transfer	Alt+t	Esc>t Esc>T
Help	Alt+h	Esc>h
Next Page	Page Down	Esc>n Esc>N
Prev Page	Page Up	Esc>p
Redraw	Alt+F9	Esc>l Esc>L
Screen-to-File	Alt+f	Esc>Shift+]
Screen-to-Printer	PrintScreen	Esc>]
On-line profile change	Alt+F12	Esc>o Esc>O

Note: You cannot transfer files to the IBM AS/400 because there is no *IND$FILE* file transfer program on the IBM AS/400.

Using an IBM RT as Gateway Between a DEC VAX and an IBM AS/400

On a DEC VAX running VMS and with appropriate TCP/IP software installed (for example the Wollongong TCP/IP Software for VMS) you can log in to an IBM RT using *telnet* and then start a 3270 emulation to an IBM AS/400 using Network 3270-PLUS. Both the VAX and the IBM RT must be connected to an Ethernet and have TCP/IP properly configured. For the 3270 connection from the IBM RT to the IBM AS/400, you can either use an SDLC connection as described in "SDLC SNA Connection to IBM AS/400 Host" on page 450 or a Token-Ring connection as described in "Token-Ring Connection to SNA IBM AS/400 Host" on page 459.

Logging In to the IBM AS/400 from the VAX

You can use a VT100, VT220 or VT320 terminal on the VAX to log in to the IBM RT with *telnet*. The terminal should emulate a VT100 terminal because the Network 3270-PLUS VTS profile that maps the keyboard of the VT terminal to an IBM 3270 terminal keyboard uses the `Esc` key in combination with other keys to emulate many of the 3270 keys. The `Esc` key exists on a VT100 terminal; not on the VT220 and VT320 terminals, but you can use the "general setup mode" of the terminals to make the VT220 and VT230 terminals run in VT100 mode. The function of the `Esc` key is then available on the `F11` of the VT220 or VT320 keyboard. To log in you:

1. Log in to the VAX.
2. Set the terminal mode of the VT220 or VT320 terminal to VT100 mode using "general setup".
3. Establish a *telnet* connection to the IBM RT.
4. Log in to the IBM RT and set the `TERM` environmant variable to `vt100` if you are working with a VT100 terminal by typing: `TERM=vt100`. Set the `TERM` environment variable to `vt220` if you use a VT220 or VT320 terminal by keying: `TERM=vt220`.
5. Type: `export TERM` to export the new value of `TERM` to the shell.
6. Start a Network 3270-PLUS 3270 session on the IBM RT.

Now you should receive the signon screen of the IBM AS/400 on your terminal.

Using a VT Terminal Keyboard with the IBM AS/400

When you are logged on from the VAX to the IBM AS/400 using Network 3270-PLUS you use the keyboard of the VT terminal to work with the IBM AS/400. You are actually going through two terminal emulations: First you use the 3270 emulation provided by Network 3270-PLUS, then you use the 5250 emulation provided by the IBM AS/400. Network 3270-PLUS provides the `vt100.s3270` VTS profile for a VT100 terminal and the `vt220.s3270` VTS profile for the VT220 and VT320 terminals to make them appear like a 3270 terminal. Both profiles can be found in the `/usr/lpp/r/comm/sna/3270/profiles` directory. The mapping done by these tables is shown in Table 59 on page 472.

When a key combination is used to emulate IBM 5250 functions, the table uses the > (greater than) character to illustrate that the two keys of a key combination must be used one after the other. For example: `PF1>Esc>1` means:

Press the PF1 key and release it, then press the Esc key and release it, then press the 1 key.

We are using the + (plus) character when you have to use the two keys of a key combination simultaneously. For example: PF2>Esc>Shift+1 means:

Press the PF2 key and release it, then press the Esc key and release it, then press the Shift key and hold it down while pressing the 1 key.

Table 59. Mapping of the 5250 Functions for Use on a VT terminal

5250 keys	**5250 keys emulated on the 3270 keyboard**	**5250 keys emulated on the VT Keyboard**
F1	PA1>PF1	PF1>Esc>1
F2	PA1>PF2	PF1>Esc>2
F3	PA1>PF3	PF1>Esc>3
F4	PA1>PF4	PF1>Esc>4
F5	PA1>PF5	PF1>Esc>5
F6	PA1>PF6	PF1>Esc>6
F7	PA1>PF7	PF1>Esc>7
F8	PA1>PF8	PF1>Esc>8
F9	PA1>PF9	PF1>Esc>9
F10	PA1>PF10	PF1>Esc>0
F11	PA1>PF11	PF1>Esc>-
F12	PA1>PF12	PF1>Esc>=
F13	PA2>PF1	PF2>Esc>Shift+1
F14	PA2>PF2	PF2>Esc>Shift+2
F15	PA2>PF3	PF2>Esc>Shift+3
F16	PA2>PF4	PF2>Esc>Shift+4
F17	PA2>PF5	PF2>Esc>Shift+5
F18	PA2>PF6	PF2>Esc>Shift+6
F19	PA2>PF7	PF2>Esc>Shift+7
F20	PA2>PF8	PF2>Esc>Shift+8
F21	PA2>PF9	PF2>Esc>Shift+9
F22	PA2>PF10	PF2>Esc>Shift+0
F23	PA2>PF11	PF2>Esc>Shift+-
F24	PA2>PF12	PF2>Esc>Shift+=
Enter	Enter	Return
Help	PF1	Esc>1
3270 Help	PF2	Esc>2
Clear	PF3	Esc>3
Print	PF4	Esc>4
Display embedded attributes	PF5	Esc>5
Test Request	PF6	Esc>6
Roll Down	PF7	Esc>7
Roll Up	PF8	Esc>8
Error Reset	PF10	Esc>0
Sys Req	PF11	Esc>-
Record Backspace	PF12	Esc>=

The special functions of Network 3270-PLUS are invoked with the key combinations shown in Table 60.

Table 60. Mapping of the special Functions of Network 3270-PLUS for Use on a VT Keyboard

Network 3270-PLUS special Function Keys	**Network 3270-PLUS special Function Keys on a VT Keyboard**
Banner	Esc>b Esc>B
Cancel	Esc>Delete
Cursor Sel (toggle)	Esc>z Esc>Z
Exit	Esc>x Esc>X
Finish/Suspend	Esc>f Esc>F
File Transfer	Esc>t Esc>T
Help	Esc>h Esc>H
Next Page	Esc>n Esc>N
Prev Page	Esc>p Esc>P
Redraw	Esc>l Esc>L
Screen-to-File	Esc>Shift+]
Screen-to-Printer	Esc>]
On-line profile change	Esc>o

Note: The IBM AS/400 does not have the *IND$FILE* file transfer program so files can not be transferred with Esc>t or Esc>T.

X.25 Profiles for 3270 Network-PLUS (SNA)

Profiles for Network 3270-PLUS and Network RJE-PLUS (SNA)

These profiles were used for SNA communication over X.25 between an IBM RT running Network 3270-PLUS (SNA) and Network RJE-PLUS (SNA) on one side and an IBM 37X5 communication controller running NPSI and attached to an IBM System/370. The VTAM Listing is shown in "Sample VTAM Listing" on page 480.

SNA Services Profiles

```
LU4_CONNECTION:
        type = CONNECTION
        profile_name = LU4
        attachment_name = qnpsia
        local_lu_name = LLU4
        network_name =
        remote_lu_name =
        stop_connection_on_inactivity = NO
        lu_type = LU2
        interface_type = EXTENDED
        remote_tpn_list_name =
        mode_list_name =
        node_verification = NO
        inactivity_timeout_value = 0
        notify = YES
        cp_sessions = NO
        parallel_sessions = SINGLE
        negotiate_session_limits = NO

LU5_CONNECTION:
        type = CONNECTION
        profile_name = LU5
        attachment_name = qnpsia
        local_lu_name = LLU5
        network_name =
        remote_lu_name =
        stop_connection_on_inactivity = NO
        lu_type = LU2
        interface_type = EXTENDED
        remote_tpn_list_name =
        mode_list_name =
        node_verification = NO
        inactivity_timeout_value = 0
        notify = YES
        cp_sessions = NO
        parallel_sessions = SINGLE
        negotiate_session_limits = NO
```

Figure 217. Network PLUS: X.25. Connection Profiles (1 of 3)

```
LU6_CONNECTION:
        type = CONNECTION
        profile_name = LU6
        attachment_name = qnpsia
        local_lu_name = LLU6
        network_name =
        remote_lu_name =
        stop_connection_on_inactivity = NO
        lu_type = LU2
        interface_type = EXTENDED
        remote_tpn_list_name =
        mode_list_name =
        node_verification = NO
        inactivity_timeout_value = 0
        notify = YES
        cp_sessions = NO
        parallel_sessions = SINGLE
        negotiate_session_limits = NO

LU7_CONNECTION:
        type = CONNECTION
        profile_name = LU7
        attachment_name = qnpsia
        local_lu_name = LLU7
        network_name =
        remote_lu_name =
        stop_connection_on_inactivity = NO
        lu_type = LU2
        interface_type = EXTENDED
        remote_tpn_list_name =
        mode_list_name =
        node_verification = NO
        inactivity_timeout_value = 0
        notify = YES
        cp_sessions = NO
        parallel_sessions = SINGLE
        negotiate_session_limits = NO

LU8_CONNECTION:
        type = CONNECTION
        profile_name = LU8
        attachment_name = qnpsia
        local_lu_name = LLU8
        network_name =
        remote_lu_name =
        stop_connection_on_inactivity = NO
        lu_type = LU3
        interface_type = EXTENDED
        remote_tpn_list_name =
        mode_list_name =
        node_verification = NO
        inactivity_timeout_value = 0
        notify = YES
        cp_sessions = NO
        parallel_sessions = SINGLE
        negotiate_session_limits = NO
```

Figure 218. Network PLUS: X.25. Connection Profiles (2 of 3)

```
qrjeLU3_CONNECTION:
        type = CONNECTION
        profile_name = qrjeLU3
        attachment_name = rjeattq
        local_lu_name = qrjeLU3
        network_name =
        remote_lu_name =
        stop_connection_on_inactivity = NO
        lu_type = LU1
        interface_type = EXTENDED
        remote_tpn_list_name =
        mode_list_name =
        node_verification = NO
        inactivity_timeout_value = 0
        notify = NO
        cp_sessions = NO
        parallel_sessions = SINGLE
        negotiate_session_limits = NO
```

Figure 219. Network PLUS: X.25. Connection Profiles (3 of 3)

```
LLU4_LOCALLU:
        type = LOCALLU
        profile_name = LLU4
        local_lu_name = IGJTNP13
        network_name =
        lu_type = LU2
        independent_lu = NO
        cp_sessions = NO
        tpn_list_name =
        local_lu_address = 4
        sscp_id = 050000001234
        number_of_rows = 24
        number_of_columns = 80
        destination_address = 0

LLU5_LOCALLU:
        type = LOCALLU
        profile_name = LLU5
        local_lu_name = IGJTNP14
        network_name =
        lu_type = LU2
        independent_lu = NO
        cp_sessions = NO
        tpn_list_name =
        local_lu_address = 5
        sscp_id = 050000001234
        number_of_rows = 24
        number_of_columns = 80
        destination_address = 0

LLU6_LOCALLU:
        type = LOCALLU
        profile_name = LLU6
        local_lu_name = IGJTNP15
        network_name =
        lu_type = LU2
        independent_lu = NO
        cp_sessions = NO
        tpn_list_name =
        local_lu_address = 6
        sscp_id = 050000001234
        number_of_rows = 24
        number_of_columns = 80
        destination_address = 0
```

Figure 220. Network PLUS: X.25. Local LU Profiles (1 of 2)

```
LLU7_LOCALLU:
        type = LOCALLU
        profile_name = LLU7
        local_lu_name = IGJTNP16
        network_name =
        lu_type = LU2
        independent_lu = NO
        cp_sessions = NO
        tpn_list_name =
        local_lu_address = 7
        sscp_id = 050000001234
        number_of_rows = 24
        number_of_columns = 80
        destination_address = 0

LLU8_LOCALLU:
        type = LOCALLU
        profile_name = LLU8
        local_lu_name = IGJTNP17
        network_name =
        lu_type = LU3
        independent_lu = NO
        cp_sessions = NO
        tpn_list_name =
        local_lu_address = 8
        sscp_id = 050000001234
        number_of_rows = 24
        number_of_columns = 80
        destination_address = 0

qrjeLU3_LOCALLU:
        type = LOCALLU
        profile_name = qrjeLU3
        local_lu_name = IGJTNP22
        network_name =
        lu_type = LU1
        independent_lu = NO
        cp_sessions = NO
        tpn_list_name =
        local_lu_address = 3
        sscp_id = 050000001234
        number_of_rows = 1
        number_of_columns = 1
        destination_address = 0
```

Figure 221. Network PLUS: X.25. Local LU Profiles (2 of 2)

```
qnpsicpt_CONTROL POINT:
        type = CONTROLPOINT
        profile_name = qnpsicpt
        xid_node_id = 234B6789
        network_name =
        cp_name =
        end_node_status = NO_BIND
        locate_gds_support = NO
        directory_services_support = NO
        resource_registration_support = NO
        registration_of_characteristics = NO
        topology_database_update_support = NO
        request_reply_cp_msus_support = NO
        unsolicited_CP_MSUs_supported = NO
        parallel_CP_CP_sessions_supported = NO
        flow_reduction_sequence_number = 0
```

Figure 222. Network PLUS: X.25. Control Point Profile

```
qnpsip_X.25 PHYSICAL:
        type = X.25PHYSICAL
        profile_name = qnpsip
        device_name = qllc0
        maximum_link_stations = 20
        local_x25_network_address = 4321987654321
```

Figure 223. Network PLUS: X.25. Physical Data Link Profile

```
qnpsil_QLLC LOGICAL:
        type = QLLCLOGICAL
        profile_name = qnpsil
        station_type = SECONDARY
        negotiable = NO
        drop_link_on_inactivity = NO
        force_disconnect_timeout = 180
        link_trace = TRACE_OFF
        trace_entry_size = SHORT
        secondary_inactivity_timeout = 30
        primary_repoll_timeout = 3
        primary_repoll_count = 10
        maximum_i_field = SYSTEM_DEFINED
        maximum_i_field_size = 1417
```

Figure 224. Network PLUS: X.25. Logical Data Link Profile

```
qnpsia_ATTACHMENT:
        type = ATTACHMENT
        profilename = qnpsia
        control_point_profile_name = qnpsicpt
        logical_link_profile_name = qnpsil
        physical_link_profile_name = qnpsip
        logical_link_type = QLLC
        stop_attachment_on_inactivity = NO
        station_type = SECONDARY
        physical_link_type = X.25
        remote_secondary_station_address = 0
        smart_modem_command_sequence =
        length_of_command_sequence = 0
        call_type = CALL
        autolisten = NO
        timeout_value = 0
        remote_link_name_ethernet =
        remote_link_name_token_ring =
        remote_link_address_token_ring = 000000000000
        selection_sequence =
        length_of_selection_sequence = 0
        network_type = SWITCHED
        access_routing = LINK_NAME
        remote_sap_address = 00
        remote_sap_address_range_lower = 00
        remote_sap_address_range_upper = 00
        virtual_circuit_type = SWITCHED
        remote_station_X.25_address = 1234567891234
        optional_X.25_facilities = NO
        logical_channel_number_of_PVC = 1
        reverse_charging = NO
        rpoa = NO
        default_packet_size = YES
        default_window_size = YES
        default_throughput_class = YES
        closed_user_group = NO
        data_network_identification_code =
        packet_size_for_received_data = 256
        packet_size_for_transmit_data = 256
        window_size_for_received_data = 7
        window_size_for_transmit_data = 7
        throughput_class_for_received_data = 9600
        throughput_class_for_transmit_data = 9600
        index_to_selected_closed_user_group =
```

Figure 225. Network PLUS: X.25. Attachment Profile

Sample VTAM Listing

```
*
*   SWITCHED MAJOR NODE FOR X25 DEVICES
*
M2X25S1  VBUILD TYPE=SWNET,MAXGRP=4,MAXNO=10
*
***********************************************************************
*       X25 DIAL IN TERMINAL
*       TO BE USED FOR LU2   TESTING
*
IGJCNP01 PU    ADDR=C1,            STATION ADDRESS                     C
               IDBLK=234,          ID FOR PC                           C
               IDNUM=B6789,                                            C
               DISCNT=YES,         HANG-UP ON LU LOGOFF                C
               MAXDATA=265,                                            C
               MAXOUT=7,                                               C
               MAXPATH=1,          NO OF DIAL-OUT PATHS                C
               PUTYPE=2,ISTATUS=ACTIVE
*
*
IGJTNP12 LU    LOCADDR=3,ISTATUS=ACTIVE,                               C
               MODETAB=C32782,     MODE TABLE                          C
               USSTAB=UT2X25,                                          C
               SSCPFM=USSSCS
*
*
IGJTNP13 LU    LOCADDR=4,ISTATUS=ACTIVE,                               C
               MODETAB=C32782,     MODE TABLE                          C
               USSTAB=UT2X25,                                          C
               SSCPFM=USSSCS
*
*
IGJTNP14 LU    LOCADDR=5,ISTATUS=ACTIVE,                               C
               MODETAB=C32782,     MODE TABLE                          C
               USSTAB=UT2X25,                                          C
               SSCPFM=USSSCS
*
*
IGJTNP15 LU    LOCADDR=6,ISTATUS=ACTIVE,                               C
               MODETAB=C32782,     MODE TABLE                          C
               USSTAB=UT2X25,                                          C
               SSCPFM=USSSCS
*
*
*
IGJTNP16 LU    LOCADDR=7,ISTATUS=ACTIVE,                               C
               MODETAB=C32782,     MODE TABLE                          C
               USSTAB=UT2X25,                                          C
               SSCPFM=USSSCS
*
*
IGJTNP17 LU    LOCADDR=8,ISTATUS=ACTIVE,                               C
               MODETAB=C32782,     MODE TABLE                          C
               USSTAB=UT2X25,                                          C
               SSCPFM=USSSCS
***********************************************************************
*       X25 DIAL IN TERMINAL
*       TO BE USED FOR LU1   TESTING
*
*       NOTE IGJTNP22 DEFINED TO JES AS RJE LU
*
*
IGJCNP02 PU    ADDR=C1,            STATION ADDRESS                     C
               IDBLK=234,          ID FOR PC                           C
               IDNUM=A2502,                                            C
               DISCNT=YES,         HANG-UP ON LU LOGOFF                C
               MAXDATA=265,                                            C
               MAXOUT=7,                                               C
               MAXPATH=1,          NO OF DIAL-OUT PATHS                C
               PUTYPE=2,ISTATUS=ACTIVE
*
*
IGJTNP22 LU    LOCADDR=3,ISTATUS=ACTIVE,                               C
               MODETAB=C32782,     MODE TABLE                          C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
IGJTNP23 LU    LOCADDR=4,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
IGJTNP24 LU    LOCADDR=5,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
IGJTNP25 LU    LOCADDR=6,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
*
IGJTNP26 LU    LOCADDR=7,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
IGJTNP27 LU    LOCADDR=8,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
***********************************************************************
*       X25 DIAL IN TERMINAL
*
*       TO BE USED FOR LU6.2 TESTING
*
*
IGJCNP03 PU    ADDR=C1,            STATION ADDRESS                   C
               IDBLK=234,          ID FOR PC                         C
               IDNUM=A2503,                                          C
               DISCNT=YES,         HANG-UP ON LU LOGOFF              C
               MAXDATA=265,                                          C
               MAXOUT=7,                                             C
               MAXPATH=1,          NO OF DIAL-OUT PATHS              C
               PUTYPE=2,ISTATUS=ACTIVE
*
*
IGJTNP32 LU    LOCADDR=3,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
IGJTNP33 LU    LOCADDR=4,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
IGJTNP34 LU    LOCADDR=5,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
IGJTNP35 LU    LOCADDR=6,ISTATUS=ACTIVE,                             C
               MODETAB=C32782,     MODE TABLE                        C
               USSTAB=UT2X25,                                        C
               SSCPFM=USSSCS
*
*
*
IGJTNP36 LU    LOCADDR=7,ISTATUS=ACTIVE,                             C
```

```
               MODETAB=C32782,     MODE TABLE                       C
               USSTAB=UT2X25,                                       C
               SSCPFM=USSSCS
*
*
IGJTNP37 LU    LOCADDR=8,ISTATUS=ACTIVE,                            C
               MODETAB=C32782,     MODE TABLE                       C
               USSTAB=UT2X25,                                       C
               SSCPFM=USSSCS
*
```

Appendix E. Network 3270-PLUS File Transfer Programs

RPG Program PIMRTUP for File Transfer to IBM AS/400

Important

The programs described in this appendix assume the IBM AS/400 to run in IBM S/36 mode. They have not been tested at the ITSC in Austin.

```
H     64                   B                                    EMIA1
FSCREEN  CP  F    1700         WORKSTN
F                                           KNUM      01
F                                           KFMTS  SCRRTPCA
FRTPCFIL O   F1700 170         DISK
E                  FG       10170                                  *
ISCREEN  HM  99
I* FORMAT-SCRRTPCA
I                                    1 170 FL0002
I                                  171 340 FL0003
I                                  341 510 FL0004
I                                  511 680 FL0005
I                                  681 850 FL0006
I                                  8511020 FL0007
I                                 10211190 FL0008
I                                 11911360 FL0009
I                                 13611530 FL0010
I                                 15311700 FL0011
C                   MOVEAFL0002  FG,1
C                   MOVEAFL0003  FG,2
C                   MOVEAFL0004  FG,3
C                   MOVEAFL0005  FG,4
C                   MOVEAFL0006  FG,5
C                   MOVEAFL0007  FG,6
C                   MOVEAFL0008  FG,7
C                   MOVEAFL0009  FG,8
C                   MOVEAFL0010  FG,9
C                   MOVEAFL0011  FG,10
C                   DO   10      X       20
C  N01              MOVELFG,X    TESTFL  3
C          TESTFL   COMP 'EOF'                01
C   01              SETON                  LR
C  N01              MOVE FG,X    FIELD1170
C  N01              EXCPTRTPC
C                   END                           OF DO LOOP       *
OSCREEN  D
O                              K8 'SCRRTPCA'
ORTPCFIL E             RTPC
O                      FIELD1  170
```

RPG Program PIMRTDN File Transfer from IBM AS/400

```
H     64                   B                                    EMIB1
FSCREEN  CP  F    1700         WORKSTN
F                                           KNUM      01
F                                           KFMTS  SCRRTPCB
FRTPCFIL ID  F1700 170         DISK
ISCREEN
IRTPCFIL HM  20
I                                    1 170 FIELD1
C   99              SETON                  LR
C   LR              GOTO FINI
C                   MOVE *BLANKS   FL0002
C                   MOVE *BLANKS   FL0003
C                   MOVE *BLANKS   FL0004
C                   MOVE *BLANKS   FL0005
C                   MOVE *BLANKS   FL0006
C                   MOVE *BLANKS   FL0007
C                   MOVE *BLANKS   FL0008
```

```
C                    MOVE *BLANKS   FL0009
C                    MOVE *BLANKS   FL0010
C                    MOVE *BLANKS   FL0011
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0002170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0003170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0004170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0005170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0006170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0007170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0008170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0009170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0010170
C                    READ RTPCFIL                 99
C   99               SETON                    02
C   99               GOTO ENDE
C                    MOVE FIELD1    FL0011170
C                    GOTO FINI
C          ENDE      TAG
C   02               MOVEL'EOF'     FL0002
C   03               MOVEL'EOF'     FL0003
C   04               MOVEL'EOF'     FL0004
C   05               MOVEL'EOF'     FL0005
C   06               MOVEL'EOF'     FL0006
C   07               MOVEL'EOF'     FL0007
C   08               MOVEL'EOF'     FL0008
C   09               MOVEL'EOF'     FL0009
C   10               MOVEL'EOF'     FL0010
C   11               MOVEL'EOF'     FL0011
C          FINI      TAG
OSCREEN  D        20
O                              K8 'SCRRTPCB'
O                        FL0002  170
O                        FL0003  340
O                        FL0004  510
O                        FL0005  680
O                        FL0006  850
O                        FL0007 1020
O                        FL0008 1190
O                        FL0009 1360
O                        FL0010 1530
O                        FL0011 1700
```

Script File PimLogup.LU# File Transfer to IBM AS/400

```
99 Login for IBM AS/400 connection
21 39 SIGN ON
99 Next Line puts your UserID at cursor position
10 0 PHILIPPE
90 0
1 0
21 184 MAIN MENU
99 Next Line specifies the Command to address the correct Work Library
10 0 SLIB PHILIPPE
90 0
1 0
21 184 MAIN MENU
99 Next Line specifies the name of the Program Procedure
10 0 PIMRTUP
90 0
1 0
99 Next Line opens the file where the data is to be read from
80 1 PimInp.LU
90 0
3
10 0 OFF
90 0
1 0
21 39 SIGN ON
98
```

Script File PimLogdn.LU# for File Transfer from IBM AS/400

```
99 Login to the IBM AS/400 connection
21 39 SIGN ON
99 Next Line uts your UserID at cursor position
10 0 PHILIPPE
90 0
1 0
21 184 MAIN MENU
99 Next Line specifies the Command to address the correct Work Library
10 0 SLIB PHILIPPE
90 0
1 0
21 184 MAIN MENU
99 Next Line specifies the name of the Program Procedure
10 0 PIMRTDN
90 0
1 0
99 Next Line opens the file the data is to be stored to
80 1 PimOut.LU
90 0
3
10 0 OFF
90 0
1 0
21 39 SIGN ON
98
```

PIM Program pimrtup.c File Transfer to IBM AS/400

```
/************************************************************************

@(#)pimrtup.c  1.8

P I M    3 2 7 0  - -   F I L E   T R A N S F E R   P R O G R A M

NAME:        pimrtup.c

SYNOPSIS:    Transfer of an AIX file from the IBM RT to an AS/400
             file using the capabilities of the PIM modules of the
             program &n327. (SNA).

DESCRIPTION: In the first step, this program establishes contact to
             the AS/400 by reading and executing the Logon file
             PimLogup.LU#, where # is the LU number specified in the
             PIM startup command: s3270 pim LU# pimrtup. On the
             AS/400 a procedure is started and an
             RPG program is activated. This program will read data
             from a 3270 screen buffer from the IBM RT.
             In the second step, the AIX file to be transferred
             is read on the IBM RT. Records are moved into a screen and
             once the screen is filled, the RPG program running on
             on the AS/400 reads it. This RPG program writes the
             corresponding 170-byte records in a sequential file.
             When the end of the file is reached a string :q.EOF:eq.
             is written in the next available record entry on the screen.
             The name of the file to be transferred is provided
             in the input file (PimInp.LU#).
             In a third step, control is passed back to the file
             PimLogup.LU#, which terminates activities on the AS/400 by
             performing a sign off.
             This program only enables the transfer of files
             containing characters of the 3270 character set.

HISTORY:     This program is based on the PIM sample program
             described in the manual IBM SC23-0776.

************************************************************************/

#include <stdio.h>      /* Standard I/O definitions.               */
#include <signal.h>     /* signal definitions.                     */
#include <errno.h>      /* errno definitions.                      */
#include "pim3270.h"    /* PIM 3270 definitions.                   */
extern  int     errno;
typedef unsigned char   byte;

#define COLS            80      /* Columns per 3270 screen.        */
#define ROWS            24      /* Lines per 3270 screen.          */
#define BUFF_SIZE       ( ROWS * COLS )   /* Buffer size.          */
#define LINELEN         175     /* Input  read limit size          */
#define FAIL            -1      /* Global flag.                    */
#define SUCCEED         0       /* Global flag.                    */
#define FALSE           0       /* Global flag.                    */
#define TRUE            1       /* Global flag.                    */
#define ATTRIBUTE       0x080   /* Attribute byte indicator.       */
#define attrUNPROT      0x020   /* Unprotected field attribute     */
#define attrMDT         0x001   /* ModifiedDataTag(MDT)attribute   */
#define HI_ASCII        '0e177' /* Highest printable character.    */
#define LO_ASCII        '0e040' /* Lowest printable character.     */
#define attrFT          0x0FC   /* FileTran attribute value        */
#define ftCNTL          0x043   /* FileTran control block indic    */
#define ftDATA          0x041   /* FileTran data block indic       */
#define NUM_RETRIES     60      /* Num of Retries on Buffer read   */
#define LogFilePrefix   "PimLogup.LU"    /* Logon file name pref   */
#define ScrFilePrefix   "PimInp.LU"      /* Input file name pref   */

/* These lines define the routines that PIM invokes each time it
   receives a signal from the device handler.  The PreHandler
   routine is invoked prior to PIM's processing of a device
   handler message. The PostHandler routine is called after PIM
   has completed its processing of a device handler message.       */
```

```
extern int PreHandler();          /* pre-exit routine.                */
extern int PostHandler();         /* post-exit routine.               */

static  char ScreenBuffer [ BUFF_SIZE ] =
   { NULL };                      /* PIM 3270 screen buffer.          */

static  char MyBuffer [ BUFF_SIZE ] =
   { NULL };                      /* My 3270 screen buffer.           */

/* These lines define the PIM session control block and initialize
   the block to run predefined values (above).  A pointer to the
   control block is passed to PIM upon initialization.               */

PIMCtlBlk pimSCB =       /* PIM Session Control Block.               */
{
        PreHandler,      /*  Pre-exit routine for PIMRespond.        */
        PostHandler,     /*  Post-exit routine for PIMRespond.       */
        ScreenBuffer     /*  Pointer to 3270 device buffer.          */
};

static  char AidError [  ] = "pimrtup: Error -%d- post AID key, LU%d\n";
static  char EventError [ ] = "pimrtup: Error -%d- post SysReq event, LU%d\n";
static  int     SessionID;      /* PIM session descriptor.           */
static  int     LU_Address;     /* 3270 logical unit address.        */
static  int     fCatch = TRUE;  /* PIM will catch signals.           */
static  int     CloseDown = FALSE;  /* Shutdown flag.                */
static  int     AlarmWakeup = FALSE;/* Signal was Alarm              */
static  FILE    *scriptp;       /* Pointer to current file           */
static  FILE    *mainscriptp;   /* MAIN Input file pointer           */
static  FILE    *altscriptp;    /* AIX Input file pointer            */
static  char    InputData [LINELEN];
static  char    text[LINELEN];            /* input data text         */
static  char    altfn[LINELEN];           /* Input data file name    */
static  char    LuSuffix[3];              /* Logon file name suffix  */
static  int     TextLgth;                 /* Length of input text    */
static  int     Count = 0;    /* Count of the input lines read       */
static  int   Enrnb = 0;      /* Number of transferred records       */
static  int   Silent = TRUE;  /* If TRUE no printf's only fprintf's  */
static  int   CursorPosn;     /* Cursor position in MyBuffer buffer  */
static  int   GetRequest;     /* Get request for screen buffer       */
static  int   PutRequest;     /* Put request to screen buffer        */
static  int   Reps;           /* Number of AIX file opens to execute */
static  int   SuppressLog;    /*Switch suppress logging ON for ALT file*/
static  int   altope = 0;     /* switch for current dataset          */

/***********************************************************************/
/*  ALRMCATCHER :       SIGNAL CATCHERS                                */
/*                                                                     */
/***********************************************************************/
static void AlrmCatcher(sig)
int sig;
{
    AlarmWakeup = TRUE;
    (void)signal(sig, AlrmCatcher);     /* set it back */
}

static void FastShutDown(sig)
int sig;
{
    extern void ShutDown();

    (void)signal(sig, FastShutDown);     /* set it back */
    printf("\npimrtup: fast shutdown\n");
    Close3270(SessionID);
    exit(0);
}

static void CloseItDown(sig)
int sig;
{
    (void)signal(sig, FastShutDown);        /* set it to FastShutDown */
    printf("\npimrtup: Closedown signal\n");
    CloseDown = TRUE;
}
```

```
/*************************************************************************/
/*  SHUTDOWN :                    SHUTDOWN                               */
/*                                                                       */
/*                       Shuts down the program.                         */
/*************************************************************************/
static  void    ShutDown ( )  /*  Close PIM test program, log errors    */
{
    if ( Close3270 ( SessionID ) == FAIL )
    {
       fprintf ( stderr, "pimrtup: Error -%d- closing LU#%d.\n",
       errno, LU_Address );
       fflush ( stderr );
       exit(0);
    }

/* Important: the PIM application should always wait for an
   acknowledgement of a close prior to exiting the program. */
    while (pimSCB.psState != stCLOSED)  pause ();
    fprintf(stderr,"\npimrtup: session terminated\n");
    exit(0);
}                                           /* end of ShutDown */

/*************************************************************************/
/*                                                                       */
/*                         U T I L I T I E S                             */
/*                                                                       */
/*************************************************************************/

/*************************************************************************/
/*  ClearScreen :       CLEAR SCREEN                                     */
/*                                                                       */
/*                Sets the 3270 buffer to NULL.                          */
/*************************************************************************/
static  void    ClearScreen ( )
{
byte    *dataptr = (byte *) MyBuffer;
byte    *endptr  = (byte *) (MyBuffer + BUFF_SIZE);

    while (dataptr < endptr)  *dataptr++ = 0;
}                                                /* end of ClearScreen */

/*************************************************************************/
/*  Decrement :               DECREMENT                                  */
/*                                                                       */
/*          Decrement function that wraps around size of screen.         */
/*************************************************************************/
static  int      Decrement(address)
int *address;

{
    if (*address == 0) *address = BUFF_SIZE - 1;    /* Wrap              */
    else (*address)--;                              /* Decrement         */
    return(*address);
}                                     /* end of Decrement */

/*************************************************************************/
/*  Increment :               INCREMENT                                  */
/*                                                                       */
/*    Increment function that wraps around size of screen.               */
/*************************************************************************/
static  int   Increment(address)
int *address;

{
    (*address)++;                          /* Increment Address     */
    if (*address >= BUFF_SIZE) *address = 0; /* check if wrapped      */
    return(*address);
}                                     /* end of Increment     */

/*************************************************************************/
/*  FindNxtUnprot :     FINDNXTUNPROT                                    */
/*                                                                       */
/*      Find next Unprotected Attribute from current field               */
/*************************************************************************/
static int FindNxtUnprot(address)
```

```
int     address;
{
    int     loopcount;
    int     i = address;
    for (loopcount = 0;loopcount < BUFF_SIZE; loopcount++, Increment(&i))
        {
            if ( ( ( MyBuffer[i] & ATTRIBUTE ) == ATTRIBUTE )
                && ( ( MyBuffer[i] & attrUNPROT ) != attrUNPROT ) )
                break;
        }
        if (loopcount != BUFF_SIZE) loopcount = i;
        return (loopcount);            /* if loopcount = BUFF_SIZE     */
                                       /* no attribute else attribute  */
                                       /* is at loopcount offset       */
} /* end of FindNxtUnprot */

/*************************************************************************/
/*  FindCurAttrib :     FINDCURATTRIB                                    */
/*                                                                       */
/*        Find Attribute for current field (Cursor Position)             */
/*************************************************************************/
static int FindCurAttrib(address)
int address;
{
    int loopcount = 0;
    int i = address;

    for ( ; loopcount < BUFF_SIZE; loopcount++, Decrement (&i) )
        {
            if ( ( MyBuffer[i] & ATTRIBUTE ) == ATTRIBUTE ) break;
        }
        if (loopcount != BUFF_SIZE) loopcount = i;
        return (loopcount);             /* if loopcount = BUFF_SIZE     */
                                        /* no attribute found else attr */
                                        /* is at loopcount offset       */
} /* end of FindCurAttribute */

/*************************************************************************/
/*  ShowStatus :               SHOW STATUS                               */
/*                                                                       */
/*         Writes terminal status information to the log file.           */
/*************************************************************************/
static  void    ShowStatus ( fptr )
FILE    *fptr;
{
   extern  PIMCtlBlk pimSCB;
   fprintf ( fptr,
       "Function=%d, Status=%d, State=%d, KBLock=%d, HWLock=%d, Cursor=%d\n",
       pimSCB.psFunction, pimSCB.psStatus, pimSCB.psState,
       pimSCB.psKBLock, pimSCB.psHWLock, pimSCB.psCursor );
   fprintf ( fptr,
       "Owner=%d, StatWord=%d, ProgChk=%d, CommChk=%d, MachChk=%d\n",
       pimSCB.psOwner, pimSCB.psStatWord, pimSCB.psProgChk,
       pimSCB.psCommChk, pimSCB.psMachChk );
   fflush ( fptr );
}                                                /* end of ShowStatus */

/*************************************************************************/
/*  DisplayData :       DISPLAYDATA                                      */
/*                                                                       */
/*            Paints the 3270 buffer onto the log file.                  */
/*************************************************************************/
static  void    DisplayData ( i )
int     i;
{
    int     loopcount;
    byte    *dataptr = (byte *) MyBuffer;

    if (i == 0) i = BUFF_SIZE;
    for (loopcount = 0 ; loopcount < i; loopcount++, dataptr++ )
        /* This routine records data that is stored in the holding buffer
        "MyBuffer". Note that because part of the screen may contain
        attributes, printable characters are checked for first.        */
    {
        if ( ( *dataptr >= LO_ASCII ) && ( *dataptr <= HI_ASCII ) )
```

```
                (void) putc ( *dataptr, stderr );
        else (void) putc ( LO_ASCII, stderr );
    }
    putc ( '\n',stderr );
}                                               /* end of DisplayData */

/*********************************************************************/
/*  SetMDT :                 SETMDT                                  */
/*                                                                   */
/*     Sets the  MDT bit ON for the attribute of the current cursor  */
/*     or screen pointer position  if n = 0                          */
/*********************************************************************/
static void    SetMDT ( n )
int     n;
{
    int     posit;
    if (n == 0)  n = (Decrement (&CursorPosn));  /* Use cursor       */
    else    n = n - 1;           /* convert to offset                */

    if ((posit = FindCurAttrib(n)) != BUFF_SIZE)
        MyBuffer[posit] |= attrMDT; /* if attribute found, set the MDT */
}                                /* end of SetMDT                    */

/*********************************************************************/
/*  FillinData :        FILLINDATA                                   */
/*                                                                   */
/*           Move Input text to pos "n" or cursor if n = 0           */
/*********************************************************************/
static  void    FillinData ( n )
int     n;
{
    int     i;

    if (n == 0) n = CursorPosn;              /* Use cursor          */
    else n = n - 1;                          /* Convert to Offset   */
    if ( TextLgth > 0)
    {
        for (i =0; i < TextLgth; i++)
        {
            MyBuffer[n] = text[i];
            Increment (&n);
        }
    }
    CursorPosn = n;
    return;
}                                               /* end of FillinData  */

/*********************************************************************/
/*  MatchSessOwn :      MATCHSESSOWN                                 */
/*                                                                   */
/*     Wait to match session owner to equal input val.               */
/*                                                                   */
/*     Possible states for sessions are defined as follows:          */
/*           0 = Undefined;           2 = Owned by SSCP;             */
/*           1 = Unowned;             3 = Owned by LU.               */
/*                                                                   */
/*********************************************************************/
static  void    MatchSessOwn ( i )
int     i;
{
    while  ( pimSCB.psOwner != i )
    {
        alarm(5);
        pause();
        alarm(0);
    }
    ClearScreen();
}                                               /* end of MatchSessOwn*/

/*********************************************************************/
/*  TabCursor :             TABCURSOR                                */
/*                                                                   */
/*     Sets the cursor to the next available unprotected field       */
/*********************************************************************/
static  void    TabCursor ( n )
```

```
int     n;
{
    int     i;
    int     posit;

    if (n == 0) n = 1;                          /* Ensure one tab    */
    for ( i = 0; i < n; i++)
    {
        if ((posit = FindNxtUnprot(CursorPosn)) != BUFF_SIZE)
        {
            Increment(&posit);
            CursorPosn = posit;
        }
    }
}                                               /* end of TabCursor   */

/*********************************************************************/
/*  WaittoPush :        WAITTOPUSH                                   */
/*                                                                   */
/*  Waits for both an absence of communication (COMM) errors and a   */
/*  NotReceive state to be reached.  If the COMM check is not made,  */
/*  unexpected Host and PIM errors may result.                       */
/*********************************************************************/
static  void    WaittoPush ( )
{
   while ( ( pimSCB.psProcState != prNRCV ) || ( pimSCB.psCommChk != 0 ) )
   {
       alarm(5);
       pause();
       alarm(0);
       if ( pimSCB.psCommChk != 0 )
       {
           fprintf( stderr, "CommChk = %d\n",pimSCB.psCommChk );
           fflush( stderr );
       }
   }
}                                                /* end of WaittoPush */

/*********************************************************************/
/*  SendAIDKey :        SENDAIDKEY                                   */
/*                                                                   */
/*      Sends the AID key represented by "i" from Logon data         */
/*********************************************************************/
static  void    SendAIDKey ( i )
int     i;
{
    int     AIDKey;

    WaittoPush();
    switch ( i )    /* case for AID key val} assignment */
    {
        case    0:      AIDKey = aiENTER;
                        break;
        case    1:      AIDKey = aiPF1;
                        break;
        case    2:      AIDKey = aiPF2;
                        break;
        case    3:      AIDKey = aiPF3;
                        break;
        case    30:     AIDKey = aiCLEAR;
                        ClearScreen;
                        break;
        case    31:     AIDKey = aiPA1;
                        break;
        case    32:     AIDKey = aiPA2;
                        break;
        case    33:     AIDKey = aiPA3;
                        break;
        case    40:     AIDKey = aiSELPEN;
                        break;
        case    41:     AIDKey = aiTESTREQ;
                        break;
        default:        AIDKey = aiENTER;
                        break;
    }
```

```
/* The following AID key error should not occur if the WAITTOPUSH
   communication error check is completed successfully.          */

    if ( PostAIDKey ( SessionID, AIDKey ) == FAIL )
       fprintf ( stderr, AidError, errno, LU_Address );
}                                                 /* end of SendAIDKey */

/***********************************************************************/
/*  SendSNAEvent :      SENDSNAEVENT                                   */
/*                                                                     */
/*      Send the SNA Event key represented by "i" from Logon data      */
/***********************************************************************/
static  void    SendSNAEvent ( i )
int     i;
{
    int     EventKey;

    /* It is important to note each of the states and conditions that
       are required for each SNA key assignment.  For example, if a
       SYSREQ is iss}d, it is essential to verify the absence of any
       communication (COMM) errors. At the conclusion of the SNA
       event case statement, a check of current states is made.    */

    switch ( i )
    {
        case 1:       EventKey = scPSA;
                      WaittoPush();       /* not nRCV not Com  */
                      break;

        case 2:       EventKey = scSYSREQ; /* not CommChk       */
                      while (pimSCB.psCommChk != 0)
                      ;
                      ClearScreen ();
                      break;

        case 3:       EventKey = scATTN;   /* not DTR and SSCP  */
                      while ( (pimSCB.psCommChk != 0) ||
                      (pimSCB.psProcState == prDTRESET) ||
                      (pimSCB.psOwner != owLU) )
                      ;
                      break;

        default:      EventKey = scPSA;
                      WaittoPush();
                      break;
    }

    /* Check for improper states */
    if ( PostSNAEvent ( SessionID, EventKey ) == FAIL )
        fprintf ( stderr, EventError, errno, LU_Address );
}                                               /* end of SendSNAEvent */

/***********************************************************************/
/*  MatchData :         MATCHDATA                                      */
/*                                                                     */
/*                                                                     */
/***********************************************************************/
static  void    MatchData ( i )
int     i;
{
    int NotFound;
    int posit;

    /* This function calls GetBuf, and waits for GetBuf to complete.
       When GetBuf returns successfully, Screen data is compared to
       the input data. If the Screen data does not match the input
       data, the process is repeated.                              */

    NotFound = TRUE;
    while ( NotFound == TRUE )
    {
        GetBuf();
        if ( CloseDown == TRUE )
        {
                NotFound = FALSE;
                break;
            }
            if (i == 0)                          /* Use cursor and backup */
            {
                if ((posit = FindCurAttrib (CursorPosn)) != BUFF_SIZE)
                {
                    Decrement(&posit);
                    posit = FindCurAttrib (posit);
                    Increment(&posit);
                }
                else posit = CursorPosn;
            }
            else posit = i - 1;
            if (strncmp(&MyBuffer[posit], text, TextLgth) == 0)  NotFound = FALSE;
            else
            {
                alarm(2);
                pause();
                alarm(0);
            }
        }
        return;
    }                                                 /* end of MatchData */

/************************************************************************/
/*  PutBuf :            PUTBUF                                          */
/*                                                                      */
/*          Moves Mybuffer into ScreenBuffer.                           */
/*                                                                      */
/*   Waits for the PreHandler routine to set PutRequest to  FALSE.      */
/*   This indicates that the PreHandler routine has moved "MyBuffer"    */
/*   into "ScreenBuffer".                                               */
/************************************************************************/
static  void    PutBuf( )
{
    while (PutRequest == TRUE)
    {
        alarm(2);
        pause();
        alarm(0);
    }
}                                                 /* end of PutBuf   */

/************************************************************************/
/*  GetBuf :            GETBUF                                          */
/*                                                                      */
/*         Moves ScreenBuffer into MyBuffer                             */
/*                                                                      */
/*   Sets GetRequest to TRUE, then waits for the PostHandler routine    */
/*   to set GetRequest to FALSE; the PostHandler routine then moves     */
/*   "ScreenBuffer" into "MyBuffer".                                    */
/*   If GetRequest is not set to FALSE within a reasonable amount       */
/*   of time, a retry counter indicates a timeout.                      */
/************************************************************************/
static  int    GetBuf( )
{
    int    Retries;
    int    Secs;

    Secs   = 2;
    Retries = NUM_RETRIES;
    GetRequest = TRUE;
    while (GetRequest == TRUE)
    {
        alarm(Secs);
        pause();
        alarm(0);
        if ( Retries == 0 ) /* Maximum number of retries reached */
        {
            fprintf( stderr, "TIMEOUT Screen data ");
            fprintf( stderr, "After %d SECONDS\n",
                             (Secs * NUM_RETRIES));
            printf("\nTimeout occurred during data transfer");
            DisplayData(0);
            GetRequest = FALSE;
            CloseDown = TRUE;
        }
        else  Retries--;                /* count down retries  */
    }
}                                                /* end of GetBuf       */

/**********************************************************************/
/*  OpenAlt :           OPENALT                                       */
/*                                                                    */
/*            Open alternate file (file to transfer)                  */
/**********************************************************************/
static  void    OpenAlt ( i )
int     i;
{
    cpyblk ( altfn, text, LINELEN );
    if ((altscriptp = fopen (altfn, "r")) == NULL)
    {
        printf ("Open error AIX input file \n");
        Close3270(SessionID);
        exit (exKILLSTRT );
    }
    mainscriptp = scriptp;
    scriptp = altscriptp;
    Reps = i;
    altope = 1;
}                                                  /* end of OpenAlt    */

/**********************************************************************/
/*  CloseAlt :          CLOSEALT                                      */
/*                                                                    */
/*            Close and possibly reopen alternate input.              */
/**********************************************************************/
static  void    CloseAlt ( )
{
    scriptp = altscriptp;
    fclose(altscriptp);
    Reps--;
    if ( Reps == 0 )
    {
        SuppressLog = FALSE;
        scriptp = mainscriptp;
        altope = 0;
    }
    else
    {
        fprintf(stderr,"********************");
        fprintf(stderr,"Log suppressed AIX file Open");
        fprintf(stderr,"********************\n");
        SuppressLog = TRUE;
        if ((altscriptp = fopen (altfn,"r")) == NULL)
        {
            printf ("Open error AIX input file \n");
            Close3270(SessionID);
            exit (exKILLSTRT );
        }
    }
}                                                /* end of CloseAlt     */

/**********************************************************************/
/*  AcceptAndSend1 :    ACCEPT AND SEND LOGON DATA ( FILE .LUn )      */
/*                                                                    */
/*  Gets input data and AID key from the input file and sends data.   */
/**********************************************************************/
static  void    AcceptAndSend1 ( )
{
    int     cmnd, nval;
    char    *temp;

    /* If there is an error when reading the input file, EINTR will
       catch the alarm and post it to errno; a reread is then tried. */
    while (TRUE)
    {
        while (((temp = fgets ( InputData, LINELEN, scriptp)) == NULL)
            && ( errno == EINTR ));
    if ((temp == NULL) && ( errno != EINTR ))
    {
        fprintf(stderr, "pimrtup: EOF or error, errno = %d\n", errno);
        fflush(stderr);
        CloseDown = TRUE;
        return;
    }
    cmnd = 99;
    nval = 0;
    text[0] = 0;

    /* Break input line down */
    if ((sscanf(InputData, "%d %d %[¬\n]",&cmnd,&nval,text)) x 3)
    {
        fprintf(stderr, "pimrtup: scan error, errno = %d\n", errno);
        contin};
    }
    TextLgth = ( strlen  ( text ) );
    break;
}
Count = Count + 1;
if ( SuppressLog == FALSE )
{
    fprintf(stderr,"# %d:cmnd=%d:nval=%d:text '%s'\n",
    Count,cmnd,nval,text);
    fflush(stderr);
}
if ( Silent == FALSE ) printf("%d ",Count); /* Used for display to the
{
    printf("%d ",Count);       /* Used for display to the screen
                                  when running in background mode  */
switch ( cmnd )
{
    case 1:  PutRequest = TRUE;
             SendAIDKey(nval);        /*  Send AID key,          */
             PutBuf();                /*  then move buffer       */
             break;

    case 2:  SendSNAEvent(nval);      /*  Send SNAEvent out      */
             break;

    case 3:  GetBuf();                /* Read screen into        */
             break;                   /* MyBuffer                */

    case 10: FillinData(nval);        /* Move data in            */
             SetMDT ( 0 );            /* Set MDT bits on         */
             break;

    case 11: SetMDT (nval);           /* Set MDT bits on         */
             break;                   /* at cursor-field or      */
                                      /* nval position           */

    case 12: CursorPosn = nval;       /* Set Cursor position     */
             break;

    case 13: CursorPosn = BUFF_SIZE - 1;    /* "home" cursor     */
             TabCursor(1);
             break;

    case 14: TabCursor(nval);          /* Move Cursor to next    */
             break;                    /* unprotected field      */
                                       /* nval times             */

    case 20: MatchSessOwn(nval);       /* Match Session Owner    */
             break;                    /* to equal nval. Val}s:  */
                                       /* 0 = undefined          */
                                       /* 1 = unowned            */
                                       /* 2 = SSCP               */
                                       /* 3 = LU                 */

    case 21: MatchData(nval);          /* Match screen buffer    */
             break;                    /* Data to input          */
                                       /* at "nval" position;    */
                                       /* if nval=0, then match to*/
                                       /* the input at the 1st   */
```

```
                                            /* char of the prior field */
                                            /* from cursor position    */

        case 80: OpenAlt(nval);             /* Open AIX input file     */
                 break;

        case 89: CloseAlt();                /* Close AIX input file    */
                 break;                     /* this must be the last   */
                                            /* command in any ALT file */

        case 90: DisplayData (nval);        /* record to log           */
                 break;

        case 91: ShowStatus(stderr);        /* store the current status*/
                 break;                     /* of the terminal on log  */

        case 92: alarm (nval);              /* loop wait for nval secs */
                 pause();
                 alarm(0);
                 break;

        case 98: CloseDown = TRUE;          /* shutdown time           */
                 break;

        case 99: if ( ( nval == 1 ) && ( Count == 1 ) )
                 {                          /* if this is the first    */
                     Silent = FALSE;        /* input record            */
                     printf("%d ",Count++);
                 }
                 if ( Silent == FALSE )
                 {
                     printf("%s ",text);     /* then toggle on display */
                 }                           /* for backround mode     */
                 break;

        default: fprintf(stderr, "Input command error %d %d %s",
                 cmnd, nval, text);
                 break;
    }
}                                              /* end of AcceptAndSend1 */

/***********************************************************************/
/*  AcceptAndSend2 :    ACCEPT AND SEND TRANSFER DATA ( FILE .Alt )    */
/*                                                                     */
/*  Gets input data from file, stores it in screen and sends it        */
/***********************************************************************/
static  void    AcceptAndSend2 ( )
{
    int     finprog;
    int     mrec, nrec;
    char    *temp;
    mrec = 10;          /* mrec = maximum nb of records per screen  */

    finprog=FALSE;
    while (finprog==FALSE)
    {
        GetBuf();
        for (nrec = 0; (nrec < mrec)&&(finprog==FALSE); nrec++)
        {
            /* If there is an error when reading the input file, EINTR will
            catch the alarm and post it to errno; a reread is then tried. */
            temp = fgets ( InputData, LINELEN, scriptp );
            if (temp==NULL)
            {
                if ( errno == EINTR )
                {
                    fprintf(stderr,"pimrtup: EOF\n");
                    fflush(stderr);
                    sprintf(text,"EOF");
                    finprog=TRUE;
                }
                else
                {
                    fprintf(stderr,"pimrtup: read error, errno = %d\n",errno);
                    fflush(stderr);
                    CloseDown = TRUE;
                    finprog= TRUE;
                }
            }
            else
            {
                /* get text from input record */
                if ((sscanf(InputData, "%[¬\n]",text)) > 3)
                {
                    fprintf(stderr,"pimrtup: scan error, errno = %d\n",errno);
                    continue;
                }
                Count = Count + 1;
                Enrnb = Enrnb + 1;
                if ( SuppressLog == FALSE )
                {
                    fprintf(stderr,"# %d:text '%s'\n",Count,text);
                    fflush(stderr);
                }
                if ( Silent == FALSE )
                {
                    printf("%d ",Count);         /* Used for display to the
                                                   screen when running in
                                                   background mode        */
                }
            }
            TextLgth = ( strlen ( text ) );
            FillinData (0);               /* move data in                  */
            SetMDT (0);                   /* set MDT bits on               */
            TabCursor (1);                /* move cursor next unprot field */
        } /*end_for*/
        PutRequest = TRUE;              /* send screen to Host by        */
        SendAIDKey(0);                  /* depressing the ENTER key      */
        PutBuf();
    } /*end_while*/
    GetBuf();                         /* wait for a screen from Host   */
    PutRequest = TRUE;                /* send screen to Host by        */
    SendAIDKey(0);                    /* depressing the ENTER key      */
    PutBuf();
    CloseAlt();
}/*end_procedure*/                       /* end of AcceptAndSend2         */

/**********************************************************************/
/*  ErrHandler :              ERROR HANDLER                           */
/*                                                                    */
/*                Error handling function for PIM.                    */
/*                                                                    */
/* The error handler function is specified via the InitPIM call and is */
/* used when PIM encounters a fatal error.  PIM executes this function */
/* before shutting down, enabling the PIM test program to close down  */
/* cleanly.                                                           */
/**********************************************************************/
static int   ErrHandler (errorcode)
int errorcode;
{
    fprintf(stderr,
        "pimrtup: ErrHandler called with errorcode %d and errno %d\n",
        errorcode, errno);
    CloseDown = TRUE;                        /* fatal error          */
    return(SUCCEED);
}                                            /* end of ErrorHandler  */

/**********************************************************************/
/*  PreHandler :              PRE HANDLER                             */
/*                                                                    */
/*                     Pre exit handler for PIM.                      */
/**********************************************************************/
static  int    PreHandler ( sd, msg_type, status_blk )
int    sd;                    /*  Session descriptor.        */
int    msg_type;              /*  Message type.              */
PIMCtlBlk *status_blk;        /*  Pointer to status block.   */
{
    if ( ( PutRequest == TRUE ) && ( pimSCB.psProcState == prNRCV ) )
    {
        PutRequest = FALSE;
        cpyblk ( ScreenBuffer, MyBuffer, BUFF_SIZE);
        pimSCB.psCursor = CursorPosn;  /* give local cursor       */
    }                              /* position to PIM         */
    return(SUCCEED);
}                                          /* end of PreHandler        */

/**********************************************************************/
/*  PostHandler :           POST  HANDLER                             */
/*                                                                    */
/*                      Post exit handler for PIM.                    */
/**********************************************************************/
static  int     PostHandler ( sd, msg_type, status_blk )
int     sd;             /*  Session descriptor.         */
int     msg_type;       /*  Message type.               */
PIMCtlBlk *status_b'k;  /*  Pointer to status block.    */
{
    extern  PIMCtlBlk pimSCB;
/*
    switch(msg_type)
    {
        case UEERROR:
                  fprintf(stderr,"pimrtup: ***** Received }ERROR\n");
                  break;
        case UESTATUS:
                  fprintf(stderr,"pimrtup: ***** Received }STATUS\n");
                  break;
        case UEREAD:
                  fprintf(stderr,"pimrtup: ***** Received }READ\n");
                  break;
        case UEWRITE:
                  fprintf(stderr,"pimrtup: ***** Received }WRITE\n");
                  break;
        default:
                  fprintf(stderr,"pimrtup: ***** Received %d\n",
                  msg_type);
    }
*/
/* Before acting on a req}st to get a buffer, it must first be ensured
   that the Keyboard is not locked and that HostWait is not locked. Once
   these conditions are met, buffers can safely be manipulated.       */

    if ( ( ( ( GetRequest == TRUE ) && ( pimSCB.psKBLock != TRUE ) )
                                && ( pimSCB.psHWLock != TRUE ) ) )
    {
        GetRequest = FALSE;
        cpyblk ( MyBuffer, ScreenBuffer, BUFF_SIZE );
        CursorPosn = pimSCB.psCursor;
    }
    /* Whenever PIM invokes the PostHandler, the PIM application must
       check to see if PIM has req}sted a Close.                     */
    if (pimSCB.psState == stCLOSED) CloseDown = TRUE;
    return ( SUCCEED) ;
}                                              /* end of PostHandler */

/****************************************************
 *              MAIN PROGRAM                        *
 *                                                  *
 *   The main processing function begins here....   *
 ****************************************************/

/* The PIM application will be called with the following set of parameters
   (some of the parameters are used internally and are not relevant to the
   PIM programmer):
                        argc - count
                        argv [1]  LU address
                        argv [2]  emulation type
                        argv [3]  device handler process ID
                        argv [4]  start process ID
                        argv [5]  write signal from device handler*/
main ( argc, argv )
int     argc;
char    *argv[];
{
    char    input[LINELEN];
   /* Once the LU address is defined, a "suffix" is assigned to
      permit the use of a different set of input and log files for
      individual LUs.  Log and input files are then opened.      */

   LU_Address = atoi ( argv [ 1 ] );
   (void) strcpy ( LuSuffix, argv [ 1 ] );
   (void) strcat( input, LogFilePrefix );
   (void) strcat( input, LuSuffix );
   (void) freopen ( input, "w", stderr );
   input[0] = 0;
   (void) strcat( input, ScrFilePrefix );
   (void) strcat( input, LuSuffix );

   if ((mainscriptp = fopen (input, "r")) == NULL)
   {
       printf ("Open error Logon file %s\n",input );
       exit (exKILLSTRT );
   }

   fprintf (stderr,"Input %s opened \n",input);

  /* This example of InitPIM passes the arguments on to PIM. It specifies
     that PIM is to catch any signals that come through from the device
     handler; that the application is non-interactive and will run in
     background mode; and specifies the name of the fatal error handler
     as "ErrHandler".                                                 */

/* Should a PIM application encounter serious problems,the PIM program
   must set an exit status to inform the other components of the IX
   3270-PLUS product of this fact. The exKILLSTRT is defined within
   pim3270.h.                                                       */
   if ( InitPIM ( argc, argv, fCatch, FALSE, ErrHandler ) == FAIL )
   {
       fprintf (stderr,"pimrtup: Error -%d- initializing PIM.\n", errno );
       exit ( exKILLSTRT );           /* Tell install to tell start pim */
   }

   /*   Set signals */
   if ( ( (int) signal(SIGHUP, CloseItDown) ) == FAIL )
   {
       fprintf (stderr,"pimrtup: Error %d in setting signal SIGHUP \n",errno);
       exit ( exKILLSTRT );           /* Tell install to tell start pim  */
   }

   if ( ( (int) signal(SIGINT, CloseItDown) ) == FAIL )
   {
       fprintf(stderr,"pimrtup: Error %d in setting signal SIGINT \n",errno);
       exit ( exKILLSTRT );           /* Tell install to tell start pim  */
   }

   if ( ( (int) signal(SIGALRM, AlrmCatcher) ) == FAIL )
   {
       fprintf(stderr,"pimrtup: Error %d in setting signal SIGALRM \n",errno);
       exit ( exKILLSTRT );           /* Tell install to tell start pim  */
   }

/* Note that we must ignore the SIGILL signal since it might be sent  */
/* by the Device Handler to determine if the application is running.  */
   if ( ( (int) signal(SIGILL, SIG_IGN) ) == FAIL )
   {
       fprintf(stderr,"pimrtup: Error %d in setting signal SIGILL \n",errno);
       exit ( exKILLSTRT );            /* Tell install to tell start pim */
   }

   /*  Open a session for our Host address.                          */
   fprintf(stderr,"\npimrtup: Opening 3270 session\n");

   SessionID = Open3270 ( LU_Address, dtTERMINAL, BUFF_SIZE, &pimSCB );
   if ( SessionID == FAIL )
   {
       fprintf (stderr,"pimrtup: Error -%d- opening LU#%d.\n",
           errno, LU_Address );
       exit ( exKILLSTRT );            /* Tell install to tell start pim */
   }
   else
```

```
    /* Before processing any data, the application msut pause until the
       emulator is marked open.                                           */
        while ( pimSCB.psState != stOPEN ) pause ( );

    /************************** Main Loop *************************/
    scriptp = mainscriptp;                /* running main script */

    while (!CloseDown)
    {
        if ( altope == 0 ) AcceptAndSend1();  /* read user Logon input file
                                             for this LU and then execute it */
        else AcceptAndSend2();                /* read user input file to
                                             transfer into screen     */
    }

    /*  Close down this logical unit.      */

    printf("\nPimrtup finished");
    printf("\n%d records transferred\n", Enrnb );
    ShutDown ( );
    exit(0);
}                                                   /* end of Main */
```

PIM Program pimrtdn.c for File Transfer from IBM AS/400

```
/*************************************************************************

@(#)pimrtdn.c  1.8

P I M   3 2 7 0   - -   F I L E   T R A N S F E R   P R O G R A M

NAME:         pimrtdn.c

SYNOPSIS:     Transfer of a sequential file from the AS/400 to the IBM RT
              using the capabilities of the PIM modules of the
              program &n327. (SNA).

DESCRIPTION: In the first step, this program establishes contact to
              the AS/400 by reading and executing the logon file
              PimLogdn.LU#, where # is the LU number specified in the
              PIM startup command: s3270 pim LU# pimrtdn. On the
              AS/400 a procedure is started and an
              RPG program is activated. This program will write data
              to a 3270 screen buffer.
              In a second step, the AS/400 file to be transferred is
              read on theAS/400 by the RPG program. Records are moved into a
              screen and, once the screen is filled, it is sent to IBM RT
              which reads it. The PIM program reades the corresponding
              170-byte records from the 3270 screen buffer and stores
              it in the AIX file on the IBM RT
              When the end of the file is reached, the RPG program
              writes an :q.EOF:eq. string on the next available
              record entry on screen. The name of the file to be
              transferred is provided in the output file PimOut.LU#.
              In a third step, control is given back to the logon file
              PimLogdn.LU# which terminates activities on the AS/400
              by performing the sign off.
              This program only enables the transfer of files
              containing characters of the 3270 character set.

HISTORY:      This program is based on the PIM sample program
              described in the manual IBM SC23-0776.

*************************************************************************/

#include <stdio.h>      /* Standard I/O definitions.                    */
#include <signal.h>     /* signal definitions.                          */
#include <errno.h>      /* errno definitions.                           */
#include "pim3270.h"    /* PIM 3270 definitions.                        */

extern  int     errno;
typedef unsigned char   byte;

#define COLS            80      /* Columns per 3270 screen.             */
#define ROWS            24      /* Lines per 3270 screen.               */
#define BUFF_SIZE       ( ROWS * COLS )     /* Buffer size.             */
#define FIELDLN         170     /* Length of a screen field             */
#define LINELEN         175     /* Input  read limit size               */
#define FAIL            -1      /* Global flag.                         */
#define SUCCEED         0       /* Global flag.                         */
#define FALSE           0       /* Global flag.                         */
#define TRUE            1       /* Global flag.                         */
#define ATTRIBUTE       0x080   /* Attribute byte indicator.            */
#define attrUNPROT      0x020   /* Unprotected field attribute          */
#define attrMDT         0x001   /* ModifiedDataTag(MDT)attribute        */
#define HI_ASCII        '\177'  /* Highest printable character.         */
#define LO_ASCII        '\040'  /* Lowest printable character.          */
#define attrFT          0x0FC   /* FileTran attribute value             */
#define ftCNTL          0x043   /* FileTran control block indic         */
#define ftDATA          0x041   /* FileTran data block indic            */
#define NUM_RETRIES     60      /* Num of Retries on Buffer read        */
#define LogFilePrefix   "PimLogdn.LU     /* Logon file name pref        */
#define ScrFilePrefix   "PimOut.LU"      /* Data file name pref         */

/* These lines define the routines that PIM invokes each time it
   receives a signal from the device handler.  The PreHandler
   routine is invoked prior to PIM's processing of a device
   handler message. The PostHandler routine is called after PIM
   has completed its processing of a device handler message.          */
extern int PreHandler();        /* pre-exit routine.                  */
extern int PostHandler();       /* post-exit routine.                 */

static  char ScreenBuffer [ BUFF_SIZE ] =
    { NULL };                   /* PIM 3270 screen buffer.            */
static  char MyBuffer [ BUFF_SIZE ] =
    { NULL };                   /* My 3270 screen buffer.             */

/* These lines define the PIM session control block and initialize
   the block to run predefined val}s (above).  A pointer to the
   control block is passed to PIM upon initialization.                */

PIMCtlBlk pimSCB =              /* PIM Session Control Block.         */
{
    PreHandler,                 /*  Pre-exit routine for PIMRespond.  */
    PostHandler,                /*  Post-exit routine for PIMRespond. */
    ScreenBuffer                /*  Pointer to 3270 device buffer.    */
};

static  char AidError [] = "pimrtdn: Error -%d- post AID key, LU%d\n";
static  char EventError [] = "pimrtdn: Error -%d- post SysReq event, LU%d\n";
static  int     SessionID;      /* PIM session descriptor.            */
static  int     LU_Address;     /* 3270 logical unit address.         */
static  int     fCatch = TRUE;  /* PIM will catch signals.            */
static  int     CloseDown = FALSE;  /* Shutdown flag.                 */
static  int     AlarmWakeup = FALSE;/* Signal was Alarm               */
static  FILE    *scriptp;       /* Pointer to current file            */
static  FILE    *mainscriptp;   /* MAIN Input file pointer (Logon file)*/
static  FILE    *altscriptp;    /* AIX Output file pointer            */
static  char    InputData [LINELEN];
static  char    text[LINELEN];  /* input data text          */
static  char    altfn[LINELEN]; /* Name of Output file      */
static  char    LuSuffix[3];    /* Logon file name suffix   */
static  int     TextLgth;       /* Length of input text               */
static  int     Count = 0;      /* Count of the input lines read      */
static  int     Enrnb = 0;      /* Number of records transferred      */
static  int     Silent = TRUE;  /* If TRUE no printf's only fprintf's */
static  int     CursorPosn;     /* Cursor position in MyBuffer buffer */
static  int     GetRequest;     /* Get request for screen buffer      */
static  int     PutRequest;     /* Put request to screen buffer       */
static  int     Reps;           /* Number of AIX file opens to execute */
static  int     SuppressLog;    /*Switch suppr. logging ON for AIX file*/
static  int     altope = 0;     /* switch for input dataset           */

/**********************************************************************/
/*  AlrmCatcher :       SIGNAL CATCHERS                               */
/*                                                                    */
/**********************************************************************/
static void AlrmCatcher(sig)
int sig;
{
    AlarmWakeup = TRUE;
    (void)signal(sig, AlrmCatcher); /* set it back                    */
}

static void FastShutDown(sig)
int sig;
{
    extern void ShutDown();

    (void)signal(sig, FastShutDown); /* set it back                  */
    printf("\npimrtdn: fast shutdown\n");
    Close3270(SessionID);
    exit(0);
}

static void CloseItDown(sig)
int sig;
{
    (void)signal(sig, FastShutDown); /* set it to FastShutDown       */
    printf("\npimrtdn: Closedown signal\n");
    CloseDown = TRUE;
}                               /* end of program header definitions */
```

```
/*************************************************************************/
/*  ShutDown :                  SHUTDOWN                                 */
/*                                                                       */
/*                          Shuts down the program.                      */
/*************************************************************************/
static  void ShutDown ()       /*  Close PIM test program, log errors */
{
    if ( Close3270 ( SessionID ) == FAIL )
    {
        fprintf(stderr,"pimrtdn: Error -%d- closing LU#%d.\n",
            errno, LU_Address );
        fflush ( stderr );
        exit(0);
    }

/* Important: the PIM application should always wait for an
   acknowledgement of a close prior to exiting the program.            */

    while (pimSCB.psState != stCLOSED) pause ( );
    fprintf(stderr,"\npimrtdn: session terminated\n");
    exit(0);
}                                           /* end of ShutDown */

/*************************************************************************/
/*                                                                       */
/*                        U T I L I T I E S                              */
/*                                                                       */
/*************************************************************************/

/*************************************************************************/
/*  ClearScreen :       CLEAR SCREEN                                     */
/*                                                                       */
/*                 Sets the 3270 buffer to NULL.                         */
/*************************************************************************/
static  void    ClearScreen ( )
{
    byte    *dataptr = (byte *) MyBuffer;
    byte    *endptr  = (byte *) (MyBuffer + BUFF_SIZE);

    while (dataptr < endptr) *dataptr++ = 0;
}                                                 /* end of ClearScreen */

/*************************************************************************/
/*  Decrement :                 DECREMENT                                */
/*                                                                       */
/*          Decrement function that wraps around size of screen.         */
/*************************************************************************/
static int       Decrement(address)
int *address;

{
    if (*address == 0) *address = BUFF_SIZE - 1;         /* Wrap around */
    else (*address)--;                                   /* Decrement   */
    return(*address);
}                                    /* end of Decrement                */

/*************************************************************************/
/*  Increment :                 INCREMENT                                */
/*                                                                       */
/*    Increment function that wraps around size of screen.               */
/*************************************************************************/
static  int   Increment(address)
int *address;

{
    (*address)++;                                  /* Increment Address */
    if (*address >= BUFF_SIZE) *address = 0;       /* Check if Wrap     */
    return(*address);
}                                    /* end of Increment                */

/*************************************************************************/
/*  FindNxtUnprot :     FINDNXTUNPROT                                    */
/*                                                                       */
/*        Find next Unprotected Attribute from current field             */
/*************************************************************************/
```

```
static int FindNxtUnprot(address)
int address;
{
    int loopcount;
    int i = address;

    for (loopcount = 0;loopcount < BUFF_SIZE; loopcount++, Increment(&i))
    {
        if ( ( ( MyBuffer[i] & ATTRIBUTE ) == ATTRIBUTE )
        && ( ( MyBuffer[i] & attrUNPROT ) != attrUNPROT ) )
        break;
    }
    if (loopcount != BUFF_SIZE) loopcount = i;
    return (loopcount);  /* if loopcount = BUFF_SIZE no attribute, else */
                         /* attribute is at loopcount offset           */
} /* end of FindNxtUnprot */

/*********************************************************************/
/*  FindCurAttrib :     FINDCURATTRIB                                */
/*                                                                   */
/*          Find Attribute for current field (Cursor Position)       */
/*********************************************************************/
static int FindCurAttrib(address)
int address;
{
    int loopcount = 0;
    int i = address;
    for ( ; loopcount < BUFF_SIZE; loopcount++, Decrement (&i) )
       if ( ( MyBuffer[i] & ATTRIBUTE ) == ATTRIBUTE ) break;

    if (loopcount != BUFF_SIZE) loopcount = i;
    return (loopcount);  /* if loopcount = BUFF_SIZE no attribute found */
                         /* else attr is at loopcount offset           */
}                                     /* end of FindCurAttribute      */

/*********************************************************************/
/*  ShowStatus :              SHOW STATUS                            */
/*                                                                   */
/*          Writes terminal status information to the log file.      */
/*********************************************************************/
static  void    ShowStatus ( fptr )
FILE    *fptr;
{
    extern  PIMCtlBlk pimSCB;
    fprintf ( fptr,
        "Function=%d, Status=%d, State=%d, KBLock=%d, HWLock=%d, Cursor=%d\n",
        pimSCB.psFunction, pimSCB.psStatus, pimSCB.psState,
        pimSCB.psKBLock, pimSCB.psHWLock, pimSCB.psCursor );
    fprintf ( fptr,
        "Owner=%d, StatWord=%d, ProgChk=%d, CommChk=%d, MachChk=%d\n",
        pimSCB.psOwner, pimSCB.psStatWord, pimSCB.psProgChk,
        pimSCB.psCommChk, pimSCB.psMachChk );
    fflush ( fptr );
}                                                /* end of ShowStatus */

/*********************************************************************/
/*  DisplayData :       DISPLAYDATA                                  */
/*                                                                   */
/*              Paints the 3270 buffer onto the log file.            */
/*********************************************************************/
static  void    DisplayData ( i )
int     i;
{
    int     loopcount;
    byte    *dataptr = (byte *) MyBuffer;

    if (i == 0) i = BUFF_SIZE;

    /* This routine records data that is stored in the holding buffer
       "MyBuffer". Note that because part of the screen may contain
       attributes, printable characters are checked for first.       */
    for (loopcount = 0 ; loopcount < i; loopcount++, dataptr++ )
    {
        if ( ( *dataptr >= LO_ASCII ) && ( *dataptr <= HI_ASCII ) )
            (void) putc ( *dataptr, stderr );
        else (void) putc ( LO_ASCII, stderr );
    }
    putc ( '\n',stderr );
}                                                    /* end of DisplayData */

/************************************************************************/
/*  SetMDT :                  SETMDT                                    */
/*                                                                      */
/*     Sets the  MDT bit ON for the attribute of the current cursor     */
/*     or screen pointer position  if n = 0                             */
/************************************************************************/
static void     SetMDT ( n )
int     n;
{
    int     posit;

    if (n == 0) n = ( Decrement ( &CursorPosn ) ); /* Use cursor   */
    else n = n - 1;                  /* convert to offset           */

    if ((posit = FindCurAttrib(n)) != BUFF_SIZE) /* if attr  found  */
        MyBuffer[posit] |= attrMDT;              /* ....set the MDT */
}                                      /* end of SetMDT              */

/************************************************************************/
/*  FillinData :        FILLINDATA                                      */
/*                                                                      */
/*          Move Script text to pos "n" or cursor if n = 0              */
/************************************************************************/
static  void    FillinData ( n )
int    n;
{
    int    i;

    if (n == 0)         n = CursorPosn;  /* Use cursor           */
    else n = n - 1;                      /* Convert to Offset    */
    if ( TextLgth > 0)
    {
        for (i =0; i < TextLgth; i++)
        {
            MyBuffer[n] = text[i];
            Increment (&n);
        }
    }
    CursorPosn = n;
    return;
}                                                    /* end of FillinData */

/************************************************************************/
/*  FilloutData :        FILLOUTDATA                                    */
/*                                                                      */
/*           Move Input text from pos "n" or cursor if n = 0            */
/************************************************************************/
static  void    FilloutData ( n )
int     n;
{
    int     i;

    if (n == 0) n = CursorPosn; /* Use cursor          */
    else        n = n - 1;      /* Convert to Offset    */
    for (i =0; i < FIELDLN ; i++)
    {
        text[i] = MyBuffer[n];
        Increment  (&n);
    }
    text[i] = '\n';              /* insert CR charact  er*/
    i = i + 1 ;
    text[i] = '\0';                          /* insert end string  */
    CursorPosn = n;
    return;
}                                                   /* end of FilloutData */

/************************************************************************/
/*  MatchSessOwn :      MATCHSESSOWN                                    */
/*                                                                      */
/*     Wait to match session owner to equal input val}                 */
/*                                                                      */
/*     Possible states for sessions are defined as follows:             */
/*          0 = Undefined;      2 = Owned by SSCP;                      */
/*          1 = Unowned;        3 = Owned by LU.                        */
/************************************************************************/
static  void    MatchSessOwn ( i )
int     i;
{
    while  ( pimSCB.psOwner != i )
    {
        alarm(5);
        pause();
        alarm(0);
    }
    ClearScreen();
}                                                  /* end of MatchSessOwn*/

/************************************************************************/
/*  TabCursor :             TABCURSOR                                   */
/*                                                                      */
/*     Sets the cursor to the next available unprotected field         */
/************************************************************************/
static  void    TabCursor ( n )
int     n;
{
    int     i;
    int     posit;

    if (n == 0)  n = 1;            /* Ensure one tab               */
    for ( i = 0; i < n; i++)
    {
        if ((posit = FindNxtUnprot(CursorPosn)) != BUFF_SIZE)
        {
            Increment(&posit);
            CursorPosn = posit;
        }
    }
}                                                  /* end of TabCursor */

/************************************************************************/
/*  WaittoPush :        WAITTOPUSH                                      */
/*                                                                      */
/*  Waits for both an absence of communication (COMM) errors and a     */
/*  NotReceive state to be reached.  If the COMM check is not made,    */
/*  unexpected Host and PIM errors may result.                         */
/************************************************************************/
static  void    WaittoPush ( )
{
    while ((pimSCB.psProcState != prNRCV) || (pimSCB.psCommChk != 0))
    {
        alarm(5);
        pause();
        alarm(0);
        if ( pimSCB.psCommChk != 0 )
        {
            fprintf( stderr, "CommChk = %d\n",pimSCB.psCommChk );
            fflush( stderr );
        }
    }
}                                                  /* end of WaittoPush */

/************************************************************************/
/*  SendAIDKey :        SENDAIDKEY                                      */
/*                                                                      */
/*        Sends the AID key represented by "i" from script             */
/************************************************************************/
static  void    SendAIDKey ( i )
int     i;
{
    int     AIDKey;

    WaittoPush();

    switch ( i )    /* case for AID key val} assignment */
    {
            case    0:      AIDKey = aiENTER;
                            break;
            case    1:      AIDKey = aiPF1;
                            break;
            case    2:      AIDKey = aiPF2;
                            break;
            case    3:      AIDKey = aiPF3;
                            break;
            case   30:      AIDKey = aiCLEAR;
                            ClearScreen;
                            break;
            case   31:      AIDKey = aiPA1;
                            break;
            case   32:      AIDKey = aiPA2;
                            break;
            case   33:      AIDKey = aiPA3;
                            break;
            case   40:      AIDKey = aiSELPEN;
                            break;
            case   41:      AIDKey = aiTESTREQ;
                            break;
            default:        AIDKey = aiENTER;
                            break;
        }
/* The following AID key error should not occur if the WAITTOPUSH
   communication error check is completed successfully.           */
    if ( PostAIDKey ( SessionID, AIDKey ) == FAIL )
       fprintf ( stderr, AidError, errno, LU_Address );
}                                                /* end of SendAIDKey */

/*********************************************************************/
/*  SendSNAEvent :      SENDSNAEVENT                                 */
/*                                                                   */
/*        Send the SNA Event key represented by "i" from script      */
/*********************************************************************/
static  void    SendSNAEvent ( i )
int     i;
{
    int     EventKey;

     /* It is important to note each of the states and conditions that
        are required for each SNA key assignment.  For example, if a
        SYSREQ is iss}d, it is essential to verify the absence of any
        communication (COMM) errors. At the conclusion of the SNA event
        case statement, a check of current states is made.            */
    switch ( i )
    {
        case 1:         EventKey = scPSA;
                        WaittoPush();         /* not nRCV not Com */
                        break;

        case 2:         EventKey = scSYSREQ;  /* not CommChk      */
                        while (pimSCB.psCommChk != 0)
                        ;
                        ClearScreen ();
                        break;

        case 3:         EventKey = scATTN;    /* not DTR and SSCP */
                        while ( (pimSCB.psCommChk != 0) ||
                        (pimSCB.psProcState == prDTRESET) ||
                        (pimSCB.psOwner != owLU) )
                        ;
                        break;

        default:        EventKey = scPSA;
                        WaittoPush();
                        break;
    }

    /* Check for improper states */
    if ( PostSNAEvent ( SessionID, EventKey ) == FAIL )
       fprintf ( stderr, EventError, errno, LU_Address );
}                                              /* END OF SENDSNAEVENT */
```

```
/**********************************************************************/
/*  MatchData :         MATCHDATA                                     */
/*                                                                    */
/*                                                                    */
/**********************************************************************/
static  void    MatchData ( i )
int     i;
{
    int NotFound;
    int posit;

    NotFound = TRUE;

    /* This function calls GetBuf, and waits for GetBuf to complete.
       When GetBuf returns successfully, Screen data is compared to
       the input test data. If the Screen data does not match the
       input data, the process is repeated.                          */
    while ( NotFound == TRUE )
    {
        GetBuf();
        if ( CloseDown == TRUE )
        {
            NotFound = FALSE;
            break;
        }
        if (i == 0)                             /* Use cursor and backup*/
        {
            if ((posit = FindCurAttrib (CursorPosn)) != BUFF_SIZE)
            {
                Decrement(&posit);
                posit = FindCurAttrib (posit);
                Increment(&posit);
            }
            else posit = CursorPosn;
        }
        else posit = i - 1;
        if (strncmp(&MyBuffer[posit], text, TextLgth) == 0) NotFound = FALSE;
        else
        {
            alarm(2);
            pause();
            alarm(0);
        }
    }
    return;
}                                                   /* end of MatchData */

/**********************************************************************/
/*  PutBuf :            PUTBUF                                        */
/*                                                                    */
/*                Moves Mybuffer into ScreenBuffer.                   */
/*                                                                    */
/*  Waits for the PreHandler routine to set PutRequest to FALSE.      */
/*  This indicates that the PreHandler routine has moved "MyBuffer"   */
/*  into "ScreenBuffer".                                              */
/**********************************************************************/
static  void    PutBuf( )
{
    while (PutRequest == TRUE)
    {
        alarm(2);
        pause();
        alarm(0);
    }
}                                               /* end of PutBuf      */

/**********************************************************************/
/*  GetBuf :            GETBUF                                        */
/*                                                                    */
/*              Moves ScreenBuffer into MyBuffer                      */
/**********************************************************************/
static  int     GetBuf( )
{
    int     Retries;
    int     Secs;
    Secs    = 2;
    Retries = NUM_RETRIES;

    /* This function sets GetRequest to TRUE, then waits for the
    PostHandler routine to set GetRequest to FALSE; the PostHandler
    routine then moves "ScreenBuffer" into "MyBuffer". If GetRequest
    is not set to FALSE within a reasonable amount of time, a retry
    counter indicates a timeout.                          */

    GetRequest = TRUE;
    while (GetRequest == TRUE)
    {
        alarm(Secs);
        pause();
        alarm(0);
        if ( Retries == 0 ) /* Maximum number of retries reached */
        {
            fprintf( stderr, "TIMEOUT Screen data ");
            fprintf( stderr, "After %d SECONDS\n",
                            (Secs * NUM_RETRIES));
            printf("\nTimeout occurred during data transfer");
            DisplayData(0);
            GetRequest = FALSE;
            CloseDown = TRUE;
        }
        else Retries--;               /* count down retries  */
    }
}                                              /* end of GetBuf       */

/*************************************************************************/
/*  OpenAlt :           OPENALT                                      */
/*                                                                   */
/*              Open AIX output file.                                */
/*************************************************************************/
static  void    OpenAlt ( i )
int     i;
{
    cpyblk ( altfn, text, LINELEN );
    if ((altscriptp = fopen (altfn, "w")) == NULL)
    {
        printf ("Open error AIX Output file \n");
        Close3270(SessionID);
        exit (exKILLSTRT );
    }
    mainscriptp = scriptp;
    scriptp = altscriptp;
    Reps = i;
    altope = 1;
}                                               /* end of OpenAlt      */

/*************************************************************************/
/*  CloseAlt :          CLOSEALT                                     */
/*                                                                   */
/*            Close and possibly reopen alternate script.            */
/*************************************************************************/
static  void    CloseAlt ( )
{
    scriptp = altscriptp;
    fclose(altscriptp);
    Reps--;
    if ( Reps == 0 )
    {
        SuppressLog = FALSE;
        scriptp = mainscriptp;
        altope = 0;
    }
    else
    {
        fprintf(stderr,"********************");
        fprintf(stderr,"Log suppressed AIX output file Open");
        fprintf(stderr,"********************\n");
        SuppressLog = TRUE;
        if ((altscriptp = fopen (altfn,"r")) == NULL)
        {
            printf ("Open error AIX output file \n");
            Close3270(SessionID);
            exit (exKILLSTRT );
        }
    }
}                                                   /* end of CloseAlt    */

/**********************************************************************/
/*  AcceptAndSendl :    ACCEPT AND SEND LOGON DATA ( FILE .LUn )      */
/*                                                                    */
/*  Gets input data and AID key from the Logon file and sends data.   */
/**********************************************************************/
static  void    AcceptAndSendl ( )
{
    int     cmnd, nval;
    char    *temp;

    while (TRUE)
    {
        /* If there is an error when reading the Logon file, EINTR will
        catch the alarm and post it to errno; a reread is then tried. */
        while (((temp = fgets ( InputData, LINELEN, scriptp)) == NULL)
             && ( errno == EINTR ));
        if ((temp == NULL) && ( errno != EINTR ))
        {
            fprintf(stderr, "pimrtdn: EOF or error, errno = %d\n", errno);
            fflush(stderr);
            CloseDown = TRUE;
            return;
        }
        cmnd = 99;
        nval = 0;
        text[0] = 0;

        /* Break input line down */
        if ((sscanf(InputData, "%d %d %[¬\n]",&cmnd,&nval,text)) > 3)
        {
            fprintf(stderr, "pimrtdn: scan error, errno = %d\n", errno);
            continue;
        }
        TextLgth = ( strlen  ( text ) );
        break;
    }
    Count = Count + 1;
    if ( SuppressLog == FALSE )
    {
        fprintf(stderr,"# %d:cmnd=%d:nval=%d:text '%s'\n",
            Count,cmnd,nval,text);
        fflush(stderr);
    }
    if ( Silent == FALSE )
    {
        printf("%d ",Count);  /* Used for display to screen when running */
                              /* in background mode                      */
    }

    switch ( cmnd )
    {
        case 1:  PutRequest = TRUE;
                 SendAIDKey(nval);          /*  Send AID key,        */
                 PutBuf();                  /*  then move buffer     */
                 break;

        case 2:  SendSNAEvent(nval);        /*  Send SNAEvent out    */
                 break;

        case 3:  GetBuf();                  /* Read screen into      */
                 break;                     /*     MyBuffer          */

        case 10: FillinData(nval);          /*  Move data in         */
                 SetMDT ( 0 );              /*  Set MDT bits on      */
                 break;

        case 11: SetMDT (nval);             /* Set MDT bits on       */
                 break;                     /* at cursor-field or    */
```

```
                                                  /* nval position          */

        case 12: CursorPosn = nval;               /* Set Cursor position    */
                 break;

        case 13: CursorPosn = BUFF_SIZE - 1;      /* "home" cursor          */
                 TabCursor(1);
                 break;

        case 14: TabCursor(nval);                 /* Move Cursor to next    */
                 break;                           /* unprotected field      */
                                                  /* nval times             */

        case 20: MatchSessOwn(nval);              /* Match Session Owner    */
                 break;                           /* to equal nval. Val}s: */
                                                  /* 0 = undefined          */
                                                  /* 1 = unowned            */
                                                  /* 2 = SSCP               */
                                                  /* 3 = LU                 */

        case 21: MatchData(nval);                 /* Match screen buffer    */
                 break;                           /* Data to file data      */
                                                  /* at "nval" position;    */
                                                  /*if nval=0, then match to*/
                                                  /* the data at the 1st    */
                                                  /* char of the prior field*/
                                                  /* from cursor position   */

        case 80: OpenAlt(nval);                   /* Open AIX output file   */
                 break;

        case 89: CloseAlt();                      /* Close AIX output file  */
                 break;                           /* this must be the last  */
                                                  /* command in any ALT file*/

        case 90: DisplayData (nval);              /* record to log          */
                 break;

        case 91: ShowStatus(stderr);              /*store the current status*/
                 break;                           /* of the terminal on log */

        case 92: alarm (nval);                    /* loop wait for nval secs*/
                 pause();
                 alarm(0);
                 break;

        case 98: CloseDown = TRUE;                /* shutdown time          */
                 break;

        case 99: if ( ( nval == 1 ) && ( Count == 1 ) )
                     {                            /* if this is the first   */
                     Silent = FALSE;              /* file record            */
                     printf("%d ",Count++);
                     }

                 if ( Silent == FALSE )
                     {
                     printf("%s ",text);          /* then toggle on display */
                     }                            /* for backround mode     */
                 break;

        default: fprintf(stderr, "Input command error %d %d %s",
                 cmnd, nval, text);
                 break;
    }
}                                                 /* end of AcceptAndSend1  */

/***********************************************************************/
/*  AcceptAndWrite :    ACCEPT AND WRITE TRANSFER DATA ( FILE .UPD )    */
/*                                                                     */
/*  Gets input data from screen and writes it to disk                  */
/***********************************************************************/
static  void    AcceptAndWrite ( )
    int     finprog;
    int     mrec, nrec;
    int     templ;
    char    *temp;
    mrec = 10;          /* mrec = maximum nb of records per screen  */

    finprog=FALSE;
    while (finprog==FALSE)
    {
        GetBuf();
        for (nrec = 0; (nrec < mrec)&&(finprog==FALSE); nrec++)
        {
            /* If there is an error when writing the output file, EINTR will
            catch the alarm and post it to errno; rewrite is then tried.  */
            FilloutData (0);            /* move data out                */.
            TabCursor (1);              /* move cursor next unprot field */

            /* put text from output record */
            templ =  strncmp ( text, "EOF", 3 );
           if ( templ != NULL )
           {
               fputs ( text, scriptp );
               Enrnb = Enrnb + 1;
           }
           else
           {
               fprintf(stderr,"pimrtdn: EOF\n");
               fflush(stderr);
               finprog = TRUE;
           }
           Count = Count + 1;
           if ( SuppressLog == FALSE )
           {
               fprintf(stderr,"# %d:text '%s'\n",Count,text);
               fflush(stderr);
           }
           if ( Silent == FALSE )
           {
               printf("%d ",Count);  /* Used for display to the screen  */
                                     /* when running in background mode */
           }
       }
       PutRequest = TRUE;              /* get screen from Host by     */
       SendAIDKey(0);                  /* depressing the ENTER key    */
       PutBuf();
   }
   CloseAlt();
}/*end_procedure*/                     /* end of AcceptAndWrite       */

/***********************************************************************/
/*  ErrHandler :              ERROR HANDLER                            */
/*                                                                     */
/*                   Error handling function for PIM.                  */
/* The error handler function is specified via the InitPIM call and is */
/* used when PIM encounters a fatal error.  PIM executes this function */
/* before shutting down, enabling the PIM test program to close down   */
/* cleanly.                                                            */
/***********************************************************************/
static int  ErrHandler (errorcode)
int errorcode;
{
    fprintf(stderr,
        "pimrtdn: ErrHandler called with errorcode %d and errno %d\n",
         errorcode, errno);
    CloseDown = TRUE;                          /* fatal error         */
    return(SUCCEED);
}                                              /* end of ErrorHandler */

/***********************************************************************/
/*  PreHandler :              PRE HANDLER                              */
/*                                                                     */
/*                    Pre exit handler for PIM.                        */
/***********************************************************************/
static  int    PreHandler ( sd, msg_type, status_blk )
int     sd;                 /*  Session descriptor.        */
int     msg_type;           /*  Message type.              */
```

```
PIMCtlBlk *status_blk;          /*  Pointer to status block.    */
{
    if ( ( PutRequest == TRUE ) && ( pimSCB.psProcState == prNRCV ) )
    {
        PutRequest = FALSE;
        cpyblk ( ScreenBuffer, MyBuffer, BUFF_SIZE);
        pimSCB.psCursor = CursorPosn;       /* give local cursor   */
    }                                     /* position to PIM     */
    return(SUCCEED);
}                                                /* end of PreHandler   */

/**********************************************************************/
/*  PostHandler :            POST  HANDLER                           */
/*                                                                    */
/*                     Post exit handler for PIM.                     */
/**********************************************************************/
static  int     PostHandler ( sd, msg_type, status_blk )
int     sd;                     /*  Session descriptor.       */
int     msg_type;               /*  Message type.             */
PIMCtlBlk *status_blk;          /*  Pointer to status block.  */
{
    extern  PIMCtlBlk pimSCB;
/*
    switch(msg_type)
    {
      case UEERROR:
                  fprintf(stderr,"pimrtdn: ***** Received }ERROR\n");
                  break;
      case UESTATUS:
                  fprintf(stderr,"pimrtdn: ***** Received }STATUS\n");
                  break;
      case UEREAD:
                  fprintf(stderr,"pimrtdn: ***** Received }READ\n");
                  break;
      case UEWRITE:
                  fprintf(stderr,"pimrtdn: ***** Received }WRITE\n");
                  break;
      default:
                  fprintf(stderr,"pimrtdn: ***** Received %d\n",
                                                        msg_type);
    }
*/

/* Before acting on a req}st to get a buffer, it must first be ensured
   that the Keyboard is not locked and that HostWait is not locked. Once
   these conditions are met, buffers can safely be manipulated.    */

    if ( ( ( ( GetRequest == TRUE ) && ( pimSCB.psKBLock != TRUE ) )
                             && ( pimSCB.psHWLock != TRUE ) ) )
    {
        GetRequest = FALSE;
        cpyblk ( MyBuffer, ScreenBuffer, BUFF_SIZE );
        CursorPosn = pimSCB.psCursor;
    }
    /* Whenever PIM invokes the PostHandler, the PIM application must
       check to see if PIM has req}sted a Close.                    */

    if (pimSCB.psState == stCLOSED) CloseDown = TRUE; /* Are we still up? */
    return ( SUCCEED) ;
}                                                 /* end of PostHandler */

/****************************************************
 *               MAIN PROGRAM                      *
 *                                                 *
 *   The main processing function begins here....  *
 ***************************************************/

/* The PIM application will be called with the following set of parameters
   (some of the parameters are used internally and are not relevant to the
   PIM programmer):

                        argc - count
                        argv [1]  LU address
                        argv [2]  emulation type
                        argv [3]  device handler process ID
                        argv [4]  start process ID
                        argv [5]  write signal from device handler*/
main ( argc, argv )
int     argc;
char    *argv[];
{
    char    input[LINELEN];

    /* Once the LU address is defined, a "suffix" is assigned to
       permit the use of a different set of input and log files for
       individual LUs.  Logging and Output files are then opened.  */

    LU_Address = atoi ( argv [ 1 ] );
    (void) strcpy ( LuSuffix, argv [ 1 ] );
    (void) strcat( input, LogFilePrefix );
    (void) strcat( input, LuSuffix );

    (void) freopen ( input, "w", stderr );

    input[0] = 0;

    (void) strcat( input, ScrFilePrefix );
    (void) strcat( input, LuSuffix );

    if ((mainscriptp = fopen (input, "r")) == NULL)
    {
        printf ("Open error Logon file %s\n",input );
        exit (exKILLSTRT );
    }
    fprintf (stderr,"Input %s opened \n",input);

  /* This example of InitPIM passes the arguments on to PIM. It specifies
     that PIM is to catch any signals that come through from the device
     handler; that the application is non-interactive and will run in
     backround mode; and specifies the name of the fatal error handler
     as "ErrHandler".                                                 */

  /* Should a PIM application encounter serious problems,the PIM program
     must set an exit status to inform the other components of the IX
     3270-PLUS product of this fact. The exKILLSTRT is defined within
     pim3270.h.                                                       */
    if ( InitPIM ( argc, argv, fCatch, FALSE, ErrHandler ) == FAIL )
    {
        fprintf (stderr,"pimrtdn: Error -%d- initializing PIM.\n", errno );
        exit ( exKILLSTRT );            /* Tell install to tell start pim */
    }

    /*   Set signals */
    if ( ( (int) signal(SIGHUP, CloseItDown) ) == FAIL )
    {
        fprintf(stderr,"pimrtdn: Error %d in setting signal SIGHUP \n",errno);
        exit ( exKILLSTRT );            /* Tell install to tell start pim */
    }

    if ( ( (int) signal(SIGINT, CloseItDown) ) == FAIL )
    {
        fprintf(stderr,"pimrtdn: Error %d in setting signal SIGINT \n",errno);
        exit ( exKILLSTRT );            /* Tell install to tell start pim */
    }

    if ( ( (int) signal(SIGALRM, AlrmCatcher) ) == FAIL )
    {
        fprintf(stderr,"pimrtdn: Error %d in setting signal SIGALRM \n",errno);
        exit ( exKILLSTRT );            /* Tell install to tell start pim */
    }

    /*Note that we must ignore the SIGILL signal since it might be sent*/
    /*by the Device Handler to determine if the application is running.*/
    if ( ( (int) signal(SIGILL, SIG_IGN) ) == FAIL )
    {
        fprintf(stderr,"pimrtdn: Error %d in setting signal SIGILL \n",errno);
        exit ( exKILLSTRT );            /* Tell install to tell start pim */
    }

    /*  Open a session for our Host address.                         */
    fprintf(stderr,"\npimrtdn: Opening 3270 session\n");

    SessionID = Open3270 ( LU_Address, dtTERMINAL, BUFF_SIZE, &pimSCB );
    if ( SessionID == FAIL )
    {
        fprintf (stderr,"pimrtdn: Error -%d- opening LU#%d.\n",
             errno, LU_Address );
        exit ( exKILLSTRT );            /* Tell install to tell start pim */
    }
    else
    /* Before processing any data, the application must pause until the
       emulator is marked open.                                      */
       while ( pimSCB.psState != stOPEN )  pause ( );

    /*************************** Main Loop *************************/
    scriptp = mainscriptp;               /* running main script */

    while (!CloseDown)
    {
        if (altope == 0) AcceptAndSend1();    /* read user Logon Input file
                                                 for this LU and then
                                                 execute it             */
        else AcceptAndWrite();               /* read screen and write
                                                 records to disk        */
    }

    /*  Close down this logical unit.    */
    printf("\nPimrtdn finished");
    printf("\n%d records transferred\n", Enrnb);
    ShutDown ( );
    exit(0);
}                                        /*      end of main        */
```

Appendix F. WHIP Profiles and Programs

S/370 Host Profiles

```
AIXDV780 IODEVICE UNIT=3278,ADDRESS=780,                            *
                  MODEL=2,                                          *
                  FEATURE=(EBKY3277,KB78KEY,AUDALRM,                *
                  DOCHAR,SELPEN,PTREAD),OFFLINE=NO
AIXDV781 IODEVICE UNIT=3279,ADDRESS=781,                            *
                  MODEL=2A,                                         *
                  FEATURE=(EBKY3277,KB78KEY,AUDALRM,                *
                  DOCHAR,SELPEN,PTREAD),OFFLINE=NO
AIXDV782 IODEVICE UNIT=3278,ADDRESS=782,                            *
                  MODEL=3,                                          *
                  FEATURE=(EBKY3277,KB78KEY,AUDALRM,                *
                  DOCHAR,SELPEN,PTREAD),OFFLINE=NO
AIXDV783 IODEVICE UNIT=3279,ADDRESS=783,                            *
                  MODEL=3B,                                         *
                  FEATURE=(EBKY3277,KB78KEY,AUDALRM,                *
                  DOCHAR,SELPEN,PTREAD),OFFLINE=NO
```

Figure 226. Sample IODEVICE Macros for MVS

```
AIXDV780 RDEVICE DEVTYPE=3278,ADDRESS=780,                          *
                  MODEL=2
AIXDV781 RDEVICE DEVTYPE=3279,ADDRESS=781,                          *
                  MODEL=2
AIXDV782 RDEVICE DEVTYPE=3278,ADDRESS=782,                          *
                  MODEL=3
AIXDV783 RDEVICE DEVTYPE=3279,ADDRESS=783,                          *
                  MODEL=3
```

Figure 227. Sample RDEVICE Macros for VM

```
AIXCT770 RCTLUNIT ADDRESS=770,UNIT=3274,                            *
                  FEATURE=16-DEVICE
AIXCT780 RCTLUNIT ADDRESS=780,UNIT=3274,                            *
                  FEATURE=32-DEVICE
```

Figure 228. Sample VM RCTLUNIT, 16-Port 5088-1 and 32-Port 3274

```
        LBUILD
AIXDV780 LOCAL TERM=3277,CUADDR=780,                                          *
               ISTATUS=ACTIVE,FEATUR2=(EDATS,ANKEY,                           *
               MODEL2,PFK,SELPEN),LOGTAB=INTTAB,                              *
               MODETAB=RSTINCLM,DLOGMOD=MOD2,                                 *
               USSTAB=USST327X,LOGAPPL=NETMON

AIXDV781 LOCAL TERM=3277,CUADDR=781,                                          *
               ISTATUS=ACTIVE,FEATUR2=(EDATS,ANKEY,                           *
               MODEL2,PFK,SELPEN),LOGTAB=INTTAB,                              *
               MODETAB=RSTINCLM,DLOGMOD=MOD2,                                 *
               USSTAB=USST327X,LOGAPPL=NETMON

AIXDV782 LOCAL TERM=3277,CUADDR=782,                                          *
               ISTATUS=ACTIVE,FEATUR2=(EDATS,ANKEY,                           *
               MODEL2,PFK,SELPEN),LOGTAB=INTTAB,                              *
               MODETAB=RSTINCLM,DLOGMOD=MOD3,                                 *
               USSTAB=USST327X,LOGAPPL=NETMON

AIXDV783 LOCAL TERM=3277,CUADDR=783,                                          *
               ISTATUS=ACTIVE,FEATUR2=(EDATS,ANKEY,                           *
               MODEL2,PFK,SELPEN),LOGTAB=INTTAB,                              *
               MODETAB=RSTINCLM,DLOGMOD=MOD3,                                 *
               USSTAB=USST327X,LOGAPPL=NETMON
```

Figure 229. Sample VTAM Local non-SNA Major Node

```
RSTINCLM MODETAB

* 3270 NON-SNA WITH EXTENDED DATA STREAM & MODEL 2 SCREEN:
*
*        PRIMARY   = 24 LINES X 80 COLUMNS (1920 BYTES)
*        SECONDARY = 24 LINES X 80 COLUMNS (1920 BYTES)

MOD2     MODEENT LOGMODE=MOD2,FMPROF=X'02',                            *
               TSPROF=X'02',PRIPROT=X'71',                             *
               SECPROT=X'40',COMPROT=X'2000',                          *
               RUSIZES=X'0000',                                        *
               PSERVIC=X'008000000000185018507F00'

* 3270 NON-SNA WITH EXTENDED DATA STREAM & MODEL 3 SCREEN:
*
*        PRIMARY   = 24 LINES X 80 COLUMNS (1920 BYTES)
*        SECONDARY = 32 LINES X 80 COLUMNS (2560 BYTES)

MOD3     MODEENT LOGMODE=MOD3,FMPROF=X'02',                            *
               TSPROF=X'02',PRIPROT=X'71',                             *
               SECPROT=X'40',COMPROT=X'2000',                          *
               RUSIZES=X'0000',                                        *
               PSERVIC=X'008000000000185020507F00'

* 3270 NON-SNA WITH EXTENDED DATA STREAM & MODEL 4 SCREEN:
*
*        PRIMARY   = 24 LINES X 80 COLUMNS (1920 BYTES)
*        SECONDARY = 43 LINES X 80 COLUMNS (3440 BYTES)

MOD4     MODEENT LOGMODE=MOD4,FMPROF=X'02',                            *
               TSPROF=X'02',PRIPROT=X'71',                             *
               SECPROT=X'40',COMPROT=X'2000',                          *
               RUSIZES=X'0000',                                        *
               PSERVIC=X'00800000000018502B507F00'

* 3270 NON-SNA WITH EXTENDED DATA STREAM & MODEL 5 SCREEN:
*
*        PRIMARY   = 24 LINES X 80 COLUMNS (1920 BYTES)
*        SECONDARY = 27 LINES X 132 COLUMNS (3564 BYTES)

MOD4     MODEENT LOGMODE=MOD5,FMPROF=X'02',                            *
               TSPROF=X'02',PRIPROT=X'71',                             *
               SECPROT=X'40',COMPROT=X'2000',                          *
               RUSIZES=X'0000',                                        *
               PSERVIC=X'00800000000018501B847F00'

*
* 3287 Printer
*
T3287    MODEENT LOGMODE=T3287,FMPROF=X'02',                           *
               TSPROF=X'02',PRIPROT=X'71',                             *
               SECPROT=X'40',COMPROT=X'2000',                          *
               RUSIZES=X'0000',                                        *
               PSERVIC=X'008000000000000000000200'
*
         MODEEND
         END
```

Figure 230. Sample MODETAB Table Entries for 3270-2, 3, 4, 5 and 3287

WHIP API Sample Program (sendkeys.c)

```
/* ******************************************************* */
/*  sendkeys.c sends keystrokes from the AIX system to      */
/*  the 3270 session screen and emulates the 3270 ENTER key */
/*  Program written by: Johnny Lauridsen, IBM Denmark       */
/* ******************************************************* */
/*  Compile/link with: cc -o sendkeys sendkeys.c -lg3270    */
/*  Run program with: sendkeys arg1 arg2                    */
/*  Up to two arguments can be specified                    */
/*  Example: sendkeys query time                            */
/*  Example: sendkeys exec jobs                             */
/* ******************************************************* */

#include "g32_api.h"
#include <g32_keys.h>

/* ----------------*/

main( argc, argv)
int  argc;
char *argv[];
{
    struct g32_api asx;           /* API information structure       */
    register struct g32_api *as;  /* working pointer to asx          */
    register int rc;              /* return codes from API functions */
/* ******************************************************* */
/*  Step 1.                                                 */
/*  Establish a logical path to the host with g32_open()    */
/* ******************************************************* */
    as = &asx;
    rc = g32_open(as, 0, 0, 0);
    if (rc < 0)
    {
        printf("g32_open returned: rc = %d\n", rc);
        printf("                   as->errcode = %d\n", as->errcode);
        printf("                   as->xerrinfo = %d\n", as->xerrinfo);
        printf("The WHIP host API is not functional.\n");
        return(0);
    }

/* ******************************************************* */
/*  Step 2.                                                 */
/*  Establish a session with the host using g32_alloc()     */
/* ******************************************************* */

    rc = g32_alloc(as, "", MODE_3270);
    if (rc < 0)
    {
        printf("g32_alloc returned: rc = %d\n", rc);
        printf("                    as->errcode = %d\n", as->errcode);
        printf("                    as->xerrinfo = %d\n", as->xerrinfo);
        g32_close(as);
        printf("The WHIP host API is not functional.\n");
        return(0);
    }
/* ******************************************************* */
/*  Step 3.                                                 */
/*  Send the actual keystrokes using g32_send_keys()        */
/* ******************************************************* */

    rc = g32_send_keys(as,argv[1]);
    if ( argc > 2 )
        {
        rc = g32_send_keys(as," ");
        rc = g32_send_keys(as,argv[2]);
        }
    rc = g32_send_keys(as,ENTER);
/* ******************************************************* */
/*  Step 4                                                  */
/*  Deallocate the session using g32_dealloc()              */
/* ******************************************************* */

    rc = g32_dealloc(as);
    if (rc < 0)
    {
        printf("g32_dealloc returned: rc = %d\n", rc);
        printf("                    as->errcode = %d\n", as->errcode);
        printf("                    as->xerrinfo = %d\n", as->xerrinfo);
        g32_close(as);
        printf("The WHIP host API is not functional.\n");
        return(0);
    }

/* ********************************************************** */
/*  Step 5                                                    */
/*  Close the logical path to the host using g32_close()      */
/* ********************************************************** */

    rc = g32_close(as);
    if (rc < 0)
    {
        printf("g32_close returned: rc = %d\n", rc);
        printf("                    as->errcode = %d\n", as->errcode);
        printf("                    as->xerrinfo = %d\n", as->xerrinfo);
        printf("The WHIP host API is not functional.\n");
        return(0);
    }

    printf("Send to host: %s %s\n", argv[1], argv[2]);
    return(0);
}

/* Entries for the compile/link - /usr/lib/libg3270.a */
g32_logon() {}
g32_logoff() {}
```

Appendix G. Local Area Networks

A local area network is a data transmission system that allows a number of independent devices to communicate with each other. It is distinguished from other types of networks in that communications are generally confined to a moderate sized geographical area such as a building or campus. The main characteristics of a Local Area Network are:

- Speeds in the megabit/second range.
- Low error rates.
- Single Owner network
- Users are in the same organization or company.

This should be contrasted with the typical Wide Area Network in which longer distances are involved, often with different companies owning the transmission media, and which may be used as public utilities by many different organizations.

Token-Ring Network

A ring is a closed path over which data travels, as in a loop. A station which resides on the circumference of this path receives data sent from its Nearest Active Upstream Neighbour (NAUN) and copies the data from the medium to its internal memory if it is the intended recipient.

The star-wired ring topology incorporates star-like characteristics with ring characteristics. That is, in a star-wired ring, data flows on a path which is circular, but the devices are connected at concentration points (called wiring concentrators) where access to the network is controlled. The wiring concentrators are normally located in wiring closets, so that the network acquires the shape of a star. The cable paths from the physical location of the work stations to the wiring concentrators actually implement two half circuits, using two wire pairs. These paths (called lobes) are terminated in the wiring concentrators by means of electromechanical circuitry that permits the control of access to the ring. The diagram below shows four devices connected to a wiring concentrator. Note that although the topology is star-like, the actual data path is a ring.

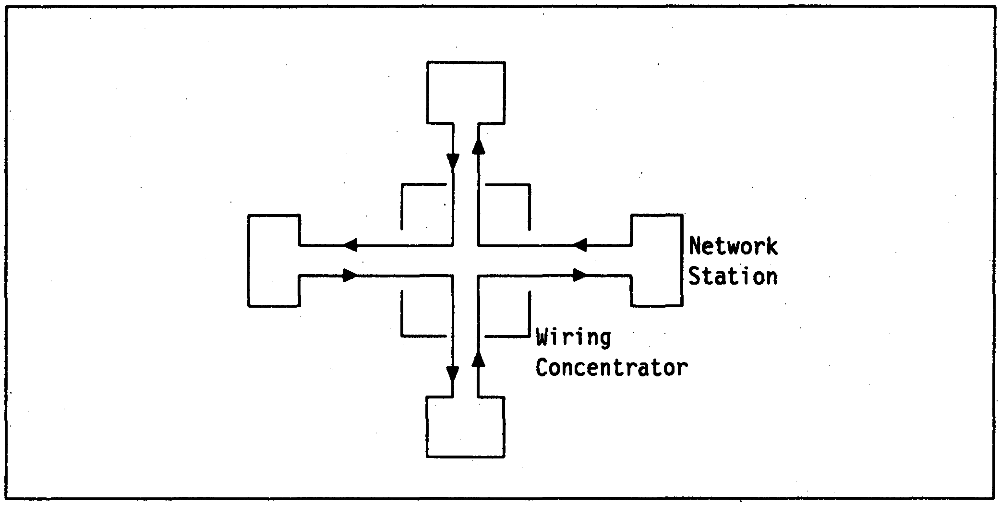


Figure 231. Star Wired Ring

Several of these wiring concentrators (each with devices on them) can be chained together to form a larger ring. The wiring concentrator for the IBM Token-Ring is called the 8228 Multi-Station Access Unit (MAU).

Protocols

In order to communicate over a Token-Ring network, a set of protocols is needed. The ones in general use are called the 802 protocols and were developed by a standards body known as the Institute of Electrical and Electronic Engineers (IEEE). Similar to the SNA approach, IEEE created a reference model with two layers which corresponded to the lowest two layers of the SNA seven-layer model, the data link layer and the physical layer. In the IEEE model, however, the data link layer is further divided into two sub-layers:

- Logical Link Control
- Medium Access Control.

The diagram below shows the relationship between the the IEEE model and the SNA layer structure:

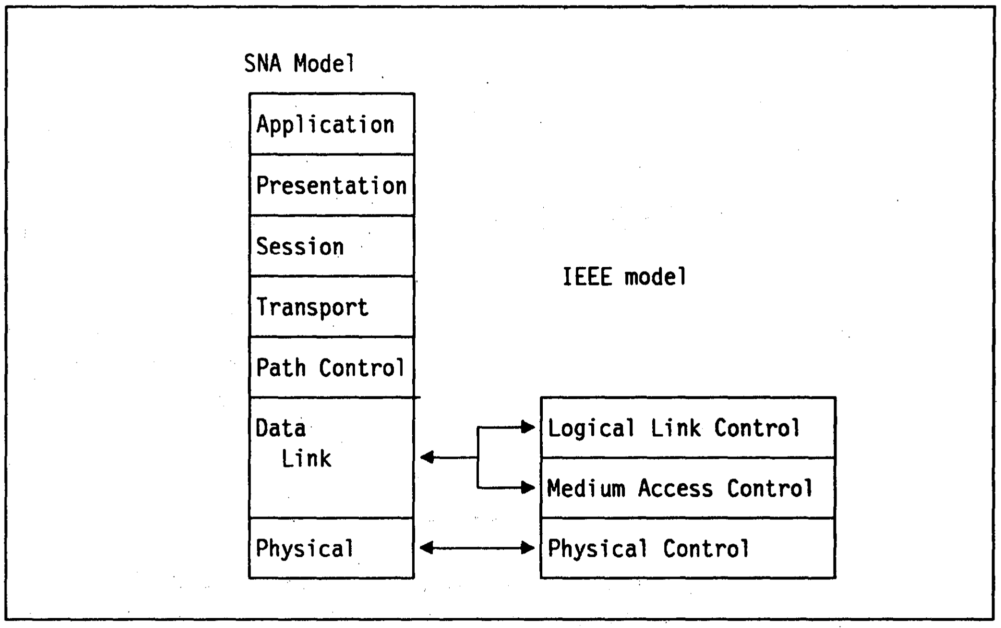


Figure 232. SNA and IEEE Model Relationship

Logical Link Control Sub-layer

The logical link control (LLC) sub-layer is the common sub-layer that provides services to layers above the data link layer, isolating them from the peculiarities of the various medium access methods and from the medium itself. LLC deals with the peer to peer protocols for the transfer of information and flow control among stations.

There may be several different LLC entities running from the same station on the network (for example, a station may be communicating with two other, different stations on the network simultaneously), and so each of these entities communicating at the LLC level are identified with a value or address that is unique to each LLC that is operating on that station. These entities are known as ***Service Access Points*** (SAP's) and are used by the higher layers to obtain services from the LLC and lower layers.

It is important to differentiate between SAP addresses, used at the LLC level, and the physical node or ring station addresses, used at the Medium Access Control (MAC) Level. These MAC addresses are covered below.

Medium Access Control Sub-layer and Physical Layer

The Medium Access Control (MAC) sub-layer is concerned with the procedures that stations must follow in order to have access to the common transport medium, so that conflicting (simultaneous) transmissions cannot occur, or if they occur, the stations detect the error and the transmissions are retried.

At the MAC level, addresses are used to identify the ring stations or nodes physically attached to the ring. According to the standard, these addresses can be two or six bytes in length. In IBMs implementations, address are six bytes long.

Hardware Requirements

Figure 233 illustrates the components used, when connecting IBM RTs to a Token-Ring Network. Figure 233

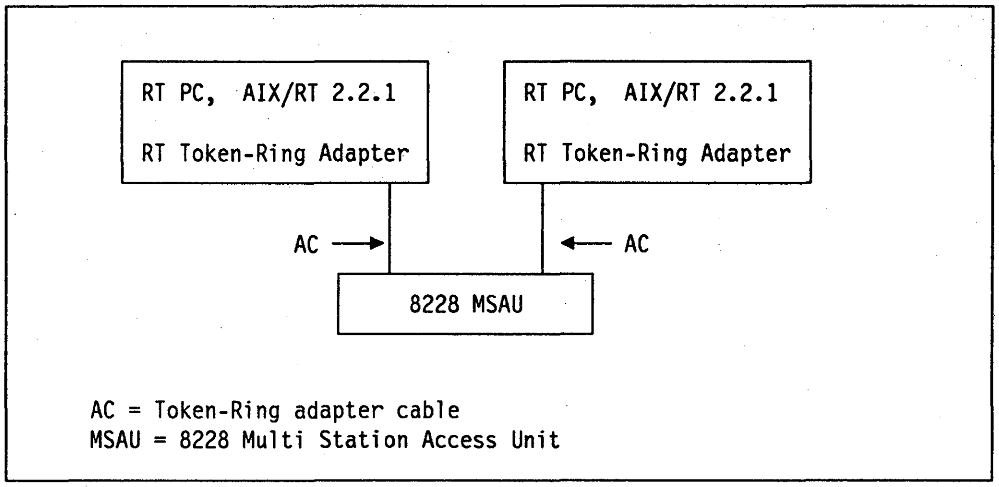


Figure 233. IBM RT Token-Ring Hardware

The IBM RT Token-Ring Adapter is unique to the IBM RT. It is ***not*** the same adapter as is used in the PC/AT.

IBM RTs attach to a Token-Ring network through an IBM 8228 Multi Station Access Unit. (This unit is often referred to as an MSAU or a MAU). Each IBM RT needs an IBM RT Token-Ring Adapter and a Token-Ring Adapter cable for attachment to the MSAU. An MSAU has ten sockets. The first socket is labeled RI (Ring In.). The other eight sockets, labeled 1 through 8, are the ports used for attaching systems. The last socket is RO (Ring Out). More than eight systems may be attached to the Token-Ring Network by introducing additional MSAUs and chaining the units together via cables between RO and RI. Token-Ring allows up to 260 devices in a single ring.

Host Connection Using Token-Ring

There are a number of different ways to connect an IBM RT to a System/370 host using Token-Ring network. The following sections describe the "37xx gateway" and the "3174 gateway". The IBM RT can also communcicate with an IBM 9370 connected to the Token-Ring network.

IBM 37xx Gateway

The IBM 37xx communication controller has the facility to connect to a Token-Ring with a special Token-Ring card, the MAC address of which can be programmed in by the operator of the IBM 37xx. This allows other stations on the Token-Ring to connect to the host via the IBM 37xx just as if they were using an ordinary telephone line except that they will get faster response times. The IBM RT can connect to the IBM 37xx network controller via the Token-Ring as illustrated in Figure 234 on page 527.

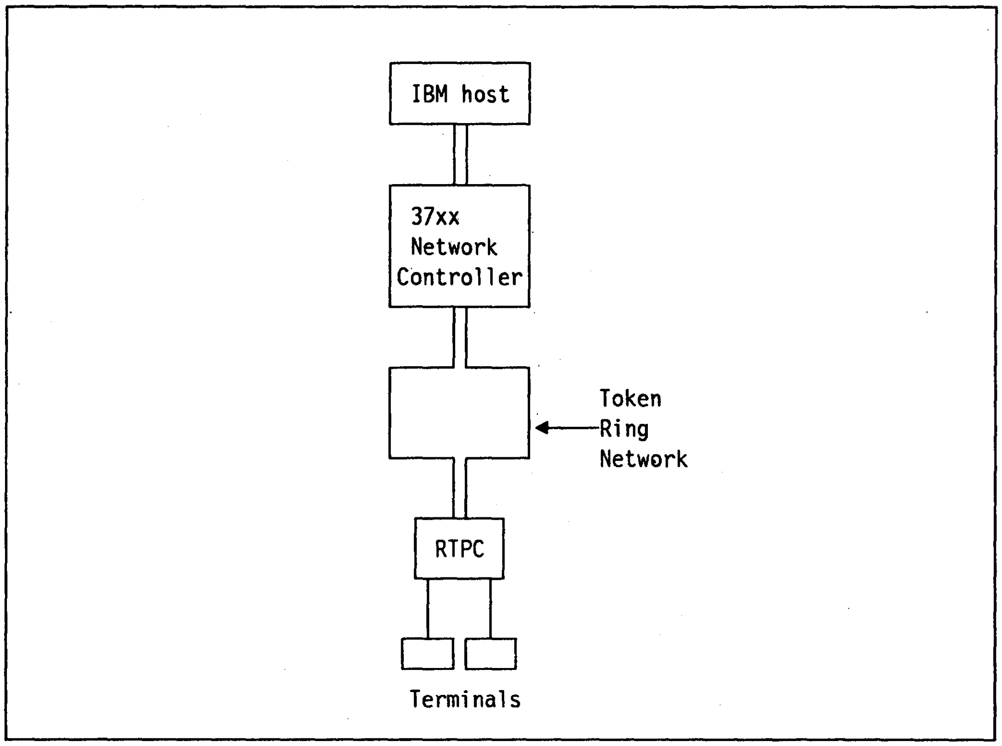


Figure 234. IBM RT Connected via Token-Ring to Mainframe Host

IBM 3174 Gateway

An IBM 3174 cluster controller, which is connected via a leased line (or local channel attached) to a System/370 host, may simultaneously be connected to a Token-Ring network. This allows other stations on the ring (such as IBM RT's) to connect to the host via the IBM 3174 cluster controller. To the host, the other stations on the Token-Ring appear as though they are multi-dropped on the same leased line as the IBM 3174. The IBM RT can connect to the IBM 3174 via Token-Ring as illustrated in Figure 235 on page 528.

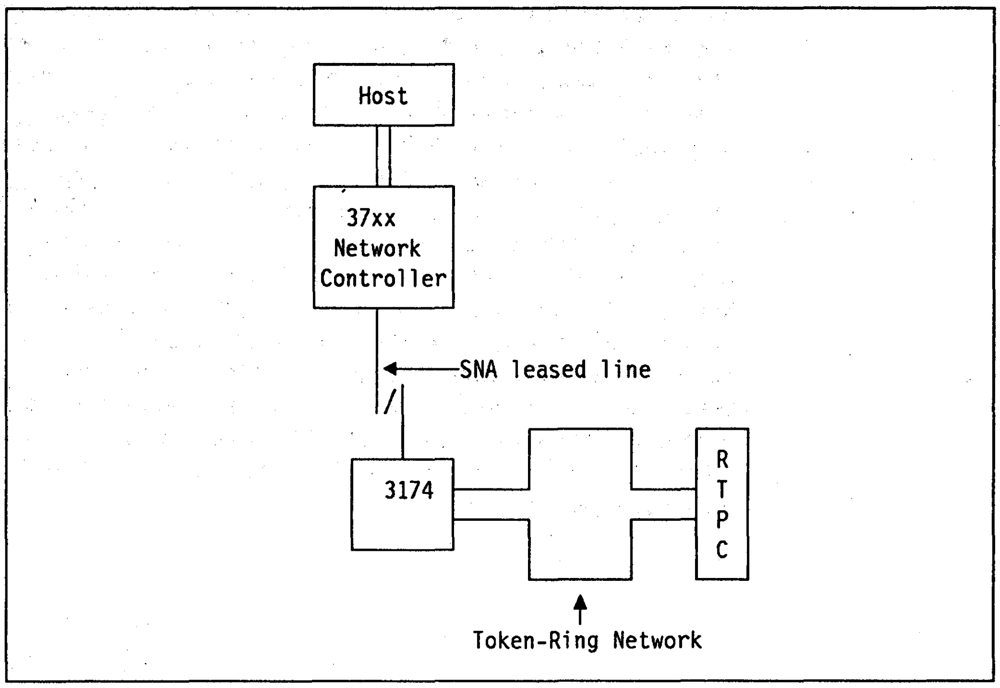


Figure 235. IBM RT Connected to an IBM 3174 over Token-Ring

Ethernet Local Area Network

A simple IBM RT Ethernet configuration is illustrated in Figure 236.

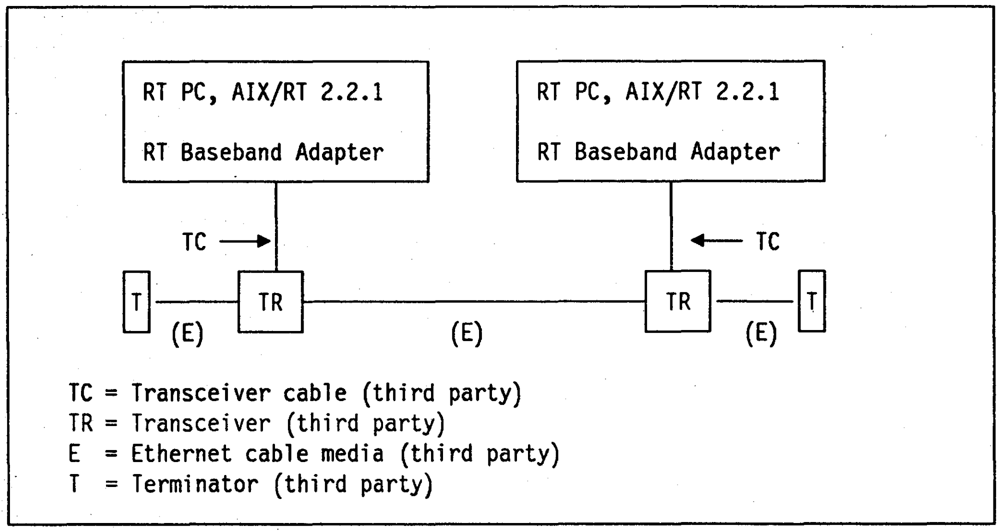

RT PC, AIX/RT 2.2.1
RT Baseband Adapter
RT PC, AIX/RT 2.2.1
RT Baseband Adapter
TC
TC
T
TR
TR
T
(E)
(E)
(E)
TC = Transceiver cable (third party)
TR = Transceiver (third party)
E = Ethernet cable media (third party)
T = Terminator (third party)

Figure 236. Ethernet Hardware

Hardware Requirements

The IBM RT attach to an Ethernet network using the IBM RT Baseband Adapter as illustrated in Figure 236. The IBM customer is responsible for network design and acquisition of the non-IBM hardware involved in setting up the network. As far as the network cable media is concerned, you can choose to implement either standard Ethernet 50-ohm coaxial cable or 50-ohm RT-58A/U thin coaxial cable. The transmission rate for both types is 10 Megabits per second, but they have different configuration restrictions and specifications. The remaining hardware and cabling requirements are dependent upon the actual network media chosen. Not only must the transceiver be compatible with the Ethernet network type, it must also have the appropriate tap or connector required by the media. A variety of connectors on transceiver cables are available. The cable must, of course, have one connector that is compatible with the transceiver being used. The other connector must be compatible with the RT PC Baseband Adapter. The RT Baseband Adapter will not accept the "sliding lock" that is used by some Ethernet cabling vendors. The cable is secured to the adapter with screws on the cable side. (This is consistent with the majority of the cables used on the IBM RT, they have screw or thumbscrew fasteners). The correct transceiver cable should have a 15-pin D-shell connector for attachment to the IBM RT Baseband Adapter. Detailed information on the IBM RT Baseband Adapter is provided in the *IBM RT Hardware Technical Reference* publication.

Appendix H. IBM RT Hardware Installation

IBM RT Hardware Installation and Verification

In order to use communications on the IBM RT, you must first install the communications adapter(s) in the systems and configure the hardware into the systems. On the IBM RT this is done by installing the communications adapter in an appropriate slot. Examples:

- The Token-Ring Adapter can be installed in slot 2,4,5,7 and 8 in a 6150 model (floorstanding model).
- The Token-Ring Adapter can be installed in slot 2,3,4, and 5 in a 6151 model (desktop model).
- The Multiprotocol Adapter can be installed in slot 2,4,5,7 and 8 in a 6150 model.
- The Multiprotocol Adapter can be installed in slot 2,3,4, and 5 in a 6151 model.
- The Baseband Adapter can be installed in slot 2,3,4,5,6 and 7 in a 6150 model.
- The Baseband Adapter can be installed in slot 1,2,3 and 4 in a 6151 model.
- The IBM PC X.25 Communications Adapter can be installed in any free slot in a 6151 or 6150 model.

The following section will give an overview of what you should be aware of, when you install a communications adapter in the IBM RT.

It is always a good idea to run the IBM RT diagnostics test when a new adapter is introduced to the system. By running ***Installation Verification*** from the diagnostics diskettes, it is possible to see if there are any DMA and/or address conflicts among the installed adapters.

See the *IBM RT User Setup Guide and Options Installation* or the *IBM RT Hardware Technical Reference* for information on setting DMA and address jumpers on the different IBM RT adapters.

The Multiprotocol Adapter

If the Multiprotocol Adapter is installed in an IBM RT system with the EESDI disk drive adapter, a DMA jumper on the Multiprotocol adapter needs to be moved. This is because the EESDI adapter uses DMA level 1, and the Multiprotocol Adapter uses this DMA level as default. By moving the DMA jumper, the Multiprotocol Adapter can use DMA level 5. See Figure 237 on page 532.

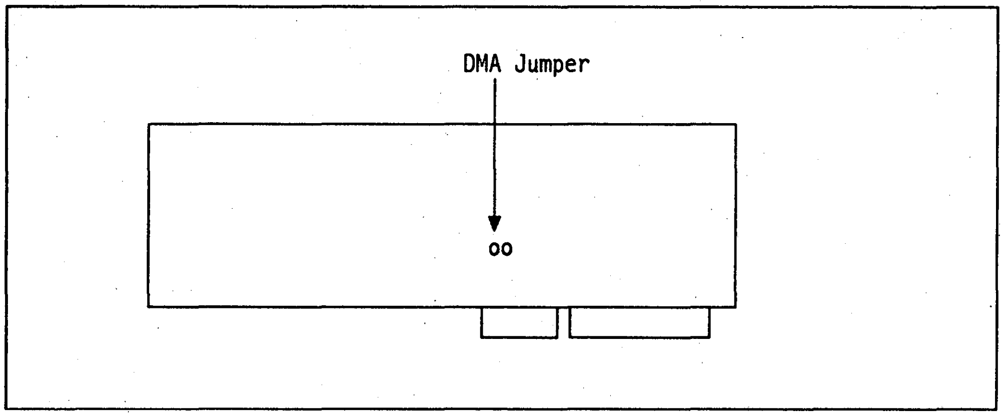


Figure 237. Setting the DMA-Channel of the Multiprotocol Adapter to 5

The Baseband Adapter

When installing the Baseband Adapter, you should check the jumpers on the card for memory I/O addresses and Interrupt level. This is important, since this information must be provided to AIX, when you later configure the adapter into the system using the *devices* command.

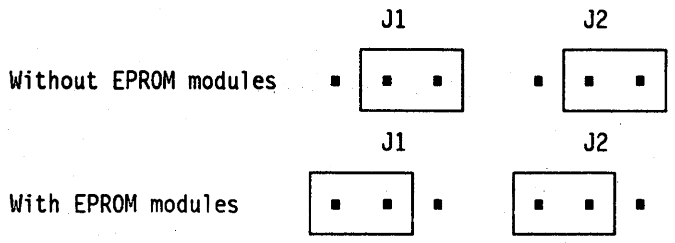
Run the Installation Verification Test (option 4 on the Diagnostic Diskettes) to make sure there are no conflicts with other adapters.

The Token-Ring Adapter

Make a note of all card settings. These settings must match the parameters set when the adapter is added with the AIX *devices* command. There are two versions of the IBM RT Token-Ring Adapter.

You can easily distinguish between the early- and later versions of the Token-Ring adapter card, since the early version has horizontal pins labeled J4 and J5. On the later version, these pins are vertical and are labeled J5 and J6. On the early version card, you should set J4 to "Internal" and J5 to "Monitored". Detailed information about the IBM RT Token-Ring Adapter is provided in the publication *IBM RT Technical Reference Token-Ring Adapter*, This publication is appropriate for hardware and program designers, programmers, and engineers. Information can also be found in the *Options Installation Manual.*

Note: In some copies of the *Options Installation Manual:ecit, the description of jumpers J1 and J2 for the "early version" Token-Ring Adapter are incorrect. The jumpers are shown backwards in these copies of the* Options Installation Manual, *but are correctly illustrated in the* Hardware, Maintenance and Service Manual: *The jumper settings are determined by the presence/absence of EPROM modules (plug-in modules) on the adapter. The* Options Installation Manual *shows the EPROM area on the adapter. The correct settings for J1 and J2 on the "early version" card are:*



Run the Installation Verification Test (option 4 on the Diagnostic Diskettes) to make sure there are no adapter conflicts. Also run the adapter test to check out the adapter and the cabling.

Configuring the Adapters Into the System

On the IBM RT, this is done with the devices command. For the Token-Ring Adapter, the following must be added to the system:

- *Add an adapter description:* `token0` *for the first Token-Ring Adapter and* `token1` *for the second. Remember to set the parameters according to the jumper settings on the Token-Ring Adapter.*
- *If the adapter is to be used for SNA connections, add a datalink description:* `trllc0` *for the first Token-Ring Adapter and* `trllc1` *for the second.*

For the Multiprotocol Adapter the following must be added:

- *Add an adapter description:* `mpd0` *for the first Multiprotocol Adapter and* `mpd1` *for the second.*
- *If the adapter is to be used for SNA connections, add a datalink description:* `sdlcllc0` *for the first Multiprotocol Adapter and* `sdlcllc1` *for the second.*

For the Baseband Adapter (Ethernet attachment) the following must be added:

- *Add an adapter description:* `net0` *for the first Baseband Adapter and* `net1` *for the second. Set the parameters in accordance with the jumper settings on the adapter. Normally, you would choose the following values for the first Baseband Adapter in the system:*
  - `rsa=98000`
  - `rea=99fff`
  - `brsa=9a000`
  - `brea=9ffff`

  Normally, you would choose the following values for the second Baseband Adapter in the system:
  - `rsa=90000`
  - `rea=91fff`
  - `brsa=92000`
  - `brea=97fff`

  If your system has the AT-Coprocessor adapter installed, you must choose the following values (remember to set the jumpers on the card accordingly. For the first Baseband Adapter, `net0`*:*
  - `rsa=e0000`
  - `rea=e1fff`
  - `brsa=e1000`
  - `brea=e7fff`

  For the second Baseband Adapter, `net1`*:*
  - `rsa=e8000`
  - `rea=e9fff`
  - `brsa=ea000`
  - `brea=effff`

  If you use Class A addresses in TCP/IP, you will also have to change the parameter `nidl` *to the value 1.*

- *If the adapter is to be used for SNA connections, add a datalink description:* `ethllc0` *for the first Baseband Adapter and* `ethllc1` *for the second.*

For the IBM PC X.25 Communications Adapter, the following must be added:

- *Add an adapter description:* `x25w0`
- *If the adapter is to be used for SNA connections, add a datalink description:* `qllc0` *if you need to run SNA over X.25.*

For further information on the X.25 adapter, see "IBM PC X.25 Communications Adapter Features" on page 312.

Once the hardware is installed and the device descriptions are added, the next step of configuring your software can begin, for example, TCP/IP, SNA Services, 3270 emulation and so on.

Appendix I. AIX/RT Performance Tuning

The hints in this appendix about tuning the AIX/RT system are collected from a variety of sources. The hints can be used "as is" but have not been through any formal testing. Neither have the information about various system options been checked for accuracy. Nevertheless, the hints are passed on to you because they have proved helpful in many installations throughout the world.

It is recommended that before you do anything to your system, make some objective observations. What is the response time? What is the load? What error messages are occurring? Write your observasions down so you have a well defined and well documented basis for comparison after trying to tune.

Maintaining System Performance

Before even attempting to tune your system by changing system options, three basic things should be done:

1. *Make sure directory files are small (less than 10 logical blocks)*
2. *Reorganize file systems*
3. *Reorganize minidisks.*

The procedure for doing this is discussed in Managing the AIX Operating System. *A brief summary follows.*

Keeping Directory Files Small

To find large directories, use the following command:

```
find / -type d -size +10 -print | print
```

Erase any unnecessary files. For example, the INed files *.bak *and* ...* *may usually be deleted. If files can not be deleted, decide on how to reorganize the large directories into smaller ones.*

In order to free the directory space, you will have to compact the directory using one of the following procedures:

- *To compact a single directory:*
  1. *Create a new directory using mkdir.*
  2. *Move (mv) or copy (cp) the files from the old directory to the new directory.*
  3. *Remove the old directory and all its files using rm -r.*
- *To compact a complete file system, use dcopy to copy the entire old file system to a new file system. This is the recommended option, as it will reorganize your files as well.*

Reorganizing File Systems

- *To reorganize both the data and the free list, which is the recommended method, there are two methods available:*

  1. *Backup the entire file system.*
  2. *Make a new file system using mkfs.*
  3. *Restore the backed up file system.*
  4. *Use dcopy*

- *To reorganize only the free list, use fsck -s or fsck -S.*

Reorganizing Minidisks

You may want to reorganize the location of your minidisks for better performance.

Generally, you should try to put the minidisks with high usage at locations close to one another. This will reduce the average distance and time for the disk heads to travel to the next block of data. The time taken for disk heads to get from one block of data to another increases in relation to the physical distance that the head has to move.

Ideally, high usage minidisks should be located near the middle of the physical disk, so that if the disk heads need to move from highly used data to less used data, they will normally only have to travel across 1/2 the radius of the disk. If the highly used data was at one extreme end of the disk (for example, outermost cylinder) and less often used data was at the other (for example, innermost cylinder), then the disk heads would need to travel across the whole disk surface.

When using the minidisks command, the first minidisk on the list is the one located at the outermost location.

Another performance tip is to try to keep the paging minidisk on a disk separated from disks with high usage files. By its nature, the paging minidisk will have a lot of activity. Minimizing the number of other active files on the same physical disk will reduce contention. This tip can be extended to any high usage file system — try to keep them on separate physical disks.

Note that to change locations of the system minidisks, you will have to use the Install/Maintenance diskette.

Using the sar Command

Serious tuning of AIX/RT always involves the use of the sar command. Run sar during peak periods and during off-peak. Run when you experience unsatisfactory response time, as well as when response time is acceptable. Then compare the results, determine if there is a system parameter that can be changed (see "The /etc/master Stanzas" on page 544), and try out the changes. Be sure to back up everything before you start changing parameters and to keep a log of the changes being made and the results. Doing this rigously will save a lot of headache later.

Running the sar Command

The sar command has two forms; one to collect data and one to present the data collected. The publication AIX Operating System Technical Reference *describes the command syntax, examples, and other commands that can be used to run sar. The most basic syntax of the sar command is:*

sar -A *-o rptfile interval number*

This command collects data number times, spaced interval seconds apart and puts the information in the file rptfile. To print the report:

sar -A *-shh mm -isec -ehh mm -frptfile* **| print**

This will print the sar report from the sar file named rptfile, starting at hh mm (-s flag), ending at hh mm (-e flag), selecting data records at intervals as close as possible to sec seconds. Omitting -isec will cause all intervals to be reported.

Interpreting sar Output

The following is a description of each sar option and some tips where available.

```
20:57:25  iget/s-namei/s dirbk/s
20:57:45       0       0       0
20:57:50       1       1       1
20:57:55       6       5       2
20:58:00      54      40      33

Average        8       4       1
```

Figure 238. Use of File Access System Routines, sar -a Option

igets/s *Calls per second to the i-node look-up routine The iget routine locates the i-node entry of a file. It first searches the i-node entry in memory. If the i-node entry is not in the table, the iget routine gets the i-node from the file system where the file resides and enters it in the i-node table in memory. iget returns a pointer to this entry.*

namei/s *Calls per second to the directory search routine. namei* calls iget, but since other routines also call iget, iget should always be greater than namei.

dirbk/s Directory blocks read per second by **namei()**. The number of directory blocks read divided by the number of namei calls estimates the average path length of files. The lower the value of this quotient, the better. If the values are very low for both namei and dirblk (less than 3, for example), the value of the quotient is not really very meaningful.

Minimizing the use of large files should keep the value of namei/dirblk low. Of course, sometimes you can't avoid the use of large files. This will cause higher values for namei/dirblk because several blocks will probably have to be read before the needed data can be found.

Keeping directories small and regularly reorganizing file systems can also help keep namei/dirblk low.

```
20:57:25 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
20:57:55       2       9      82       0       2      91       0       0
20:58:00       3     240      99       2      11      84       0       0
20:58:05      10      60      83       5       8      35       0       0

Average        1      21      95       1       3      80       0       0
```

Figure 239. Buffer Activity, sar -b Option. Buffer activity for transfers, accesses and cache hit ratios.

lread/s Number of logical reads per interval, issued by the system I/O block devices.

lwrit/s Number of logical writes per interval, issued by the system to block devices.

bread/s Number of block reads per interval. This is number of times data was read from block devices into system buffers.

bwrit/s Number of block writes per interval. This is the number of times data was written from the system buffers to block devices.

%rcache Cache read ratio for reads.

%wcache Cache hit ratio for writes. The cache hit ratio is the ratio of block I/O to logical I/O and is a measure of the effectiveness of system buffers. Ideally, the cache hit ratio should be as high as possible, but this is highly dependent on the type of application. A percentage in the 90's is a good number to aim for, although some database applications have been tuned to get cache hit ratios up to 95% or more. Often the cache hit ratio for read is higher than for write.

The **kbuffers** value in /etc/master can be changed to make the amount of system memory allocated to I/O buffers either bigger or smaller. The trade-off is that as **kbuffers** increases, there is less memory available for user programs to run in. If you increase **kbuffers**, keep watch on the cycle/s parameter, which will tell you the degree of paging.

If a system has a high cache hit rate but is doing heavy paging, your value for **kbuffers** is probably too high.

pread/s Reads per interval on seekable raw devices.

pwrit/s Writes per interval on seekable raw devices.

```
20:57:25 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
20:57:30      17       3       4    0.00    0.00   62926     220
20:57:35      58       2       3    0.20    0.40   59535     201
20:57:40      17       2       3    0.00    0.00   58744     202
20:57:45      11       3       3    0.00    0.00   58371     194
20:57:50      58       2       3    0.20    0.40   58748     198
20:57:55     120       2       3    0.20    0.20   59740     199
20:58:00     274      37       4    2.19    2.58   60778     214

Average      236      40      17    0.34    0.33   68193    1205
```

Figure 240. Buffer Call Activity, sar -c Option

scall/s Total number of system calls per second

sread/s	Number of system read calls.
swrit/s	Number of system write calls.
fork/s	Number of fork system calls per second.
exec/s	Number of exec system calls per second.
rchar/s	Character transferred per interval by read call.
wchar/s	Character transferred per interval by write call.

If rchar/s and wchar/s are low (50,000 or so), effectivity of buffering may not be significant towards performance.

The ratio of [***wchar/s*** + ***rchar/s***] to [***sread/s*** + ***swrit/s***] gives the number of bytes transferred per call. The higher this value, the better.

```
20:57:25 ksched/s kproc-ov kexit/s
20:57:30       0        0       0
20:57:35       0        0       0
20:57:40       0        0       0
21:04:15       0        0       0

Average        0        0       0
```

Figure 241. DS Kernel Activity, sar -k Option. The traced processes are used only by Distributed Services.

ksched/s	Number of kernel processes assigned to tasks per second.
kproc-ov	Number of overflows occurring between sampling points.
kexit/s	Number of kernel processes terminating per second.

```
20:57:25   msg/s  sema/s
20:57:40    0.00    0.00
20:58:45    0.00    0.20
20:59:00    0.00    0.00
20:59:05    0.00    0.82

Average     0.02    0.02
```

Figure 242. Message and Semaphore Activity, sar -m Option. The traced processes are used only by Distributed Services.

msg/s IPC message primitives per second. This value gives an idea of the amount of interprocess communication going on. A sudden high value may indicate an abnormal condition. The **msg...** stanzas in /etc/master effect these numbers.

sema/s IPC semaphore primitives per second. Indicates semaphore activity. Again, a sudden high value may indicate that something is wrong. The **sem...** stanzas in /etc/master effect these numbers.

```
20:57:25 runq-sz %runocc
20:57:30     1.0      43
20:57:35     1.0      20
20:57:40     1.0      20
20:57:45     1.0      39
20:57:50     1.0      20
20:57:55     1.5      40
20:58:00     2.3      60
20:58:05     3.4     104

Average      3.3      81
```

Figure 243. Queue Lengths, sar -q Option. Average queue length while occupied, and percentage of time occupied.

runq-sz Queue queue of processes in memory and runable, that is, not waiting for I/O, etc. In conjunction with %runocc, this indicates how busy the processor is. If this value is large and %runocc is 100%, it may indicate a bottleneck in the processor (assuming the values for paging are okay).

%runocc Percentage of queues that are in memory and runable. Is a measurement of the user application wait time. The **slice** stanza in /etc/master will effect this time.

A value of 100 for this counter is ideal for a loaded system. This means that there is always work waiting for the processor. A number less than 100 may indicate a light workload; on the other hand it may indicate a need for scheduling. If less than 100, check the value of runq-sz. If it is high, consider the possibility of rescheduling some jobs.

```
20:57:25    slots cycle/s fault/s  odio/s
20:57:30     4701    0.00    2.13   27.02
20:57:35     4672    0.00    1.81   11.28
20:57:40     4667    0.00    0.40    2.58
20:57:45     4667    0.00    0.00    0.00
20:57:50     4667    0.00    0.60    1.79
20:57:55     4656    0.00    1.81    8.46

Average      3102       0       3      18
```

Figure 244. VRM Paging Statistics, sar -r Option

slots The number of free pages on the paging minidisk. The size of the page space will effect the number of free pages. The recommendation is for this value not to go below 25%, that is, the used pages on the paging minidisk should generally not exceed 75% of total pages. The available pages will have to be computed based on the size of the page space minidisk. Pages or slots are 2K, or roughly 4 blocks.

cycle/s The number of page replacement cycles per second. The size of real memory will effect cycle/s. A page replacement cycle is when all pages in memory are replaced by pages from the page space minidisk (on disk). This value indicates the degree of thrashing. A value of 0 is ideal. A value of 1 is high. A high value indicates that additional memory is needed.

fault/s The number of page faults per second. Real memory will effect fault/s. A page fault occurs when needed data is not available in memory, and the system must bring the data in. The rate of page faults is application dependent. Changing the **kbuffers** value in /etc/master will influence the number of page faults. A large number of page faults may be an indication that additional memory could improve performance.

odio/s The number of nonpaging disk I/Os per second.

```
20:57:25    %usr    %sys    %wio   %idle
20:57:50       5       0       3      91
20:57:55       6       6       9      79
20:58:00      21      43       6      30
20:58:05       1      97       2       0
20:58:10       9      68      22       2
20:58:15      26      39      22      13
20:58:20      32      52      12       4
20:58:25      40      35      16       9
20:58:30      29      41      12      18

Average       39      28       3      29
```

Figure 245. CPU Activity, sar -u Option

%usr Percentage of CPU time devoted to the user.

%sys Percentage of CPU time executed in privilege mode. If this value is high (above 30%), it's an indication that the system is spending too much time doing I/O, paging, swapping, etc.. It may also indicate that the system is thrashing.

%wio Percentage of CPU time waiting for block I/O to complete. Value should preferably be less than 7%. If very close to zero percent indicates that CPU is bottleneck. A value of 20% may be considered high. A value of more than 30% may be an indication of an abnormal condition. Increasing the **kbuffers** value in /etc/master, reorganizing file systems, or spreading out high usage minidisks may decrease this value, which may in turn increase performance.

%idle Percentage of CPU time idle. A loaded system should not have idle time. If idle time is greater than 0 and response time bad at other times, try rescheduling some jobs.

```
20:57:25 text-sz proc-sz inod-sz file-sz text-ov proc-ov inod-ov file-ov
20:57:30  17/ 60 426/512  54/1000 104/1000       0       0       0       0
20:57:35  18/ 60 427/512  55/1000 105/1000       0       0       0       0
20:57:40  17/ 60 426/512  54/1000 104/1000       0       0       0       0
20:57:45  17/ 60 426/512  54/1000 104/1000       0       0       0       0
```

Figure 246. Used Versus Allocated Table Entries, sar -v Option. Status of text, process, i-node, and file tables.

text-sz Text table entries. The /etc/master stanza **texttab** sets the number of allocated entries.

proc-sz Number of kernel processes. The /etc/master stanza **proc** sets the number of kernel processes. The displayed value includes the /etc/master value for **kprocs**.

inod-sz Number of i-node entries. The /etc/master stanza **inodetab** sets number of i-node entries.

file-sz Number of open files. The /etc/master stanze **filetab** sets maximum number of open files.

text-ov Overflows occurring for text table at sample point. If greater than zero, try increasing the value of **texttab** in /etc/master.

proc-ov Overflows occurring for proc table at sample point. If greater than zero, try increasing the **procs** value in /etc/master.

inod-ov Overflows occurring for inode table at sample point. If greater than zero, try increasing the **inodetab** value in /etc/master.

file-ov Overflows occurring for file table at sample point. If greater than zero, try increasing the **filetab** value in /etc/master.

The ratio of text-sz, proc-sz, inod-sz, and file-sz indicate the utilization of the table entries allocated for these values in the kernel. Values of above 80% mean that you should increase the tables. Values of 50% or less may indicate that you could reduce the number of entries, but all you gain is reducing the kernel with a (probably) insignificant number of bytes.

```
20:57:25 pswch/s
20:57:30       4
20:57:35       5
20:57:40       5
20:57:45       3
20:57:50       5
20:57:55       9
20:58:00      41

Average       62
```

Figure 247. System Switching Activity, sar -w Option

pswch/s Process switches per second; or the number of times the switcher was invoked. This happens when:

1. A system call resulted in a road block.
2. An interrupt occurred resulting in awakening a higher priority process.
3. A one-second clock interrupt occurs.

If you have a heavily loaded system and pswch/s is quite high, increasing the **slice** value in /etc/master may decrease the time it takes to run an application, since the processor spends less time switching from one task to another.

```
20:57:25 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
20:57:30       0       0     116       0       3       0
20:57:35       3       2     110       3       2       0
20:57:40       0       0     106       0       2       0
20:57:45       3       0      99       3       2       0
20:57:50       0       3     103       0       2       0
20:57:55       1       1     107       1       2       0

Average        0       0     141       2       3       0
```

Figure 248. TTY Device Activity, sar -y Option

rawch/s	TTY raw input queue characters per second.
canch/s	TTY canonical input queue characters per second.
outch/s	TTY output queue characters per second.
rcvin/s	TTY receive interrupts per second. Receiver hardware interrupts per second for terminal devices (TTY). Indicates receive activity of terminal devices.
xmtin/s	TTY transmit interrupts per second for terminal device (TTY). Indicates transmit activity of terminal devices.
mdmin/s	TTY modem interrupts per second. Similar to revin and xmtin.

Characters generated by devices operating in the "cooked" mode, such as terminals, are counted in both rawch/s and (as edited) in canch/s, but characters from raw devices, such as communication processors, are counted only in rawch/s.

rawch/s and canch/s are input values, thatis, characters sent by the user to the system. A terminal screen is approximately 2K bytes if this value is on the low side.

Rebuilding the Kernel

Rebuilding the kernel will activate any changes to the system configuration file `/etc/master`. To rebuild the kernel:

1. Log in as superuser.
2. Edit `/etc/master` as needed.
3. Type: `cd /usr/sys`
4. Type: `make`
5. Save the old kernel: `mv /unix /unix.old`
6. Copy new kernel to root: `mv unix.std /unix`
7. Type: `shutdown -rf` to reboot the system.

If the new kernel does not work, see *Managing the AIX Operating System* for the recovery procedure in the section, "Maintaining the System".

Quick Tuning

For a quick attempt at tuning your system, you may want to follow this checklist:

1. Backup and restore all file systems (reorganize).
2. Put busy minidisks on separate physical disks.
3. Watch %cache hit ratios - effectiveness of buffering. See if increasing the **kbuffers** value in `/etc/master` helps.
4. Watch cycle/s - thrashing indicator. If high, add memory.
5. Watch text/sz, inod-sz, file-sz and proc-sz. If ration is close to 1, increase the corresponding values in `/etc/master`.
6. Watch runq-sz. If runq-sz is high and %runocc is 100%, try a faster processor.
7. Check slots. If low, increase page space minidisk size.

If the above doesn't work, it's time to do it seriously!

Miscellaneous Performance Tips

Performance may also be improved by trying some of the following methods.

1. Disable any system-wide disk I/O's, cron for example. In most cases the application should take care of regularly "committing" its data, so there may be no need to have AIX refresh buffers to disk periodically.
2. Experiment with different access methods, such as "hash", instead of "btree", "cbtree", etc., if this choice is available.
3. Have sufficient work space available in the data subdirectories. Sometimes when an application must search around for work space in places other than its current directory, performance may be impacted.
4. Use a device-driver backend instead of a daemon process for application data locking mechanisms, if available.

The /etc/master Stanzas

This section describes the stanzas in /etc/master and gives hints about their use and how to change them to fit your installation. Always check this information against the /README file supplied with your AIX/RT operating system.

node = "\"m135\""
Sets the node name of your system. It is recommended that this field always be changed to reflect the name you give your system whenever using TCP/IP, SNA Services, Basic Networking Utilities or other networking options of the AIX system.

callouts = 50
Certain activities require an interval between one action and another. The kernel's callout mechanism schedules these activities. The callouts parameter specifies the maximum number of timed actions that can be pending concurrently. If this maximum is reached, the system halts.

The callout mechanism is used most commonly by the display station device drivers. The value for callouts should be 25-100% more than the number of devices on the system.

Distributed Services sets the maximum number of timed events that can be scheduled concurrently. The value of callouts should be approximately: (maxnode - dsnkprocx + 50).

charlists = 64
Specifies the number of character list the terminal driver uses. Gives a bigger buffer in the kernel for ASYNC devices. If the buffer area is not large enough, the system will give the message "***TTY Hog errors***".

dsnkprocs = 20
Distributed Services sets the maximum number of kernel processes available for use by DS. The value of dsnkprocs should equal the number of concurrent connections. It is normally set to 2 less than **kprocs**.

dumpdev = hd4
: Specifies the target device for kernel dumps. dumpdev must be a minidisk, that is, hd4.

enhedstack = 8
: Specifies the number of commands that are held in the command buffer for reuse. The *stty enhedit* options needs to be specified to allow command recall.

filetab = 250
: Filetab specifies the maximum number of open files. When the maximum is reached, attempts to open new files will fail until running processes close some of their files. The value of filetab should be about the same as inodetab.

  The *sar* value file-sz gives the utilization of the file table.

  Distributed Services sets the maximum number of files (local and remote) that can be open simultaneously. The values of filetab and inode should be approximately equal.

floating = software
: Indicates whether the kernel should attempt to use floating-point hardware, if it is present. The default, software, means there is no optional floating-point accelerator hardware.

hashbuffers = 128
: This is the area that the system uses to map filenames to the inodes for quick lookup. This may need to be increased if there are a large number of files open.

hftbuffers = 17
: Specifies the number of virtual terminals.

inodetab = 250
: Inodetab specifies the size of the i-node table in memory. The i-node table contains copies of the i-nodes for all active files (either an open file, the current directory of a process, or if a file system is mounted on it). When the i-node table is full, new system activity is delayed until space is freed from the i-node table.

  The *sar* value inod-sz gives an indication of the utilization of the i-node table. On most systems, an allocation of 5 to 6 i-nodes for each process is adequate. For example, if maxprocs is 40, inodetab usually should be about 250.

  Distributed Services sets the maximum number of i-nodes that can be active simultaneously. The value of inodetab should equal the number of files (local and remote) that are open simultaneously. The values of inodetab and filetab should be approximately equal.

iobuffers = 8
: Specifies the number of physical I/O buffers the kernel supports. These are the read ahead/write behind buffers.

kbuffers = 0
: Almost all disk I/O goes through a group of buffers. To minimize physical I/O, copies of the disk blocks used most frequently are kept in memory. The amount of memory reserved for this purpose in the kernel is controlled by the KBUFFERS value ("kernel buffers").

As the value of kbuffers is increased, more disk blocks are stored in memory causing less physical disk accesses. This should increase response time. The trade-off is that as more space in memory is used for buffering, less space is available for other purposes. The values to watch in the *sar* report are therefore, %rcache and %wcache for effectiveness of buffering, and cycle/s. If cycle/s goes up, then the improvement due to more efficient I/O may be negated by thrashing. Note that if the data volume (see *sar* values rchar and wchar) is low, increasing kbuffers may not improve performance.

Each buffer is 2048 bytes. If kbuffers = 0, (the default), the system uses the physical size of the installed memory and the processor being used to determine the number of buffers. If your system has less than 2mb of memory, the system uses 75. For 2mb, 150 is the default, and for more than 2mb, the system seems to default to 20% of the total system memory. No specific information was found indicating the differences in these defaults with regard to the processor model. Generally, the recommendation is to use between 10% and 25%. If the applications running are very I/O dependent, for example, multiuser database applications, performance may be improved further by increasing kbuffers to more than 25% assuming there is enough memory left over for other purposes.

Note for benchmarks: often a benchmark is conducted using small files. Unrealistically good performance may be observed if kbuffers is large enough to keep the whole file in memory.

For Distributed Services the kbuffers parameter determines how many kernel buffers are reserved for disk I/O and i-nodes of open files. Increasing the value of the kbuffers parameter can improve the speed of I/O operations, depending upon the amount of memory and disk space available. Beyond a certain point, however, increasing the value of kbuffers degrades performance. The default value of kbuffers (0) reserves 100 buffers (2048 bytes each) for every M byte of memory installed. The default value is normally a good choice.

kdmabuffers = 64
Specifies the number of buffer headers used by the **exec** system call.

kmap = 100
Specifies the number of elements in resource map array for internal kernel storage.

kprocs = 22
Used by Distributed Services. The maximum number of concurrent, remote request that a server can handle is limited by the maximum number of kprocs (internal kernel processes) that the server can run. Two conditions that may indicate an inadequate number of kprocs are:

1. Slower performance in handling local requests.
2. Connection failures.

Also, check the *sar* report value **kproc-ov**.

maxnode = 20
Used by Distributed Services. Sets the maximum number of nodes that can be connected in the DS network.

maxprocs = 40
Specifies the maximum number of processes that can be run by any particular user. Maxprocs controls the number of processes for each user in the same way that procs controls them for the kernel.

mountab = 16
Specifies the number of entries in the mount table used by the kernel for file system mounting. This value sets the limit of the number of file systems that can be mounted by the kernel. This number is the number of local mounts.

Mountab specifies the maximum number of filesystems that can be mounted at the same time. The recommended value is one or two more than the number of filesystems you expect to have mounted at the same time.

msgmap = 100
Specifies the number of elements in the message array. This array assists in the allocation of space within the message data area. Elements of the array represent free space in the message data area. This space is created by the return of active messages.

msgmax = 8192
Specifies the maximum number of bytes allowed for a single message.

msgqid = 50
Specifies the maximum number of active message queues.

msgqmax = 32768
Specifies the maximum number of bytes allowed for a single message queue.

msgseg = 2048
Specifies the number of segments in the message data area. This number must be less than 32,768.

Note: Multiplying the msgseg by the msgsegsize gives you the size of the data area. Increasing one or both of these values enlarges the message data area. When adjusting the data area, increase the number of segments before increasing the size of segments.

msgsegsize = 8
Specifies the size of each segment in the message data area. The size is in bytes and should be a multiple of four. This is the minimum allocation of space for a message. Regardless of the space required, an entire segment is used. Messages which use less than an allocated segment waste space. A small msgsegsize is most efficient; 8 bytes is an adequate specification.

msgtql = 400
Specifies the maximum number of active messages permitted in the system.

netnoone = 65535
Used by Distributed Services. Specifies the DS user ID or group ID value to use when no user ID from the local node maps to a remote file's owner ID.

netsomeone = 65534
Used by Distributed Services. Specifies the DS user ID or group ID value to use when more than one local ID maps to a remote file's owner ID, and one particular ID cannot be selected (for example, because of wildcard mappings).

nflocks = 1000
Specifies the maximum number of simultaneously locked file regions.

nid = "(0)"
Specifies the node ID to generate into the system. This keyword is currently unused.

nncb = 25
Used by Distributed Services. Sets the amount of memory allocated for DS translation tables. The value of nncb should equal the number of concurrent connections.

pinkbuffers = 0
Specifies the number of kbuffers to pin to real memory. These buffers would then not be paged out of real memory.

pipedev = hd0
Names the stanza that defines the file system used for FIFO files.

power = false
Indicates whether the kernel has power warning code.

procs = 100
Specifies the total number of simultaneous processes the kernel supports. The procs parameter specifies the maximum number of processes the kernel will support. If this number is reached, any attempt to start a new process will produce a message telling the user to try again later.

The *sar* proc-sz value indicates how many kernel processes are actually being run.

pslotkill = 200
Specifies the threshold at which the system begins to kill processes in order to recover paging space. Pslotkill is specified in slots, where a slot is 2048 bytes (four blocks) of a paging minidisk.

pslotpanic = 100
Specifies the threshold at which to stop AIX and attempt a system dump because paging space has almost been exhausted. Note that the system dump itself cannot finish because of the lack of paging space. Pslotpanic is specified in slots, where a slot is 2048 bytes (four blocks) of a paging minidisk.

pslotwarn = 350
Specifies the threshold at which the system displays a message warning that paging space is running low. When the system displays this message, it also:

1. Performs a sync to write all changes to disk.
2. Enters sync mode, in which disk I/O is not buffered.
3. Sends all processes the SIGDANGER signal to warn them that the system is likely to crash any moment.

ptybuffers = **16**
Specifies the number of pseudo-terminals that can be present in the system. The maximum value is 256.

rootdev = **hd0**
Names the stanza in the /etc/system file that defines the root file system device.

rsbuffers = **80**
Specifies the number of buffers allocated for the asy terminal driver.

semadjmax = **16384**
Specifies the maximum value allowed for semaphore adjust value on exit.

semid = **120**
Specifies the number of distinct semaphore identifiers the kernel supports.

semmap = **10**
Specifies the number of entries in a semaphore map array.

semmax = **120**
Specifies the maximum of simultaneous semaphores allowed and supported by the kernel.

semopmax = **10**
Specifies the maximum of operations allowed for each **semop** system call.

semsetmax = **25**
Specifies the maximum of semaphores allowed in a set.

semunmax = **30**
Specifies the number of semaphore undo structures the kernel supports.

semunpmax = **10**
Specifies the maximum number of undo entries for each process.

semvalmax = **32767**
Specifies the maximum value allow for each semaphore.

shlibtab = **30**
Specifies the number of shared libraries that an application can use.

shmid = **100**
Specifies the number of distinct shared memory identifiers the kernel supports.

shmmax = **33554432**
Specifies the maximum number of kilobytes for shared memory allowed per shared segment.

shmmin = **1**
Specifies the minimum number of kilobytes for shared memory allowed per shared segment.

shmsegs = **11**
Specifies the number of segment registers that may be used to support shared memory.

shmsysmax = **32768**

Provides compatibility with other UNIX systems. This value is otherwise ignored.

slice = **20**

Specifies the percentage of time in quanta that a process can run before it must relinquish control of the processor. Each quantum on the RT system is equal to 333 milliseconds.

texttab = **40**

Specifies the number of shared text segment entries in the text table. AIX/RT allows processes running the same program to share a single copy of that program in memory. The text table contains one entry for each active shared program(text) segment. Texttab specifies the size of the text table. If the text table is full, no other text segments can be shared until some processes complete and create some room in the text table.

The value of texttab is usually 30-50% of the number of processes running in the system. Utilization of texttab can be seen by the *sar* value text-sz.

Devices Parameters

Generally, increasing the number of device buffers may:

1. Improve performance
2. Increase the number of concurrent remote request that a server can handle
3. Eliminate connection failures.

nobodr

Determines the number of buffers in the device ring. When the number of requests to a server exceeds the capacity of the server's internal message queues, some message transmissions fail.

norbosr

Determines the number of receive buffers in the SLIH ring. The value of norbosr should be the same as that of the nobodr parameter.

nobibp

Determines the number of buffers in the network device buffer pool. A larger value can enable a server to handle more concurrent remote requests and may eliminate some connection failures. Use the following formula as a guide for changing the value of nobibp:

```
nobibp = (nobodr + n * (tw * 2) + 2 * n)
```

where:

nobodr	is the value of that device parameter. dt.n
	is the number of nodes in the network.
tw	is the value of the SNA Services transmit window parameter.

For Distributed Services, the values for the above device descripture parameters should be set according to the configuration of your DS network. Connections fail if the values if these parameters are too small.

mnonid

Sets the maximum number of ***net IDs*** - not to be confused with node IDs - a server can handle. This is a parameter used by TCP/IP but ***not*** by Distributed Services. The value of mnonid should be equal to the number of clients on the network.

mnoal

Sets the maximum number of gateways that the server can handle, including the base network. The value of mnoal should be equal to the number of network adapters installed in the server. This is a parameter used by TCP/IP but ***not*** by Distributed Services.

Glossary

0 - B

3270 Device Emulation. Support that allows a local or remote device on one system to appear as a 3270 device to another system.

absolute address. (1) An address that, without the need for further evaluation, identifies a storage location or a device.

absolute value. The numeric value of a real number regardless of its algebraic sign (positive or negative).

access. The manner in which files or data sets are referred to by the computer.

access. (1) To obtain data from or put data in storage. (2) The manner in which files or data sets are referred to by the computer.

access level. In computer security, the level of authority an operator has while using a secured file or library.

access method. (1) A technique for moving data between main storage and input/output devices. (2) The way the system refers to records in files. The reference can be sequential (records are referred to one after another in the order in which they appear in the file), or it can be random (the individual records can be referred to in any order). (3) A software component in a processor for controlling the flow of information through a network.

access permission. (1) A group of designations that determine who can access a particular AIX file and how the user may access the file. (2) All access rights a user has regarding an object.

access procedure. The procedure or protocol used to gain access to a shared resource. In a local area network, the shared resource is the transmission medium. The medium access procedures specified by the IEEE 802 standard are CSMÀ/CD token, bus, and token ring.

ACF. See *Advanced Communications Function.*

ACF/NCP. See *Advanced Communications Function for the Network Control Program.*

ACF/VTAM. See *Advanced Communications Function for the Virtual Telecommunications Access Method.*

ACK. See *acknowledgement character.*

ACK0. A transmission control character for even positive acknowledgement; indicates that text was received without transmission errors.

ACK1. A transmission control character for odd positive acknowledgement; indicates that text was received without transmission errors.

acknowledge. Answer. To respond to a poll, address, or message.

Acknowledge Timeout. The number of seconds that a station should wait for an acknowledge from a remote station after sending data.

acknowledgment character (ACK). In binary synchronous communications, a transmission control character sent as an affirmative response to a data transmission.

active gateway. A gateway that is treated like a network interface, in that it is expected to exchange routing information, and if it does not do so for a period of time, the route associated with the gateway is deleted.

ACTLU. Activate Logical Unit.

ACTPU. Activate Physical Unit.

ACU. Automatic calling unit. See *autocall unit.*

adapter. (1) An mechanism for connecting two unlike parts or machines. (2) A printed circuit card that modifies the system unit to allow it to operate in a particular way. See also *communications adapter.*

address. (1) A name, label, or number identifying a location in storage, a device in a system or network, or any other data source. (2) The telephone number that remote systems use to call the system. (3) A value that identifies a register, a particular part of storage, a data source, or a data sink. The value is represented by one or more characters. (4) To refer to a device or an item of data by its address. (5) In word processing, the location, identified by an address code, of a specific section of the recording medium or storage. (6) In data communication, the unique code assigned to each device or work station connected to a network.

address field. The part of a packet containing addressing information. See packet.

Address Resolution Protocol (ARP). One of the protocols provided by TCP/IP that dynamically maps between Internet addresses, Baseband Adapter addresses, and Token-Ring Adapter addresses on a local area network.

addressing. (1) In data communications, the way that the sending or control station selects the station to which it is sending data. (2) A means of identifying storage locations. (3) Specifying an address or location within a file. (4) The assignment of addresses to the instructions of a program.

adjust. The process of moving text to fit between the left and right margins.

Advanced Communications Function (ACF). A group of IBM licensed programs (principally ACF/VTAM and ACF/NCP) that use the concepts of Systems Network Architecture (SNA), including distribution of function and resource sharing.

Advanced Communications Function for the Network Control Program (ACF/NCP). A licensed program that provides communication controller support in an SNA network.

Access Method (ACF/VTAM)

Advanced Communications Function for the Virtual Telecommunications. A licensed program that provides host processor support in an SNA network.

Advanced Program-to-Program Communication (APPC). A communications architecture that allows transaction programs to exchange information on a peer-to-peer basis. SNA LU 6.2 allows APPC architecture to operate on an SNA network. APPC is the way that a system puts the IBM SNA LU 6.2 protocol into effect.

AIX Operating System. The operating system that exists on the IBM RT, PS/2 and System/370 computers between the hardware and the application programs. It consists of a kernel and a shell, or command interpreter.

alert. An error message sent to the system services control point (SSCP) at the host system.

alias. (1) An alternate name for a node or a file that can be used in place of the real name of the node or file. (2) An alternate label for a data element or point in a computer program. (3) An alternate name for a member of a partitioned data set. (4) In pulse code codulation, a spurious signal resulting from beats between the signal frequencies and the sampling frequency. (5) Unofficial name used for the network.

All Points Addressable (APA) display. (1) A display that allows each pel to be individually addressed. An APA display allows for images to be displayed that are not made up of images predefined in character boxes. Contrast with *character display*. (2) In computer graphics, pertaining to the ability to address and display or not display each pel on a display surface.

allocate. (1) To assign a resource, such as a disk file or a diskette file, to perform a specific task. (2) A request to allocate a session between the local LU and a remote LU.

allocation. The assignment of a resource, such as a disk or diskette file, to perform a task.

alphabetic. Pertaining to a set of letters A through Z (uppercase and lowercase).

alphanumeric. Pertaining to the numbers, characters, and symbols normally found on a keyboard, especially the printing character set supported by the ASCII keyboard.

alphanumeric character. Consisting of letters, numbers and often other symbols, such as punctuation marks and mathematical symbols.

alternate routing. Using a secondary or backup path to transmit data when the usual path is not available.

American National Standard Code for Information Interchange (ASCII). The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

American National Standards Institute (ANSI). An organization sponsored by the Computer and Business Equipment Manufacturers Association by which accredited organizations create and maintain voluntary industry standards.

ampersand. The character "&".

ANSI. See *American National Standards Institute*.

answering station. The station responding to a dialed call on a switched channel.

APA. See *All-Points-Addressable Display*.

APPC. See *Advanced Program-to-Program Communications (APPC)*.

append. (1) The action that causes data to be added to the end of existing data. (2) In word processing, to attach a file to the end of another file.

application. (1) A particular task, such as inventory control or accounts receivable. (2) A program or group of programs that apply to a particular business area, such as the Inventory Control or the Accounts Receivable application. (3) The use to which an information processing system is put, such as a network application or an airline reservation application.

application program. (1) A program used to perform an application or part of an application. (2) A program written for or by a user to perform the user's work; in an SNA network, an end user. (3) Software that performs a particular task such as word processing, project planning, or inventory control. (4) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

Application Program Interface (API). (1) A set of runtime routines and/or system calls that allows an application program to use a particular service provided by either the operating system or another licensed program. (2) The formally-defined programming language interface which is between an IBM system control program or a licensed program and the user of the program.

archive. (1) To store programs and data for safekeeping. (2) A copy of one or more files or a copy of a data base that is saved for future reference in case the original data is damaged or lost.

archive library. A place where programs are stored for safekeeping.

argument. (1) Numbers, letters, or words that expand or change the way commands work. (2) In a function call, an expression that represents a value the calling function passes to the function specified in the call. (3) An independent variable. (4) Any value of an independent variable, such as a search key; a number identifying the location of an item in a table. (5) A parameter passed between a calling program and a called program.

argument list. A string of arguments.

ARP. See *Address Resolution Protocol.*

ARPA. See *Advanced Research Projects Association.*

array. (1) A variable that contains an ordered group of data objects. All objects in an array have the same data type. (2) An arrangement of data in one or more dimensions: a list, a table, or a multidimensional arrangement of items. (3) In programming languages, an aggregate that consists of data objects, with identical attributes, each of which may be referenced uniquely by subscripting.

array element. A single data item in an array.

ascending key sequence. The arrangement of data in order from the lowest value of the key field to the highest value of the key field, according to the rules for comparing data items. Contrast with *descending key sequence*.

ASCII. See *American National Standard Code for Information Interchange.*

ASCII flat file. See *flat file.*

assembler language. A symbolic programming language in which the set of instructions includes the instructions of the machine and whose data structures correspond directly to the storage and registers of the machine.

async. See *asynchronous transmission.*

asynchronous execution. The type of execution of a list of pipelines where the shell does not wait for one pipeline to finish before beginning the next one.

asynchronous terminal emulation (ATE). Enables the connection to a remote terminal system, emulation it, and use of its commands and programs.

asynchronous transmission. Data transmission in which transmission of a character or block of characters occur can begin at any time but in which the bits that represent the character or block have equal time duration. Contrast with *synchronous transmission.*

Compared to other communications methods, asynchronous communications tend to be slower and cheaper.

ATE. See asynchronous terminal emulation.. See *asynchronous terminal emulation.*

atomic operation. An operation where signals cannot occur between the operations of setting the masks and waiting for the signal. ??

attachment. (1) The physical connection to the network including supporting VRM code to make it work. (2) A type of resource that controls CPs, Link Control, and Physical Link Control. (3) Supporting materials attached to a document. An attachment is generally referred to within the text of the document but not incorporated, and provides supporting data or reference material.

attachment profile. Contains parameters that associate other defined profiles to the attachment of the LU to the network. These parameters also define the type of network being used.

attended operation. An application in which human operators are required at both stations to establish the connections and transfer the modems from talk (voice) mode to data mode. For example, one person dials a location, waits for the other person to answer, then both persons push a button to switch from voice to data communications.

attribute. (1) A property or characteristic of one or more entities. For example, the attribute of a displayed field could be blinking. (2) A terminal display language or transformation definition language (TDL) keyword that specifies a particular quality for the TDL object with which it is associated.

authorize. (1) To grant to a user the right to communicate with, or make use of, a computer system or display station. (2) To give a user either complete or restricted access to an object, resource, or function.

auto. Automatic.

auto carrier return. The system function that places carrier returns automatically within the text and on the display. This is accomplished by moving whole words that exceed the line end zone to the next line.

auto listen attachment. In SNA, a local attachment that automatically accepts incoming calls from a remote station.

autoanswer. The ability of a station to receive a call over a switched line without operator action. Contrast with *manual answer*.

autocall. The ability of a station to place a call over a switched line without operator action. Contrast with *manual call*.

autocall unit (ACU). See *automatic calling unit*.

autodialer (ADU). See *automatic dialing unit*.

autoexec. A command or list of commands executed at login time.

automatic calling unit (ACU). A device that allows a host to automatically dial the number of a remote device.

automatic dialing unit (ADU). A device capable of automatically generating dialing digits.

backend. The program that sends output to a particular device. There are two types of backends: friendly and unfriendly.

background activity. See *background process*.

background process. (1) A process that does not require operator intervention that can be run by the computer while the work station is used to do other work. (2) A mode of program execution in which the shell does not wait for program completion before prompting the user for another command.

backup copy. A copy, usually of a file or group of files, that is kept in case the original file or files are unintentionally changed or destroyed.

backup diskette. A diskette containing information copied from a fixed disk or from another diskette. It is used in case the original information becomes unusable.

bad block. A portion of a disk that can never be used reliably.

bandwidth. The difference, expressed in hertz, between the highest and the lowest frequencies of a range of frequencies.

Base System Program. That part of the AIX Operating System that contains operating system files and data. The AIX Operating System consists of the Base System Program, which contains the operating system files, and the Virtual Resource Manager, which manages RT PC hardware and software resources. See also *Virtual Resource Manager*.

baseband system. A communications system whereby information is encoded, modulated, and impressed on the transmission medium without shifting or altering the frequency of the information signal. At any point on the medium only one information signal at a time is present.

basic conversation. A connection between two transaction programs that allows them to exchange logical records that contain a two-byte prefix that specifies the length of the record. LUs 1, 2, and 3 do not use the two-byte prefix; however, LU 1, 2, and 3 conversations must be basic conversations. This conversation type is used by service transactions and LU 1, 2, and 3 application transaction programs. Contrast to *mapped conversation*.

batch compilation. A method of compiling programs, as a background process, without the continual attention of an operator.

batch file. A number of similarly grouped programs or data to be input to the computer for processing in a single run.

batch printing. Queueing one or more documents to print in a separate job as a background process. The operator can type or revise additional documents at the same time.

batch processing. (1) The processing of data or the accomplishment of jobs accumulated in advance, in such a manner that the user cannot further influence processing while it is in progress. (2) The processing of data accumulated over a period of time.
(3) Loosely, the execution of programs serially.
(4) Pertaining to the technique of executing a set of programs such that each is completed before the next program in the set is started. (5) In realtime systems, the processing of related transactions that that have been grouped together. (6) A processing method in which a program or programs process records with little or no operator action. Contrast with *interactive processing*.

baud. (1) The number of changes in signal levels, frequency, or phase per second on a communication channel. If each represents 1 bit of data, baud is the same as bits per second. However, it is possible for one signal change (1 baud) to equal more than 1 bit of data. (2) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one-half dot cycle per second in Morse code, one bit per second in a train of binary signals, and one 3-bit value per second in a train of signals, each of which can assume one of eight different states. (3) In asynchronous transmission, the unit of modulation rate corresponding to one unit interval per second; for example, if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

bid. In the contention form of invitation or selection, an attempt to gain control of a line in order to transmit data.

binary. (1) Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. (2) Pertaining to a selection, choice or condition that has two possible different values or states, such as on/off or yes/no. (3) Pertaining to a fixed radix numeration system having a radix of two.

binary file. A file that contains codes that are not part of the ASCII character set. Binary files utilize all 256 possible values for each byte in the file.

binary synchronous communication (BSC). (1) Communication using binary synchronous line discipline. Bisynchronous communications tend to be faster and more reliable than asynchronous communications, but it can be more expensive to build a binary synchronous network from scratch. (2) A form of telecommunication line control that uses a standard set of transmission control characters and control character sequences, for binary synchronous transmission of binary-coded data between stations.

BIND. See *Bind Session (BIND)*.

bind. (1) To associate a variable with an absolute address, identifier, or virtual address, or with a symbolic address or label in a program.

bind image. In SNA, the session parameters that the system services control point (SSCP) sends to the primary logical unit (PLU) and the PLU sends in the BIND request to the secondary logical unit (SLU); these parameters specify the proposed protocol options for an LU-LU session.

BIND password. One of the two communication security passwords. In an LU-LU session, it is the password that the system checks against the remote system to verify that the program to which the user is connected is the correct one. See also *node verification* and *communication authority password*.

Bind Session (BIND). In SNA products, a request to activate a session between two logical units.

bit. (1) Either of the binary digits 0 or 1 used in computers to store information. See also *byte*.
(2) Synonym for binary digit.

bit BLT. Bit block-level transfer. In graphics, the movement of a rectangular region of image data from one location to another (possibly with logical operations such as **or**).

bit clocking. In an R-232-C interface, the field that indicates which piece of equipment, either the modem (DCE) or the computer (DTE), provides the clock signal for synchronized data transactions.

bit field. A member of a structure or union that contains one or more named bits.

bit rate. The speed at which bits are transmitted, usually expressed in bits per second.

bit-mapped display. A display with a display adapter that has a hardware representation of each separately addressable point on the display. The hardware representation can be processor memory or adapter memory.

block. (1) A group of contiguous records recorded or processed as a unit. Blocks are spearated by interblock gaps and each block may. contain one or more records. (2) In data communications, a group of records that is recorded, processed, or sent as a unit. (3) In programming languages, a compound statement that coincides with the scope of at least one of the declarations contained within it. A block may also specify storage allocation or segment programs for other purposes.

block-check character (BCC). (1) In longitutinal redundancy checking and cyclic redundancy checking, a character that is transmitted by the sender after each message block and is compared with a block-check character computed by the receiver to determine whether if the transmission was successful.
(2) In BSC, the character used to check that all of the bits transmitted were also received.

block device. One of the types of files in the AIX file system, described by an i-node.

block special file. A special file that provides access to a device which is capable of supporting a file system.

border. A visual boundary that separates a displayed object from everything else on a screen.

bps. Bits per second. In serial transmission, the instantaneous bit speed with which a device or channel transmits a character.

BPS. Bytes per second.

branch. (1) In a computer program an instruction that selects one of a number of alternative sets of instructions. A conditional branch occurs only when a specified condition is met. (2) In a network, a path that connects two adjacent nodes and that has no intermediate nodes. (3) A set of instructions that are executed between two successive branch instructions. (4) Loosely, a conditional jump.

breakpoint. (1) A place in a computer program, usually specified by an instruction, where execution may be interrupted by external intervention or by a monitor program. (2) An instruction in a program for halting execution. Breakpoints are usually established at positions in a prgram where halts, caused by external intervention, are convenient for restarting. (3) A place in a program, specified by a command or condition, where the system halts execution and gives control to the work station user or to a specified program.

bridge. (1) In the connection of local loops, channels, or rings, the equipment and techniques used to match circuits and facilitate accurate data transmission. (2) A functional unit that connects two local area networks (LANs) that use the same logical link control (LLC) procedure but may use different medium access control (MAC) procedures.

broadband. Transmission media and techniques that use a broad frequency range, divided into sub-bands of narrower frequency, so that different kinds of transmission can occur at the same time.

broadcast. Simultaneous transmission of data to more than one destination.

broadcast topology. The topology in which all stations are connected in parallel with the medium and are capable of concurrently receiving a signal transmitted by any other station connected to the medium.

BSC. See *binary synchronous communication*.

buffer. (1) A temporary storage unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of storage, temporarily reserved for performing input or output, into which data is read, or from which data is written. (3) A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another. (4) An isolating circuit used to prevent a driven circuit from influencing the driving circuit.

bug. An error in a program or a logic problem in the intent of the program. Also, a special on-screen marker serving as a cursor for a mouse input system.

bus. (1) In a processor, a physical facility on which data is transferred to all destinations, but from which only addressed destinations may read in accordance with appropriate conventions. (2) A computer configuration in which processors are interconnected in series. (3) One or more conductors used for transmitting signals or power.

button. (1) A word or picture on the screen that can be selected. Once selected and activated, a button begins an action in the same manner that pressing a key on the keyboard may begin an action. (2) A key on the mouse that is used to select buttons on the display screen or to scroll the display image.

byte. (1) The amount of storage required to represent one character; a byte is 8 bits. (2) A binary character operated on as a unit and usually shorter than one word. (3) String consisting of a certain number of bits, usually eight, treated as a unit, and representing a character. (4) A group of eight adjacent binary digits representing one EBCDIC character.

C - D

C. Celsius.

C language. A general-purpose programming language that is the primary language of the &bos..

C library. A system library that contains common C language subroutines for file access, string operators, character operations, memory allocation, and other functions.

cable. The physical media for transmitting signals; includes copper conductors and optical fibers.

cache. A high speed buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

call. (1) To activate a program or procedure, usually by specifying the entry conditions and jumpting to an entry point. (2) To transfer control to a procedure, program, routine, or subroutine. (3) In data communications, the action necessary in making a connection between two stations on a switched line. (4) T attempt to contact a user, regardless of whether the attempt is successful.

call-back. A characteristic of the the UUCP file USERFILE that tells a remote system whether the local system it tries to access will call back to check its identity.

call-in line. A UUCP communication feature that receives files and commands sent from another system.

callout. An &ix. kernel parameter establishing the maximum number of scheduled activities that can be pending simultaneously.

call-out line. A UUCP communications feature that sends files and commands to another system.

cancel. To end a task before it is completed.

caps (and small caps). Capital letters, an upper-case font. "Caps and small caps" is a printing style using two type sizes of a single upper-case font. The smaller size is used where lower-case font would ordinarily be used.

card. An electronic circuit board that is installed by plugging it into a slot in the system unit.

carrier. In data communication, a continuous frequency that can be modulated or impressed with an information-carrying signal.

carrier return. (1) In text data, the action that indicates to continue printing at the left margin of the next line. Equivalent to the carriage return of a typewriter. (2) A keystroke generally indicating the end of a command line.

carrier sense. In a local area network, a signal provided by the physical layer to the medium access control sublayer, which indicates that one or more stations are currently transmitting on the medium.

carrier sense multiple access with collision detection (CSMA/CD). The generic term for a class of medium access procedures that (1) allows multiple stations to access the medium at will without explicit prior coordination, (2) avoids contention by way of carrier sense and deference, and (3) resolves contention by way of collision detection and transmission.

carrier signal. A signal with a constant frequency that can be modulated to carry a data signal.

case sensitive. The ablility to distinguish between uppercase and lowercase letters.

cathode ray tube (CRT). A vacuum tube in which a beam of electrons can be moved to draw lines or to form characters or symbols on its luminescent screen.

CCITT. See *Consultative Committee on International Telegraphy and Telephone*.

CDSTL. See *connect data set to line*.

CE cylinder. A cylinder for use as a save area for a dump of control storage, the history file, and main storage.

centrex. Central office telephone equipment serving subscribers at one location on a private automatic branch echange basis. The system allows such services as direct inward dialing, direct distance dialing, and console switchboards.

channel. (1) A path along which signals, or data passes. (2) The portion of a storage medium that is accessible to a given reading or writing station. (3) In data communication, a means of one-way transmission. (4) A functional unit, controlled by the processor, that handles the transfer of data between processor storage and local peripheral equipment. (5) A device connecting the processor to input and output devices. (6) One of 32 bits in a table used to represent which event classes are active or inactive. The most significant bit is called channel 0 and the least significant bit is called channel 31.

character. (1) A symbol used in printing, including Greek and Roman alphabetic, numeric, punctuation, mathematical notation, tabbing, spacing, and other symbols; especially any of the characters defined by ASCII. (2) A member of a set of elements used for the representation, organization, or control of data. (3) A letter, digit, or other symbol used as part of the organization, control, or representation of data. A character is often in the form of a spacial arrangement or adjacent or connected strokes.

character class. (1) Ranges of characters that match a single character in the input stream. (2) A set of characters enclosed in sequence, or square, brackets.

character constant. (1) A character with a constant value. (2) A character or an escape sequence enclosed in single quotation marks. Some compilers allow more than one character or escape sequence in a character constant. The &cnamef. compiler allows you to place one to four characters in a character constant. The character constant, however, can occupy no more than four bytes of storage. Thus, although you can place four 1-byte characters in a character constant, you can place only two 2-byte characters in a character constant.

character delete. In text data, the action that erases the character at the current cursor location and moves any trailing text one character position to the left.

character device. A type of file in the AIX file system, described by its i-node.

character display. A display that uses a character generator to display predefined character boxes of images (characters) on the screen. This kind of display cannot address the screen any less than one character box at a time. Contrast with *All Points Addressable display*.

character generation. (1) In phototypesetting, the process of creating characters by projection of light from a source through an imaging film onto a photosensitive medium. (2) The process of creating a video image on the (phosphor coated) back of the face of a CRT by projection of electronically generated pixils.

character graphics. (1) The visual representation of a character, defined by toned or intoned picture elements (pels). (2) Graphics that are composed of symbols printed in a monospace font. Some symbols are stand-alone, others are intended for assembling larger figures. A popular type of printed character graphics is a print graphics character set that corresponds to some video graphics character set. This allows the appearance of a video display screen to be accurately reproduced on paper.

character key. (1) A keyboard key that allows the user to enter the character shown on the key. Compare with *function keys*. (2) In word processing, a control used to process text, one character at a time.

character pacing. The sending of a character and waiting for the character to be returned. Contrast with *line pacing*.

character position. On a display, each location that a character or symbol can occupy.

character set. (1) A group of characters used for a specific reason; for example, the set of characters a printer can print or a keyboard can support.

character special file. A special file that provides access to an input or output device. The character interface is used for devices that do not use block I/O. See also *block special file*.

character string. (1) A sequence of consecutive characters. (2) A string consisting solely of characters.

character translation. In AIX international character support, the dd command and various conversion subroutines that translate between extended characters and ASCII escape strings to preserve unique character information.

checksum. (1) the sum of a group of data associated with the group and used for checking purposes.
(2) On a diskette, data written in a section for error detection purposes.

check digit. (1) A digit usef for the purpose of performing a check. (2) A check key consisting of a single digit. (3) The right-most digit of a self-check field used to check the accuracy of the field.

child process. In the &osix., a child is a process spawned by a parent process that shares resources of parent process.

child resource. (1) Pertaining to a secured resource, either a file or library, that uses the user list of a parent resource. A child resource can have only one parent resource. Contrast with *parent*.

choice. An option in a pop-up or menu used to influence the operation of the system.

chord. In graphics, a short line segment whose end points lie on a circle. Chords are a means for producing a circular image from straight lines. The higher the number of chords per circle, the smoother the circular image.

CICS. Customer Information Control System.

CICS/VS. Customer Information Control System for Virtual Storage, which operates on a host system.

CID. See *connection identifier*.

circuit switching. A process that, on demand, connects two or more data terminal equipments (DTEs) and permits the exclusive use of a data circuit between them until the connection is released. Synonymous with *line switching*.

class. Pertaining to the I/O characteristics of a device. &ix. devices are classified as block or character.

client. On a network, the computer requesting services or data from another computer.

clock. (1) A device that generates periodic signals used for synchronization. (2) In data communication, equipment that provides a time base used in a transmission system to control the timing of certain functions such as sampling, and to control the duration of signal elements.

clocking. (1) In binary synchronous communication, the ue of clock pulses to control synchronization of data and control characters. (2) In data communications, a method of controlling the number of data bits sent on a communications line in a given time.

close. (1) To end an activity and remove that window from the display. (2) A data manipulation function that ends the connection between a file and a program. Contrast with *open*.

cluster. (1) Any configuration of interconnected workstations for the purpose of sharing resources (for example, Local Area Networks, host attached workstations, etc.) (2) A group of storage locations allocated at one time. (3) A station that consists of a control unit (cluster controller) and the terminals attached to it.

cluster controller. A device that can control the input/output operations of more than one device connected to it. A cluster controller can be controlled by a program stored and executed in the unit, or it can be entirely controlled by hardware.

cluster controller node. A peripheral node that can control a variety of devices.

coaxial cable. A cable consisting of one conductor, usually a small copper tube or wire, within and insulated from another conductor of larger diameter, usually copper tubing or copper braid.

code. (1) Instructions for the computer. (2) To write instructions for the computer; to program. (3) A representation of a condition, such as an error code. (4) To represent data or a computer program in a symbolic form that can be accepted by a data processor.

code page. (1) An assignment of graphic characters and control function meanings to all code points. (2) Arrays of code points representing characters that establish ordinal sequence (numeric order) of characters. &ix. uses 256-character code pages. Code page P0 consists of &1c.s that represent the ASCII, ISO, and EBCDIC character sets and additional characters and symbols. Lower code page P0 (0-127 ordinal) is the ASCII character set. Additional code pages consist of code points for 2-byte character representations. See *code point* and *extended character.*

code point. (1) A 1-byte code representating one of 256 potential characters. (2) A 1- or 2-byte representation of a character. A byte can contain a single-shifted bit that indicates that the second byte is a part of the same code point, and indicates the code page of the character. The second byte (only byte in the case of a 1-byte character) places the character in the code page array.

code segment. See *segment*.

collating sequence. The sequence in which characters are ordered within the computer for sorting, combining, or comparing.

collation. The process of character and string sorting based on alphabetical order and equivalence class.

collation table. Provides an ordered character set and character equivalence classes used by functions.

collision. A unwanted condition caused by concurrent transmissions on the medium, which results in garbled data.

color display. A display device capable of displaying more than two colors and the shades produced via the two colors, as opposed to a monochrome display.

color table. A table that maps color names or values to the actual color on the display.

color expansion operation. A graphics programmming operation that occurs automatically when the source pixel map data area contains only one byte per pixel, and the destination pixel map data area is a color display adaptor buffer frame defined to have more than one bit per pixel.

column. (1) A vertical arrangement of characters or other expressions. (2) A character position within a print line or on a display. The positions are numbered from 1, by 1, starting at the leftmost character position and extending to the rightmost position.

command. (1) A request from a terminal to perform an operation or execute a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command. (2) An instruction that directs a control unit or device to perform an operation or set of operations. (3) In data communication, an instruction represented in the control filed of a frame and transmitted by a primary or combined station. It causes the addressed station to execute a data link control function.

command bar. The horizontal area at the top of the screen that contains commands that you can use in the current window. This line appears when the WINDOWS window, an APPLICATIONS window, a FILES window, or a TOOLS window is active.

command interpreter. A program that sends instructions to the kernel; also called an interface. See *shell.*

command line. (1) On a display screen, a display line in which only a command can be entered. (2) The area to the right of a prompt for entering commands or program names.

command line editing keys. Keys for editing the command line.

command name. (1) The first or principal term in a command. A command name does not include parameters, arguments, flags, or other operands. (2) The full name of a command when an abbreviated form is recognized by the computer (for example, print working directory for pwd).

command pop-up. A pop-up area in which you type in commands. The command pop-up appears at the bottom of the screen when you press the Command or Previous Command key.

command programming language. Facility that allows programming by the combination of commands rather than by writing statements in a conventional programming language.

command substitution. The ability to capture the output of any command as an argument to another command by placing that command line within grave accents (\` \`). The shell first executes the command or commands enclosed within the grave accents and then replaces the whole expression, including grave accents, with their output. This feature is often used in assignment statements.

command word. The name of the 16-bit units used for storing graphic primitive strings. The first command word determines the primitive type and sets the length of the string. Subsequent command words contain information in multiples of quid, or four bits of data.

comment. (1) In programming languages, a language construct for the inclusion of text in a program that has no impact on the execution of the program. (2) A statement used to document a program or file, which may be helpful in running a job or reviewing an output listing. (3) A comment contains text that the compiler ignores. Comments begin with the /* characters, end with the */ characters, and span any number of lines. Comments cannot be nested.

commit. To cause all changes that have been made to the data base file since last committment operation to become permanent and the records to be unlocked so they are available to other users.

commit operation. An operation that saves a file to permanent storage.

common carrier. Any government-regulated company that provides communication services to the general public.

communication authority password. One of the two communications security passwords. It controls access to communication configuration menus so that only authorized persons can change the profiles, encrypt a portion of the communication profile data base, or control the startup of SNA processes. The password must be a 30-to-80-character phrase, with interior blanks allowed. See also *BIND password*.

communication channel. An electrical path that facilitates transmission of information from one location to another.

communication controller. (1) A device that directs the transmission of data over the data links of a network; its operation may be controlled by a program executed in a processor to which the controller is connected, or it may be controlled by a program executed within the device. (2) A type of communication control unit with operations that are controlled by one or more programs stored and executed in the unit, for example the IBM 3725 Communications Controller.

communications. See *data communications*.

communications adapter. A hardware feature that enables a computer or device to become a part of a data communications network.

Communications Authority Password. Used to control access to communications configuration menus so that only authorized persons may change the profiles, encrypt a portion of the communications profile data base, or control the startup of &snanames. processes.

communications line. The line over which data communications takes place; for example, a telephone line.

communications link. See *data link*.

compatibility. (1) The ability to perform tasks identically in different environments without major modifications. (2) The capability of a functional unit to meet the requirements of a specified interface.

compilation time. The time during which a source program is translated from a high-level language (such as C language) into a machine language.

compile. (1) To translate a program written in a high-level programming language into an intermediate language, assembly language, or a machine language. (2) The computer actions required to transform a source file into an executable object file. (3) To prepare a machine language program from a computer program written in another language by making use of the overall logic structure of the program.

compiler. (1) A program that translates a source program into an executable program (an object program). (2) A program that translates instructions written in a high-level programming language into machine language.

complement of a number. The value that can be added to the number to equal a given value.

compress. (1) To move files and libraries together on disk to create one continuous area of unused space.. (2) In data communications, to delete a series of duplicate characters in a character string. (3) To save storage space by eliminating gaps, empty fields, redundancy, or unnecessary data to shorten the length of records or files.

compressed output. See *compression*.

compression. (1) A technique for removing strings of duplicate characters, gaps, empty fields, and trailing blanks before transmitting data. (2) In SNA, the replacement of a string of up to 64 characters by an encoded coltrol byte to reduce the length of the data stream sent to the LU-LU session partner.

concatenate. (1) To link together. (2) To join two character strings.

concurrent groups. The ability to access files from many groups at the same time.

condition. (1) One of a set of specified values that a data item can assume. (2) An expression that can be evaluated to a value of either true or false when a program is compiled or run.

conditioning. (1) The use of indicators to control when calculations or output operations are to be performed. (2) In data communications, the addition of equipment to a nonswitched voice-graded channel to provide minimum values of line characteristics required for data transmission.

conditioning indicator. An indicator used to indicate when calculations are done or which attributes apply to a format or format field.

conduit. A pipe for protecting electrical wires or cables.

configuration. (1) The arrangement of a computer system or network as defined by the nature, number, and chief characteristics of its functional units. The term can refer to both hardware and software configurations. (2) The group of machines, devices, and programs that make up a data processing system or network.

configuration file. A file that specifies the characteristics of a system or subsystem, for example, the &ix. queueing system.

configure. To describe to a system the devices, optional features, and program products installed on a system.

CONFIRM. A request which asks the remote transaction program to tell whether the last transmission was successfully received.

confirmation. A transmission by a receiver that permits a sender to continue.

CONFIRMED. A response to the CONFIRM request indicating that the remote site received the transmission without detecting any errors.

connect data set to line (CDSTL). In SNA, an option that determines how the DTR (Data Terminal Ready) signal to the modem operates. It is used if DTR indicates an unconditional command from the DTE (data terminal equipment) to the attached DCE (data circuit terminating equipment) to connect to or remove itself from the network.

connect-time accounting. The record of the amount of time each user spends logged into the system.

connection. (1) The linking together of two or more logical units (LUs) for the purpose of providing communications channels between them for the application programs running at the respective LUs. (2) The network path that links together two LUs in different nodes to enable them to establish communications.

connection identifier (CID). A value used to identify a resource. It is returned to the connecting program after connect processing has established a session and must be used on subsequent requests to the resource.

connection profile. A data management file that contains parameters that associate other defined profiles to the connection of two logical units.

connector. (1) An electrical part used to join two other electrical parts. (2) A flowchart symbol that represents a break in a flowline and indicates where the flowline is continued. (3) A means of establishing electrical flow.

consecutive processing. The processing of records in the order in which they exist in a file. Same as *sequential processing*. See also *random processing*.

consistent. Pertaining to a file system, without internal discrepancies.

console. (1) A part of a computer used for communications between the operator or maintenance engineer and the computer. (2) The main &ix. display station. (3) A device name associated with the main &ix. display station.

console bell. See *BEL*.

console display. A display that can be requested only at the system console, from which an operator can display, send, and reply to messages and use all control commands.

constant. (1) A data item with a value that does not change. Contrast with *variable*. (2) Data that have an unchanging, predefined value to be used in processing.

constant expression. An expression having a value that is determined during compilation and that cannot be changed during execution.

constant field. A field defined by a display format to contain a value that does not change.

constant pool. In VRM TOC object module format, the address of a common area of constants and pointers used by each routine.

constant-width characters. A character set designed so each character is of the same width as the other characters.

Consultative Committee on International Telegraphy and Telephone (CCITT). A United Nations Specialized Standards group, whose membership includes common carriers concerned with devising and proposing recommendations for international telecommunications representing alphabets, graphics, control information, and other fundamental information interchange issues.

contact port. See *well-known port.*

contention. (1) In a local area network, a condition on a communications channel when two or more stations are allowed by the protcol to start transmitting concurrently and thus risk collision. (2) A condition on a session when two programs try to start a conversation at the same time.

contention resolution. The process of resolving contention (medium access control conflicts) according to a defined algorithm.

context address. A regular expression enclosed in / (slashes).

context search. A search through a file whose target is a character string.

continuation line. A line of a source statement into which characters are entered when the source statement cannot be contained on the previous line of lines.

control block. (1) A storage area used by a program to hold control information. (2) the circuitry that performs the control functions such as decoding microinstructions and generating the internal control signals that perform the operations requested.

control character. (1) A character, occurring in a particular context, that initiates, modifies, or stops any operation that affects the recording, processing, transmission, or interpretation of data (such as carriage return, font change, and end of transmission). (2) A non-printing character that performs formatting functions in a text file.

control commands. Commands that allow conditional or looping logic flow in &shell. procedures. A control command does not run a procedure and cannot be used in a procedure.

Control Point Profile Name. The name of the control point profile that defines the node ID of the physical unit associated with the attachment.

control program (CP). Part of the &osix. system that determines the order in which basic functions should be performed.

control station. The primary or controlling computer on a multipoint line. The control station controls the sending and receiving of data.

Control Unit Terminal (CUT) mode. An IBM protocol used for communications with an IBM 3174 or 3274 Control Unit. In this protocol, the &o. is emulating a dumb 3278/79 terminal, and the 3274 is responsible for enforcing the protocol.

controlled cancel. The system action that ends the job step being run, and saves any new data already created. The job that is running can continue with the next job step.

control terminal. Any active terminal at which the user is authorized to enter commands affecting system operation.

converged pheripheral node. A type of physical unit that has limited addressing and path control routing capabilities. It provides general connectivity to other SNA nodes, and supports parallel sessions, multiple sessions per LU, primary and secondary LUs, and multiple lines per node.

conversation. (1) The logical connection between a pair of transaction programs for serially sharing a session between type 6.2 logical units from transaction to transaction. While a conversation is active, it has exclusive use of an LU-LU session as delimited by a distinct bracket; successive conversations may use the same session. (2) An interchange of information between two application programs. (3) A pathway between two application programs that allows them to transfer information between each other. (4) Interaction between a computer and a user by means of a keyboard.

conversation correlator. An internal SNA identifier used by the LU services to track which applications are using which conversations. A one- to eight-byte identifier that is assigned by the attach function and maintained by LU services.

conversion. (1) In programming languages, the transformation between values that represent the same data item but belong to different data types. (2) A change in the form of a value. For example, when you add values having different data types, the compiler converts both values to the same form before adding the values.

conversion code. In a print function call, a specification of the type of the value, as the value is to be printed (in octal format, for example).

conversion modifier. In a print function call, a specification of how a value is to be printed (left justified, for example).

conversion specification. In a print function call, a specification of how the system is to place the value of zero or more format parameters in the output stream. Each conversion specification contains a % symbol. The % is followed by conversion modifiers and a conversion code.

copy. (1) The action by which the user makes a whole or partial duplicate of an already existing data object. (2) To read data from a source, leaving the source data unchanged, and to write the same data elsewhere in a physical form that may differ from the source.

copy-on-write. An option that creates a mapped file with changes that are saved in the system paging space, instead of saving the changes to the copy of the file on the disk.

counter. A register or storage location used to accumulate the number of occurrences of an event.

coupler. A device connecting a modem to a telephone network.

CP. See *control program*.

crash. An unexpected interruption of computer service, usually due to a serious hardware or software malfunction.

CRC. See *cyclical redundancy check character*.

creation date. The date at the time a file is created.

crosstalk. The disturbance caused in a circuit by an unwanted transfer of energy from another circuit.

CRQ. Call request.

CRT. See *cathode ray tube*.

CSMA/CD. See *carrier sense multiple access with collision detection* .

currency time. The time at which a user reads news items. The news command considers only the items posted after this time to be current for the user.

current directory. the currently active directive; the directory that is searched when you enter a filename without indicating the directory that contains the fielname. When you specify a file name without specifying a directory, the system assumes that the file is in the current directory.

current file. (1) The file you are editing. If you are using multiple windows, it the file containing the cursor. (2) In make, the file that the make command is working with at a given moment; make replaces the macro $* with the name of the current file.

current host. See *local host*.

current line. (1) The line on which the cursor is located. (2) Usually the last line that is affected by a command.

current record. (1) The record pointed to by the current line pointer. (2) The record that is currently available to the program.

current screen. In &lcur., the actual image that is currently on the terminal.

current working directory. See *current directory*.

curses. The system library that contains the control functions for writing dataa to and getting data from the terminal screen. (Do not use.) See also *extended curses*.

cursor. (1) A movable symbol (such as an underline) on a display, used to indicate to the operator where the next typed character will be placed or where the next action will be directed. In &usys., the cursor is called a text cursor. (2) A marker that indicates the current data access location within a file. See also *pointing cursor*.

cursor movement keys. The directional keys used to move the cursor without altering text.

Customer Information Control System (CICS). An IBM licensed program product that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining data bases.

customize. (1) To describe (to the system) the devices, programs, users, and user defaults for a particular data processing system or network. (2) to design a data processing installation or network to meet the requirements of particular users.

customize helper program. A program that locates, converts, and constructs the Define Device structure unique to the VRM device driver being loaded into the character string it receives. It must additionally locate, convert, and construct the AIX kernel device driver that calls the AIX device driver initialization routine.

CUT. See *control unit terminal*.

cycle time. (1) The time elapsed during one cycle of the processor. Cycle time varies from one type of processor to another. (2) the minimum time interval between starts of successive read- write cycles of a storage device.

cyclic redundancy check character (CRC). A character code used in a modified cyclic code for error sensing and correction.

cylinder. (1) All fixed disk or diskette tracks that can be read or written without moving the disk drive or diskette drive read/write mechanism. (2) In a disk pack, the set of all tracks with the same nominal distance from the axis about wich the disk rotates.

daemon process. (1) A background process that is usually started at system start, runs continuously, and performs a function required by other processes. (2) A process begun by the root or the root shell that can be stopped only by the root. Daemon processes generally provide services that must be available at all times, such as sending data to a printer.

DARPA. Defense Advanced Research Projects Agency.

data block. See *block*.

data channel. A device that connects a processor and main storage with I/O control units. Synonym for input/output channel.

data circuit. (1) A pair of associated transmit and receive lines that provide a means of two-way data communications. (2) In SNA, synonym for link connection.

data circuit-terminating equipment (DCE). In a data station, the equipment installed at the user's premises that provides all the functions required to establish, maintain, and terminate a connection, and the signal conversion and coding between the data terminal equipment (DTE) and the line.

data communications. The transmission of data according to a protocol between computers and or remote devices, usually over a long distance.

data link. (1) The assembly of parts of two data terminal equipment that are controlled by a link protocal, and the interconnecting data circuit, that enable data to be transferred from a data source to a data sink. (2) The interconnecting data circuit and the link protocol between two or more equipments; not including the data source or data sink. (3) The physical connection and the connection protocols between units that exchange data over a telecommunications line.

data link control (DLC) protocol. In SNA, the set of rules used by two nodes on a data link to accomplish an orderly exchange of information.

data link control layer. In SNA and X.25, the layer that consists of the link stations that schedule data transfer over a link between two nodes and perform error control for the link.

data link escape (DLE) character. In BSC, a transmission control character usually used in transparent text mode to indicate that the next character is a transmission control character.

data lock. (1) The ensurance of data availability to a single application program as a protection agains conflicting updates to a data record. (2) The system lock that locks data segment into memory.

data stream. (1) All information (data and control information) transmitted over a data chnnel in a single read or write operation. (2) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

data terminal equipment (DTE). The part of data processing unit that serves as a data source, data sink, or both.

data transfer. The movement, or copying, of data from one location and the storage of the data at another location.

data type. (1) In programming languages, a set of values together with a set of permitted operations. (2) The mathematical properties and internal representation of data and functions. The four basic types are integer, real, complex, and logical. (3) An attribute used for defining data as numeric or character. (4) A category that identifies the mathematical qualities and internal representation of data.

datagram. In packet switching, a self-contained packet, independent of other packets, that carries information sufficient for routing from the originating data terminal equipment (DTE) to the destination DTE without relying on earlier exchanges between the DTEs and the network.

DCE. See *data circuit-terminating equipment*.

DD. Device driver.

dead letter file. A file containing messages that could not be sent to a proper destination file.

deadlock. (1) An error condition in which processing cannot continue because each of two elements of the process is waiting for an action by or a response from the other. (2) Unresolved contention for the use of a resource. (3) An impasse that occurs when multiple processes are waiting for the availability of a resource that will not become available because it is being held by another process that is in a similar wait state.

DEALLOCATE. A request to remove the allocation of the specified conversation from the local transaction program.

debug. (1) To detect, locate, and correct errors in a program. (2) To find the cause of problems detected in software.

debugger. A device used to detect, trace, and eliminate errors in computer programs or software.

decimal. (1) Pertaining to a system of numbers to the base 10; decimal digits range from 0 through 9. (2) Characterized by a selection, choice, or condition that has ten possible different values or states.

default. A value, attribute, or option that is assumed when no alternative is specified by the operator.

default value. See *default.*

DEL. See *delete character*

delete. To remove. For example, to delete a file.

delete character (DEL). (1) A control character uesd primarily to obliterate an erroneous or unwanted character. (2) A character that identifies a record to be removed from a file.

delimiter. (1) A character or sequence of characters that marks the beginning or end of character string or unit of data. (2) A character that groups or separates words or values in a line of input.

de-select. To cancel the selection of a button. With a mouse, you de-select a highlighted area with the Select (left) button. Otherwise, you can use the Select key on the keyboard. To de-select a default button, select an alternate button in the selection list.

device. A mechanical, electrical or electronic machine that is designed for a specific purpose and that attaches to your computer, such as a printer, plotter, or disk drive.

device driver. (1) A program that operates a specific device, such as a printer, disk drive, or display. (2) A collection of subroutines that control the interface between I/O device adapters and the processor. (3) A set of routines installed as a part of the AIX kernel to control the transmission of data to and from a device.

device name. (1) the logical or symbolic name reserved by the system that refers to a specific device. (2) The AIX Operating System device name of the network adapter as that name appears in **/etc/system**. Communication functions use this name to get information that defines the interface to the network adapter.

DFT. Distributed function terminal.

diagnostic. Pertaining to the detection and isolation of errors in programs and faults in equipment.

diagnostic aid. A tool (procedure, program, reference manual) used to detect and isolate a device or program malfunction or error.

diagnostic output. Error or status messages produced by processes, in addition to standard output. Synonym for *error output.*

diagnostic routine. A computer program that recognizes, locates, and explains either a fault in equipment or a mistake in a computer program.

dialing directory. In ATE, a list of telephone numbers that can be called with Asynchronous Terminal Emulator (ATE). It is similar to a page in a telephone directory.

dialog. In an interactive system, a series of related inquiries and responses similar to a conversation between two people.

digit. (1) A character that represents a non-negative integer. (2) A symbol that represents one of the non-negative intergers smaller than the radix. (3) Any of the numerals from 0 through 9.

digital signals. (1) Computers and terminals employ digital signals to handle data and control information. These digital pulses are timed and shaped very precisely to indicate whether a bit is on or off, providing the binary notation of 1 or 0 respectively. (2) A discrete or discontinuous signal; a signal whose various states are discrete intervals apart.

direct connection. The attachment of a system, terminal, or other I/O device througha selected communication interface and a limited- length cable. No modem is required.

direct memory access (DMA) device. (1) A component that can read or write to system storage directly, without processor intervention. Two device types are identified: 1) an alternate controller resides on a hardware adapter and 2) a system DMA controller resides on the system planar board. DMA capability permits simultaneous use of input/output devices and the processor. (2) The transfer of data between memory and input/output units without processor intervention.

directory. (1) A type of file containing the names and controlling information for other files or other directories. (2) A table of identifiers and references to the corresponding items of data. (3) An index used by a control program to locate blocks of data that are stored in separate areas of a data set in direct access storage.

disable. To make nonfunctional. In interactive communications, to disconnect or stop a subsystem. Contrast with *enable*.

disabled port. In ATE, a port configuration indicating that a port is ready to call out. Contrast with *enabled port*.

DISC. Disconnect.

discipline. Pertaining to the order in which requests are serviced, for example, first-come-first-served (fcfs) or shortest job next (sjn).

disconnect signal. A signal transmitted to a receiving station to indicate that the channel is to be disconnected.

disconnected mode. In SDLC, a response from a secondary station indicating that it is disconnected and wants to be online.

disk. A storage device made of one or more flat, circular plates with magnetic surfaces on which information can be stored.

disk buffering. The ability of the operating system to temporarily store recently accessed data blocks in memory for increased efficiency of I/O operations.

disk drive. The mechanism used to read and write information on disk.

disk I/O. Fixed-disk input and output.

disk-usage accounting. The record of the number of disk blocks occupied by a user's files. Disk-usage accounting is performed by the acctdisk command.

diskette. A thin, flexible magnetic plate that is permanently sealed in a protective cover. It can be used to store information copies from the disk or another diskette.

diskette drive. The mechanism used to seek, read, and write information on diskettes.

display device. An output unit that gives a visual representation of data.

display screen. The part of the display device that displays information visually.

display station. A input/output device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or received from the computer.

display symbols. The set of character symbols that can be displayed on an HFT display device in KSR mode.

distortion. In data communications, an undesirable change in a waveforms that may occur between two points in a transmission system. The six major forms of distortion are bias, characteristic, delay, end, fortuitous, and harmonic.

distributed computation. Interprocess communication.

distributed function terminal. A terminal that performs operations previously accomplished by the processing unit, such as managing data links, controlling devices, and formatting data.

Distributed Services. An IBM licensed progream that provides the ability to create configuration profiles for remote nodes.

DLC. See *data link control*.

DMA. See *Direct Memory Access (DMA) device*.

DOS library. A system library on the RT PC that contains the DOS (Disk Operating System) functions.

DOS Services. In addition to AIX, a command interpreter that can be used on the RT PC.

DOS Shell prompt. A letter followed by the > character displayed by the DOS Shell to indicate that it is ready to process commands from the standard input device (usually the keyboard).

dot. A symbol (.) that indicates the current directory in a relative path name.

dot dot. A symbol (..) in a relative path name that indicates the parent directory.

dotted decimal. A common notation for Internet addresses, which divides the 32-bit address into four 8-bit fields. The value of each field is specified as a decimal number and the fields are separated by periods (for example, 010.002.000.052, or 10.2.0.52).

download. To transfer data from one computer for use on another one. Typically, users download from a larger computer to a diskette or fixed disk on a smaller computer.

DR1I. Definite response 1 indicator.

DR2I. Definite response 2 indicator.

drive specification. In DOS, the letter-colon pair that specifies a drive.

DSR. (1) Data set ready. (2) See *device status report.*

DTE. See *data terminal equipment (DTE).*

DTR. Data terminal ready. A signal to the modem.

dump. (1) To copy the contents of all or part of storage onto another data medium or to an output device. (2) Data that has been dumped.

dump diskette. A diskette that contains a dump or is prepared to receive a dump.

dump formatter. Program for analyzing a dump.

duplex. Pertains to communications data that can be sent and received at the same time. Same as *full duplex.* Contrast with *half duplex.*

E - F

EBCDIC. See *extended binary-coded decimal interchange code.*

EBCDIC character. Any one of the symbols included in the eight-bit EBCDIC set.

ECB. (1) See *event control bit.* (2) Event control block. (3) Electronic codebook.

ECC. (1) Error checking and correction. (2) Error correction code.

echo. (1) A reflected signal on a communications channel. (2) In computer graphics, the immediate notification of the current values provided by an input device to the operator at the display console.
(3) Inword processing, to print or display each character or line as it is keyed in.

ECMA. European Computer Manufacturers' Association.

edit. To add, change, delete, rearrange, or modify the form or format of data.

edit buffer. A temporary storage area used by an editor.

editor program. A program used to enter and modify programs, text, and other types of documents and data.

effective root directory. When searching for a file, the starting point whose path name begins with / (slash). The chroot system call causes the directory named by the path parameter to become the effective root directory.

embedded blanks. Blanks that are surrounded by any other characters.

emphasized. A form of double-strike printing in which characters are printed in multiple passes (usually two) with a slight offset. The effect is to create an artificial bold type. Emphasized printing is used to fill gaps and rough appearance in dot-matrix character forms. It also prints a bold font without changing the mounted character set.

emulation. (1) The imitation of all or part of one system by another, primarily by hardware, so that the imitating system accepts the same data, executes the same programs, and achieves the same results as the imitated computer system. (2) The use of programming techniques and special machine features to permit a computing system to execute programs written for another system. (3) Imitation; for example, when one computer imitates the characteristics of another computer.

enable. (1) To make functional. (2) In interactive communications, to load and start a subsystem. Contrast with *disable.*

enabled port. In ATE, a port configuration indicating that a port is ready to receive calls. Contrast with *disabled port*

encrypt. To convert clear data into ciphertext.

encryption key. A key generated by the makekey command to use with programs that perfrom encryption. Its input and output are usually pipes.

end signal. In an online conference, a mutually agreed upon character that indicates the end of a comment by a participant. Common end signals are o and oo.

enqueue. To place items in a queue.

enter. (1) To send information to the computer by pressing the Enter key. (2) To place on the line a message to be transmitted from a terminal to the computer.

In programming languages, a language construct within a procedure, designating the start of the execution sequences of the procedure.

entry. (1) An element of information in a table, list, queue, or other organized structure of data or control information. (2) A single input operation on a work station.

entry point. (1) An address or label of the first instruction performed upon entering a computer program, a routine, or a subroutine. A program may have several different entry points, each corresponding to a different function or purpose. (2) In a routine, any place to which control can be passed.

environment. (1) The settings for shell variables and paths set when the user logs in. These variables can be modified later by the user. (2) A named collection of logical and physical resources used to support the performance of a function.

EOF. End of file.

erase. To remove text from a data medium, leaving the medium available for recording new text.

erase character. A character that indicates that the previous character on the command line has been erased. Deprecated term for delete character.

error condition. The state that results from an attempt to execute instructions in a computer program that are invalid or that operate on invalid data.

error counter. A type of error entry generated by device driver components. Certain device drivers can generate retries if an operation is not successful on the first attempt. They use counters to monitor the number and cause of retries, and they contain algorithms that decide when these counters should be sent to the error log.

error device driver. The device driver used by the error daemon process (errdemon) to write error records to the event log.

error entry. A data structure containing a header of identifying information, in addition to several bytes of defined data. Error entries are generated by error points and written to an error log file.

error ID. Part of the data required by an error entry. It is a unique combination of three hexadecimal digits that identifies the component that generated the error entry.

error identifier. A three-character code used to identify error templates and to specify which error entries the error formatter should process. This code is based on the error ID; however, it uses alphanumeric characters instead of hexadecimal digits.

error log. (1) A data set or file in a product or system where error information is stored for later access. (2) A form in a maintenance library that is used to record error information about a product or system. (3) A data set used in a processor to record information about certain hardware and programming events. (4) A record of machine checks, device errors, and volume statistical data.

error message. An indication that an error has been detected.

error output. See *diagnostic output file*.

error point. A group of code statements that generates an error entry from within a software program. Error entries are generated when a software or hardware component encounters an error.

error template. A template in the error log facilities that provides a format for an error report. Error templates can be either pre-defined or user- defined.

error type. One of six categories of errors. The type of an error is determined by the software program that generates the error. When you format an error log, you can specify which types of errors you want to format.

error-correct backspace. An editing key that performs editing based on a cursor position; the cursor is moved one position toward the beginning of the line, the character at the new cursor location is deleted, and all characters following the cursor are moved one position toward the beginning of the line (to fill the vacancy left by the deleted element).

escape character (ESC). (1) The backslash character, used to indicate to the shell that the next character is not intended to have the special meaning normally assigned to it by the shell. (2) A character that suppresses or selects a special meaning for one or more characters that follow. (3) A code extension character used to indicate by some convention that the coded representation following are to be interpreted according to a different code or coded character set.

Ethernet. A 10-megabit baseband local area network using CSMA/CD (Carrier Sense Multiple Access with Collision Detection). The network allows multiple stations to access the medium at will without prior coordination; avoids contention by using carrier sense and deference; and resolves contention by using collision detection and transmission.

event. (1) The enqueueing or dequeueing of an element. (2) An occurrence of significance to a task.

event class. A number assigned to a group of trace points that relate to a specific subject or system component. The defined event classes are listed in the trace profile.

event control bit (ECB). A bit assigned to each queue to signal the arrival or departure of an element.

event log. Two files of a predetermined size, which contain entries about system errors and other system events. When one file is full, the system begins to log events in the second file. When the second file is full, the system begins to overwrite data in the first file with new event entries.

exception. (1) In programming languages, an abnormal situation that may arise during execution, that may cause a deviation from the normal execution sequence, and for which facilities exist in a programming language to define, raise, recognize, ignore, and handle it. (2) An abnormal condition such as an I/O error encountered in processing a data set or a file. (3) One of five types of errors that can occur during a floating point exception: invalid operation, overflow, underflow, division by zero, and inexact results. (4) Contrast to *interrupt* and *signal*. See also *floating point exception*.

exception handler. A set of routines used to detect deadlock conditions or to process abnormal condition processing. This allows the normal execution of processes to be interrupted and resumed.

exception trapping. Performing a user-defined action when an exception occurs.

Exchange Identification (XID) frame. In a logical link control (LLC) header, the frame that conveys the characteristics of the sending host.

exec. See *fork*.

exit value. (1) A code sent to either standard output or standard error on completion of the command. (2) A numeric value that a command returns to indicate whether it completed successfully. Some commands return exit values that give other information, such as whether a file exists. Shell programs can test exit values to control branching and looping.

explicit route (ER). In SNA, the path control network components, including a specific set of one or more transmission groups and any intermediate nodes that connect two subarea nodes. An explicit route is identified by an origin subarea address, a destination subarea address, an explicit-route number, and a reverse explicit-route number. See also *path*.

expression. (1) A representation of a value. For example, variables and constants appearing alone or in combination with operators. (2) In programming languages, a language construct for computing a value form one or more operands; for example, literals, identifiers, array references, and function calls. (3) A configuration of signs.

extended binary-coded decimal interchange code (EBCDIC). A code used on the IBM System/370, and developed for the representation of textual data. EBCDIC consists of a set of 256 eight-bit characters.

extended character. A character other than a 7-bit ASCII character. An extended character can be a 1-byte code point with the 8th bit set (ordinal 128-255) or a 2-byte code point (ordinal 256 and greater).

extended curses. A system library that contains the control functions for writing data to and getting data from the terminal screen. It supports color, multiple windows, and an enhanced character set.

extended interface. Provides a set of full function system calls (readx and writex) to communicate with SNA. These calls contain an extra parameter on the call (a pointer to the structure containing extra function requests). See *interface* and *limited interface*.

extent. A continuous space on disk or diskette that is occupied by or reserved for a particular data set, data space, or file.

external clocking. In data communications, the ability of a modem to provide data clocking.

external modem. A modem that is separate from the unit with which it operates.

external name. (1) A name that can be referred to by any control section or separately assembled or compiled module; a control section name or an entry name in another module. (2) In a program, a name whose scope is not necessarily confined to one block and its contained blocks.

extract. To obtain. For example, to extract information from a file.

FD or FDX. Full duplex. See *duplex*.

feature. A programming or hardware option, usually available at an extra cost.

fiber. A transmission medium that utilizes optical fiber.

field. (1) An area in a record or panel used to contain a particular category of data. (2) The smallest identifiable component of a record. (3) An area in a presentation space into which the program accepts input.

field advance. The action that moves a data cursor from field to field in a forward direction, as determined by the panel layout.

field return. The action that moves a data cursor from field to field in a reverse direction, as determined by the panel layout.

FIFO. See *first-in-first-out*.

file. A collection of related data that is stored and retrieved by an assigned name.

file lock. A means to limit or deny access to a file by other users. A file lock can be a read lock or a write lock.

file name. (1) A name assigned or declared for a file. (2) The name used by a program to identify a file. See also *label*.

file name substitution. The process of the shell recognizing a word (character string) that contains any of the characters *, ?, [, or {, or begins with ~, and replacing it with an alphabetically sorted list of file names that match the pattern of the word. Synonym for globbing.

file owner. The user who has the highest level of access authority to a file, as defined by the file. name.

file specification (filespec). The name and location of a file. A file specification consists of a drive specifier, a path name, and a file name.

file synchronization. (1) The operation that causes pending writes to be written to disk. (2) The coordination of access to a file by multiple processes.

file system. The collection of files and file management structures on a physical or logical mass storage device, such as a diskette or minidisk.

File Transfer Protocol (FTP). In TCP/IP, the protocol that makes it possible to ttransfer data among hosts and to use foreign hosts indirectly.

file type. In AIX, the possible types of files are: ordinary file, directory, block device, character device, and first-in-first-out (FIFO or named pipe).

filetab. An AIX kernel parameter establishing the maximum number of files that can be open simultaneously.

fill characters. (1) A character used to fill a field in storage. (2) Visual representations of enterable character positions on the display (for example, dots in each position or vertical bars between positions).

filter. (1) A command that reads standard input data, modifies the data, and sends it to the display screen. (2) A device or program tht separates data, signals, or materials is accordance with specified criteria.

first level interrupt handler (FLIH). A routine that receives control of the system as a result of a hardware interrupt. One FLIH is assigned to each of the six interrupt levels.

first-in-first-out (FIFO). (1) A named permanent pipe. A FIFO allows two unrelated processes to exchange information using a pipe connection. (2) A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

fixed disk. A flat, circular, nonremoveable plate with a magnetizable surface layer on which data can be stored by magnetic recording.

fixed disk drive. The mechanism used to read and write information on a fixed disk.

flag. (1) A modifier that appears on a command line with the command name that defines the action of the command. Flags in the AIX Operating System almost always are preceded by a dash. (2) An indicator or parameter that shows the setting of a switch. (3) A character that signals the occurence of some condition, such as the end of a word.

flat file. (1) A file that has no hierarchical structure. (2) A one-dimensional or two-dimensional array: a list or table of items. (3) In a relational data base, synonym for relation.

flattened character. An ASCII character created by translating an extended character to its ASCII equivalent in appearance. The code point information is lost; the character cannot be retranslated to an extended character.

FLIH. See *first level interrupt handler*.

flush. (1) Left-justified. Aligned vertically at the left margin, with no indentation. (2) To close a data stream.

FM Header. Function Management Header.

font. A family or assortment of characters of a given size and style.

forced attachment stop. Stops an SNA Services attachment regardless of the possible pending state of the attachment or the connections associated with the attachment. See also *normal attachment stop*.

foreground. (1) A mode of program execution in which the shell waits for the program specified on the command line to complete before returning your prompt. (2) In multiprogramming, the environment in which high-priority programs are executed.

foreign host. Any host on the network except the one at which a particular operator is working; sometimes called *remote host*.

foreign socket. The 16-bit port number.

fork. To create and start a child process.

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, or files. (2) The pattern which determines how data is recorded. (3) To arrange such things as characters, fields, and lines. (4) In programming languages, a language construct that specifies the representation, on character form, of data objects in a file.

formatted diskette. A diskette on which track and control information for a particular computer system has been written but which may or may not contain any data.

formatted file. A file displayed and arranged with particular characteristics, such as line spacing, headings, and number of characters and lines per page. Contrast with unformatted *file*.

frame buffer. A display adapter frame buffer is memory storage containing a representation of a display image.

free list. A list of available space on each file system. This is sometimes called the free-block list.

free-block list. See *free list*.

free space. Space reserved within the control intervals of a key-sequenced data set or file, used for inserting new records into the data set or file in key sequence; also, whole control intervals reserved in a control area for the same purpose. Also called distributed free space.

FTP. (1) File Transfer Program. (2) See *File Transfer Protocol.*

full backup. Backup copies of all the files on the system.

full duplex. Same as *duplex*.

full image dump. Writes all the system's real memory to diskettes (up to four). The dump formatter cannot be used to inspect the data produced by a full image dump.

full path name. The name of any directory or file expressed as a string of directories and files beginning with the root directory.

fully-qualified name. A qualified name that includes all names in the hierarchical sequence above the structure member to which the name refers, as well as the name of the member itself.

function. (1) A synonym for procedure. The C language treats a function as a data type that contains executable code and returns a single value to the calling function. (2) A specific purpose of an entity, or its characteristic action. (3) A machine action such as carriage return or linefeed. (4) A subroutine that returns the value of a single variable and that usually has a single exit, such as subroutines that compute mathematical functions.

function call. An expression that moves the path of execution from the current function to a specified function and evaluates to the return value provided by the called function. A function call contains the name of the function to which control moves and a parenthesized list of arguments.

function keys. Keys that request actions but do not display or print characters. Included are the keys that normally produce a printed character, but when used with the code key produce a function instead. Compare with *character key*.

function management (FM) profile. In SNA, a specification of various data flow control protocols (such as RU chains and data flow control requests) and FDM options (such as the use of FM headers, compression, and alternate codes) supported for a particular session. Each function management profile is identified by a number.

G - H

gateway. (1) An entity operating above the link layer, which translates, when required, the interface and protocol used by one network into those used by another distinct network. See also *active gateway.* (2) The network connecting hosts.

gateway host. A host that connects independent networks. It has multiple interfaces, each with a different name and address.

Gateway-to-Gateway Protocol (GGP). the protocol used by a gateway to determine connectivity to networks and neighbor gateways, and to implement the shortest-path routing algorithm.

gather. For input/output operations, reading data from noncontiguous memory I locations to write to a device. See *scatter* (antonym).

A means of referencing items in terms of time and ancestry so that an item without antecedents is designated as the first (n-th) generation and subsequent derivations are designated as n-1, n-2, and so on.

generation. (1) For some remote systems, the translation of configuration information into machine language.

GGP. *See Gateway-to-Gateway Protocol.*

GID. See *group number*.

global. (1) In programming languages, pertaining to the relationship between a language object and a block in which the language object has a scope extending beyond that block but contained within an encompassing black. (2) Pertaining to that which is defined in one subdivision of a computer program and used in at least one other subdivision of the program. (3) Pertains to information available to more than one program or subroutine.

global action. An action having general applicability, independent of the context established by any task.

global area. (1) A storage area used for communication between two or more main programs. (2) An uninitalized portion of a partition accessible by any program of a task set in the partition at a given time. The same area may be used by other task sets that execute in the same partition.

global character. The special characters * and ? that can be used in a file specification to match one or more characters. For example, placing ? in a file specification means any character can be in that position.

global data. Data that can be addressed by any process while in kernel mode. It contains such tables as the open file table and process table, and other data such as buffer pointers, maintained by the kernel.

global search. In word processing, the process of having the system look through a document for specific characters, words, or groups of characters.

GPS. See *graphics primitive string.*

grammar rules. The structure rules in a parser program. See also *parser* and *ambiguous grammar rules.*

grandchild process. A second generation child process; a proces forked from a child process. ?? (PT & I, 4-21)

granularity. The extent to which a larger entity is subdivided. For example, a yard broken into inches has finer granularity than a yard broken into feet.

graphic character. A character that can be displayed or printed.

graphics. A type of data created from fundamental drawing units such as lines, splines, curves, polygons, and so forth.

graphics primitive string (GPS). (1) A single item of graphics operation. gd.The format used for storing graphics file data. A GPS is composed of up to five types of graphical data: comments, lines, arcs, text, and hardware.

Graphics Support Library (GSL). An application programming interface to various output devices.

Graphics Kernel System (GKS). An ANSI standard for interfacing graphics applications programs to computer systems and computer terminals.

group. (1) A set of related records that have the same value for a particular field in all records. (2) A series of records logically joined together. (3) A series of lines repeated consecutively as a set on a full-screen form or full screen panel. (4) A collection of users who can share access authorities for protected resources. (5) A list of names that are known together by a single name.

group ID (GID). See *group number.*

group leader process. The first accessible process in a process group. For example, when a user logs in, the shell process is the group leader process, and any process descendents are in the process group.

group name. A name that uniquely identifies a group of users to the system.

group number (GID). A unique number assigned to a group of related users. The group number can often be substituted in commands that take a group name as an argument.

GSL. See Graphics Support Library.

half duplex. Pertains to communications in which data can be sent in only one
direction at a time. Contrast with *duplex.*

half-session. A component that provides data flow control and transmission control at one end of a session.

handler. A software routine that controls a program's reaction to specific external events, such as an interrupt handler.

handset. The part of a telephone used for talking and listening.

handshaking. Before transmitting data, modems and equipment must establish an electrical path and synchronization. The process used to establish the path and synchronization is called handshaking.

hard copy. A printed copy of machine output in a visually readable form; for example, printed reports, listings, documents, and summaries.

hardware. The physical equipment of computing and computer-directed activities. The physical components of a computer system.

HD or HDX. See *half duplex.*

HDR. See *header label.*

header. (1) Constant text that is formatted to be in the top margin of printed pages in a document. (2) System-defined control information that precedes user data. (3) The portion of a message that contains control information for the message such as destination fields, originating station, and priority level.

header label (HDR). A special set of records on a diskette describing the contents of the diskette.

header record. A record at the beginning of a file that details the sizes, locations, and other information that follows in the file.

help. One or more display images that describe how to use application software or how to do a system operation.

help file. A file, separate from the source code of a program, that contains help definitions in a special help format that can be ready by message services.

help pop-up. A pop-up area on the screen produced by pointing to an object and pressing the help key.

helper. A program used by the &ednames. editor to provide extra function for a particular type of data file.

here document. Data contained within a &shell. program or procedure (also called *inline input*).

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. See *hexadecimal*.

hexadecimal. Pertaining to a system of numbers to the base sixteen; hexadecimal digits range from 0 (zero) through 9 (nine) and A (ten) through F (fifteen).

hexadecimal constant. The characters **0x** or **0X** (zeroX) followed by any digits **0** through **9** and uppercase or lowercase letters **A** through **F**.

HFT. See *high function terminal*.

hidden file. A file that is not displayed by a directory listing. The name of a hidden file usually begins with a period.

high function terminal (HFT). A virtual terminal device that, in addition to displays and keyboards, supports locations, valuations, lighted programmable keys, and sound generators.

higher layer (level). The conceptual level of control or processing logic existing in the hierarchical structure of a station that is above the link layer, and upon which the performance of data link functions are dependent (for example, device control, buffer allocation, station management).

highlight. To emphasize an area on the display screen by any of several methods, such as brightening the area or reversing the color of characters within the area.

high-order. Most significant; leftmost. For example, bit 0 in a register.

history file. (1) A file that exists for each licensed program product supplied by IBM that documents the version, release, and level of the product. Because information is added to this file whenever updates are made to the program product, the history file reflects all changes made to the currently installed version and release of the program product. (2) A file containing a log of system actions and operator responses. (3) A file that displays all versions of a structured file.

H&J. In *C.A.T* terminology, the (usually automatic) process of hyphenation and justification. See also *hyphenation routine*.

hog factor. In system accounting, an analysis of how many times each command was run, how much processor time and memory it used, and how intensive that use was.

hole in a file. See *sparse file*.

home directory. (1) A directory associated with an individual user. (2) The user's current directory on login or after issuing the **cd** command with no argument. (3) A parameter that supplies the full path name of the home directory for the transaction program.

hook ID. A unique number assigned to a specific trace point. All trace entries include the hook ID of the originating trace point in the trace entry header. Pre-defined trace points use assigned hook IDs ranging from 0 to 299. User-defined trace points can choose hook IDs ranging from 300 to 399.

hop count. In the IBM Token-Ring Network, the number of bridges through which a frame passes on the way to its destination.

hop count metric. (1) In a gateway, the indicator that the next string represents the hop count to the destination host or network. (2) The number of host-to-host connections in a route.

host. (1) The primary or controlling computer in the communications network. (2) A computer attached to a network.

hot-key. Colloquially, the ability for the user to switch between two or more active applications by pressing a key sequence.

housekeeping commands. Operations or routines that do not contribute directly to the solution of a problem but contribute directly to the operation of the computer.

Hz. Hertz.

I - K

ICMP. See *Internet Control Message Protocol*.

icon. A picture or graphical representation of an object on a display screen to which a user can point to a with a device such as a mouse in order to select a particular operation or perform a certain action.

ID. (1) Identification. (2) See *identifier*.

identifier (ID). (1) A name you use to reference a data object, containing letters, digits, and/or underscores. A digit, however, cannot appear as the first character in an identifier. (2) In programming languages, a lexical unit that names a language object, such as the names of arrays, records, labels, and procedures. An identifier usually begins with a letter optionally followed by letters, digits, or other characters. (3) A sequence of bits or characters that identifies a program, device, or system to another program, device, or system.

idle list. A list of secondary stations on a network that are polled less often by the primary station due to their inactivity.

idle time. The part of operable time during which a functional unit is not operated.

IEEE. Institute of Electrical and Electronics Engineers.

I-field. Information field.

Ignore. In DOS, the option to disregard the device error and continue processing.

i-list. In a file system, blocks 2 through n compose the i-list, which contain structures (i-nodes) that relate a file to the data blocks or disk. The size of the i-list depends on the size of the mounted file system. See also *i-node* and *superblock*.

image. A faithful likeness of the subject matter of the original.

immediate message. A message that appears on the screen associated with a program. It is usually in response to an action taken by the user. Contrast to *queued message*.

IMS. Information management system.

IMS/VS. Information Management System/Virtual Storage.

incremental backups. The process of copying files that have been opened for reasons other than read-only access since the last backup was created, and that meet the backup frequency criteria. Compare with *full backup*.

index. (1) A table containing the key value and location of each record in an indexed file. (2) A computer storage position or register, whose contents identify a particular element in a set of elements. (3) A list of the contents of a file or a document, together with keys or references for locating the contents. (4) In computer or assembly language programs, an integer used as a relative address. (5) A symbol or a numeral used to identify a particular quantity in an array of similar quantities.

index key. The field within a record that identifies that record in an indexed file.

indexed fields. An area in a structured data file that contains tree data paths.

indexed file. (1) A file in which the key and the position of each record are recorded in a separate portion of the file called an index. (2) In a data base, a file whose access path is built on key values, Each record in the file is identified by a key field.

indicator. (1) An internal switch that communicates a condition between parts of a program or procedure. (2) A device that may be set into a prescribed state, usually according to the result of a previous process or on the occurrence of a specified condition in the equipment. It usually gives an indication of the existence of the prescribed state, and may be used to determine the selection among alternative processes.

information field (I-field) overflow. Occurs when the size of the information field in the receive data exceeds the primary station's buffer capacity, and some of the information field was lost.

informational message. (1) A message providing information to the operator but does not require a response. (2) A message that is not the result of an error condition.

initial program load (IPL). (1) The process of loading the system programs and preparing the system to run jobs. (2) the initialization procedure that causes an operating system to commence operation. (3) The process by which a configuration image is loaded into storage at the beginning or a work day or after a system malfunction.

initialize. (1) In programming languages, to set the starting value of a data object. (2) To set counters, switches, addresses, or contents of storage to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine. (3) To prepare for use, such as initializing a diskette.

initialize system. The preparation of the system for operation. After loading the kernel into memory, the system runs internal checks, initializes all memory and some devices, and analyzes the root file system.

i-node. The internal structure for managing files in the system. I-nodes describe the individual files in the file system; there is one i-node for each file. They contain all of the information pertaining to the node, type, owner, and location of a file. A table of i-nodes is stored near the beginning of a file system.

i-node number. A number specifying a particular i-node file in the file system. Also called i-number.

inodetab. An AIX kernel parameter that establishes a table in memory for storing copies of i-nodes for all active files.

input. (1) Data to be processed. (2) Pertaining to a functional unit or channel involved in an input process or to the data involved in such a process.

input device. Physical devices used to provide data to a computer.

input field. (1) An area in a display file into which you can type or change data. (2) In computer graphics, an unprotected field on a display surface in which data can be entered, modified, or erased.

input file. A file opened in order to allow records to be read.

input list. A list of variables to which values are assigned from input data.

input mode. An open mode in which records can be read from the file.

input stream. The sequence of operation control statements and data given to the system from an input device.

input-output channel controller (IOCC). A hardware component that supervises communication between the input/output bus and the processor.

input-output code number (IOCN). A value supplied by the virtual machine to a VRM component. This number uniquely identifies the code associated with a component and can be considered a module name.

input-output device number (IODN). A value assigned to a device driver by the guest operating system or to the virtual device by the virtual resource manager. This number uniquely identifies the device regardless of whether it is real or virtual.

input-output file. A file opened for input and output use.

input/output (I/O). (1) Pertaining to either input, output, or both between a computer and a device. (2) Pertaining to a device whose parts can perform an input process and an output process at the same time. (3) Pertaining to a functional unit or channel involved in an input process, an output process, or both, concurrently or not, and to the data involved in such a process.

input/output subsystem. That part of the VRM comprised of processes and device managers that provides the mechanisms for data transfer and I/O device management and control.

inquiry. (1) A request for information in storage. (2) A request that puts a display station into inquiry mode. (3) In data communications, a request for information from another system.

inquiry mode. A mode during which the job currently running from a display station is interrupted so that other work can be done.

inquiry program. (1) A program that allows an operator to get information from a disk file. (2) A program that runs while the system is in inquiry mode.

insert mode. (1) A form of keyboard operation that puts new text within existing text. New text is inserted at the cursor position. (2) The source entry utility operation during which source statements are keyed in and added as new records in a source member. (3) In the IBM Token-Ring Network, to make an attaching device an active part of a ring.

integer. A positive or negative whole number or zero.

interactive. Pertains to an activity involving requests and replies, such as between a system user and a program, or between two programs.

interactive processing. A processing method in which each system user action causes response from the program or the system. Contrast with *batch processing*.

interface. (1) A common boundary, but not of internal connections. May be a hardware component to link two devices, or a portion of storage or registers accessed by two or more computer programs. (2) A shared boundary betw.een tow functional units, defined by functional characteristics, common physical interconnection characteristics, signal characteristics, and other characteristics as appropriate. (3) Hardware, software, or both, that links systems, programs, or devices. (4) A synonym for *shell*.

internal clocking. In data communications, the ability of an adapter to provide data clocking.

internal modem. A communications hardware device that fits inside the RT system unit.

international character support. The character sets and date and time string handling provided by the AIX operating system for languages and places other than English and the U.S. It contains conversion subroutines for translation between various character sets and date and time string formats.

Internet Control Message Protocol (ICMP). A protocol used by a gateway to communicate with a source host, for example, to report an error in datagram. It is an integral part of Internet Protocol (IP).

Internet Protocol (IP). The protocol that provides the interface from the higher level host-to-host protocols to the local network protocols. Addressing at this level is usually from host to host.

Internet Router. Enables an IP host to act as a gateway for routing data between separate networks that use either the IBM RT PC Baseband Adapter for use with Ethernet, or the IBM Token-Ring Network RT PC adapter.

inter-process communication (IPC). Ways for programs to communicate data to each other and to synchronize their activities. Semaphores, signals, and internal message queues are common methods of inter-process communication.

interrupt. (1) To temporarily stop a process. (2) In data communications, to take an action at a receiving station that causes the sending station to end a transmission. (3) A signal sent by an I/O device to the processor when an error has occurred or when assistance is needed to complete I/O. An interrupt usually suspends execution of the currently executing program.

interval pacing. One of the two types of pacing supported by ATE. In it, the send routine begins to transfer data as soon as it is started. When it finds a linefeed, it converts it to a carriage return and waits a specified time before it sends the next line. Also, the receive routine looks for data as soon as it is started, and it times out if it does not receive data within 30 seconds.

i-number. A number specifying a particular i-node on a file system. Note: Do not use. See *i-node number.*

inverse. A square array that results from a mathematical operation on a square array such that the two arrays can be multiplied together to obtain a square array with a determinant of one.

invert on zero. A transmission coding method in which the DTE changes the signal to the opposite state to send a binary 0 and leaves it in the same state to send a binary 1.

invoke. To start a command, procedure, or program.

I/O. See *input/output.*

IOCC. See *input-output channel controller.*

IOCN. See *input-output code number.*

IODN. See *input-output device number.*

IP. See *Internet Protocol.*

IP socket. The port concatenated with the Internet Protocol (IP) address.

IPC. See *interprocess communication.*

IPL. See *initial program load.*

ISO. International Organization for Standardization.

Japanese Shift-In start delimiter. In SNA Services, an optional feature supported by X.21 Physical Link Control.

JEP. Job Entry Program.

JES. Job entry subsystem.

job. (1) A unit of work defined by a user to be done by a system. The term job is sometime used to refer to a representation of the job, such as a set of programs, files and control statements to the operating system. (2) One or more related procedures or programs grouped into a procedure, identified by appropriate job control statements.

job queue. A list, on disk, of jobs waiting to be processed by the system.

justify. To print a document with even right and left margins.

kbuffers. An AIX kernel parameter establishing the number of buffers that can be used by the kernel.

K-byte. See *kilobyte.*

kernel. (1) The part of an operating system that contains programs that control how the computer does its work, such as input/output, management and control of hardware, and the scheduling of user tasks.
(2) The memory-resident part of the AIX Operating System containing functions needed immediately and frequently. The kernel supervises the input and output, manages and controls the hardware, and schedules the user processes for execution. (3) A

part of a program that must be in main storage in order to load other parts of the program.

kernel mode. Contrast with *user mode*

kernel parameters. Variables that specify how the kernel allocates certain system resources.

key. (1) One or more characters used to identify a record and establish the record's order within an indexed file. (2) A unique identifier (of type **key_t**) that names the particular interprocess communications member. (3) Identifies the name of the shared library text image. (4) An identifier within a set of data elements. (5) A character string that matches a definition in a key table.

key pad. A physical grouping of keys on a keyboard (for example, numeric key pad, and cursor key pad).

keyboard. An input device consisting of various keys allowing the user to input data, control cursor and pointer locations, and to control the user/work station dialogue.

keyboard click (key click). Transient pulses or surges on a transmission line set up by the opening or closing of keying circuit contacts.

keyboard send-receive. See *KSR mode.*

keylock feature. A security feature in which a lock and key can be used to restrict the use of the display station.

keyword. (1) One of the predefined words of a programming language; a reserved word. (2) In programming languages, a lexical unit that characterizes some language construct. A keyword normally has the form of an identifier. (3) A name or symbol that identifies a parameter. (4) Part of a command operand that consists of a specific character string.

keyword argument. One type of variable assignment that can be made on the command line.

keyword parameter. (1) A parameter that consists of a keyword, followed by one or more values. See also *positional parameter* (2) Setting a shell parameter for the duraction of a command.

kg. Kilogram.

kill. An AIX command that stops a process.

kill character. The character that is used to delete a line of characters entered after the user's prompt.

kilobyte. 1024 bytes.

kprocs. An AIX kernel parameter establishing the maximum number of processes that the kernel can run simultaneously.

KSR Mode. KSR (keyboard send-receive) Mode causes a virtual terminal to act like a standard ASCII terminal, with some extensions, for both input and output.

kVA. Kilovolt-ampere.

L - M

label. (1) The name in the disk or diskette volume table of contents that identifies a file. See also *file name*. (2) The field of an instruction that assigns a symbolic name to the location at which the instruction begins, or such a symbolic name. (3) In programming languages, a language construction naming a statement and including an identifier.

LAN. See *local area network.*

layer. Levels in the network design. The OSI model has seven layers: *physical*, *data link*, *network*, *transport*, *session*, *presentation*, and *application*.

LC. See *link control.*

LCC. See *logical link control.*

leased facility. See *nonswitched line.*

LED. See *light-emitting diodes.*

library. (1) A collection of functions, calls, subroutines, or other data. (2) A data file that contains copies of a number of individual files and control information that allows them to be accessed individually. (3) Collections of commonly used functions and declarations. (4) The set of publications for a product. (5) A named area on disk that can contain programs and other related information (not files). A library consists of different sections called members.

licensed programs (LP). (1) Software programs that remain the property of the manufacturer, for which customers pay a license fee. (2) A separately priced program and its associated materials that bear an IBM copyright and and are offered to customers under the terms and conditions of a licensing agreement.

light-emitting diodes (LED). A semiconductor chip that gives off visible or infrared light when activated.

limited interface. A set of system calls that provides a limited function interface to communicate with SNA. See *interface* and *extended interface*.

line. (1) A horizontal display on a screen. See *communications line*. Contrast with *column*. (2) the portion of a data circuit external to data circuit-terminating equipment (DCE) that connects the DCE to a data switching exchange (DSE), connects a DEC to one or more other DCEs, or a DSE to another DSE. (3) A string of characters accepted by a system as a single block of input from a terminal, such as all characters entered before a carriage return.

line adapter. (1) A modem that is a feature of a particular device. (2) A functional unit that converts the serial-by-bit output of a station to a parallel bit form and from a parallel bit form to serial-by-bit input to a station.

line editor. An editor that modifies the contents of a file one line at a time.

line pacing. The sending of a line followed by a waiting interval before continuing transmission. Contrast with *character pacing*.

line printer. A printer that prints output one line of characters at a time, as a unit. Output of line printers is in constant-width characters.

linefeed. An ASCII character that causes an output device to move forward one line.

link. (1) A connection between an i-node and one or more file names associated with it. (2) A transmission medium and data link control component that together transmit data between adjacent nodes. (3) A connector between a file name and the file itself. (4) A programming, the part of a program that passes control and parameters between separate portions of the computer program. (5) To interconnect items of data or portions of one or more computer programs, such as linking object programs by a linkage editor or linking data items by pointers.

link control (LC). See *logical link control* and *physical link control*.

Link Trace. A link trace is a sequential log of events that occur on the link that may be helpful in finding the source of a recurring error.

linkage editor. A program that resolves cross-references between separately assembled object modules, then assigns final addresses to create a single relocatable load module. If a single object module is linked, the linkage editor simply makes it relocatable. Synonym for overlay linkage editor.

linker. See *linkage editor*.

linking. (1) Sharing a disk owned by another user, on either a temporary or permanent basis. the sharing is normally read-only and may require a password to access the data. (2) Terminal-to-terminal communication.

list. (1) A data object consisting of a collection of related records. (2) A sequence of one or more pipelines separated by separators and terminators. (3) An ordered set of data. (4) To print or otherwise display items of data that meet specified criteria.

listen attachment. In SNA, an attachment that accepts incoming calls from a remote station.

literal. (1) A symbol or a quantity in a source program that is itself data, rather than a reference to data. (2) In programming languages, a lexical unit that directly represents a value; for example 14 represents the integer 14.

literal string. A string that does not contain wildcard characters and can therefore be interpreted just as it is. Contrast to *regular expression*.

LLC. See *logical link control (LLC)*.

LNS. See *LU Network Services Component*.

load. (1) To transfer data or programs from an internal storage to a register,. or from a register to another register. (2) To place a diskette into a diskette drive, or a magazine into a diskette magazine drive. (3) To insert paper into a printer.

load module. See *run file*.

loader. A program that reads run files into main storage, thus preparing them for execution.

local. (1) Pertaining to a device, file, or system that is accessed directly from your system, without the use of a communications line. Contrast with *remote*. (2) In programming languages, pertaining to the relationship between a language object and block such that the language object has a scope contained in that block. (3) Pertaining to that which is defined and used only in one subdivision of a computer program.

local area network (LAN). (1) A network in which communications are limited to a moderate-sized geographic area (1 to 10 km) such as a single office building, warehouse, or campus. A local network depends upon a communications medium capable of moderate to high data rate (1 to 20 M bytes per second), and normally operates with a consistently low error rate. (2) A data network located on the user's premises in which serial transmission is used for direct data communication among data stations.

local host. The host on the network at which a particular operator is working; sometimes called *current host*.

locator. In computer graphics, an input device that provides coordinate data; for example, a mouse, tablet, or thumbwheel.

lock file. In a shared DASD environment under VSE, a system file on disk used by the sharing systems to control their access to shared data.

lock. A serialized mechanism by means of which a resource is restricted for use by the holder of the lock. See also *physical lock, file lock,* and *record lock*.

log. (1) To record; for example, to log all messages on the system printer. A list of this type is called a log, such as an error log. (2) A collection of messages or message segments placed in an auxiliary storage device for accounting or data collections purposes.

log in. To begin a session at a display station.

log in shell. The program, or command interpreter, started for a user at log in.

log off. To end a session at a display station.

log out. To end a session at a display station.

logger. (1) A functional unit that records events and physical conditions, usually with respect to time. (2) A device that enables a user entity to log in, for example, identify itself, its purpose, and time of entry, and log out with the corresponding data. This enables the appropriate accounting procedures to be carried out in accordance with the operating system.

logical device. (1) A file for conducting input or output with a physical device. (2) A file for mapping user I/O between virtual and real devices.

logical expression. An expression consisting of logical operators and/or relational operators that can be evaluated to a value of either true or false.

logical link control protocol (LCC). (1) In a local area network, the protocol that governs the assembling of transmission frames and their exchange between data stations, independently of the medium access control protocol. (2) A protocol process that allows data transfer on a given physical link. A logical link control (LLC) may reside in the operating system or the VRM and is controlled by the block I/O device manager.

logical unit (LU). (1) A type of network addressable unit that enables end users to communicate with each other and gain access to network resources. (2) In SNA, a port through which an end user accesses the SNA network in order to comunicate with another end user, and through which the end user accesses the functions provided by system services control points (SSCPs). An LU can support at least two sessions, one with an SSCP and one with a another LU, and may be capable of supporting many sessions with other LUs.

Logical Unit (LU) type 1. Describes an SNA session that supports communication between an application and multiple input/output devices. This communication could occur in an interactive or batch environment.

Logical Unit (LU) type 2. Describes an SNA session that supports communication between an application and a display using a 3270 device data stream.

Logical Unit (LU) type 3. Describes an SNA session that supports communication between an application and a printer using a 3270 device data stream.

Logical Unit (LU) type 6.2. (1) Describes an SNA session between two applications in a distributed data processing environment. LU 6.2 sessions provide communication between two type 5 nodes, a type 5 and a type 2.1 node, and two type 2.1 nodes. (2) Most recent LU type used for SNA program-to-program communications.

login directory. The directory you access when you log into the system.

loop. (1) A sequence of instructions performed repeatedly until an ending condition is reached. (2) A closed unidirectional signal path connecting input/output devices to a system.

lowercase. The uncapitalized letters of a font.

low-order. Least significant; rightmost. For example, in a 32 bit register (0-31), bit 31 is the low-order bit.

LP. See *licensed program.*

LPFK. See *lighted programmable function key.*

LU. See *logical unit (LU).*

LU 1. See *logical unit (LU) type 1.*

LU 2. See *logical unit (LU) type 2.*

LU 3. See *logical unit (LU) type 3.*

LU 6.2. See *logical unit (LU) type 6.2.*

LU, dependent. A logical unit that cannot start a conversation but must wait for the host system to start the conversation.

LU, independent. A logical unit that can start a conversation with another logical unit.

LU network services component (LNS). Initiates and terminates LU-LU sessions in response to requests from the resource manager and from the remote LU. It also activates and deactivates CP-LU sessions.

LU-LU session. In SNA, a session between two logical units (LUs) of the same type that supports communication between two end users, or between an end user and an LU services component.

m. Meter.

M, M byte, Mb. Megabyte. When referring to storage capacity, two to the twentieth power; 1,048,576 in decimal notation.

macro. (1) A name or label used in place of a number of other names. (2) Sequence of instructions or statements a macrogenerator executes when replacing a macroinstruction. (3) Set of statements defining name of, format of, and conditions for generating a sequence of assembler statements from a single source statement. (4) A label that is declared at the start of a program or file. The label can then be used to represent the values assigned to the label in the declaration. (5) See also *macro instruction*.

macro call. An identifier followed by a parenthesized list of arguments that the preprocessor replaces with the replacement code located in a preprocessor define statement. Synonynm for macro instruction.

magic number. A numeric or string constant in a file that indicates the file type.

mail. Correspondence in the form of messages transmitted between work stations over a network. Synonym for electronic mail.

mail box. A storage location in a network that is associated with a user name and to which messages are sent to that user.

main program. (1) The highest-level program involved in a run unit. (2) The first program unit to receive control when a program is run. See also *subprogram*. (3) The part of the processing unit where programs are run. (4) A program that performs primary functions, passing control to routines and subroutines for the performance of more specific functions.

main storage. (1) Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing. (2) The part of internal storage into which instructions and other data must be loaded for execution or processing. (3) The part of the processing unit where programs are run.

mainframe. A large computer, particularly one to which other computers can be connected so that they can share facilities the mainframe provides. The term usually refers to hardware only.

maintenance analysis procedure (MAP). Documentation used by IBM customer engineers and by service representatives to repair IBM equipment. A MAP contains yes/no questions and procedures that direct the user to the failing part of the equipment.

maintenance mode. State where a product or system can be serviced by a service representative. Synonym for service mode. Contrast to *maintenance mode*

maintenance system. A special version of the AIX Operating System which is loaded from diskette and used to perform system management tasks.

major device number. A system identification number for each device or type of device.

manual answer. (1) An answer in which a call is established only if the called user signals a readiness to receive the call by means of a manual operation. (2) A line type requiring operator actions to receive a call over a switched line. Contrast with *autoanswer*.

manual call. (1) A call that permits the entry of selection signals from a calling data station at an undefined character rate. (2) In data communications, the operator actions required to place a call over a switched line. Contrast with *auto-call*.

manual dialing. In making an ATE connection, dialing the number manually over a telephone line. Contrast to *automatic dialing*

MAP. See *maintenance analysis procedure*.

mapped conversation. A temporary connection between an application program and an advanced program-to-program communication (APPC) session in which the system provides all the protocol information. It allows the two programs to exchange data records of any length and in any format specified by the transmission programs. Only LU 6.2 sessions allow mapped conversation; it is used primarily for application transaction programs. Contrast to *basic conversation*

mapped file. (1) A file that can be accessed using direct memory operations, rather than reading it from disk each time it is accessed. (2) Files on the fixed-disk that are accessed as if they are in memory.

marker. (1) A visual symbol within a non-interactive pane indicating the location of the cursor when the pane was last interactive. (2) In computer graphics, a glyph with a specified appearance that is used to identify a particular location.

mask. A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

matrix. (1) A rectangular array of elements arranged in rows and columns, that can be manipulated based on matrix algebra rules. (2) In computers, a logic network in the form of an array of input and output leads with logic elements joined at some of their intersections. (3) By extension, an array of any number of dimensions.

maxprocs. An AIX kernel parameter establishing the maximum number of processes that can be run simultaneously by a user.

m-byte. See *megabyte*.

measurement. See *scale*.

medium (media). The material in or on which data may be represented (for example, twisted pairs, coaxial cables, and optical fibers).

megabyte (M byte). A unit of measure for storage capacity. One megabyte equals 1,048,576 bytes. Loosely, one million bytes.

megahertz. A unit of measure of frequency. The measurement of 1 megahertz equals 1,000,000 hertz.

member. (1) A data object in a structure or a union, or in a library. (2) A partition of a partitioned data set.

memory. (1) Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing. (2) Storage on electronic chips. Examples of memory are random access memory, read only memory, or registers. See *storage*.

memory allocation. See allocation.

memory dump. The means by which the RT OC records its state at the time of a failure. There are three types of memory dumps: Virtual Resource Manager Dump, kernel dump, and full image dump.

memory image. The logical layout of a process's parts in memory.

memory-mapped file. See *mapped files*.

menu. A displayed list of items from which an operator can make a selection.

message. (1) A response from the system to inform the operator of a condition which may affect further processing of a current program. (2) An error indication, or any brief information that a program writes to standard error or a queue. (3) Information sent from one user in a multi-user operating system to another. (4) A general method of communication between two processes. (5) A group of characters and control bit sequences transferred as an entity.

message pop-up. A pop-up area on the screen caused by an activity associated with another pane.

message queue ID. An identifier assigned to a message queue for use within a particular process. It is similar in use to a file descriptor of a file.

message services. A set of routines to help create, update and display messages from a program.

metacharacter. A character used to specify another character.

metafile. A data file representation of a graphics picture which can be transmitted and re-edited.

MHz. See *megahertz*.

mil. A measurement of thickness: 1/1000 inch.

millisecond. One one-thousandth of a second.

minidisk. (1) A logical division of a fixed disk that may be further subdivided into one or more partitions. (2) A disk or diskette having a diameter smaller than the conventional diameter.

minor device number. A number used to specify various types of information about a particular device, for example, to distinguish among several printers of the same type.

mm. Millimeter.

mnemonic. (1) A symbol chosen to assist the human memory. (2) The field of an assembler instruction that contains the acronym or abbreviation for a machine instruction. Using mnemonics frees the programmer from having to remember the machine's numeric operator codes.

mode. (1) A method of operation. (2) The set of rules and protocols to be used for a session.

mode name. (1) The name of an entry in the in the logon mode table. (2) In SNA, identifies the set of rules and protocols to be used for the session when the local LU is connected as a dependent LU to a host statement.

mode word. An i-node field that describes the type and state of the i-node.

modem. See *modulator-demodulator*.

modem eliminator. A device that connects a terminal directly to a computer port through a wired connector with a specific pin arrangement. When two devices both function as DTEs (data terminal equipment), the cable that connects them must transmit send/receive signals using a modem eliminator.

modulation. Changing the frequency or size of one signal by using the frequency or size of another signal.

modulator-demodulator (modem). A device that converts data from the computer to an analog signal that can be transmitted on a telecommunications line, and converts the analog signal received to data for the computer.

module. (1) A discrete programming unit that usually performs a specific task or set of tasks. Modules are subroutines and calling programs that are assembled separately, then linked to make a complete program. (2) In programming languages, a language construct that consists of procedures or data declarations and that interact with other such constructs. (3) A packaged functional hardware unit designed for use with other components. (4) See *load module*. and *run file*.

monitor. (1) A device that observes and verifies operations of a data processing system. (2) A functional unit that observes and records selected activities for analysis within a data processing system. Possible uses are to show significant departures from the norm, or to determine levels of utilization or particular functional units. (3) Synonym for visual display unit.

monitor mode. (1) A mode of the network control program in which the host is immediately notified when an attention, disconnect, or other unusual status occurs on a designated line. (2) A secondary state operating mode in which the station is receiving and analyzing loop traffic, but is not inserted into the loop. (3) A console display mode in which an application program can directly access the display adapter without conflict with the standard virtual terminal output mechanism.

monochrome display. A display device that has only one color.

mount. To make a file system accessible.

mountab. An AIX kernel parameter establishing the maximum number of file systems that can be mounted simultaneously.

mouse. A hand-held locator operated by moving it on a flat surface. It generally contains a control ball or a pair of wheels, and it allows you to select objects and scroll the display screen by means of buttons.

mouse threshold. Determines the granularity of input events reported, or the amount of horizontal or vertical mouse movements required before an event occurs.

msqid. See *message queue ID*.

multi-byte control. One of the two types of controls valid in a character stream data. Also called escape sequences and control sequences.

multidrop. (1) Stations connected to a multipoint channel at one location. (2) A network configuration in which there are one or more intermediate nodes on the path between an central node and an endpoint node.

multiline. More than one communications line.

multiplexed device. (1) A device that takes several input signals and combines them into a single output signal so that each of the input signals can be recovered. (2) A device capable of interleaving events of two or more activities or capable of distributing events of an interleaved sequence to the respective activities.

multipoint. In data communications, pertains to a network that allows two or more stations to communicate with a single system on one line.

multipoint link. A circuit that interconnects several stations. Synonymous with multipoint line.

multiprogramming. (1) Pertaining to the processing of two or more programs at the same time by a single processor. (2) A mode of operation that provides for interleaved execution of two or more computer programs by a single processor.

multiuser mode. A mode of operation that enables two or more users to use the services of a processor within a given period of time; usage is usually serial unless otherwise specified. Contrast to *maintenance mode*

multivolume file. (1) A diskette file occupying more than one diskette. (2) A file contained on more than one storage medium.

MVS. Multiple Virtual Storage.

N - P

Name Server Protocol. A protocol used by TCP/IP to resolve host names into internet addresses or vice versa.

NAU. See *network addressable unit*.

NCCF. Network Communications Control Facility.

NCP. Network control program.

negative response. In data communications, a reply indicating that data was not received correctly or that a command was incorrect or unacceptable.

nest. (1) To incorporate a structure or structures into a structure of the same kind. For example, to nest one loop (the nested loop) within another loop (the nesting loop); to nest one block (the nested subroutine) within another block (the nesting block). (2) To place subroutines or data in other subroutines or data at a different heirarchical level so that the subroutines can be executed as recursive subroutines or so that the data can be accessed recursively.

network. (1) A collection of data processing products connected by communication lines for information exchange between stations. (2) An arragement of nodes and connecting branches. Connections are made between data stations.

network adapter. Circuitry that allows devices using a directly attached network to communicate with the system.

network addressable unit (NAU). In SNA, a logical unit, a physical unit, or a system services control point. The NAU us the origin or the destionation of information transmitted by the path control network.

Network Information Center (NIC). The publication distribution center for DARPA TCP/IP information.

network management. The conceptual control element of a data station that interfaces with all of the layers of that data station and is responsible for the resetting and setting of control parameters, obtaining reports of error conditions, and determining if the station should be connected to or disconnected from the medium.

new-line character (NL). A control character that causes the print or display position to move to the first position on the next line.

new-process image. A new program laid over the current program by the exec system call.

NIC. See *Network Information Center*.

nice value. A positive number that determines a process's CPU priority. A higher number results in a lower priority.

nickname. See *alias*.

node. (1) An individual element of a full pathname. Nodes are separated by slashes (/). (2) An end point of a link, or a junction common to two or more links in a network. Nodes can be processors, controllers, or work stations. Nodes can vary in routing and other functional capabilities. (3) The portion of a hardware component, along with its associated software components, that implement the functions of the seven architectural layers (SNA). (4) The representation of a state or event by means of a point on a diagram. (5) In a tree structure. a point at which subordinate items of data originate.

node ID. A unique string of characters that identifies the node on a network.

node verification. Provides a level of security beyond that provided by the network addressing scheme to help ensure that a connection reaches the correct remote station. It is available on LU 6.2 connections only. See also *BIND password*.

noise. (1) Undesirable electrical signals on the communications channel that can interfere or distort data signals. (2) Random variations of one or more characteristics of any entity such as voltage, current, or data. (3) A random signal of known statistical properties of amplitude, distribution, and spectral density.

nondisplay. A field attribute that prevents the displaying of data.

non-return-to-zero inverted (NRZI). Similar to non-return-to-zero (NRZ); however, instead of changing signal whenever the binary designation changes from 1 to 0 or from 0 to 1 (as in non-return-to-zero), the signal changes only whenever a 1 is transmitted.

non-return-to-zero (NRZ). A binary code system in which a signal condition must be sustained for the full time interval and does not revert to a standby or quiescent state between signal elements. The use of NRZ permits the maximum data signalling rate on the channel which, theoretically, is twice the bandwidth (Nyquist's theorem). See also *non-return-to-zero inverted (NRZI)*.

nonspacing character sequence. For accented characters, a two-part sequence consisting of one of 13 valid diacritics, followed by an alphabetic character or a space, converted by the virtual terminal subsystem into a single code point resulting in the alphabetic character with the specified diacritic mark.

nonswitched line. A connection between computers or devices that does not have to be established by dialing. Contrast with *switched line*.

nonswitched network. A connection between computers or devices on a network that does not have to be established by dialing.

nonvolatile random access memory. A portion of random access storage that retains its contents after electrical power to the machine is shut off.

normal attachment stop. See *normal stop*.

normal mode. See *multi-user mode*.

normal stop. One of two ways to stop an attachment, the other way being forced stop. If the attachments or any connections of the attachments are in a pending state, SNA Services rejects the normal storp attachment action. See also *forced attachment stop*.

NRZ. See *non-return-to-zero (NRZ)*.

NRZI. See *non-return-to-zero inverted (NRZI)*.

NUL. See *null character*.

null. Empty, having no value, containing nothing.

null character (NUL). (1) The character hex 00, used to represent the absence of a printed or displayed character. (2) A control character used to accomplish media-fill or time-fill that can be inserted into or removed from a sequence of characters without affecting the meaning of the sequence; however, the control of equipment or the format may be affected by the character.

null character string. Two consecutive single quotation marks that specify a character constant of no characters.

null modem. See *modem eliminator*.

null signal. A signal parameter of 0 (zero).

numeric. Pertaining to any of the digits 0 through 9.

NVRAM. See *nonvolatile random access memory*.

object code. (1) Machine-executable instructions, usually generated by a compiler from source code written in a higher level language (such as C language). For programs that must be linked, object code consists of relocatable machine code.
(2) Output from a compiler or assembler which is itself executable machine code or is suitable for processing to produce executable machine code.

object file. A member file in an object library.

object library. An area on a direct access storage device used to store object programs and routines.

object module. A set of instructions in machine language. The object module is produced by a compiler or assembler from a subroutine or source module and can be input to the linker. The object module consists of object code. See *module*.

octal. (1) A base eight numbering system. (2) Pertaining to a fixed radix numeration having a radix of eight.

off-hook. Activated, with regard to a telephone set. By extension, a data set automatically answering on a public switched system is said to go off-hook. Contrast with *on-hook*.

offline. (1) Pertaining to the operation of a functional unit when not under the direct control of a computer. (2) Neither controlled directly by, nor communicating with, the computer, or both. Contrast with *online*.

on-hook. Deactivated, in regard to a telephone set. A telephone not in use is on-hook. Contrast with *off-hook*.

online. (1) Being controlled directly by, or directly communicating with, the computer, or both. Contrast with *offline*. (2) Pertaining to the operation of a functional unit when under the direct control of a computer.

open. (1) To prepare and make a file available to a program for processing.

open architecture. A computer architecture that is documented so that programmers can write code to run on that computer.

operand. (1) An instruction field that represents data (or the location of data) to be manipulated or operated upon. Not all instructions require an operand field. (2) An identifier, a constant, or an expression that is grouped with an operator. (3) An entity on which an operation is performed. (4) Information entered with a command name to define the data on which a command processor opeates and to control the execution of the command processor.

operating system (OS). (1) Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, input/output control, and data management. (2) A set of programs that controls how the system works.

operating system state. One of two virtual machine protection states that run in the processor's unprivileged state. The kernel is the only entity that runs in the operating system state. See *problem state*.

operational state. One of the stages of the operation of runacct, the main daily accounting shell procedure.

operator. (1) A symbol (such as +, -, *) that represents an operation (in this case, addition, subtraction, multiplication). (2) A person who operates a device.

option. (1) An item of either hardware or software that may be purchased in addition to the basic system. An option may include cables, an adapter, a warranty, and other items. (2) A specification in a statement that may b used to influence the exection of the statement.

orphaned files. Files that cannot be reached by the fsck command.

OS. See *operating system.*

OSI. Open Systems Interconnect. Communications architecture defined by the International Standards Organization (OSI).

output. (1) The result of processing data. (2) Pertaining to a functional unit or channel involved in an output process to to the data or involved in such a process. (3) Data transferred from storage to an output device.

output devices. Physical devices used by a computer to present data to a user. Synonym for output unit.

output file. (1) A file that is opened by a program so that the program can write to that file. (2) A file that contains the results of processing.

output list. A list of variables from which values are written to a file or device.

output mode. An open mode in which records can be written to a file.

output redirection. The specification of an output destination other than the standard one.

output stream. Messages and other output data, displayed on output devices by an operating system or a processing program.

overlay. (1) To write over (and therefore destroy) an existing file. (2) A program segment that is loaded into main storage, replacing all or part of a previously loaded program segment. (3) The technique of repeatedly using the sameareas of internal storage during different states of a program. (4) A collection of predefined data such as lines, shading, text, boxes, or logos, that can be merged with variable data on a page while printing. Synonym for medium overlay.

overlay linkage editor. See *linkage editor*.

override. (1) A parameter or value that replaces a previous parameter or value. (2) To replace a parameter or value.

overwrite. To record into an area of storage so as to destroy the data that was previously stored there.

overwrite mode. A form of system operation that puts characters typed from the keyboard in place of existing characters.

owner. The user who has the highest level of access authority to a data object or action, as defined by the object or action.

pacing. (1) A technique by which a receiving component controls the rate of transmission by sending a component to prevent overrun or congestion. (2) A file transfer protocol required by some systems. It controls data transmission by waiting for a specified character, or waiting a specified number of seconds between lines. This protocol prevents the loss of data when the block size is too large or data is sent too quickly for the system to process. See *character pacing* and *interval pacing*.

pacing response. In SNA, an indicator that signifies the readiness of a receiving component to accept another pacing group. The indicator is carried in a response header (RH) for session-level pacing, and in a transmission header (TH) for virtual-route pacing.

packet. (1) The data of one transaction between a host and its network. A packet usually contains a network header, followed by one or more headers used by high-level protocols, followed by data blocks. (2) In data communications, a sequence of binary digits, including data and control signals, that is transmitted and switched as a composite whole.

pad. (1) To fill unused positions in a field with dummy data, usually zeros or blanks. (2) A device used to introduce transmission loss into a circuit. It may be inserted to introduce loss or match impedances.

page. (1) A block of instructions, data, or both. (2) The number of lines that can fit into a window. (3) In a virtual storage system, a fixed-length block that has a virtual address and is trnsfixed as a unit between real storage and auxiliary storage.

page fault. A program interruption that occurs when a page that is not in memory is referred to by an active page.

page frame. (1) In real storage, a storage location having the size of a page. (2) An area of main storage used to hold a page.

page space minidisk. The area on a fixed disk that temporarily stores instructions or data currently being run. See also *minidisk*.

paging. (1) The action of transferring instructions, data, or both between real storage and external page storage. (2) Moving data between memory and disk as the data is needed.

paging space. An area on disk that the system uses to store information that is resident in virtual memory, but is not currently being accessed.

panel. (1) A set of logically related information displayed on the screen for the purpose of communicating information to or from a computer user. (2) A group of one or more panes that are treated as a unit. The panes of a panel are displayed together, erased together, and usually represent a unit of information to a person using the application. A panel is represented on the display as a rectangular area that is tiled (completely filled) with panes.

panning. (1) In computer graphics, viewing an image that is too large to fit on a single screen by moving from one part of the image to another. (2) Progressively translating an entire display image to give the visual impression of lateral movement of the image.

parallel processing. The condition in which multiple tasks are being performed simultaneously within the same activity. Contrast with serial processing.

parallel transmission. (1) Transmitting all bits of a character simultaneously. (2) In data communication, the simultaneous transmissions of a certain number of signal elements constituting the same telegraph or data signal.

parameter. (1) Information that the user supplies to a panel, command, or function. (2) A variable that is given a constant value for a specified application and that may denote the application. (3) Data passed between programs or procedures.

parent directory. The directory one level above the current directory.

parity. The state of being either even-numbered or odd-numbered.

parser. A program that interprets user input and determines what to do with the input.

partition. (1) A logical division of storage on a fixed disk. (2) A fixed-size division of storage.

partner. In data communications, the remote application program or the remote computer.

passive gateway. A gateway that does not exchange routing information. Its routing information is contained indefinitely in the routing tables and is included in any routing information that is transmitted.

password. In computer security, a string of characters known to the computer system and a user, that when entered along with identification, allows the user access to the system.

password security. A program product option that helps prevent the unauthorized use of a display station, by checking the password entered by each operator at sign-on.

path. (1) In SNA the series of path control network components traversed by the information exchanged between two network addressable units (NAUs). A path consists of a series of path control elements, data link control elements, and links. See also *explicit route*. (2) In a network, any route between any two nodes. (3) In a data base, a sequence of segment occurences from the root segment to an individual segment.

path control (PC) layer. In SNA, the layer that manages the sharing of link resources of the SNA network and routes basic information units (BIUs) through it. Path control routes message units between network addressable units (NAUs) in the network and provides the paths between them. It converts the BIUs from transmission control (possibly segmenting them) into path information units (PIUs) and exchanges basic transmission units (BTUs) with data link control.

path name. A file name specifying all directories leading to that file. See *full path name* and *relative path name*.

pattern matching. The process of specifying a pattern of characters that the shell compares against the file names in a directory.

pattern-matching character. Special characters such as * or ? that can be used in a file specification to match one or more characters. For example, placing a ? in a file specification means that any character can be in that position.

pattern string. Strings of regular expressions (REs) composed of special pattern matching characters used in addresses to specify lines, and in some subcommands, portions of a line.

PCS. See *programmable character set.*

PDN. See *public data network (PDN).*

peer-to-peer communications. Pertaining to data communications between two nodes that have equal status in the interchange. Either node can begin the conversation. See also *Logical Unit (LU) type 6.2.*

pel. See *picture element* and *pixel.*

pending. Waiting, as in an operation that is pending.

pending state. A condition of a server program in which it has received a request for an action (start, stop, or suspend) but has not as yet performed that action.

per-process data. In kernel mode, a portion of the user process stack segment. This area is paged with the process and it contains process information such as the current directory of files opened by the process or input in I/O mode. This information occupies the top of the stack segment.

peripheral device. With respet to a particular processing unit, any equipment that can communicate directly with that unit. Synonym for peripheral unit.

permanent error. An error thatn cannot be eliminated by retrying a read/write operation.

permanent storage. A storage device withose contents cannot be modified.

permission code. A three-digit octal code, or a nine-letter alphabetic code, indicating the access permissions. The access permissions are read, write, and execute.

permission field. (1) One of the three-character fields within the permissions column of a directory listing indicating the read, write, and run permissions for the file or directory owner, group, and all others. (2) Codes that determine how the file can be used by any users who work on the system.

phase. (1) One of several stages file system checking and repair performed by the **fsck** command. (2) A distinct part of a process in which related operations are performed. (3) A part of a sort/merge program, for example, sort phase, merge phase. (4) A part of a data call.

phase modulation. (1) Altering the phase of a carrier signal to convey data signals. (2) Modulation that varies the phase angle of a sinusoidal carrier from a reference carrier phase angle by an amount proportional to the instantaneous amplitude of the modulating signal.

phrase password. A type of BIND password that consists of a string of ASCII characters between 30 and 80 characters long, including internal spaces. You use it only when communicating with other RT PCs with SNA Services installed.

physical block. See *block*.

physical device. See *device*.

physical file. (1) An indexed file containing data for which one or more alternative indexes have been created. (2) A data base file that describes how data are to be presented or received from a program and how data are actually stored in the data base. A physical file contains one record format and one or more members.

physical layer (or level). The lowest layer of network design as specified by the ISO Open System Interconnection (OSI) reference model. This layer is responsible for interfacing with the medium, detecting and generating signals on the medium, and converting and processing signals received from the medium and from the data link layer.

physical record. (1) A group of records recorded or processed as a unit. Same as *block*. (2) A unit of data moved into or out of the computer. (3) A record whose characteristics depend on the manner or form in which it is stored, retrieved, or moved. A physical record may contain all or part of one or more logical records.

Physical Unit (PU). In SNA, a set of programs that controls the actual physical hardware associated with a node.

picture element (pel). (1) In computer graphics, the smallest element of a display space that can be independently assigned color and intensity. (2) The area of the finest detail that can be reproduced effectively on the recording medium. (3) An element of a raster pattern about which a toned area on a photoconductor can appear. (4) A point in the frame buffer or on the display.

PID. Process ID of a program. See also *process ID*.

pin. A area of memory reserved for certain functions.

PIP. See *Program Initialization Parameters (PIP)*.

pipe. To direct the data so that the output from one process becomes the input to another process.

pipeline. (1) A direct, one-way connection between two or more processes. (2) A serial arrangement of processors or a serial arrangement of registers within a processor. Each processor or register performs part of a task and passes results to the next processor; several parts of different tasks can be performed at the same time. (3) To perform processes in a series. (4) To start execution of an instruction sequence before the previous instruction sequence is completed to increase processing speed.

pix map. See *pixel map*.

pixel. See *picture element*.

pixel format. A definition for a pixel map that indicates that data stored in this format has all bits for a pixel stored together. The data starts with the origin and increases first in the x direction and then in the y direction.

pixel map. A 32-bit array of integers that defines the characters of a rectangle. Also called a pixmap.

plane format. A definition of a pixel map that indicates that each of the bits that compose a pixel is store in a separate consecutive plane in memory. The most significant bit is first followed by the next most significant, and so forth. The bits within a plane are packedd together 8 bits per byte.

plane mask. Determines shich of the display adapter storage places are modified by the output functions.

plotter. A printing device externally attached with cables to the system unit, used to print two-dimensional graphs and charts.

plug. A device that connects the wires of an electrical circuit to an electrical source. The plug is designed to be inserted into a jack.

pointing. The action of positioning the pointing cursor on a displayed object.

pointing cursor. A cursor used to point to and select buttons, to activate panes, and to scroll the contents of the active pane. The pointing cursor is controlled by a mouse.

point-to-point line. A switched or nonswitched communications line that connects a single remote station to a computer.

point-to-point link. A switched or nonswitched link that connects a single remote link station to a node or to another station.

poll. (1) An interrogation to determine if a station is ready to transmit information. (2) To execute a polling sequence.

poll list. A list of terminal addresses in computer storage that specifies the; sequences in which stations are to be polled.

polling. (1) A method for determining whether each of the stations sharing a communications line has data to send. (2) On a multipoint connection or a point-to-point connection, the process whereby data stations are invited one at a time to transmit. (3) Interrogation of devices so as to avoid contention, to determine operational status, or to determine readiness to send or receive data.

pop-up. A box on the display screen that displays information or asks you to make choices.

pop-up icon. A button that indicates to you that additional values are defined for another button.

pop-up panel. A screen which may be any size smaller than the regular screen that is superimposed on another screen to present some choices to be made or information needed at that point in the procedure.

POR. See *power-on reset*.

port. (1) A part of the system unit or remote controller to which cables for external devices (display stations, terminals, printers) are attached. The port is an access point for data entry or exit. (2) An entrance to or exit from a network. (3) To make the programming changes necessary to allow a program that runs on one type of computer to run on another type of computer. (4) An access point for data input to or data output from a computer system. (5) The logical connection between the RT PC and another unit.

positional parameter. (1) A shell facility for assigning values from the command line to variables in a program. (2) A paramenter that must appear in a specified location relative to other positional parameters.

POST. See *power-on self test*.

power-on light. The light on the operator panel that indicates that the DC power in the system unit is functioning.

power-on reset. A way to make a computer execute its power-on self test without actually turning the computer off and then on.

power-on self test (POST). A series of internal diagnostic tests activated each time the system power is turned on.

primary station. (1) On a point-to-point channel, the station that gains control of the channel first, on a multipoint channel, the station controlling communications. (2) In high level data link (HLDL), the part of a data station that supports the primary control functions of the data link, generates commands for transmission, and interprets received responses. (3) In SNA, the station on an SDLC data link that is responsible for control of the data link. There must be only one primary station on a data link. All traffic over the data link is between the primary station and a secondary station.

primary-to-secondary communications. Pertaining to data communications between two nodes that do not have equal status in the interchange. It allows a remote transaction program to send data to a logical device without being involved in the control of the network.

print queue. A file containing a list of the names of files waiting to be printed.

priority. (1) A rank assigned to a task that determines its precedence in receiving system resources. (2) The relative significance of one job to other jobs in competing for allocation of resources.

privileged user. The account with superuser or supervisor authority.

problem determination. The process of identifying the source of a problem. Often this process identifies programs, equipment, data communications facilities, or user errors as the source of the problem.

problem determination procedure. A prescribed sequence of steps aimed at recovery from, or circumvention of, problem conditions.

problem state. (1) One of two virtual machine protection states that run in the processor's unprivileged state. User-written application programs typically run in the problem state. See *operating system state*. (2) A state during which the processing unit cannot execute input/ output and other privileged instructions. Contrast with *supervisor state*.

procedure. (1) See *shell procedure*. (2) In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a procedure call. (3) The description of the course of action taken for the solution of a problem. (4) A set of related control statements that cause one or more programs to be performed.

process. (1) A sequence of actions required to produce a desired result. (2) An entity receiving a portion of the processor's time for executing a program. (3) An activity within the system begun by entering a command, running a shell program, or being started by another process. When a program is running, it is called a process. (4) In a computer system, a unique, finite course of events defined by its purpose or by its effect, achieved under given conditions. (5) Any operation or combination of operations on data. (6) In the operating system, the current state of a program that is running. This includes a memory image, the program data, variables used, general register values, the status of opened files used, and the current directory. Programs running in a process must be either operating system programs or user programs.

process accounting. An analysis of the use each process makes of the processing unit, memory, and I/O resources.

process ID (PID). A unique number assigned to a process that is running.

process image. See *new process image*.

process lock. Allows the calling process to lock or unlock both its text and data segments into memory.

profile. (1) A file containing customized settings for a system or user. (2) Data describing the significant features of a user, program, or device. (3) In security, a description of the characteristics of an entity to which access is controlled. (4) A description of the control available to a particular network operator.

program. (1) A file containing a set of instructions, conforming to a particular programming language syntax. (2) A sequence of instructions suitable for processing by a computer. Processing may include the use of an assembler, a compiler, an interpreter, or a translator to prepare the program for execution, as well as to execute it. (3) In programming languages, a logical assembly of one or more interrelated modules. (4) To design, write, and test computer programs.

program date. The date associated with a program or job step. See also *creation date*, *session date* and *system date*.

Program Initialization Parameters (PIP). Data passed to a program when it starts running that modify the actions taken by that program or the environment in which that program runs.

program temporary fix (PTF). A temporary solution to, or bypass of, a defect in a current release of a licensed program.

program text. The executable part of a program. See *text*.

program unit. A main program or a subprogram. Synonym for module.

programmable terminal. (1) A user terminal that has computational capabilities. (2) A terminal that can be programmed to performed user-determined functions.

Programmers' Hierarchical Interactive Graphics System (PHIGS). A proposed ANSI and ISO standard. PHIGS defines an application programming interface designed for interactive two-dimensional and three-dimensional graphics applications.

prompt. A displayed symbol or message that requests information or operator action.

propagation time. The time necessary for a signal to travel from one point on a communications line to another.

protected field. A displayed field in which operators cannot enter, modify, or erase data.

protection. An arrangement for restricting access to or use of all or part of a computer system.

protection state. An arrangement for restricting access to or use of all or part of the hardware instruction set. Hardware protection states are privileged state and unprivileged state. Virtual machine protection states are operating system state and problem state.

protocol. (1) In SNA, the meaning of, and the sequencing rules for, requests and responses used for managing a network, transferring data, and synchronizing the states of network components. (2) A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication.

pseudo-terminal device (PTY). A pair of bi-directional character device drivers that implement a psuedo terminal, which can act as a keyboard and a display to existing software that uses the standardd terminal device interface termio.

PSDN. See *public switched data network.*

PSN. See *public switched network.*

PSTN. See *public switched telephone network.*

PTF. See *program temporary fix.*

PTN. See *public telephone network (PTN).*

PTY. See *pseudo-terminal device.*

PU. See *physical unit.*

public data network (PDN). A communications common carrier network providing data communications services over switched or nonswitched lines.

public switched data network (PSDN). See *public switched network (PSN).*

public switched network (PSN). A communications service through which users can be connected by dialing specific service address numbers.

public switched telephone network (PSTN). See *public telephone network (PTN).*

public telephone network (PTN). A communications common carrier network that provides voice and data communications services over switched or non-switched lines.

pulse dialing. Rotary dialing.

pulse modulation. Transmitting data by altering (modulating) a pulsed or intermittent carrier.

PUT 2.0 or PUT 2.1. A peripheral node that has limited addressing and path control routing capabilities. A PUT 2.0 node depends on subarea nodes (PUT 4 and PUT 5) to translate between its local addressing and network addressing. PUT 2.0 does not support the full capabilities of LU 6.2; PUT 2.1 does. SNA Services operates only as either a PUT 2.0 or a PUT 2.1 peripheral node.

PUT 4 or PUT 5. A subarea node that provides network-wide addressing and control data flow within a subarea (the subarea node and all peripheral nodes connected to it). PUT 4 does not contain an SSCP component; PUT 5 does. SNA Services cannot perform the functions of a PUT 4 or a PUT 5 subarea node.

Q - R

qdaemon. The daemon process that maintains a list of outstanding jobs and sends them to the specified device at the appropriate time.

qualified name. (1) A name made unique by the addition of one or more qualifiers. (2) A data name explicitly accompanied by a specification of the class to which it belongs in a specified classification system.

qualifier. (1) A name used to uniquely identify another name. (2) A modifier that makes a name unique. (3) All names in a qualified name other than the rightmost, which is called the simple name.

query. (1) The action of searching data for desired information. (2) The process by which a master station asks a slave station to identify itself and to give its status. (3) In interactive systems, an operation at a terminal that elicits a response from the system. (4) A request for information from a file based on specific conditions.

queue. (1) A line or list formed by items waiting to be processed. (2) To arrange in or form a queue.

queue element. A block of data, or item, in a queue.

queued message. A message from the system that is added to a list of messages stored in a file for viewing by the user at a later time. Backgroundd processes produce queued messages; programs that operate directly with the user do not. These programs produce messages that are sent directly to the screen for the user to see immediately.

quit. A key, command, or action that tells the system to return to a previous state or stop a process.

quote. To mask the special meaning of certain characters; to cause them to be taken literally.

race condition. The condition in the signal system call when the signal occurs during the time period when the signal action is set to SIG_DFL, and the signal-catching function has not had time to establish itself as the catcher for this signal.

RAM. Random access memory.

random access. An access mode in which records can be read from, written to, or removed from a file in any order.

raster array. (1) In computer graphics, a predetermined arrangement of lines that provide uniform coverage of a display space. (2) The coordinate grid that divides the display area of a display device.

RCS. Revision Control System manages multiple revisions of text files. This system, written by W.F.Tichy at Purdue University, was designed to control frequently revised text such as programs, form letters, and papers. It features automatic identification, storage, logging, retrieval, and merging of file revisions. See also *SCCS* and *sccstorcs*.

read lock. Prevents any other process from setting a write lock on any part of the protected area. See also *write lock* and *locks*.

read-only file. Pertaining to file system mounting, a condition that allows data to be read, but not copied, printed, or modified. Contrast to *editable file*.

real memory. The active physical memory on any system. Contrast with *virtual memory*.

real time. (1) Pertaining to the processing of data by a computer in connection with another process outside the computer, according to time requirements imposed by the outside process. (2) Used to describe systems operating in conversational mode and processes that can be influenced by human intervention while in progress. (3) Pertaining to an application such as a process control system or a computer-assisted instruction system in which response to input is fast enough to affect subsequent input.

real-time system. A system that receives and processes data so the data or result is available for immediate use.

receive pacing. In SNA. the pacing of message units that a component is receiving. See also *send pacing*.

receive time-out. In data communications, the result of no data being received in a given period of time.

Recommendation X.25 (Geneva 1980). A Consultative Committee on International Telegraph and Telephone (CCITT) recommendation for the interface between data terminal equipment and packet-switched data networks.

record. (1) In programming languages, an aggregate that consists of data objects, possibly with different attributes, that usually have identifiers attached to them. (2) A set of data treated as a unit. (3) A collection of fields treated as a unit.

record lock. A lock that read or write locks some portion or all of a file.

recovery procedure. (1) An action performed by the operator when an error message appears on the display screen. Usually, this action permits the program to continue or permits the operator to run the next job. (2) The method of returning the system to the point where a major system error occurred and running the recent critical jobs again. (3) A process in which a specified data station attempts to resolve conflicting erroneous conditions arising during the transfer of data.

recursion. (1) The process of using a function to define itself. (2) the performance of an operation in several steps, with each step using the output of the preceding step.

redirect. To divert data from a process to a file or device to which it would not normally go.

Reduced Instruction Set Computer (RISC). A computer design that uses a small simplified set of frequently used instructions for rapid execution.

reentrant module. A module that allows the same copy of itself to be used concurrently by two or more tasks. Contrast with *serially reusable*.

reference count. In an i-node, a record of the total number of directory entries that refer to the i-node.

regular expression. (1) A set of characters, metacharacters, and operators that define a string or group of strings in a search pattern. (2) A string that contains wildcard characters and operations that define a set of one or more possible strings. Contrast with *literal string*.

reject. To cause portions of applied updates to not become permanent parts of the product. based on the results of a test period. Contrast to *commit*.

relational expression. (1) A logical statement describing the relationship (such as greater than or equal) of two arithmetic expressions or data items. (2) An expression that consists of an arithmetic expression followed by a relational operator, followed by another arithmetic expression, and that can be reduced to a value that is true or false.

relational operator. (1) The reserved words or symbols used to express a relational condition or a relational expression. (2) An operator that operates on at least two operands and yields a truth value.

relative directory. A directory whose name does not begin with / (slash).

relative path name. The name of a directory or file expressed as a sequence of directories followed by a file name, beginning from the current directory. Relative path names do not begin with / (slash), but are relative to the current directory.

release number. The number identifying a distribution of a new product or new function and APAR fixes for an existing product. The first version of a product is announced as Release 1, Modification, Level 0.

relocatable. (1) A value, expression, or address is relocatable if it does not have to be changed when the program is relocated. (2) Attribute of a set of codes whose address constants can be altered to make up for a change in origin.

remote. Pertaining to a system or device that is accessed through a communications line. Contrast with *local*.

remote host. Any host on the network except the one at which a particular operator is working.

remote job entry (RJE). Submission of a job through an input unit that has access to a computer through a data link.

remote transaction program name (RTPN). The name of a transaction program at the other (remote) end of a conversation.

repetitive tiling operation. In a pixel map definition, an operation that consists of repeatedly copying a 16x16-pixel tile rectangle, pointed to by the tile pixel map data address, to fill a rectangle area of a size specified by the height and width parameters of this call. The format of the tile data is determined by the format defined in the flags filed of the tile pixel map structure.

reply. (1) A response to an inquiry. (2) In SNA, a request unit sent only in reaction to a received request unit. For example, Quiesce Complete is the reply sent after receipt of Quiesce At End of Chain.

request. (1) A message unit that signals initiation of a particular action or protocol. (2) A directive, by means of a basic transmission unit, from an access method that causes the network control program to perform a data- transfer operation or auxiliary operation. (3) In SNA, a message unit that signals initiation of an action or protocol.

requester. A display station or interactive communications session that requests a program to be run.

required parameter. A parameter having no value automatically supplied. The user must provide a value.

required value. See *required parameter*.

reserved character. A character or symbol that has a special (non-literal) meaning unless quoted.

reset. (1) To cause a counter to take the state corresponding to a specified initial number. (2) To put all or part of a data processing device back to a prescribed state. (3) On a virtual circuit, reinitialization of data flow control. (4) To return a device or circuit to a clear state.

An answer to an inquiry.

response. (1) In SNA, a message unit that acknowledges receipt of a request; a response consists of a response header (RH), a response unit (RU), or both.

response time. The time it takes for a data communications system to respond to a request. For example, if you enter a customer number on a terminal keyboard, response time begins when you press the last key and ends when the first character of your answer is displayed at the terminal.

restore. To return to an original value or image. For example, to restore a library from diskette.

retransmit. To repeat the transmission of a message or segment of a message.

retry. To try the operation again that caused the device error message.

return code. (1) A value that is returned to a program to indicate the results of an operation issued by that program. (2) A code used to influence the execution of succeeding instructions.

return value. See *return code*.

ring. A method used to distribute data in a LAN. See also *ring network*.

ring buffer. A virtual terminal in Monitor Mode can share a ring buffer with an application and place data from input devices in the buffer. The ring buffer mechanism dramatically shortens the imput data path from the virtual terminal to the application.

ring network. (1) A network in which every node has two branches connected to it. (2) A network configuration in which devices areconnected by unidirectional transmission links to form a closed path.

RIP. See *Routing Information Protocol*.

RISC. See *Reduced Instruction Set Computer*.

RJE. See *remote job entry*.

roller ball. The sphere inside a mechanical mouse that contacts a desktop or other hard surface.

root. The user name for the system user with the most authority. See also *superuser*.

root directory. The directory, created when the disk is formatted, that contains all other directories in the AIX file system.

root file system. The basic AIX file system, which contains operating system files and onto which other file systems can be mounted. The root file system is the file system that contains the files that are run to start the system running.

root segment. (1) In an overlay operation, the part of a program that must remain in main storage when other overlay segments are executed; the first segment of a program with overlays. The root segment remains in main storage at all times while the program is being run. (2) In a hierarchical data base, the highest segment in the tree structure.

route. A path defined for sending data across a network.

routine. A set of statements in a program causing the system to perform an operation or a series of related operations. See also *macro* and *subroutine*.

routing. (1) In SNA, the forwarding of a message unit along a particular path through a network as determined by parameters carried in the message unit, such as the destination network address in a transmission header. (2) The assignment of the path by which a message will reach its destination.

Routing Information Protocol (RIP). A variant of the Xerox NS Routing Information Protocol, used to maintain current kernel routing table entries.

routing table. A structure in memory that describes the routes that are currently defined. Used for Internet routing.

RRI. Request/response indicator.

RS-232C. An EIA interface standard that defines the physical, electronic, and functional characteristics of an interface line connecting a modem and associated work station. It uses a 25-pin connector, and an unbalanced line voltage.

RS-366. An EIA interface standard that defines the physical, electronic, and functional characteristics of an interface line connecting data terminal equipment and automatic calling equipment. This system automatically dials a remote location when the data terminal equipment activates the appropriate interface circuits.

RS-422A. An EIA interface standard that defines the physical, electronic and functional characteristics of an interface line connecting data terminal equipment and data communications equipment. It uses a balanced line voltage for noise reduction and longer distance capability. The RT PC uses the send and receive pins from the set of 40 pins defined by the RS-422A interface.

RTI. Response type indicator.

RTPN. See remote transaction program name.

run. (1) A performance of one or more jobs or programs. (2) To cause a program, utility, or other machine function to be performed.

run file. The output of the linkage editor. A program file in a format that is suitable for being loaded into main storage and run.

run-time environment. A collection of subroutines and shell variables that provide commonly used functions and information for system components.

S - T

SAP. Service Access Point.

scaling. (1) In computer graphics enlarging or reducing all or part of a display image by multiplying the coordinates of the image by a constant value. (2) In programming, indicating the number of digit positions in object code to be occupied by the fractional portion of a fixed-point or floating-point constant.

scatter. For input/output operations, reading data from a device and locating it in noncontiguous memory addresses. See *gather* (antonym).

scratch file. A file, usually used as a work file, that exists temporarily, until the program that uses it ends.

screen. (1) See *display screen*.

scroll. To move information vertically or horizontally to view information that is outside the display or pane boundaries.

SCSI. See *small computers system interface*.

sdb. See *symbolic debugger*.

SDLC. See *synchronous data link control*.

SDLC primary station. A station that has responsibility for the data link; it issues commands to secondary stations.

SDLC secondary station. A station that responds to requests from another station (the primary station) and has little control over data link operations.

SDT. See *static debugger trap*.

second level interrupt handler (SLIH). A routine that handles the processing of an interrupt from a specific adapter. An SLIH is called by the first level interrupt handler associated with that interrupt level.

secondary command processor. A copy of the command processor which assumes control from the original command processor.

secondary station. A data station that executes data link control functions as instructed by the primary station. It interprets received commands and generates responses for transmission.

sector. (1) An area on a disk track or a diskette track reserved to record information. (2) The smallest amount of information that can be written to or read from a disk or diskette during a single read or write operation. (3) On disk or diskette storage, an addressable subdivision of a track used to record one block of a program or data. (4) The part of a track or band on a magnetic drum, magnetic disk, or disk pack that can be accessed by the magnetic heads in the course of a predetermined angular displacement of the data medium.

security. The protection of data, system operations, and devices from accidental or intentional ruin, damage, or exposure.

segment. (1) A contiguous area of virtual storage allocated to a job or system task. A program segment can be run by itself, even if the whole program is not in main storage. (2) Virtual memory is divided into segments which are linearly- addressable spaces of one or more 2K-byte pages up to a maximum size of 2 to the 28th power bytes. (VRM PR, 3-21) (3) A portion of a computer program that may be executed as an entity without the entire program being maintained in main storage. (4) A group of display elements.

segmenting of BIUs. An optional function of path control that divides a basic information unit (BIU) received from transmission control into two or more path information units (PIUs). The first PIU contains the request header of the BIU and usually part of the RU; the remaining PIU or PIUs contain the remaining parts of the RU.

select. (1) To choose a button on the display screen. (2) To select, place the cursor on an object (name or command) and press the Select (left) button on the mouse or the **Select** key on the keyboard.

selection. (1) Addressing a terminal or a component on a selective calling circuit. (2) The process by which a computer requests a station to send it a message. (3) See *addressing*.

semaphore. (1) Entity used to control access to system resources. Processes can be locked to a resource with semaphores if the processes follow certain programming conventions. (2) Provides a general method of communication between two processes that is an extension of the features of signals.

semaphore ID. An integer that points to a set of semaphores and a data structure that contains information about the semaphores.

semid. See *semaphore ID*

send pacing. In SNA, pacing of message units that a component is sending. See also *receive pacing*.

sense code. A value sent or received or a negative response to indicate what error occurred.

separator. A punctuation character used to separate parts of a command or file, or to delimit character strings.

sequential access. (1) An access method in which records are read from, written to, or removed from a file based on the logical order of the records in the file. (2) The facility to obtain data from or enter data into a storage device so that the process depends on the location of the data and on a reference to data previously accessed.

sequential execution. the execution of a list of pipelines where the shell waits for a pipeline to finish before executing the next one.

sequential I/O model. A model of the operating system for all accesses to system network resources. When SNA supports this model, it simplifies access to the network, allows programs to be designed for portability, and allows programs to use network resources via redirection.

sequential processing. (1) The processing of logical records in the order in which they are accessed. (2) The processing of records in the order in which they exist in a file. Synonym for *consecutive processing*. See also *random processing*.

serial device. A device that performs functions sequentially, such as a serial printer that prints one character at a time. Contrast with *parallel device*.

serial transmission. Transmitting each bit of a data character separately over the same electrical path.

serially reusable load module. A module that can be accessed by only one task at a time. It cannot be used by a second task until the first task is complete. Contrast with *reentrant*.

server. (1) An application program that usually runs in the background (*daemon*) and is controlled by the System Resource Manager. (2) On a network, the computer that contains the data or provides the facilities to be accessed by other computers on the network. (3) A program that handles protocol, queuing, routing, and other tasks necessary for data transfer between devices in a computer system.

service request number. A group of numbers used by service technicians to determine the failing area of the system.

service transaction program. (1) A program that provides a function which is internal to SNA Services. (2) A transaction program implemented by a transaction prôcessing system. Service transaction programs perform such functions as providing access to remote data bases and remote queues.

session. (1) The period of time during which programs or devices can communicate with each other. (2) A name for a type of resource that controls local LU's, remote LU's, modes, and attachments. (3) In network architecture, an association of facilities necessary for establishing, maintaining, and releasing connections for communication between stations. (4) The period of time during which the user of a terminal can communicate with an interactive system, usually elapsed time between logon and logoff. (5) In SNA, a logical connections between two network addressable units (NAUs) that can be activated, tailored to provide various protocols, and deactivated as requested.

session date. The date associated with a session. See also *creation date*, *program date*, and *system date*.

session records. In the accounting system, a record of time connected and line usage for connected display stations, produced from log in and log out records.

session-level pacing. In SNA, a flow control technique that permits a receiving half-session to control the data transfer rate (the rate at which it receives request units). It is used to prevent overloading a receiver with unprocessed requests when the sender can generate requests faster than the receiver can process them.

set-group-ID bit. In setting file access permissions, sets the process's effective group ID to the file's group on execution.

set-user-ID bit. In setting file access permissions, sets the process's effective user ID to the file's owner on execution.

severity code. A code that indicates how serious an error condition is.

shared code libraries. Libraries that contain common subroutines that multiple programs can access at runtime as a memory-mapped file. This eliminates the need for each program to have a copy of the subroutine linked into its object module.

shared memory. A area of memory that more than one cooperating process can access simultaneously.

shared memory ID. An identifier assigned to the shared segment for use within a particular process. It is similar in use to a *file descriptor* of a file.

shared printer. A printer that is used by more than one work station.

shell. (1) A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens and communicate them to the operating system. (2) Software that allows a kernel prgram to run under different operating system environments. (3) The command interpreter providing the user with an interface to the AIX kernel. See *shell program*.

shell control command. A command that enables the user to pass control to various parts of a shell procedure or control how a procedure ends.

shell procedure. A series of commands combined in a file that carry out a particular function when the file is run or when the file is specified as an argument to the sh command. Shell procedures are frequently called shell scripts.

shell program. A program that accepts and interprets commands for the operating system (there is an AIX shell program and a DOS shell program).

shell prompt. The character string on the command line indicating the the system can accept a command (typically the $ character).

shell script. See *shell* procedure.

shell variables. Facilities of the shell program for assigning variable values to constant names.

shielded twisted pair. A transmission medium of two twisted conductors with a foil or braid shield.

shmid. See *shared memory ID*.

shutdown. The process of ending operation of a system or a subsystem, following a defined procedure.

SID. SCCS identification; the name assigned to a delta.

sign-off. To end a session at a display station.

sign-on. To begin a session at a display station.

signal. (1) A simple method of communication between two processes. One process can inform the other process when an event occurs. (2) In AIX operations, a method of inter-process communication that simulates software interrupts.

signal handler. A subroutine called when a signal occurs.

signal mask. Defines the set of signals currently blocked form delivery to a process.

signal stack. A alternate stack on which signals are to be processed.

Simple Mail Transfer Protocol (SMPT). In the IP protocol used with TCP/IP, the protocol that handles transferring and forwarding mail.

single-step instruction execution. A method of operating a computer in which each instruction is performed in response to a single manual operation. No sequential execution of instructions is allowed.

size. The screen management action that changes the size of a window.

size field. In an i-node, a field that indicates the size, in bytes, of the file associated with the i-node.

sleeping process. The state of a process that is waiting for input or output to complete, time slices, an event to occur, or signals from other processes. When a process is sleeping, it may be paged out of memory.

SLIH. See *second level interrupt handler*.

slot. A long electrical socket inside the system unit into which an electronic circuit board (card) is installed.

slow list. A list of secondary stations on a multidrop network that are polled less often by the primary station due to their inactivity.

SMPT. See *Simple Mail Transfer Protocol*.

SNA. See *Systems Network Architecture*.

SNA network. In SNA, the part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network addressable units (NAUs), boundary-function components, and the path control network.

socket. (1) A unique identifier created by the concatenation of a port identifier with an Internet address. (2) A port identifier. (3) A 16-bit port number.

software. Programs, procedures, rules, and any associated documentation pertaining to the operation of a system. Contrast with hardware.

sort. To rearrange some or all of a group of items based upon the contents or characteristics of those items.

sort utility. The part of the program product used to arrange records (or their relative record numbers) in a sequence determined by data contained in one or more fields within the records.

source. (1) A system, a program within a system, or a device that makes a request to a target. Contrast with *target*. (2) In advanced program-to-program communictions, the system or program that starts jobs on another system.

source code. The input to a compiler or assembler, written in an source language. Contrast to *object code*.

source diskette. The diskette containing data to be copied, compared, restored, or backed up. Compare with *target diskette*.

source file. A file that contains source statements for such items as high level language problems and data description specifications.

source module. The statements or codes that form input to the assembler.

source program. (1) A computer program expressed in a source language. (2) A set of instructions written in a programming language, that must be translated to machine language before the program can be run.

space. (1) A site intended for storage of data, such as a location in a storage medium. (2) A basic unit of area, usually the size of a single character. (3) One or more space characters. (4) In a neutral circuit, an impulse that causes the loop to open or causes absence of signal. In a polar circuit, it causes the loop current to flow in a direction opposite to that for a mark impulse. A space impulse is equal to a binary zero.

special character. A character other than a letter or number. For example; *, +, and % are special characters.

special file. Special files are used in the AIX system to provide an interface to input/output devices. There is at least one special file for each device connected to the computer. Contrast with *directory* and *file*. See also *block special file* and *character special file*.

special requirement file. In the updating program, used by the updatep command to determine the grouping of descriptive titles to be displayed by the user during the update procedure.

spool file. (1) A disk file containing output that has been saved for later printing. (2) Files used in the transmission of data among devices.

spooling (simultaneous peripheral operation online). (1) The use of auxiliary storage as a buffer storage to reduce processing delays when transferring data between peripheral equipment and the processors of a computer. (2) Reading and writing input and output streams on an intermediate device in a format convenient for later processing. (3) Performing a peripheral operation such as printing while the computer is busy with other work.

SRC. See *System Program Controller.*

SRN. Service request number.

SSCP. See *System Services Control Point.*

stack. (1) An area in storage that stores temporary register information and returns addresses of subroutines. (2) A list constructed and maintained so that the last data element stored is the first data element retrieved. (3) In kernel mode, an area that is paged with the user process. The kernel maintains a stack for each process. It saves the process information such as the call chain and local variables used by the kernel for the user process.

stack buffer. A storage area that stores retrievable data in sequence. The last text stored is the first text removed.

stand-alone shell. A limited version of the shell program used for system maintenance.

stand-alone work station. A work station that can be used to preform tasks independent of (without being connected to) other resources such as servers or host systems.

standard error. The place where many programs place error messages.

standard input. The primary source of data going into a command. Standard input comes from the keyboard unless redirection or piping is used, in which case standard input can be from a file or the output from another command.

standard output. The primary destination of data coming from a command. Standard output goes to the display unless redirection or piping is used, in which case standard output can be to a file or another command.

stanza. A group of lines in a file that together have a common function or define a part of the system. Stanzas are usually separated by blank lines, and each stanza has a name.

start/stop. Asynchronous transmission such that a group of signals representing a character is preceded by a start element and followed by a stop element. See also *asynchronous transmission.*

state. (1) In a circuit, a state in which the circuit remains until application of a suitable pulse. Synonym for stable state. (2) One of the separate, restartable portions into which the runacct command (the main daily accounting shell procedure) breaks its processing.

statement. (1) An instruction in a program or procedure. (2) In programming languages, a language construct that represents a step in a sequence of actions or a set of declarations. (3) In computer programming, a symbol string or other arrangement of symbols.

static debugger trap (SDT). A trap instruction placed in a pre-defined point in code invokes the debugger at that point. The trap instruction causes a program check when executed; debugger is invoked as a result of the program check.

station. (1) A computer or device that can send or receive data. (2) An input or output point of a system that uses telecommunication facilities; for example, one or more systems, computers, terminals, devices, and associated programs at a particular location that can send or receive data over a telecommunication line. (3) A location on a device at which an operation is performed. (4) In SNA, a link station.

status. (1) The current condition or state of a program or device. For example, the status of a printer. (2) The condition of the hardware or software, usually represented in a status code.

stderr. See *standard error.*

stdin. See *standard input.*

stdout. See *standard output.*

stop bit. (1) In start-stop transmission, a signal at the end of a character that prepares the receiving device for reception of a subsequent character. (2) A signal to a receiving mechanism to wait for the next signal. Synonym for stop signal.

storage. (1) The location of saved information. (2) In contrast to memory, the saving of information on physical devices such as disk or tape. See *memory.* (3) A unit into which recorded text can be entered, retained, and processed, and from which it can be retrieved. (4) The action of placing data into a storage device.

storage device. (1) A functional unit for storing and/or retrieving data. (2) A facility into which data can be retained.

store. To place information in a storage decvice (in memory or onto a diskette, fixed disk, or tape) so that it is available for retrieval and updating.

stream. (1) Sequential input or output from an open file descriptor. (2) A continuous stream of data elements being transmitted, or intended for transmission, using a defined format. (3) All data transmitted through a data channel in a single read or write operation. Synonym for data stream.

stream editor. The sed command, which modifies lines from a specified file, according to an edit script, and writes them to a standard output.

streaming tape drive. A magnetic tape unit that stores large amounts of data and is designed to make a nonstop dump or restore of magnetic disks without using interblock gaps.

string. A linear sequence of entities such as characters or physical elements. Examples of strings are alphabetic string, binary element string, bit string, character string, search string, and symbol string.

structured field. (1) A mechanism that permits variable length data to be encoded for transmission in the data stream. (2) See *field*.

structured file. A special type of INed file that contains specialized data such as, information about the structure of the data in the file, and history information about changes that have been made to the file. Structured files can contain hierarchical data that is displayed and edited by using forms.

su. See *superuser*.

subarea node. In data communication, a node that uses network addresses for routing and whose routing tables are affected, therefore, by changes in the configuration of the network. Subarea nodes can provide boundary function support for peripheral nodes.

subcommand. A request for an operation that is within the scope of work requested by a previously issued command.

subdirectory. A directory contained within another directory in the file system hierarchy.

subhost. A communications system that controls attached terminals in addition to communicating with another (usually higher level) system.

subprogram. A program invoked by another program, such as a subshell.

subroutine. (1) A sequenced set of statements that may be used in one or more computer programs and at one or more points in a computer program. (2) A routine that can be part of another routine.

subscript. (1) An integer or variable whose value refers to a particular element in a table or an array. (2) A symbol associated with the name of a set to identify a particular subset or element. (3) Characters printed one-half line below the normal printing line.

subscript declarator. In an array definition or declaration, the bracketed expressions following the array name. A subscript declarator specifies the number elements in an array dimension.

subserver. A system resource or program that is directly controlled by a server program running under control of the Systems Resource Manager.

subshell. An instance of the shell program started from an existing shell program.

substring. A part of a character string.

subsystem. (1) A secondary or subordinate system, usually capable of operating independently of, or synchronously with, a controlling system. (2) The part of communications that handles the requirements of the remote system, isolating most system-dependent considerations from the application program.

suffix. (1) A character string attached to the end of a file name that helps identify its file type. (2) A code dialed by a caller who is already engaged in a call.

superuser. A system user with super user privileges; a user who operates without restrictions.

superuser privileges. The unrestricted ability to access and modify any part of the operating system associated with the user who manages the system.

superblock. In a file system layout, refers to Block 1, which is used to keep track of the file system and is the most critical part of the file system. It contains information about every allocation or deallocation of a block in the file system. See also *i-list*.

supervisor. The part of the AIX control program that coordinates the use of resources, and maintains the flow of processing unit operations.

supervisor call (SVC). An instruction that interrupts the program being executed and passes control to the supervisor so it can perform a specific service indicated by the instruction.

suspended state. (1) The resource is temporarily not receiving a request. A start action request will return the resource to the state it was in prior to being suspended. (2) A software state in which a task is not dispatched by the system and is not contending for the processor.

SVC. See *supervisor call.*

swapping. (1) The process of temporarily removing an active job from main storage, saving it on disk, and processing another job in the area of main storage formerly occupied by the first job. (2) A process that interchanges the contents of an area of real storage with the contents of an area in auxiliary storage. (3) In a system with virtual storage, a paging technique that writes the active pages of a job to auxiliary storage and reads pages of another job from auxiliary storage into real storage.

switched line. In data communications, a connection between computers or devices established by dialing. Contrast with *nonswitched line*.

switched link. A link between two nodes that is established by dialing.

switched network backup (SNBU). In data communications, a technique providing a switched line connection when a nonswitched line fails.

symbolic debugger (sdb). A command that debugs programs written in certain high-level languages.

SYN. See *synchronization character*.

synchronization (SYN) character. In binary synchronous communications, the transmission control character that provides a signal to the receiving station for timing.

synchronous. (1) Pertaining to two or more processes that depend upon the occurrences of specific events such as common timing signals. (2) Occurring with a regular or predictable time relationship or sequence.

Synchronous Data Link Control (SDLC). (1) A form of communications line control using commands to control the transfer of data over a communications line. Compare with *binary synchronous communications.*

On the RT PC, SDLC uses SNA protocols. An SDLC connection tends to be faster and more reliable than an asynchronous connection; however, the modems for SDLC may be more expensive. (2) A discipline conforming to subsets of the ADCCP of the ANSI and the HDLC of the International Organization for Standardization. It manages synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half- duplex over switched or non-switched links. The configuration of the link connection may be point-to-point, multipoint, or loop.

synchronous transmission. (1) In data communications, a method of transmission in which the sending and receiving of characters is controlled by timing signals. Contrast with *asynchronous transmission*. (2) Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time base.

synopsis. Synonym for *syntax*.

syntax. (1) The rules for the construction of a command, statement, or program. (2) The relationship among characters or groups of characters, independent of their meanings or the manner of their interpretation and use. (3) structure of expressions in a language. (4) The relationship among symbols.

syntax diagram. A diagram for each AIX operating system command that displays how to enter the command correctly on the command line.

system. The computer and its associated devices and programs.

system board. The main circuit board in the system unit that supports a variety of basic system devices, such as a keyboard, a mouse, and so forth. The system board also supplies other basic system functions.

system call. A request by an active process for a service by the system kernel.

system customization. A process of specifying the devices, programs, and users for a particular data processing system.

system date. The date assigned by the system user during setup and maintained by the system.

system dump. A copy of storage from all active programs (and their associated data) whenever an error stops the system. Contrast with *task dump*.

system event log. A collection of records about system events and activities that is maintained by a transaction processing system.

system image. An operating system image, or core image, or the running kernel.

system management. The tasks involved in maintaining the system in good working order and modifying the system to meet changing requirements.

system parameters. See *kernel parameters*.

system profile. A file containing the default values used in system operations.

System Program Controller. A system program that controls the operation of other application programs that run in the background (daemons).

system scheduler. In kernel mode, performs the basic time-sharing that enables the processor to be shared among many users.

system services control point (SSCP). In SNA, the focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other the session services for network end users. Multiple SSCPs, cooperating as peers, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its domain.

system unit. The part of the system that contains the processing unit, the disk drive and the disk, and the diskette drive.

system user. A person, device, or system that uses the facilities of a computer system.

Systems Network Architecture (SNA). (1) An IBM protocol for controlling the transfer of information in a data communications network. (2) The description of the logical structure, formats, protocols, and operating sequences for transmitting information units through, and controlling the configuration and operation of, networks.

table. (1) An array of data, each item of which can be unambiguously identified by means of one or more arguments. (2) A two-dimensional array in which each item and its position with respect to other items is identified.

tablet. A special flat surface with a mechanism for indicating positions thereon, normally used as a locator.

tape drive. A mechanism for moving magnetic tape and controlling its movement.

target. A system, a program within a system, or a device that interprets, rejects or satisfies, and replies to requests received from a source. Contrast with *source*.

target diskette. The diskette to be used to receive data from a source diskette.

target file. A file created by the make program that contains a completed program.

task. (1) A basic unit of work to be performed. Examples are a user task, a server task, and a processor task. (2) A process together with the procedures that run the process. (3) In a multiprogramming or multiprocessing environment, one or more sequences of instructions treated by a control program as an element of work to be accomplished by a computer.

task scheduler. In a task control block structure, a non-preemptive, round-robin scheduler that loops through the circular list of tasks until it finds one that can be run and then switches to that task.

TCP/IP. Transmission Control Protocol/Internet Protocol.

telecommunication. (1) The transmission of control signals and information between two or more locations. (2) The transmission of data between computer systems over telecommunication lines and between a computer system and remote devices.

teleprocessing. Processing data that is received from or transmitted to a remote location via communication channels. Synonym for remote access data processing.

telnet. In TCP/IP, the protocol that opens the connection to the system.

template. (1) A representation of a keyboard that includes functions not engraved on the keyboard. (2) Each command line stored in the buffer. (3) In enhanced edit mode, a special character buffer associated with the terminal.

temporary error. An error that requires an operation to be retriedd a number of times before being successfully completed.

terminal. (1) A device, usually equipped with a keyboard and a display device, capable of sending and receiving information over a communications line. See *work station*. (2) A point in a system or communication network at which data can either enter or leave. (3) In curses and extended curses, a special screen that represents what the work station's display screen currently looks like. The terminal screen is identified by a window named **curscr**, which should not be accessed directly by the user. Instead, changes should be made to **stdscr** (or a user-defined screen) and then **refresh** (or **wrefresh**) should be called to update the terminal.

terminal screen. See *display screen*.

terminator. The part of the program product that performs the action necessary to end a job or program.

text. (1) A type of data consisting of a set of linguistic characters (for example, alphabet, numbers, and symbols) and formatting controls. (2) The executable portion of a program. (3) In kernel mode, contains kernel program code that executes. It is read only by a user process. (4) In ASCII and data communications, a sequence of characters treated as an entity if preceded by one STX and terminated by one ETX communication control character. (5) In word processing, information for human comprehension that is intended for presentation in a two-dimensional form, such as data printed on paper or displayed on a screen. (6) The part of a message that is not the header or control information.

text application. A program defined for the purpose of processing text data (for example, memos, reports, and letters).

text cursor. A cursor that indicates where to type the next character. The text cursor is controlled by the keyboard.

text editing program. See *editor* and *text application*.

text table. In kernel mode, a table maintained by the system when text segments are shared by processes. It is used to track shared text segments.

texttab. A kernel parameter establishing the size of the text table, in memory, that contains one entry each active shared program text segment.

thrashing. In a virtual storage system, a condition in which the system is doing so much paging that little useful work can be done.

threshold. (1) A logic operator having the property that if P is a statement, Q is a statement, R is a statement, ..., then the threshold of P,Q,R, ... is true if at least N statements are true, and false if less than N statements are true. N is a specified nonnegative integer called the threshold condition. (2) In computer graphics, a level above which all gray-scale image data can be represented as white and below which all gray-scale image data can be represented as black.

time sharing. (1) A technique developed to share a computer's resources among several users, so that users can execute programs concurrently and interact with the program during execution. (2) An operting technique of a computer system that provides for the interleaving in time of two or more processes in one processor.

time out. (1) Measurement of time interval allotted for certain events to occur (such as a response to polling or other controls) before corrective (recovery) action is taken. (2) An event that occurs at the end of a predetermined period of time that began at the occurrence of another specified event. (3) A terminal feature that logs off a user if an entry is not made within a specified period of time.

timing-dependent. In a program, relying on the amount of time it takes to perform a certain function.

TLB. See *translation lookaside buffer*.

toggle. (1) A switching device such as a toggle key on a keyboard. (2) Pertaining to any device having two stable states. (3) To switch between two modes on a computer or network.

token. (1) The smallest independent unit of meaning as defined by either the parser or the lexical analyzer. A token can contain data, a language keyword, an identifier, or other parts of a language syntax. (2) In M4, any string of letters and digits that **m4** recognizes. (3) A type of macro that the typesetting preprocessor replaces with an assigned string value. See also *string register*. (4) Tells the parser which pattern is being sent to it by the input routine. (5) In a local area network, the symbol of authority passed among data stations to indicate the station temporarily in control of the transmission medium.

token expansion. The process whereby a token is replaced with a string value.

token numbers. Nonnegative integers that represent the names of tokens.

Token-Ring. A token access procedure used with a sequential (ring) topology.

tone dialing. the use of keys or pushbuttons instead of a rotary dial to generate a sequence of digits that establishes a circuit connection. The signal form is usually tones.

TPN. See *Transaction Program Name (TPN)*.

trace. (1) To record data that provides a history of events occurring in the system. (2) A record of the execution of a computer program. It exhibits the sequences in which the instructions were executed. (3) To monitor system performance or aid in debugging programs.

trace channel. Each bit in a table that is set up in memory by the trace daemon, after it reads the trace profile. Each event class is represented by one or more bits.

trace daemon. The part of the system's trace facilities that reads the trace profile to determine which event classes should be active, opens the trace log file and reads the trace buffers, and as they become full, writes them out to the trace log file.

trace entry. A data structure containing a header of identifying information plus up to 20 bytes of defined data. Trace entries are generated by trace points and written to a trace log file.

trace ID. One of the input parameters in a trace subroutine. It contains a channel number and hook ID to identify which event class the trace entry belongs to and which trace point generated the call.

trace log. A file formatted by the trace formatter, to which trace entries are written.

trace point. A group of code statements that generates a trace entry from within a software program. Trace points are assigned to an event class which can be active or inactive. Trace points with active event classes can generate trace entries.

trace profile. An ASCII file that can be modified to activate or deactivate the various event classes. The trace profile is used by the trace daemon to set up three channel tables that show which event classes are active.

trace table. A storage area into which a record of the performance of computer program instructions is stored.

trace template. Used by the trace formatter to determine how the data contained in trace entries should be formatted.

track. (1) A circular path on the surface of a fixed disk or diskette on which information is magnetically recorded and from which recorded information is read. (2) The path or one of the set of paths on a data medium associated with a single reading or writing component as the data medium moves past the component.

transaction. (1) An exchange between a work station and a program, two work stations, or two programs that accomplish a particular action or result; for example, the entry of a customer's deposit and the updating of the customer's balance. (2) In a batch or remote batch entry, a job or job step.

transaction program. A program that processes transactions in an SNA network. There are two kinds of transaction programs: application transaction programs and service transaction programs. See also *conversation*.

Transaction Program Name (TPN). The name of an application program that uses data communications to send or receive data to or from another application program.

transcript. In remote communications, a file that contains the written record of commands you enter on the remote system, and the remote system's response to those commands.

transfer. To send data to one place and to receive data at another place.

translation look-aside buffer (TLB). Hardware that contains the virtual-to-real address mapping.

transmission control characters. Special characters that are included in a message to control communication over a data link. For example, the sending station and the receiving station use transmission control characters to exchange information; the receiving station uses transmission control characters to indicate errors in data it receives.

Transmission Control Protocol (TCP). Used in ARPA Internet and any network following the U.S. Department of Defense standards for inter-network protocol. Provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected system of such networks. It assumes that IP is the underlying protocol.

transparent. (1) In communications, pertaining to transmissions that have no possibility of interference with data link control, regardless of format or content. Transparent transmissions are unrecognized by data link controls. (2) In data transmission, pertaining to information not recognized by the receiving program or device as transmission control characters. (3) Pertaining to operations or data that are of no significance to the user.

trap. An unprogrammed, hardware-initiated, conditional jump to a specific address.It occurs as a result of an error or certain other conditions. A recording is made of the location from which the jump occurred.

trap handler. A user-defined trap routine used when an exception occurs. See also *exception*.

tree stucture. A hierarchical calling sequence, that consists of both a root segment and also one or more levels of the segments called via the root segment.

tree-structured directories. A method for connecting directories such that each directory is listed in another directory except for the root directory, which is at the top of the tree.

tributary station. (1) On a multipoint connection or a point-to-point connection using basic mode link control, any data station other than the control station. (2) A secondary device on a multipoint line.

Trivial File Transfer Protocol. Transfers files between hosts using minimal protocol.

true color adapters. In GSL, color adapters in which the pixel color value drives the digital-to-analog converters (DACs) without the level of indirection forced by a the VLT. Contrast to *VLT-based adapters*.

truncate. (1) to terminate a computational process in accordance with some rule. (2) To remove the beginning or ending elements of a string. (3) to drop data tht cannot be printed or displayed in the line width specified or available. (4) To shorten a field or statement to a specified length.

tty. In the AIX, any device that uses the **termio** standard terminal device interface. *tty* devices typically perform input and output on a character-by-character basis.

turnaround. Changing a communications line from transmit mode to receive mode or from receive mode to transmit mode.

turnaround time. (1) The time interval required to reverse the direction of transmission from send to receive, or vice versa, over a communication line. (2) Elapsed time between submission of a job and return of the complete output.

two-digit display. Two seven-segment light-emitting diodes (LEDs) on the operator panel used to track the progress of power-on self-tests (POSTs).

two's complement. Representation of negative binary numbers. Formed by subtracting each digit of the number from zero, then adding one to the result.

type style. the form of characters of a given size, style, and design within the set of the same font.

typematic key. A key that repeats its function multiple times when held down.

U - Z

UID. See *user number*.

umask. A command that set the file creation permission code mask.

unformatted file. A file displayed with the data not arranged with particular characters. Contrast with *formatted file*.

Unix-to-Unix Copy Program (UUCP). See *UUCP*.

update. (1) An improvement for some part of the system. (2) To add, change, or delete items. (3) To modify a master file with current information according to a specified procedure.

update file. A file that adds or revises information in a program product already resident on the VRM minidisk. An update file documents the version, release, and level of updates to be installed and is required for program product update diskettes that use VRM. Install update facilities. This file is not used for programs updated from the AIX Operating System.

upload. To transfer data from one computer to another. Typically, users upload from a small computer to a larger one.

user. (1) The name associated with an account. (2) Anyone requiring the services of a computing system.

user account. See *account*.

user area. The parts of main storage and disk available to the user.

user data segment. In kernel mode, the data structure that contains user data, consisting of initialized data variables.

User Datagram Protocol (UDP). Allows a datagram made of packet switched communications in the environment of an interconnected set of computer networks. It assumes that IP is the underlying protocol.

user identification (user ID). (1) A unique string of characters identifying an operator to the system. This string of characters limits the functions and information the operator is allowed to use. The user's ID can often be substituted in commands that take a user's login name as an argument. See *user number*. (2) A parameter that specifies the user ID under which the application or transaction program runs.

user interface. Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device.

user limit. *ulimit*, the maximum file size AIX will handle. Can be changed on a system wide or per user basis.

user mode. Contrast with *kernel mode*.

user name. A name that uniquely identifies a user to the system.

user number (Uid). (1) A unique number identifying an operator to the system. This string of characters limits the functions and information the operator is allowed to use. The Uid can often be substituted in commands that take a user's name as an an argument.

user profile. A file containing a description of user characteristics and defaults (for example, printer assignment, formats, group ID) to be conveyed to the system while the user is signed on.

user-defined variable. A shell variable to which the user assigns a character string as a value.

utility. (1) A service; in programming, a program that performs a common service function. (2) The capability of a system, program, or device to perform the functions for which it is designed.

UUCP (Unix-to-Unix Copy Program). (1) A group of programs and files within the Extended Services facility of AIX that functions as an AIX background process. It includes a set of directories, files, programs, and commands that allow the user to communicate with a remote system over a dedicated line or a telephone line. (2) The command (**uucp**) that starts file copying from one or more sources to a single destination.

WAN. *See wide area network*

wide area network. A network that provides data communication capability in geographic areas larger than those serviced by local area networks.

Index

GG24-3381-0 **READER'S COMMENT FORM**

AIX Communications Handbook

You may use this form to communicate your comments about this publication, its organization, or subject matter with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment include: Clarity, Accuracy, Completeness, Organization, Coding, Retrieval, Legibility.

If you would like a reply, please give your name, company, mailing address and date:

____

____

____

____

What is your occupation? ______________________________

Most recent Newsletter associated with this publication: ______________________________

Thank you for your cooperation.

Cut or Fold Along Line

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Technical Support Center
Dept 948/Bldg 983
11400 Burnet Road
Austin
Texas 78758
U.S.A.

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

GG24-3381-0 **READER'S COMMENT FORM**

AIX Communications Handbook

You may use this form to communicate your comments about this publication, its organization, or subject matter with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment include: Clarity, Accuracy, Completeness, Organization, Coding, Retrieval, Legibility.

If you would like a reply, please give your name, company, mailing address and date:

____

____

____

____

What is your occupation? ____________________

Most recent Newsletter associated with this publication: ____________________

Thank you for your cooperation.

Reader's Comment Form

Cut or Fold Along Line

Fold and tape **Please Do Not Staple** Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Technical Support Center
Dept 948/Bldg 983
11400 Burnet Road
Austin
Texas 78758
U.S.A.

Fold and tape **Please Do Not Staple** Fold and tape

IBM®

READER'S COMMENT FORM

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Technical Support Center
Dept 948/Bldg 983
11400 Burnet Road
Austin
Texas 78758
U.S.A.

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

GG24-3381-0 **READER'S COMMENT FORM**

AIX Communications Handbook

You may use this form to communicate your comments about this publication, its organization, or subject matter with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment include: Clarity, Accuracy, Completeness, Organization, Coding, Retrieval, Legibility.

If you would like a reply, please give your name, company, mailing address and date:

____

____

____

____

What is your occupation? ______________________________

Most recent Newsletter associated with this publication: ______________________________

Thank you for your cooperation.

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Technical Support Center
Dept 948/Bldg 983
11400 Burnet Road
Austin
Texas 78758
U.S.A.

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

GG24-3381-00

AIX Communications Handbook GG24-3381-00 PRINTED IN THE U.S.A.

IBM®

GG24-3381-00