

AIX 5L Version 5.1



System Management Guide: Communications and Networks

AIX 5L Version 5.1



System Management Guide: Communications and Networks

Fifth Edition (April 2001)

Before using the information in this book, read the general information in "Appendix. Notices" on page 451.

This edition applies to AIX 5L Version 5.1 and to all subsequent releases of this product until otherwise indicated in new editions.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to Publications Department, Internal Zip 9561, 11400 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial Internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

(c) Copyright AT&T, 1984, 1985, 1986, 1987, 1988, 1989. All rights reserved.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California. We acknowledge the following institutions for their role in its development: the Electrical Engineering and Computer Sciences Department at the Berkeley Campus.

The Rand MH Message Handling System was developed by the Rand Corporation and the University of California. Portions of the code and documentation described in this book were derived from code and documentation developed under the auspices of the Regents of the University of California and have been acquired and modified under the provisions that the following copyright notice and permission notice appear:

Copyright Regents of the University of California, 1986, 1987, 1988, 1989. All rights reserved.

Redistribution and use in source and binary forms are permitted provided that this notice is preserved and that due credit is given to the University of California at Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. This software is provided "as is" without express or implied warranty.

© Copyright International Business Machines Corporation 1997, 2001. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About This Book	xiii
Who Should Use This Book	xiii
Highlighting	xiii
ISO 9000	xiii
Related Publications	xiii
Trademarks	xiii
Chapter 1. Communications and Networks Overview	1
Communications Functions Introduction	1
Network Introduction	1
Physical Networks Introduction	3
System Communications Support	3
Protocols	4
Addresses	4
Domains	4
Gateways and Bridges	4
Routing	4
Local and Remote Nodes	5
Client and Server	5
Communicating with Other Operating Systems	5
Chapter 2. Mail	7
Mail Management Tasks	7
Configuring the /etc/rc.tcpip File to Start the sendmail Daemon	8
Managing Mail Aliases	8
/etc/mail/aliases File	8
Creating Local System Aliases for Mail	9
Building the Alias Database	9
Managing the Mail Queue Files and Directories	10
Printing the Mail Queue	10
Mail Queue Files	10
Specifying Time Values in sendmail	12
Forcing the Mail Queue	12
Setting the Queue Processing Interval	13
Moving the Mail Queue	13
Starting the sendmail Daemon	13
Stopping the sendmail Daemon	14
Managing Mail Logging	14
Managing the Log	15
Logging Traffic	15
Logging Mailer Statistics	15
Displaying Mailer Information	16
Debugging sendmail	16
Internet Message Access Protocol (IMAP) and Post Office Protocol (POP)	17
Configuring IMAP and POP Servers	18
syslog Facility	19
Mail Reference	19
List of Mail Commands	19
List of Mail Files and Directories	20
List of Internet Message Access Protocol and Post Office Protocol Commands	21
Chapter 3. Transmission Control Protocol/Internet Protocol	23
Planning Your TCP/IP Network	23

Installation and Configuration for TCP/IP	24
Configuring TCP/IP	24
TCP/IP System Manager Commands	25
Configuring a TCP/IP Network Checklist	25
TCP/IP Protocols	26
Internet Protocol (IP) Version 6 Overview	29
Packet Tracing	35
Network Interface Packet Headers	35
Internet Network-Level Protocols	37
Internet Transport-Level Protocols	42
Internet Application-Level Protocols	46
Assigned Numbers	50
TCP/IP Local Area Network Adapter Cards	50
Installing a Network Adapter	51
Configuring and Managing Adapters	51
Configuring and Using Virtual Local Area Networks (VLANs)	52
Using ATM Adapters	54
TCP/IP Network Interfaces	63
Automatic Configuration of Network Interfaces	64
Implications of Multiple Network Interfaces on the Same Network	67
Managing Network Interfaces	67
Interface-Specific Network Options	68
TCP/IP Addressing	70
Internet Addresses	70
Subnet Addresses	73
Broadcast Addresses	76
Local Loopback Addresses	76
TCP/IP Address and Parameter Assignment - Dynamic Host Configuration Protocol (DHCP)	77
The DHCP Server	78
Planning DHCP	80
Configuring DHCP	81
DHCP and the Dynamic Domain Name System (DDNS)	86
DHCP Compatibility with Older Versions	88
DHCP Server File Known Options	88
Preboot Execution Environment (PXE) Vendor Container Suboption	91
DHCP Server File Syntax for General Server Operation	92
DHCP Server File Syntax for db_file Database	95
DHCP and Network Installation Management (NIM) Suggestions	110
Preboot Execution Environment Proxy DHCP Daemon (pxed)	111
The PXE Proxy DHCP Server	111
Configuring the PXED Server	112
PXE Vendor Container Suboptions	115
PXED Server File Syntax for General Server Operation	116
PXED Server File Syntax for db_file Database	118
Boot Image Negotiation Layer Daemon (BINLD)	127
The BINLD Server	127
Configuring BINLD	127
BINLD Server File Syntax for General Server Operation	131
BINLD Server File Syntax for db_file Database	133
Configuring TCP/IP	140
Prerequisites	140
Updating the Hosts List	141
TCP/IP Daemons	141
Subsystems and Subservers	141
System Resource Control (SRC)	142
Configuring the inetd Daemon	143

Client Network Services	144
Server Network Services	145
TCP/IP Name Resolution	146
Naming.	146
Performing Local Name Resolution (/etc/hosts)	152
Planning for DOMAIN Name Resolution.	153
Configuring Name Servers.	154
Configuring a Forwarder	163
Configuring a Forward Only Name Server	164
Configuring a Host to Use a Name Server	166
Configuring Dynamic Zones on the DNS Name Server	168
Planning and Configuration for LDAP Name Resolution	169
TCP/IP Routing.	171
Static and Dynamic Routing	171
Gateways	172
Planning for Gateways	174
Configuring a Gateway	175
Restricting Route Use	176
Dead Gateway Detection	176
Manually Removing Dynamic Routes.	177
Configuring the routed Daemon.	177
Configuring the gated Daemon	178
Getting an Autonomous System Number	180
Path MTU Discovery	180
SLIP.	181
Configuring SLIP over a Modem	181
Configuring SLIP over a Null Modem Cable	183
Deactivating a SLIP Connection.	185
Removing a TTY	185
Asynchronous Point-to-Point Protocol (PPP) Subsystem.	185
User-Level Processes	185
Configuring the Asynchronous Point-to-Point Protocol.	186
PPP and SNMP	188
TCP/IP Quality of Service (QoS)	189
QoS Models	190
Supported Standards and Draft Standards	190
QoS Installation	191
QoS Configuration.	191
QoS Problem Determination	193
Policy Specification	194
Guidelines for DiffServ Environments.	195
Sample policyd Configuration File	195
Loading Policies into IBM SecureWay Directory Server	197
System Configuration	198
Standards Compliance	198
IPv6 Support.	199
Controlling the Policy Daemon	199
QoS Reference.	199
TCP/IP Security	199
Operating System-Specific Security	200
TCP/IP-Specific Security	201
TCP/IP Command Security	201
Trusted Processes	204
Network Trusted Computing Base (NTCB)	204
Data Security and Information Protection	206
TCP/IP Problem Determination	206

Communication Problems	207
Name Resolution Problems	207
Routing Problems	208
Problems with SRC Support	209
telnet or rlogin Problems	210
Configuration Problems	212
Common Problems with Network Interfaces	212
Problems with Packet Delivery	215
Problems with Dynamic Host Configuration Protocol (DHCP)	215
TCP/IP Reference	216
List of TCP/IP Commands	216
List of TCP/IP Daemons	216
List of Methods	217
List of TCP/IP Files	217
List of RFCs	217
Chapter 4. Internet Protocol (IP) Security	219
IP Security Overview	219
Benefits of a Virtual Private Network (VPN)	219
IP Security and the Operating System	220
IP Security Features	220
Security Associations	221
Tunnels and Key Management	222
Native Filtering Capability	223
Digital Certificate Support	224
IP Security Installation	224
Loading IP Security	224
Planning IP Security Configuration	224
Tunnels versus Filters	225
Tunnels and Security Associations	226
Choosing a Tunnel Type	226
Using IKE with DHCP or Dynamically Assigned Addresses	227
Configuring IKE Tunnels	227
Basic Configuration Wizard	227
Advanced IKE Tunnel Configuration	228
Examples of IKE Tunnel Configurations	234
Digital Certificate Configuration	234
Using the IBM Key Manager Tool	237
Creating a Key Database	238
Adding a CA Root Digital Certificate	239
Establishing Trust Settings.	239
Deleting a CA Root Digital Certificate.	240
Requesting a Digital Certificate	240
Adding (Receiving) a New Digital Certificate	241
Deleting a Digital Certificate	241
Changing a Database Password	242
Creating IKE Tunnels using Digital Certificates	242
Configuring Manual Tunnels	244
Setting Up Tunnels and Filters	244
Creating a Manual Tunnel on Host A	244
Creating a Manual Tunnel on Host B	246
Setting Up Filters	246
Static Filter Rules and Examples	247
Autogenerated Filter Rules and User Specified Filter Rules	249
Predefined Filter Rules	250
Subnet Masks	250

Host-Firewall-Host	251
Logging Facilities	251
Labels in Field Entries	254
IP Security Problem Determination	255
Troubleshooting Manual Tunnel Errors	255
Troubleshooting IKE Tunnel Errors	256
Tracing Facilities	262
ipsecstat	262
IP Security Reference	263
List of Commands	263
List of Methods	264
Chapter 5. TTY Devices and Serial Communications	265
TTY Overview	265
TERM Values for Different Displays and Terminals	265
Setting TTY Characteristics	266
Setting Attributes on the Attached TTY Device	266
Managing TTY Devices	266
Dynamic Screen Utility	268
dscreen Terminal Configuration Information File	268
Key Action Assignments	268
Dynamic Screen Assignment	269
dsinfo File	270
Modems	273
Modem Overview	273
Generic Modem Setup	275
Adding a TTY for the Modem	276
Configuring the Modem	276
Hayes and Hayes-Compatible Modems	278
Troubleshooting Modem Problems	279
Software Services Modem Questionnaire	279
AT Command Summary	280
ATE Overview	283
Setting Up ATE Overview	283
Customizing ATE	283
Setting Up ATE	285
Prerequisites	285
Procedure	285
TTY Troubleshooting	286
Respawning Too Rapidly Errors	286
Error Log Information and TTY Log Identifiers	287
Chapter 6. Micro Channel, ISA, and PCI Adapters	291
Micro Channel Wide Area Network (WAN) Adapters	291
Supported Multiport/2 Adapters	291
Supported Portmaster Adapters	291
Device Driver Support	291
Configuring Multiport/2 and Portmaster Adapters	291
ISA/PCI Wide Area Network (WAN) Adapters	292
Multiport Model 2 Overview	292
Configuring the Multiport Model 2 Adapter	294
Multiport Model 2 Adapter Object Information and Attributes	295
Multiport Model 2 Power Management	296
2-Port Multiprotocol HDLC Network Device Driver Overview	297
Configuring the 2-Port Multiprotocol Adapter	297
ARTIC960HX PCI Adapter Overview	297

Configuring the MPQP COMIO Emulation Driver over the ARTIC960HX PCI Adapter	298
Chapter 7. Data Link Control	299
Generic Data Link Control Environment Overview	299
Meeting the GDLC Criteria	301
Implementing the GDLC Interface	301
Installing GDLC Data Link Controls	302
GDLC Interface ioctl Entry Point Operations	302
Service Access Point.	303
Link Station	303
Local-Busy Mode	303
Short-Hold Mode	304
Testing and Tracing a Link.	304
Statistics	304
GDLC Special Kernel Services	304
Managing DLC Device Drivers	305
Chapter 8. Basic Networking Utilities	307
BNU Overview	307
How BNU Works	308
BNU File and Directory Structure	308
BNU Security	310
BNU Daemons	312
Configuring BNU	314
Prerequisites.	314
Information to Collect before Configuring BNU	314
Procedure.	315
Setting Up Automatic Monitoring of BNU	317
Setting Up BNU Polling of Remote Systems	318
Using the /etc/uucp/Systems File	318
Editing Devices Files for Hardwired Connections	319
Editing Devices File for Autodialer Connection	319
Editing Devices File for TCP/IP	320
Maintaining BNU	320
Working with BNU Log Files	320
BNU Maintenance Commands	322
Monitoring a BNU Remote Connection	323
Monitoring a BNU File Transfer	324
Debugging BNU Problems.	324
Debugging BNU Login Failures Using the uucico Daemon	327
Contacting Connected UNIX Systems Using the tip Command	328
BNU Configuration Files	330
BNU Configuration for a TCP/IP Connection Example	331
BNU Configuration for a Telephone Connection Example	333
BNU Configuration for a Direct Connection Example	334
BNU Files, Commands, and Directories Reference.	336
BNU Directories	336
BNU Files.	336
BNU Commands	337
BNU Daemons	337
Chapter 9. Network Management	339
SNMP for Network Management	339
SNMP Access Policies	339
SNMP Daemon.	340
Configuring the SNMP Daemon.	340

SNMP Daemon Processing	341
Message Processing and Authentication	341
Request Processing	341
Response Processing	342
Trap Processing	343
SNMP Daemon Support for the EGP Family of MIB Variables.	345
Examples	355
SNMP Daemon RFC Conformance	357
SNMP Daemon Implementation Restrictions	358
SNMP Daemon Logging Facility	358
Logging Directed from the snmpd Command Line	359
Logging Directed from the Configuration File	359
Logging by the syslogd Daemon	360
Problem Determination for the SNMP Daemon	361
Daemon Termination Problem	361
Daemon Failure Problem	361
MIB Variable Access Problem	362
MIB Variable Access in Community Entry Problem	362
No Response from Agent Problem.	363
noSuchName Problem	363
Chapter 10. Network File System	365
Network File System Overview	365
NFS Services	365
NFS Access Control Lists (ACL) Support	366
Cache File System (CacheFS) Support	367
NFS Mapped File Support.	368
Three Types of Mounts	368
NFS Mounting Process	369
/etc/exports File	369
/etc/xtab File	370
Implementation of NFS	370
Controlling NFS	371
NFS Installation and Configuration.	373
Checklist for Configuring NFS	373
Configuring an NFS Server	374
Configuring an NFS Client.	374
Exporting an NFS File System	374
Unexporting an NFS File System	375
Changing an Exported File System	375
Enabling Root User Access to an Exported File System	376
Mounting an NFS File System Explicitly	376
Using AutoFS to Automatically Mount a File System	377
Establishing Predefined NFS Mounts	378
Unmounting an Explicitly or Automatically Mounted File System	381
Removing Predefined NFS Mounts	381
PC-NFS	381
PC-NFS Authentication Service	382
PC-NFS Print-Spooling Service	382
Configuring the rpc.pcnfsd Daemon	382
Starting the rpc.pcnfsd Daemon.	382
Verifying the rpc.pcnfsd Daemon Is Accessible	383
WebNFS	383
Network Lock Manager	384
Network Lock Manager Architecture	384
Network File Locking Process	384

Crash Recovery Process	384
Starting the Network Lock Manager	385
Troubleshooting the Network Lock Manager	385
Secure NFS	387
Secrecy	387
Secrecy in NFS.	389
Naming Network Entities for DES Authentication	391
/etc/publickey File	391
Booting Considerations of Public Key Systems	391
Performance Considerations	392
Administering Secure NFS Checklist	392
Configuring Secure NFS	393
Exporting a File System Using Secure NFS	393
Mounting a File System Using Secure NFS	394
NFS Problem Determination	395
Identifying Hard-Mounted and Soft-Mounted File Problems	395
Identifying NFS Problems Checklist	395
Asynchronous Write Errors	396
NFS Error Messages.	396
Identifying the Cause of Slow Access Times for NFS	398
NFS Reference.	402
List of Network File System (NFS) Files.	402
List of NFS Commands	402
List of NFS Daemons	402
NFS Subroutines	403
Chapter 11. AIX Fast Connect	405
AIX Fast Connect Overview	406
Features	406
Requirements	406
Packaging and Installation.	407
Limitations	409
Windows Networking Concepts (NetBIOS, SMB, WINS)	410
AIX Fast Connect Configuration and Administration	413
Overview	413
Configurable Parameters	414
Configuration of File and Print Shares (Exports)	414
User Administration	415
Basic Server Administration	417
NetBIOS Name Service (NBNS)	419
Configuring Client PCs for use with AIX Fast Connect	419
TCP/IP Configuration.	419
NetBIOS Name Resolution	421
Workgroups, Domains, and User Accounts.	422
Enabling Windows Clients for Plain Text Passwords	423
Browsing the Network	424
Mapping Drives.	425
Using AIX Fast Connect Printers	425
Support for Windows 2000 Clients.	426
Support for Windows Terminal Server	426
Advanced AIX Fast Connect Features	426
AIX-based User Authentication (Plain Text Passwords)	427
CIFS Password Encryption Protocols.	427
NT Passthrough Authentication	428
Network Logon to AIX Fast Connect	429
DCE/DFS Support.	429

Guest Logon	430
Share-Level Security	430
User Name Mappings	431
AIX Fast Connect User Management and File Access	432
Mapping Long AIX File Names to 8.3 DOS File Names	434
Support for DOS File Attributes	435
Specifying NetBIOS Aliases for HACMP support.	435
Performance Considerations	436
AIX Fast Connect Problem Determination	437
Traces	437
Logs	438
Troubleshooting Connection Problems	438
Configuring Network Logon for AIX Fast Connect	440
Configuration Options	441
Enabling the Network Logon Feature	441
Setting Up Startup Scripts	441
Setting Up Home Directories (Profile Directories)	442
Windows Configuration Policy Files	442
Configuring Win 95/98 Clients for Network Logon	442
Configuring Network Logon for NT clients from Remote Subnets.	442
Configuring LanServer (OS/2) Clients for Network Logon	443
AIX Fast Connect NetLogon Limitations	443
AIX Fast Connect Configurable Parameters for the net Command	443
Migrating to AIX Fast Connect from AIX Connections	448
Saving ACONN Configuration Data Before ACONN Uninstall	448
Appendix. Notices	451
Index	453

About This Book

This book is for system administrators who maintain the operating system's network connections. Familiarity with the Base Operating System and the material covered in *AIX 5L Version 5.1 System Management Guide: Operating System and Devices* and *AIX 5L Version 5.1 System User's Guide: Communications and Networks* is necessary.

Who Should Use This Book

This book is intended for system administrators who perform system management tasks that involve communication within a network.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, keywords, files, directories, and other items whose names are predefined by the system.
<i>Italics</i>	Identifies parameters whose actual names or values are to be supplied by the user.
Monospace	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Related Publications

The following book contains information about or related to communications:

- *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*
- *AIX 5L Version 5.1 System User's Guide: Communications and Networks*
- *AIX 5L Version 5.1 General Programming Concepts: Writing and Debugging Programs*
- *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*
- *AIX 5L Version 5.1 Commands Reference*
- *AIX Version 4.3 Installation Guide*

Trademarks

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

- AIXwindows
- IBM
- Portmaster
- SecureWay

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

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

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

Chapter 1. Communications and Networks Overview

This chapter presents the conceptual foundation for understanding computer networking in general. System administrators unfamiliar with general networking principles need to read this chapter. Those familiar with UNIX networking can safely skip this chapter.

A network is the combination of two or more computers and their connecting links. A *physical* network is the hardware (equipment such as adapter cards, cables, concentrators, and telephone lines) that makes up the network. The software and the conceptual model make up the *logical* network.

This overview provides the following information on networks:

- Communications Functions Introduction
- Network Introduction
- Physical Networks Introduction
- System Communications Support
- Communicating with Other Operating Systems.

Communications Functions Introduction

Networks allow for several user and application communication functions. They enable a user to:

- Send electronic mail (e-mail)
- Emulate another terminal or log in to another computer
- Transfer data
- Run programs that reside on a remote node.

One of the most popular applications for computer networks is e-mail which allows a user to send a message to another user. The two users may be on the same system (in which case a communications network is not needed), different systems in different buildings, or even in different countries.

Through a communications network, one computer can mimic another and access information as if it were a different type of computer or terminal. Remote login capabilities allow users to log in to a remote system and access the same programs and files as if they were using the machine locally.

Networks also allow for the transfer of data from one system to another. Files, directories, and entire file systems can be migrated from one machine to another across a network, enabling remote backup of data, as well as assuring redundancy in case of machine failure.

Several different protocols have been devised to allow users and applications on one system to invoke procedures and applications on other systems, which is useful when distributing the burden for computer-intensive routines.

Network Introduction

The complexity of modern computer networks has given rise to several conceptual models for explaining how networks work. One of the most common of these models is the International Standards Organization's Open Systems Interconnection (OSI) Reference Model, also referred to as the OSI seven-layer model. The seven layers of the OSI model are numbered beginning at the lowest layer (Physical).

- 7 **Application**
- 6 **Presentation**

- 7 **Application**
- 5 **Session**
- 4 **Transport**
- 3 **Network**
- 2 **Data Link**
- 1 **Physical**

Levels 1 through 3 are network specific, and differ depending on what physical network you are using. Levels 4 through 7 comprise network-independent, higher-level functions. Each layer describes a particular function (instead of a specific protocol) that occurs in data communications. The seven layers function from lowest level (machine level) to highest level (the level at which most human interaction takes place), as follows:

Physical	Describes the physical media of the network. For example, the fiber optic cable required for a Fiber Distributed Data Interface (FDDI) network is part of the physical layer.
Data Link	Provides reliable delivery of data across the physical layer (which is usually inherently unreliable).
Network	Manages the connections to other machines on the network.
Transport	Assures error-free data transmission.
Session	Manages the connections between applications.
Presentation	Ensures that data is presented to the applications in a consistent fashion.
Application	Comprises the applications that use the network.

Note that while the OSI Reference Model is useful for discussing networking concepts, many networking protocols do not closely follow the OSI model. For example, when discussing Transmission Control Protocol/Internet Protocol (TCP/IP), the Application and Presentation Layer functions are combined, as are the Session and Transport Layers and the Data Link and Physical Layers.

Each layer in the OSI model communicates with the corresponding layer on the remote machine as shown in the OSI Reference Model figure.

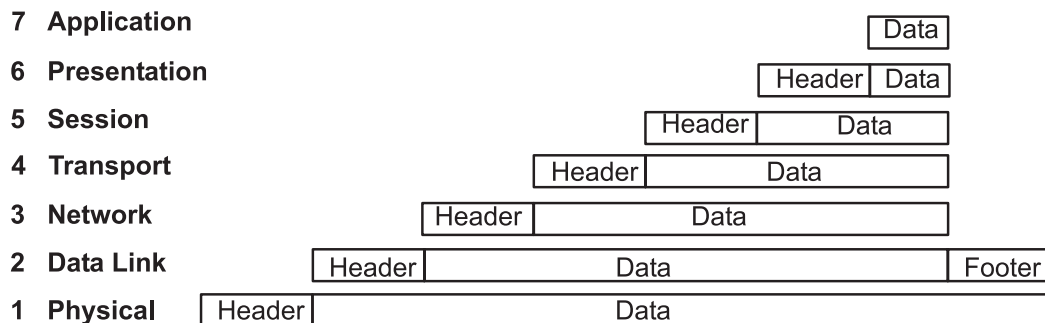


Figure 1. OSI Reference Model. This illustration shows the various communication levels of the OSI Model as described in the above text.

The layers pass data only to the layers immediately above and below. Each layer adds its own header information (and footer information, in the case of the Data Link), effectively encapsulating the information received from the higher layers.

Individual users as well as organizations use networks for many reasons, including:

- Data entry
- Data queries
- Remote batch entry

- Resource sharing
- Data sharing
- Electronic mail.

Data entry consists of entering data directly into either local or remote data files. Increased accuracy and efficiency are natural by-products of a one-step data transfer. Data queries entail searching data files for specified information. Data updating involves altering, adding, or deleting data stored in local or remote files. Remote batch entry consists of entering batches of data from a remote location, an activity often performed at night or during periods of low system usage. Because of such diverse capabilities, communications and networks are not only desirable but necessary.

Sharing resources is another function of networks. Users can share data as well as programs, file-storage space, and peripheral devices like printers, modems, terminals, and fixed disks. Sharing of system resources is cost effective because it eliminates the problems of keeping multiple copies of programs and it keeps data consistent (in the case of program and file sharing).

Physical Networks Introduction

The physical network consists of the cables (coaxial cable, twisted pair, fiber optic, and telephone lines) that connect the different hardware residing on the network, the adapter cards used on the attached hosts, and any concentrators, repeaters, routers or bridges used in the network. (A *host* is a computer attached to the network.)

Physical networks vary both in size and in the type of hardware used. The two common kinds of networks are *local area networks* (LANs) and *wide area networks* (WANs). A LAN is a network where communications are limited to a moderately sized geographic area of 1 to 10 km (1 to 6 miles), such as a single office building, warehouse, or campus. A WAN is a network providing data communications capability throughout geographic areas larger than those serviced by LANs; for example, across a country or across continents. An intermediate class of networks exists also, called *metropolitan area networks* (MANs). This guide does not generally distinguish MANs; they are grouped with WANs.

LANs commonly use Standard Ethernet, IEEE 802.3 Ethernet, or token-ring hardware for the physical network, while WANs and asynchronous networks use communications networks provided by common carrier companies. Operation of the physical network in both cases is usually controlled by networking standards from organizations such as the Electronics Industry Association (EIA) or the International Telecommunication Union (ITU).

System Communications Support

All network communications involve the use of hardware and software. *Hardware* consists of the physical equipment connected to the physical network. *Software* consists of the programs, and device drivers pertaining to the operation of a particular system.

The system hardware consists of adapter cards or other devices that provide a path or interface between the system software and the physical network. An adapter card requires an input/output (I/O) card slot in the system. Other devices, such as modems, can be attached to one of the standard ports on the computer.

Adapter cards support the standards required by the physical network (for example, EIA 232D, Smartmodem, V.25 bis, EIA 422A, X.21, or V.35) and may, at the same time, support software *protocols*, for example, synchronous data link control (SDLC), high-level data link control (HDLC), and bisynchronous protocols. If the adapter does not contain software support, then this support must be provided by the adapter device driver.

Protocols

All communications software use *protocols*, sets of semantical and syntactical rules that determine the behavior of functional units in achieving communication. Protocols define how information is delivered, how it is enclosed to reach its destination safely, and what path it follows. Protocols also coordinate the flow of messages and their acknowledgments.

Protocols exist at different levels within the kernel and cannot be manipulated directly. However, they are manipulated indirectly by what the user chooses to do at the application programming interface (API) level. The choices a user makes when invoking file transfer, remote login, or terminal emulation programs define the protocols used in the execution of those programs.

Addresses

Addresses are associated with both software and hardware. The address is the means by which the sending or control station selects the station to which it sends data. Addresses identify receiving or storage locations. A physical address is a unique code assigned to each device or workstation connected to a network.

For example, on a token-ring network, the **netstat -iv** command displays the token-ring card address. This is the physical network address. The **netstat -iv** command also displays class-level and user-level address information. Addresses are often defined by software but can be created by the user as well.

Domains

An aspect of addresses common to many communications networks is the concept of *domains*. For example, the structure of the Internet illustrates how domains define the Internet Protocol (IP) address. The Internet is an extensive network made up of many different smaller networks. To facilitate routing and addressing, Internet addresses are hierarchically structured in domains, with very broad categories at the top such as `com` for commercial users, `edu` for educational users, and `gov` for government users.

Within the `com` domain are many smaller domains corresponding to individual businesses; for example, `ibm`. Within the `ibm.com` domain are even smaller domains corresponding to the Internet addresses for various locations, such as `austin.ibm.com` or `raleigh.ibm.com`. At this level, we start seeing names of *hosts*. A host, in this context, is any computer connected to the network. Within `austin.ibm.com`, there may be hosts with the names `hamlet` and `lear`, which are addressed `hamlet.austin.ibm.com` and `lear.austin.ibm.com`.

Gateways and Bridges

A wide variety of networks reside on the Internet, often using different hardware and running different software. *Gateways* and *bridges* enable these different networks to communicate with each other. A bridge is a functional unit that connects two LANs that possibly use the same logical link control (LLC) procedure, such as Ethernet, but different medium access control (MAC) procedures. A gateway has a broader range than a bridge. It operates above the link layer and, when required, translates the interface and protocol used by one network into those used by another distinct network. Gateways allow data transfers across the various networks that constitute the Internet.

Routing

Using domain names for addressing and gateways for translation greatly facilitates the *routing* of the data being transferred. Routing is the assignment of a path by which a message reaches its destination. The domain name effectively defines the message destination. In a large network like the Internet, information is routed from one communications network to the next until that information reaches its destination. Each

communications network checks the domain name and, based on the domains with which that network is familiar, routes the information on to the next logical stop. In this way, each communications network that receives the data contributes to the routing process.

Local and Remote Nodes

A physical network is used by the hosts that reside on that network. Each host is a *node* on the network. A node is an addressable location in a communications network that provides host-processing services. The intercommunication of these various nodes are defined as *local* or *remote*. *Local* pertains to a device, file, or system accessed directly from your system, without the use of a communications line. *Remote* pertains to a device, file, or system accessed by your system over a communications line. Local files reside on your system, while remote files reside on a file server or at another node with which you communicate using a physical network, for example, Ethernet, token-ring, or phone lines.

Client and Server

Related to the concepts of local and remote are those of *client* and *server*. A server is a computer that contains data or provides facilities to be accessed by other computers on the network. Common server types are file servers, which store files; name servers, which store names and addresses; and application servers, which store programs and applications; print servers, which schedule and direct print jobs to their destination.

A client is a computer requesting services or data from a server. A client, for example, could request updated program code or the use of applications from a code server. To obtain a name or address, a client contacts a name server. A client could also request files and data for data entry, inquiry, or record updating from a file server.

Communicating with Other Operating Systems

Different types of computers can be connected on a network. The computers can be from different manufacturers or be different models from the same manufacturer. Communication programs bridge the differences in operating systems of two or more types of computers.

Sometimes these programs require that another program has previously been installed on the network. Other programs may require that such communications connectivity protocols as TCP/IP or Systems Network Architecture (SNA) exist on the network.

For example, with AIX 4.3.2 and later, AIX Fast Connect lets PC clients access operating system files and printers using native PC networking client software. PC users can use remote operating system file systems directly from their machines as if they were locally stored. They can print jobs on printers using the operating system spooler, view available printers, and map a printer as a network printer.

Chapter 2. Mail

The mail facility provides a method for exchanging electronic mail (e-mail) with users on the same system or on multiple systems connected by a network. This section documents the mail system, the standard mail user interface, the Internet Message Access Protocol (IMAP), and the Post Office Protocol (POP).

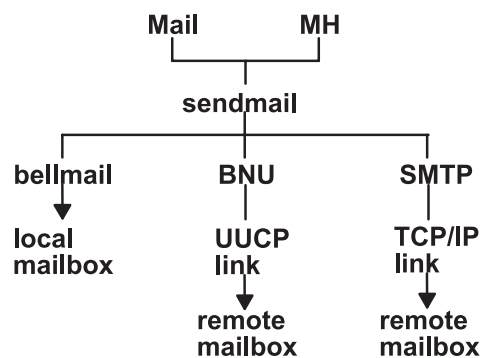
The mail system is an internetwork mail delivery facility that consists of a user interface, a message routing program, and a message delivery program (or mailer). The mail system relays messages from one user to another on the same host, between hosts, and across network boundaries. It also performs a limited amount of message-header editing to put the message into a format that is appropriate for the receiving host.

A mail *user interface* enables users to create and send messages to, and receive messages from, other users. The mail system provides two user interfaces, **mail** and **mhmail**. The **mail** command is the standard mail user interface available on all UNIX systems. The **mhmail** command is the Message Handler (MH) user interface, an enhanced mail user interface designed for experienced users.

A *message routing program* routes messages to their destinations. The mail system message routing program is the **sendmail** program, which is part of the Base Operating System (BOS) and is installed with BOS. The **sendmail** program is a daemon that uses information in the **/etc/mail/sendmail.cf** file, the **/etc/mail/aliases** file to perform the necessary routing.

Note: In versions earlier than AIX 5.1, the **sendmail.cf** and **aliases** files are located in **/etc/sendmail.cf** and **/etc/aliases**, respectively.

Depending on the type of route to the destination, the **sendmail** command uses different *mailers* to deliver messages.



As the figure illustrates:

- To deliver local mail, the **sendmail** program routes messages to the **bellmail** program. The **bellmail** program delivers all local mail by appending messages to the user's system mailbox, which is in the **/var/spool/mail** directory.
- To deliver mail over a UNIX-to-UNIX Copy Program (UUCP) link, the **sendmail** program routes messages using Basic Network Utilities (BNU).
- To deliver mail routed through Transmission Control Protocol/Internet Protocol (TCP/IP), the **sendmail** command establishes a TCP/IP connection to the remote system then uses Simple Mail Transfer Protocol (SMTP) to transfer the message to the remote system.

Mail Management Tasks

The following is a list of the tasks for which you, the mail manager, are responsible.

1. Configure the **/etc/rc.tcpip** file so that the **sendmail** daemon will be started at system boot time. See the instructions immediately following this list.
2. Customize the configuration file **/etc/mail/sendmail.cf**. The default **/etc/mail/sendmail.cf** file is configured so that both local mail and TCP/IP mail can be delivered. In order to deliver mail through BNU, you must customize the **/etc/mail/sendmail.cf** file. See the **sendmail.cf File in AIX 5L Version 5.1 Files Reference** for more information.
3. Define system-wide and domain-wide mail aliases in the **/etc/mail/aliases** file. See **Managing Mail Aliases** for more information.
4. Manage the Mail Queue. See **Managing the Mail Queue Files and Directories** for more information.
5. Manage the Mail Log. See **Managing Mail Logging** for more information.

Configuring the /etc/rc.tcpip File to Start the sendmail Daemon

To configure the **/etc/rc.tcpip** file so that the **sendmail** daemon will be started at system boot time:

1. Edit the **/etc/rc.tcpip** file with your favorite text editor.
2. Find the line that begins with **start /usr/lib/sendmail**. By default, this line should be uncommented, that is, there is no # (pound sign) at the beginning of the line. However, if it is commented, delete the pound sign.
3. Save the file.

With this change, the system will start the **sendmail** daemon at boot time.

Managing Mail Aliases

Aliases map names to address lists using personal, system-wide, and domain-wide alias files. You can define three types of aliases:

personal	Defined by individual users in the user's \$HOME/.mailrc file.
local system	Defined by the mail system administrator in the /etc/mail/aliases file. These aliases apply to mail handled by the sendmail program on the local system. Local system aliases rarely need to be changed.
domainwide	By default, sendmail reads /etc/alias to resolve aliases. To override the default and use NIS, edit or create /etc/netsvc.conf and add the line: <code>aliases=nis</code>

/etc/mail/aliases File

Note: In versions earlier than AIX 5.1, the **aliases** file is located in **/etc/aliases**.

The **/etc/mail/aliases** file consists of a series of entries in the following format:

Alias: Name1, Name2, ... NameX

where *Alias* can be any alphanumeric string that you choose (not including special characters, such as @ or !). *Name1* through *NameX* is a series of one or more recipient names. The list of names can span one or more lines. Each continued line begins with a space or a tab. Blank lines and lines beginning with a # (pound sign) are comment lines.

The **/etc/mail/aliases** file must contain the following three aliases:

MAILER-DAEMON	The ID of the user who is to receive messages addressed to the mailer daemon. This name is initially assigned to the root user: <code>MAILER-DAEMON: root</code>
----------------------	---

postmaster The ID of the user responsible for the operation of the local mail system. The **postmaster** alias defines a single mailbox address that is valid at each system in a network. This address enables users to send inquiries to the **postmaster** alias at any system, without knowing the correct address of any user at that system. This name is initially assigned to the root user:

```
postmaster: root
```

nobody The ID that is to receive messages directed to programs such as **news** and **msgs**. This name is initially assigned to **/dev/null**:

```
nobody: /dev/null
```

To receive these messages, define this alias to be a valid user.

Whenever you change this file, you must recompile it into a database format that the **sendmail** command can use. See Building the Alias Database.

Creating Local System Aliases for Mail

To create or modify local system aliases:

1. Edit the **/etc/mail/aliases** file using your favorite editor.
2. On a blank line, add an alias, followed by a colon (:), followed by a list of comma-separated recipients. For example, the following entry defines the **writers** alias to be the names of people in that group:

```
writers: geo, mark@zeus, ctw@athena, brian
```

This definition could also be contained on several lines, as long as each added line begins with a space or a tab, for example:

```
writers: geo,  
        mark@zeus,  
        ctw@athena,  
        brian
```

3. Create an owner for any distribution list aliases. If the **sendmail** command has trouble sending mail to the distribution list, it sends an error message to the owner of that list. For example, the following set of entries in the **/etc/mail/aliases** file defines a distribution list named **editors**, whose owner is **glenda@hera**:

```
editors: glenda@hera, davidm@kronos, perryw@athena  
owner-editors: glenda@hera
```
4. Recompile the **/etc/mail/aliases** file by following the instructions in the section, "Building the Alias Database".

Building the Alias Database

The **sendmail** command does not use directly the alias definitions in the local system **/etc/mail/aliases** file. Instead, the **sendmail** command reads a processed database manager (dbm) version of the **/etc/mail/aliases** file. You can compile the alias database using one of the following methods:

- Run the **/usr/sbin/sendmail** command using the **-bi** flag.
- Run the **newaliases** command. This command causes the **sendmail** command to read the local system **/etc/mail/aliases** file and create a new file containing the alias database information. This file is in the more efficient Berkeley format:

/etc/mail/aliases.db

(Versions earlier than AIX 5.1 created two database files, **/etc/aliases.dir** and **/etc/aliases.pag**.)

- Run the **sendmail** command using the **Rebuild Aliases** flag. This rebuilds the alias database automatically when it is out-of-date. Auto-rebuild can be dangerous on heavily loaded machines with large alias files. If it might take more than the rebuild time-out (normally five minutes) to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

Notes:

1. If these files do not exist, the **sendmail** command cannot process mail and will generate an error message.
2. If you have multiple alias databases specified, the **-bi** flag rebuilds all the database types it understands (for example, it can rebuild Network Database Management (NDBM) databases but not NIS databases).

The **/etc/netsvc.conf** file contains the ordering of system services. To specify the service ordering of aliases, add the following line:

```
aliases=service, service
```

where *service* can be either *files* or *nis*. For example:

```
aliases=files, nis
```

tells the **sendmail** command to try the local alias file first; and if that fails, try *nis*. If *nis* is defined as a service, it should be running.

For further information on the **/etc/netsvc.conf** file, see *AIX 5L Version 5.1 Files Reference*.

Managing the Mail Queue Files and Directories

The mail queue is a directory that stores data and controls files for mail messages that the **sendmail** command delivers. By default, the mail queue is **/var/spool/mqueue**.

Mail messages might be queued for many reasons.

For example:

1. The **sendmail** command can be configured to process the queue at certain intervals, rather than immediately. If this is so, mail messages must be stored temporarily.
2. If a remote host does not answer a request for a mail connection, the mail system queues the message and tries again later.

Printing the Mail Queue

The contents of the queue can be printed using the **mailq** command (or by specifying the **-bp** flag with the **sendmail** command).

These commands produce a listing of the queue IDs, the sizes of the messages, the dates the messages entered the queue, and the senders and recipients.

Mail Queue Files

Each message in the queue has a number of files associated with it. The files are named according to the following conventions:

TypefID

where *ID* is a unique message queue ID, and *Type* is one of the following letters indicating the type of file:

- d** The data file containing the message body without the heading information.
- q** The queue-control file. This file contains the information necessary to process the job.
- t** A temporary file. This file is an image of the **q** file when it is being rebuilt. It is quickly renamed to the **q** file.
- x** A transcript file that exists during the life of a session and shows everything that happens during that session.

For example, if a message has a queue ID of AA00269, the following files are created and deleted in the mail queue directory while the **sendmail** command tries to deliver the message:

dfAA00269	Data file
qfAA00269	Control file
tfAA00269	Temporary file
xfAA00269	Transcript file

q Control File

The **q** control file contains a series of lines, each beginning with a code letter:

- B** Specifies the body type. The remainder of the line is a text string defining the body type. If this entire field is missing, the body type is 7-bit by default, and no special processing is attempted. Legal values are **7BIT** and **8BITMIME**.
- C** Contains the controlling user. For recipient addresses that are a file or a program, **sendmail** performs delivery as the owner of the file or program. The controlling user is set to the owner of the file or program. Recipient addresses that are read from a **.forward** or **:include:** file will also have the controlling user set to the owner of the file. When **sendmail** delivers mail to these recipients, it delivers as the controlling user, then converts back to root.
- F** Contains envelope flags. The flags are any combination of **w**, which sets the **EF_WARNING** flag; **r**, which sets the **EF_RESPONSE** flag; **8**, which sets the **EF_HAS8BIT** flag; and **b**, which sets the **EF_DELETE_BCC** flag. Other letters are silently ignored.
- H** Contains a heading definition. There can be any number of these lines. The order in which the **H** lines appear determines their order in the final message. These lines use the same syntax as heading definitions in the **/etc/mail/sendmail.cf** configuration file. (For versions earlier than AIX 5.1, this file is **/etc/sendmail.cf**.)
- I** Specifies the inode and device information for the **df** file; this can be used to recover your mail queue after a disk crash.
- K** Specifies the time (as seconds) of the last delivery attempt.
- M** When a message is put into the queue because an error occurred during a delivery attempt, the nature of the error is stored in the **M** line.
- N** Specifies the total number of delivery attempts.
- O** Specifies the original message transfer system (MTS) value from the ESMTP. It is used for Delivery Status Notifications only.
- P** Contains the priority of the current message. The priority is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.
- Q** Contains the original recipient as specified by the **ORCPT=** field in an ESMTP transaction. Used exclusively for Delivery Status Notifications. It applies only to the immediately following **R** line.
- R** Contains a recipient address. There is one line for each recipient.
- S** Contains the sender address. There is only one of these lines.
- T** Contains the message creation time used to compute when to time out the message.
- V** Specifies the version number of the queue file format used to allow new **sendmail** binaries to read queue files created by older versions. Defaults to version **zero**. Must be the first line of the file, if present.
- Z** Specifies the original envelope ID (from the ESMTP transaction). Used for Delivery Status Notifications only.
- \$** Contains a macro definition. The values of certain macros (**\$r** and **\$s**) are passed through to the queue run phase.

The **q** file for a message sent to amy@zeus would look similar to:

```
P217031
T566755281
MDeferred: Connection timed out during user open with zeus
Sgeo
Ramy@zeus
H?P?return-path: <geo>
Hreceived: by george (0.13 (NL support)/0.01)
           id AA00269; Thu, 17 Dec 87 10:01:21 CST
H?D?date: Thu, 17 Dec 87 10:01:21 CST
```

H?F?From: geo
Hmessage-id: <8712171601.AA00269@george>
HTo: amy@zeus
Hsubject: test

Where:

P217031	Priority of the message
T566755281	Submission time in seconds
MDeferred: Connection timed out during user open with zeus	Status message
Sgeo	ID of the sender
Ramy@zeus	ID of the receiver
H lines	Header information for the message

Specifying Time Values in sendmail

To set the message time-out and queue processing interval, you must use a specific format for the time value. The format of a time value is:

`-qNumberUnit`

where *Number* is an integer value and *Unit* is the unit letter. *Unit* can have one of the following values:

s	Seconds
m	Minutes
h	Hours
d	Days
w	Weeks

If *Unit* is not specified, the **sendmail** daemon uses minutes (**m**) as the default. Here are three examples illustrating time-value specification:

```
/usr/sbin/sendmail -q15d
```

This command tells the **sendmail** daemon to process the queue every 15 days.

```
/usr/sbin/sendmail -q15h
```

This command tells the **sendmail** daemon to process the queue every 15 hours.

```
/usr/sbin/sendmail -q15
```

This command tells the **sendmail** daemon to process the queue every 15 minutes.

Forcing the Mail Queue

In some cases, you might find that the queue is clogged for some reason. You can force a queue to run using the **-q** flag (with no value). You can also use the **-v** flag (verbose) to watch what happens:

```
/usr/sbin/sendmail -q -v
```

You can also limit the jobs to those with a particular queue identifier, sender, or recipient using one of the queue modifiers. For example, **-qRsally** restricts the queue run to jobs that have the string **sally** in one of the recipient addresses. Similarly, **-qSstring** limits the run to particular senders, and **-qIstring** limits it to particular queue identifiers.

Setting the Queue Processing Interval

The value of the **-q** flag when the daemon starts determines the interval at which the **sendmail** daemon processes the mail queue.

The **sendmail** daemon is usually started by the **/etc/rc.tcpip** file, at system startup. The **/etc/rc.tcpip** file contains a variable called the queue processing interval (QPI), which it uses to specify the value of the **-q** flag when it starts the **sendmail** daemon. By default, the value of **qpi** is 30 minutes. To specify a different queue processing interval:

1. Edit the **/etc/rc.tcpip** file with your favorite editor.
2. Find the line that assigns a value to the **qpi** variable, such as:

```
qpi=30m
```
3. Change the value assigned to the **qpi** variable to the time value you prefer.

These changes will take effect at the next system restart. If you want the changes to take effect immediately, stop and restart the **sendmail** daemon, specifying the new **-q** flag value. See Stopping the sendmail Daemon and Starting the sendmail Daemon for more information.

Moving the Mail Queue

When a host goes down for an extended period, many messages routed to (or through) that host might be stored in your mail queue. As a result, the **sendmail** command spends a long time sorting the queue, severely degrading your system performance. If you move the queue to a temporary place and create a new queue, the old queue can be run later when the host returns to service. To move the queue to a temporary place and create a new queue:

1. Stop the **sendmail** daemon by following the instructions in Stopping the sendmail Daemon.
2. Move the entire queue directory by entering:

```
cd /var/spool  
mv mqueue omqueue
```
3. Restart the **sendmail** daemon by following the instructions in Starting the sendmail Daemon.
4. Process the old mail queue by entering:

```
/usr/sbin/sendmail -oQ/var/spool/omqueue -q
```

The **-oQ** flag specifies an alternate queue directory. The **-q** flag specifies to run every job in the queue. To get a report about the progress of the operation, use the **-v** flag.

Note: This operation can take a long time.

5. Remove the log files and the temporary directory when the queue is empty by entering:

```
rm /var/spool/omqueue/*  
rmdir /var/spool/omqueue
```

Starting the sendmail Daemon

To start the **sendmail** daemon, enter either of the following commands:

```
startsrc -s sendmail -a "-bd -q15"  
/usr/lib/sendmail -bd -q15
```

If the **sendmail** daemon is already active when you enter one of these commands, you see the following message on the screen:

```
The sendmail subsystem is already active. Multiple instances are not supported.
```

If the **sendmail** daemon is not already active, then you see a message indicating that the **sendmail** daemon has been started.

Stopping the sendmail Daemon

To stop the **sendmail** daemon, run the **stopsrc -s sendmail** command.

If the **sendmail** daemon was not started with the **startsrc** command:

- Find the **sendmail** process ID.
- Enter the **kill sendmail_pid** command (where *sendmail_pid* is the process ID of the **sendmail** process).

Managing Mail Logging

The **sendmail** command logs mail system activity through the **syslogd** daemon. The **syslogd** daemon must be configured and running for logging to occur. Specifically, the **/etc/syslog.conf** file should contain the uncommented line:

```
mail.debug          /var/spool/mqueue/log
```

If it does not, use your favorite editor to make this change; be certain that the path name is correct. If you change the **/etc/syslog.conf** file while the **syslogd** daemon is running, refresh the **syslogd** daemon by typing the following command at a command line:

```
refresh -s syslogd
```

If the **/var/spool/mqueue/log** file does not exist, you must create it by typing the following command:

```
touch /var/spool/mqueue/log
```

Messages in the log file appear in the following format:

Each line in the system log consists of a time stamp, the name of the machine that generated it (for logging from several machines over the local area network), the word **sendmail:**, and a message. Most messages are a sequence of *name=value* pairs.

The two most common lines logged when a message is processed are the **receipt** line and the **delivery attempt** line. The **receipt** line logs the receipt of a message; there will be one of these per message. Some fields may be omitted. These message fields are:

from	Specifies the envelope sender address.
size	Specifies the size of the message in bytes.
class	Indicates the class (numeric precedence) of the message.
pri	Specifies the initial message priority (used for queue sorting).
nrcpts	Indicates the number of envelope recipients for this message (after aliasing and forwarding).
proto	Specifies the protocol used to receive the message, for example ESMTP or UNIX-to-UNIX Copy Program (UUCP).
relay	Specifies the machine from which it was received.

The **delivery attempt** line is logged each time there is delivery attempt (so there can be several per message if delivery is deferred or there are multiple recipients). These fields are:

to	Contains a comma-separated list of the recipients to this mailer.
ctladdr	Specifies the <i>controlling user</i> , that is, the name of the user whose credentials are used for delivery.
delay	Specifies the total delay between the time this message was received and the time it was delivered.
xdelay	Specifies the amount of time needed in this delivery attempt.
mailer	Specifies the name of the mailer used to deliver to this recipient.
relay	Specifies the name of the host that actually accepted (or rejected) this recipient.
stat	Specifies the delivery status.

Because such a large amount of information can be logged, the log file is arranged as a succession of levels. Beginning at level 1, the lowest level, only very unusual situations are logged. At the highest level, even the insignificant events are logged. As a convention, log levels ten and under the most useful information. Log levels above 64 are reserved for debugging purposes. Levels from 11-64 are reserved for verbose information.

The types of activities that the **sendmail** command puts into the log file are specified by the **L** option in the **/etc/mail/sendmail.cf** file. (For versions earlier than AIX 5.1, this file is **/etc/sendmail.cf**.)

Managing the Log

Because information is continually appended to the end of the log, the file can become very large. Also, error conditions can cause unexpected entries to the mail queue. To keep the mail queue and the log file from growing too large, run the **/usr/lib/smdemon.cleau** shell script. This script forces the **sendmail** command to process the queue and maintains four progressively older copies of log files, named **log.0**, **log.1**, **log.2**, and **log.3**. Each time the script runs it moves:

- **log.2** to **log.3**
- **log.1** to **log.2**
- **log.0** to **log.1**
- **log** to **log.0**

Running this script allows logging to start over with a new file. Run this script either manually or at a specified interval with the **cron** daemon.

Logging Traffic

Many Simple Mail Transfer Protocols (SMTPs) implementations do not fully implement the protocol. For example, some personal computer-based SMTPs do not understand continuation lines in reply codes. These can be very hard to trace. If you suspect such a problem, you can set traffic logging by using the **-X** flag. For example:

```
/usr/sbin/sendmail -X /tmp/traffic -bd
```

This command logs all traffic in the **/tmp/traffic** file.

Because this command logs a lot of data very quickly, it should never be used during normal operations. After running the command, force the **errant** implementation to send a message to your host. All message traffic in and out of **sendmail**, including the incoming SMTP traffic, will be logged in this file.

Using **sendmail**, you can log a dump of the open files and the connection cache by sending it a **SIGUSR1** signal. The results are logged at **LOG_DEBUG** priority.

Logging Mailer Statistics

The **sendmail** command tracks the volume of mail being handled by each of the mailer programs that interface with it. Those mailers are defined in the **/etc/mail/sendmail.cf** file. (For versions earlier than AIX 5.1, this file is **/etc/sendmail.cf**.)

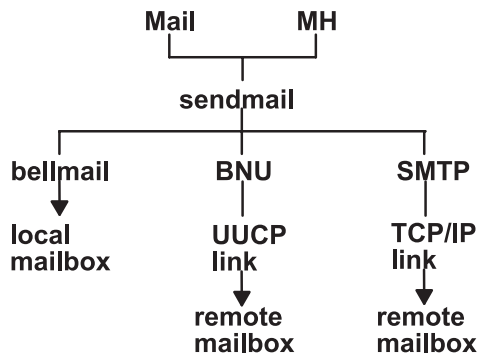


Figure 2. Mailers Used by the Sendmail Command. This illustration is a type of top-down organizational chart with Mail and MH at the top. Branching from them are bellmail, BNU and SMTP. Underneath the previous level are local mailbox, UUCP link, and TCP/IP link respectively. Beneath UUCP link is remote mailbox and under TCP/IP link is remote mailbox.

To start the accumulation of mailer statistics, create the **/etc/mail/statistics** file by typing the following:

```
touch /etc/mail/statistics
```

If the **sendmail** command encounters errors when trying to record statistics information, the command writes a message through the **syslog** subroutine. These errors do not affect other operations of the **sendmail** command.

The **sendmail** command updates the information in the file each time it processes mail. The size of the file does not grow, but the numbers in the file do. They represent the mail volume since the time you created or reset the **/etc/mail/statistics** file.

Note: In versions earlier than AIX 5.1, statistics were kept in the **/var/tmp/sendmail.st** file.

Displaying Mailer Information

The statistics kept in the **/etc/mail/statistics** file are in a database format that cannot be read as a text file. To display the mailer statistics, type the following at a command prompt:

```
/usr/sbin/mailstats
```

This reads the information in the **/etc/mail/statistics** file, formats it, and writes it to standard output. For information on the output of the **/usr/sbin/mailstats** command, read its description in the *AIX 5L Version 5.1 Commands Reference*.

Debugging sendmail

There are a large number of debug flags built into the **sendmail** command. Each debug flag has a number and level, where higher levels print more information. The convention is levels greater than nine print out so much information that they are used only for debugging a particular piece of code. Debug flags are set using the **-d** flag as shown in the following example:

```
debug-flag:      -d debug-list
debug-list:      debug-flag[.debug-flag]*
debug-flag:      debug-range[.debug-level]
debug-range:     integer|integer-integer
debug-level:     integer

-d12             Set flag 12 to level 1
-d12.3          Set flag 12 to level 3
-d3-17          Set flags 3 through 17 to level 1
-d3-17.4        Set flags 3 through 17 to level 4
```


The available debug flags are:

-d0	General debugging.
-d1	Show send information.
-d2	End with <i>finis</i> ().
-d3	Print the load average.
-d4	Enough disk space.
-d5	Show events.
-d6	Show failed mail.
-d7	The queue file name.
-d8	DNS name resolution.
-d9	Trace RFC1413 queries.
-d9.1	Make host name canonical.
-d10	Show recipient delivery.
-d11	Trace delivery.
-d12	Show mapping of relative host.
-d13	Show delivery.
-d14	Show header field commas.
-d15	Show network get request activity.
-d16	Outgoing connections.
-d17	List MX hosts.

Note: There are now almost 200 defined debug flags in **sendmail**.

Internet Message Access Protocol (IMAP) and Post Office Protocol (POP)

AIX provides two Internet-based mail protocol server implementations for accessing mail remotely:

- Post Office Protocol (POP)
- Internet Message Access Protocol (IMAP)

Both the POP and IMAP servers store and provide access to electronic messages. Using these mail access protocols on a server eliminates the requirement that, to receive mail, a computer must always be up and running.

The POP server provides an off-line mail system, whereby a client, using POP client software, can remotely access a mail server to retrieve mail messages. The client can either download the mail messages and immediately delete the messages from the server, or download the messages and leave the messages resident on the POP server. After the mail is downloaded to the client machine, all mail processing is local to the client machine. The POP server allows access to a user mailbox one client at a time.

The IMAP server provides a superset of POP functionality but has a different interface. The IMAP server provides an off-line service, as well as an on-line service and a disconnected service. The IMAP protocol is designed to permit manipulation of remote mailboxes as if they were local. For example, clients can perform searches and mark messages with status flags such as "deleted" or "answered." In addition, messages can remain in the server's database until explicitly removed. The IMAP server also allows simultaneous interactive access to user mailboxes by multiple clients.

Both the IMAP and POP servers are used for mail access only. These servers rely on the Simple Mail Transfer Protocol (SMTP) for sending mail.

Both IMAP and POP are open protocols, based on standards described in RFCs. The IMAP server is based on RFC 1730, and the POP server is based on RFC 1725. Both are connection-oriented using TCP sockets. The IMAP server listens on port 143, and the POP server listens on port 110. Both servers are handled by the **inetd** daemon.

Configuring IMAP and POP Servers

Prerequisites

You must have root authority.

Procedure

1. Uncomment the **imapd** and **pop3d** entries in the **/etc/inetd.conf** file.
2. Refresh the **inetd** daemon by running the following command:

```
refresh -s inetd
```

Running Configuration Tests

Run a few tests to verify your **imapd** and **pop3d** servers are ready for operation.

First, verify the servers are listening on their ports. To do this, type the following commands at a command prompt, pressing the **Enter** key after each command:

```
netstat -a | grep imap
netstat -a | grep pop
```

The following is the output from the **netstat** commands:

```
tcp    0      0  *.imap2      *.*          LISTEN
tcp    0      0  *.pop3       *.*          LISTEN
```

If you do not receive this output, recheck the entries in the **/etc/inetd.conf** file and rerun the **refresh -s inetd** command.

To test the configuration of the **imapd** server, telnet into the **imap2**, port 143. When you connect via telnet, you will get the **imapd** prompt. You can then enter the IMAP Version 4 commands as defined in RFC 1730. To run one of the commands, type a period (**.**), followed by a space, and then the command name. For example:

```
. CommandName
```

Note that passwords are echoed when you telnet into the **imapd** server.

In the following telnet example, you must provide your own password where *id_password* is indicated in the **login** command.

```
telnet e-xbelize 143
Trying...
Connected to e-xbelize.austin.ibm.com.
Escape character is '^]'.
* OK e-xbelize.austin.ibm.com IMAP4 server ready
. login id id_password
. OK
. examine /usr/spool/mail/root
* FLAGS (\Answered \Flagged \Draft \Deleted \Seen)
* OK [PERMANENTFLAGS (\Answered \Flagged \Draft \Deleted \Seen \*)]
* 0 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 823888143]
. OK [READ-ONLY] Examine completed
. logout
* BYE Server terminating connection
. OK Logout completed
Connection closed.
```

To test the configuration of the **pop3d** server, telnet into the Post Office Protocol Version 3 (POP3) port, 110. When you telnet in, you should get the pop3d prompt. You can enter the POP commands that are defined in RFC 1725. To run one of the commands, type a period (.), followed by a space, and then the command name. For example:

```
. CommandName
```

Note that passwords are echoed when you telnet into the **pop3d** server.

In the following telnet example, you must provide your own password where *id_password* is indicated in the **pass** command.

```
telnet e-xbelize 110
Trying...
Connected to e-xbelize.austin.ibm.com.
Escape character is '^]'.
+OK e-xbelize.austin.ibm.com POP3 server ready
user id
+OK Name is a valid mailbox
pass id_password
+OK Maildrop locked and ready
list
+OK scan listing follows
.
stat
+OK 0 0
quit
+OK
Connection closed.
```

syslog Facility

The IMAP and POP server software sends log messages to the **syslog** facility.

To configure your system for IMAP and POP logging through the **syslog** facility, you must be the root user. Edit the **/etc/syslog.conf** file, and add an entry for ***.debug** as follows:

```
*.debug /usr/adm/imapd.log
```

The **usr/adm/imapd.log** file must exist before the **syslogd** daemon re-reads the **/etc/syslog.conf** configuration file. To create this file, type the following at a command line prompt and press **Enter**:

```
touch /usr/adm/imapd.log
```

Refresh the **syslogd** daemon to re-read its configuration file. Type the following at a command line prompt and press **Enter**:

```
refresh -s syslogd
```

Mail Reference

This section provides a quick reference to the various Mail commands, files, and directories.

List of Mail Commands

The following is a list of mail management commands.

bugfiler	Stores bug reports in specific mail directories.
comsat	Notifies users of incoming mail (daemon).
mailq	Prints the contents of the mail queue.
mailstats	Displays statistics about mail traffic.
newaliases	Builds a new copy of the alias database from the /etc/mail/aliases file.

rmail	Handles remote mail received through the uucp command of the Basic Networking Utilities (BNU).
sendbug	Mails a system bug report to a specific address.
sendmail	Routes mail for local or network delivery.
smdemon.cleanu	Cleans up the sendmail queue for periodic housekeeping.

List of Mail Files and Directories

This list of files and directories is arranged by function.

Note: In versions earlier than AIX 5.1, the **sendmail.cf** and **aliases** files are located in **/etc/sendmail.cf** and **/etc/aliases**, respectively.

Using the Mail Program

/usr/share/lib/Mail.rc	Sets local system defaults for all users of the mail program. A text file you can modify to set the default characteristics of the mail command.
\$HOME/.mailrc	Enables the user to change the local system defaults for the mail facility.
\$HOME/mbox	Stores processed mail for the individual user.
/usr/bin/Mail, /usr/bin/mail, or /usr/bin/mailx	Specifies three names linked to the same program. The mail program is <i>one</i> of the user interfaces to the mail system.
/var/spool/mail	Specifies the default mail drop directory. By default, all mail is delivered to the /var/spool/mail/UserName file.
/usr/bin/bellmail	Performs local mail delivery.
/usr/bin/rmail	Performs remote mail interface for BNU.
/var/spool/mqueue	Contains the log file and temporary files associated with the messages in the mail queue.

Using the sendmail Command

/usr/sbin/sendmail	The sendmail command.
/usr/ucb/mailq	Links to the /usr/sbin/sendmail . Using mailq is equivalent to using the /usr/sbin/sendmail -bp command.
/usr/ucb/newaliases	Links to the /usr/sbin/sendmail file. Using newaliases is equivalent to using the /usr/sbin/sendmail -bi command.
/etc/netsvc.conf	Specifies the ordering of certain name resolution services.
/usr/sbin/mailstats	Formats and prints the sendmail statistics as found in the /etc/sendmail.st file if it exists. The /etc/sendmail.st file is the default, but you can specify an alternative file.
/etc/mail/aliases	Describes a text version of the aliases file for the sendmail command. You can edit this file to create, modify, or delete aliases for your system.
/etc/aliasesDB	Describes a directory containing the aliases database files, DB.dir and DB.pag , that are created from the /etc/mail/aliases file when you run the sendmail -bi command.
/etc/mail/sendmail.cf	Contains the sendmail configuration information in text form. Edit the file to change this information.
/usr/lib/smdemon.cleanu	Specifies a shell file that runs the mail queue and maintains the sendmail log files in the /var/spool/mqueue directory.
/etc/mail/statistics	Collects statistics about mail traffic. This file does not grow. Use the /usr/sbin/mailstats command to display the contents of this file. Delete this file if you do not want to collect this information.
/var/spool/mqueue	Describes a directory containing the temporary files associated with each message in the queue. The directory can contain the log file.

/var/spool/cron/crontabs

Describes a directory containing files that the **cron** daemon reads to determine which jobs to start. The **root** file contains a line to start the **smdemon.cleanu** shell script.

List of Internet Message Access Protocol and Post Office Protocol Commands

/usr/sbin/imapd

The Internet Message Access Protocol (IMAP) server process.

/usr/sbin/pop3d

The Post Office Protocol Version 3 (POP3) server process.

Chapter 3. Transmission Control Protocol/Internet Protocol

This chapter describes the Transmission Control Protocol/Internet Protocol (TCP/IP) suite of networking software. TCP/IP is a powerful and flexible industry-standard way of connecting multiple computers to other machines.

The topics discussed in this chapter are:

- Planning Your TCP/IP Network
- Installation and Configuration for TCP/IP
- TCP/IP Protocols
- TCP/IP Local Area Network Adapter Cards
- TCP/IP Network Interfaces
- TCP/IP Addressing
- TCP/IP Address and Parameter Assignment - Dynamic Host Configuration Protocol (DHCP)
- Preboot Execution Environment Proxy DHCP Daemon (pxed)
- Boot Image Negotiation Layer Daemon (BINLD)
- Configuring TCP/IP
- TCP/IP Daemons
- TCP/IP Name Resolution
- TCP/IP Routing
- Path MTU Discovery
- Serial Line Interface Protocol (SLIP)
- Asynchronous Point-to-Point Protocol (PPP) Subsystem
- TCP/IP Quality of Service (QoS)
- TCP/IP Security
- TCP/IP Problem Determination
- TCP/IP Reference

Note: Most of the tasks discussed in this chapter require root authority.

Planning Your TCP/IP Network

Because TCP/IP is such a flexible networking tool, you can customize it to fit the specific needs of your organization. The following are the major issues you need to consider when planning your network. The details of these issues are discussed at length later. This list is intended only to introduce you to the issues.

1. Decide which type of network hardware you want to use: token-ring, Ethernet Version 2, IEEE 802.3, Fiber Distributed Data Interface (FDDI), Serial Optical Channel (SOC), or Serial Line Interface Protocol (SLIP).
2. Plan the physical layout of the network.
Consider which functions each host machine will serve. For example, you must decide which machine or machines will serve as gateways before you cable the network.
3. Decide whether a *flat* network or a *hierarchical* network organization best fits your needs.
If your network is fairly small, at a single site, and consists of one physical network, then a flat network probably suits your needs. If your network is very large or complex with multiple sites or multiple physical networks, a hierarchical network might be a more efficient network organization for you.
4. If your network is to be connected to other networks, you must plan how your gateways should be set up and configured. Things to consider are:

- a. Decide which machine or machines will serve as gateways.
 - b. Decide whether you need to use static or dynamic routing, or a combination of the two. If you choose dynamic routing, decide which routing daemons each gateway will use in light of the types of communications protocols you need to support.
5. Decide on an addressing scheme.
If your network will not be part of a larger internetwork, choose the addressing scheme that best fits your needs. If you want your network to be connected to a larger internetwork such as the Internet, you will have to obtain an official set of addresses from your internet service provider (ISP).
 6. Decide whether your system needs to be divided into subnets. If so, decide how you will assign subnet masks.
 7. Decide on a naming scheme. Each machine on the network needs its own unique host name.
 8. Decide whether your network needs a name server for name resolution or if using the `/etc/hosts` file will be sufficient.
If you choose to use name servers, consider the type of name servers you need and how many you need to serve your network efficiently.
 9. Decide the types of services you want your network to provide to remote users; for example, mail services, print services, file sharing, remote login, remote command execution, and others.

Installation and Configuration for TCP/IP

For information on installing Transmission Control Protocol/Internet Protocol (TCP/IP), see the *AIX 5L Version 5.1 Installation Guide*.

Configuring TCP/IP

After the TCP/IP software is installed on your system, you are ready to begin configuring your system.

Many TCP/IP configuration tasks can be performed in more than one way, either by:

- Using the Web-based System Manager Network application (fast path **wsm network**)
- Using the System Management Interface Tool (SMIT)
- Editing a file format
- Issuing a command at the shell prompt.

For example, the **rc.net** shell script performs required minimum host configuration for TCP/IP during the system startup process (the **rc.net** script is run by the configuration manager program during the second boot phase). By using Web-based System Manager or SMIT to perform the host configuration, the **rc.net** file is configured automatically.

Alternatively, you can configure the `/etc/rc.bsdnet` file using a standard text editor. With this method, you can specify the traditional UNIX TCP/IP configuration commands such as **ifconfig**, **hostname**, and **route**. See List of TCP/IP Commands for further information. If using the file edit method, you must enter **smit configtcp** fast path and then select **BSD Style rc Configuration**.

A few tasks, such as configuring a name server, cannot be done using Web-based System Manager or SMIT.

Configuring Hosts

Each host machine on your network must be configured to function according to the needs of the end users and the network as a whole. For each host on the network, you must configure the network interface, set the Internet address, and set the host name. You also must set up static routes to gateways

or other hosts, specify daemons to be started by default, and set up the **/etc/hosts** file for name resolution (or set up the host to use a name server for name resolution).

Configuring Hosts as Servers

If the host machine will have a specific function like serve as a gateway, file server, or name server, you must perform the necessary configuration tasks after the basic configuration is complete.

For example, if your network is organized hierarchically and you want to use the **Domain Name** protocol to resolve names into Internet addresses, you will need to configure at least one name server to provide this function for your network.

Remember, a server host does not have to be a dedicated machine, it can be used for other things as well. If the name server function for your network is fairly small, the machine might also be used as a workstation or as a file server for your network.

Note: If your system has either NIS or NIS+ installed, these services can also provide name resolution. For more information, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

Configuring Gateways

If your network is going to communicate with other networks, you will need to configure at least one gateway host. You must consider which communications protocols you want to support, and then use whichever routing daemon (the **routed** or **gated** daemon) that supports those protocols.

TCP/IP System Manager Commands

The following list contains the commands used to configure and manage a TCP/IP network:

arp	Displays or changes the Internet address to hardware address translation tables used by the Address Resolution protocol.
finger	Returns information about users on a specified host.
host	Shows the Internet address of a specified host or the host name of a specified Internet address.
hostname	Shows or sets the Internet name and address of the local host.
ifconfig	Configures network interfaces and their characteristics.
netstat	Shows local and foreign addresses, routing tables, hardware statistics, and a summary of packets transferred.
no	Sets or shows current network kernel options.
ping	Determines whether a host is reachable.
route	Permits you to manipulate the routing tables manually.
ruptime	Shows status information on hosts that are connected to local physical networks and are running the rwhod server.
rwho	Shows status information for users on hosts that are connected to local physical networks and running the rwhod server.
setclock	Reads the network time service and sets the time and date of the local host accordingly.
timedc	Returns information about the timed daemon.
trpt	Reports protocol tracing on TCP sockets.
whois	Provides the Internet name directory service.

Configuring a TCP/IP Network Checklist

Use the following procedure as a guide for configuring your network. Ensure that you have read and understood the appropriate material.

After you bring your network up and it is running properly, you might find it useful to refer to this checklist for the purpose of debugging.

Prerequisites

1. Network hardware is installed and cabled. See TCP/IP Network Adapter Cards
2. TCP/IP software is installed. See the *AIX 5L Version 5.1 Installation Guide*.

Procedure

1. Read TCP/IP Protocols for the basic organization of TCP/IP. You should understand:
 - the layered nature of TCP/IP (that is, different protocols reside at different layers)
 - how data flows through the layers
2. Minimally configure each host machine on the network. This means adding a network adapter, assigning an IP address, and assigning a hostname to each host, as well as defining a default route to your network. Read TCP/IP Network Interfaces, TCP/IP Addressing, and Choosing Names for Hosts on Your Network. Follow the instructions in Configuring TCP/IP.

Note: Each machine on the network needs this basic configuration whether it will be an end-user host, a file server, a gateway, or a name server.

3. Configure and start the **inetd** daemon on each host machine on the network. Read TCP/IP Daemons and then follow the instructions in Configuring the inetd Daemon.
4. Configure each host machine to perform either local name resolution or to use a name server. If you are setting up a hierarchical Domain Name network, configure at least one host to function as a name server. Read and follow the instructions in TCP/IP Name Resolution.
5. If your network will communicate with any remote networks, configure at least one host to function as a gateway. The gateway can use static routes or a routing daemon to perform internetwork routing. Read and follow the instructions in TCP/IP Routing.
6. Decide which services each host machine on the network will use. By default, all services are available. Follow the instructions in Client Network Services if you wish to make a particular service unavailable.
7. Decide which hosts on the network will be servers, and which services a particular server will provide. Follow the instructions in Server Network Services to start the server daemons you wish to run.
8. Configure any remote print servers you will need. See Printer Overview in *AIX 5L Version 5.1 Guide to Printers and Printing* for more information.
9. If desired, configure a host to use or to serve as the master time server for the network. See the **timed** daemon in the *AIX 5L Version 5.1 Commands Reference* for more information.

TCP/IP Protocols

The topics discussed in this section are:

- IP6 Overview
- Packet Tracing
- Network Interface Packet Headers
- Internet Network-Level Protocols
- Internet Transport-Level Protocols
- Internet Application-Level Protocols
- Assigned Numbers.

Protocols are sets of rules for message formats and procedures that allow machines and application programs to exchange information. These rules must be followed by each machine involved in the communication in order for the receiving host to be able to understand the message.

The **TCP/IP** suite of protocols can be understood in terms of layers (or levels).

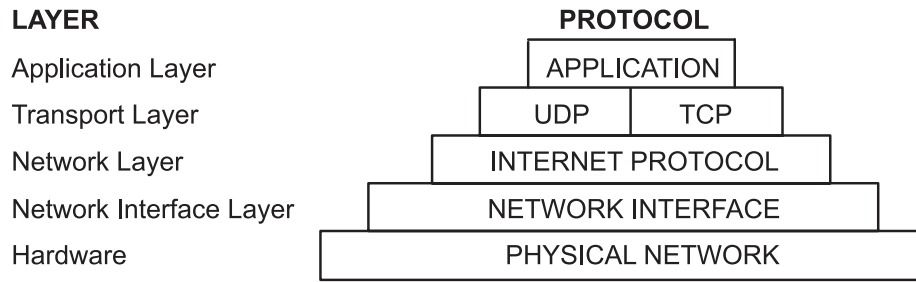


Figure 3. TCP/IP Suite of Protocols. This illustration depicts the layers of the TCP/IP protocol. From the top they are, Application Layer, Transport Layer, Network Layer, Network Interface Layer, and Hardware.

TCP/IP carefully defines how information moves from sender to receiver. First, application programs send messages or streams of data to one of the Internet Transport Layer Protocols, either the **User Datagram Protocol (UDP)** or the **Transmission Control Protocol (TCP)**. These protocols receive the data from the application, divide it into smaller pieces called *packets*, add a destination address, and then pass the packets along to the next protocol layer, the Internet Network layer.

The Internet Network layer encloses the packet in an **Internet Protocol (IP)** datagram, puts in the datagram header and trailer, decides where to send the datagram (either directly to a destination or else to a gateway), and passes the datagram on to the Network Interface layer.

The Network Interface layer accepts **IP** datagrams and transmits them as *frames* over a specific network hardware, such as Ethernet or Token-Ring networks.

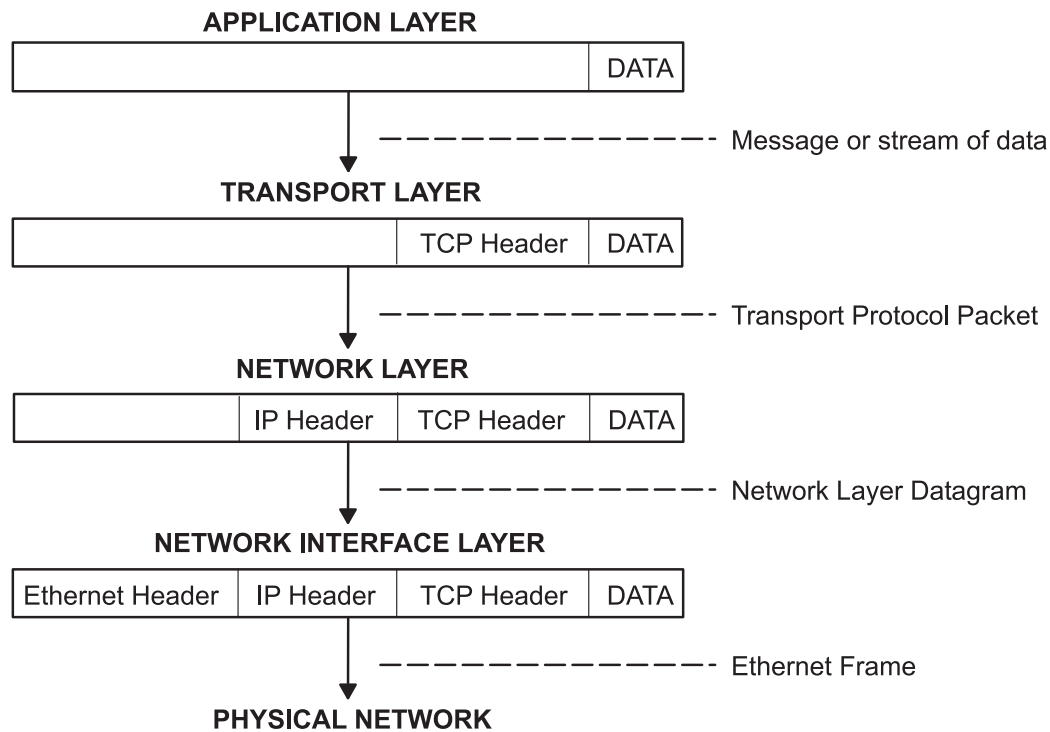


Figure 4. Movement of Information from Sender Application to Receiver Host. This illustration shows the flow of information down the TCP/IP protocol layers from the Sender to the Host.

Frames received by a host go through the protocol layers in reverse. Each layer strips off the corresponding header information, until the data is back at the application layer.

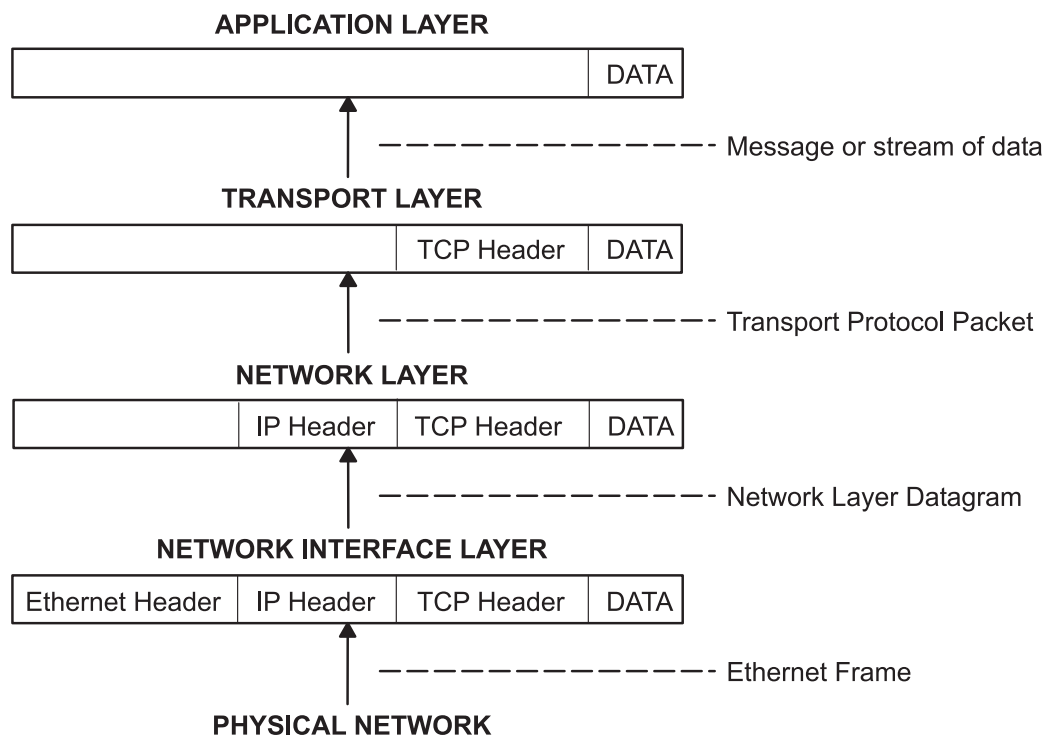
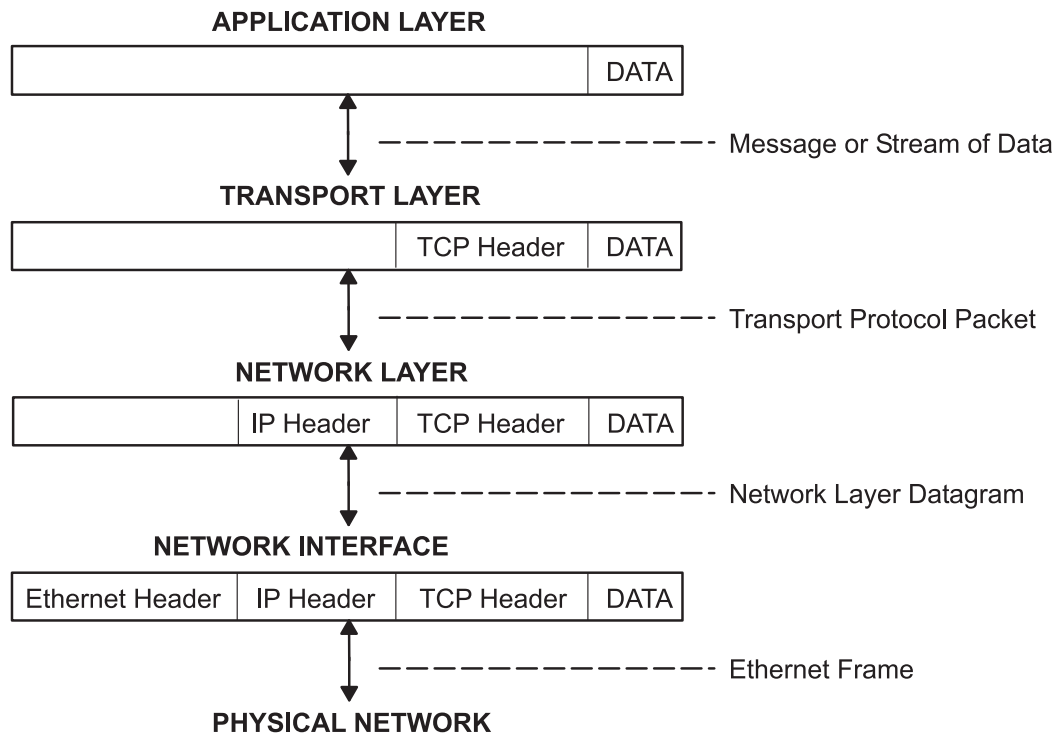


Figure 5. Movement of Information from Host to Application. This illustration shows the flow of information up the TCP/IP protocol layers from the Host to the Sender.

Frames are received by the Network Interface layer (in this case, an Ethernet adapter). The Network Interface layer strips off the Ethernet header, and sends the datagram up to the Network layer. In the Network layer, the Internet Protocol strips off the IP header and sends the packet up to the Transport layer. In the Transport layer, the **TCP** (in this case) strips off the **TCP** header and sends the data up to the Application layer.

Hosts on a network send and receive information simultaneously. The "Host Data Transmission and Reception" figure below more accurately represents a host as it communicates.



Note: Headers are added and stripped in each protocol layer as data is transmitted and received by a host.

Figure 6. Host Data Transmissions and Receptions. This illustration shows data flowing both ways through the TCP/IP layers.

Internet Protocol (IP) Version 6 Overview

Internet Protocol (IP) Version 6 (IPv6 or IPng) is the next generation of **IP** and has been designed to be an evolutionary step from **IP** Version 4 (IPv4). While IPv4 has allowed the development of a global Internet, it is not capable of carrying much farther into the future because of two fundamental factors: limited address space and routing complexity. The IPv4 32-bit addresses do not provide enough flexibility for global Internet routing. The deployment of Classless InterDomain Routing (CIDR) has extended the lifetime of IPv4 routing by a number of years, but the effort to better manage the routing will continue. Even if IPv4 routing could be scaled up, the Internet will eventually run out of network numbers.

The Internet Engineering Task Force (IETF) recognized that IPv4 would not be able to support the phenomenal growth of the Internet, so the IETF IPng working group was formed. Of the proposals that were made, **Simple Internet Protocol Plus (SIPP)** was chosen as an evolutionary step in the development of IP. This was renamed to IPng, and RFC1883 was finalized in December of 1995.

IPv6 extends the maximum number of Internet addresses to handle the ever increasing Internet user population. As an evolutionary change from IPv4, IPv6 has the advantage of allowing the new and the old to coexist on the same network. This coexistence enables an orderly migration from IPv4 (32 bit addressing) to IPv6 (128 bit addressing) on an operational network.

This overview is intended to give the reader a general understanding of the IPng protocol. For detailed information, please see RFCs 1883, 1884, 1885, 1886, 1970, 1971, and 2133.

Expanded Routing and Addressing

IPv6 increases the **IP** address size from 32 bits to 128 bits, thereby supporting more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler autoconfiguration of addresses.

IPv6 has three types of addresses:

unicast	<p>A packet sent to a unicast address is delivered to the interface identified by that address. A unicast address has a particular scope: link-local, site-local, global. There are also two special unicast addresses:</p> <ul style="list-style-type: none">• <code>::/128</code> (unspecified address)• <code>::1/128</code> (loopback address)
multicast	<p>A packet sent to a multicast address is delivered to all interfaces identified by that address. A multicast address is identified by the prefix <code>ff::/8</code>. As with unicast addresses, multicast addresses have a similar scope: node-local, link-local, site-local, and organization-local.</p>
anycast	<p>An anycast address is an address that has a single sender, multiple listeners, and only one responder (normally the "nearest" one, according to the routing protocols' measure of distance). For example, several web servers listening on an anycast address. When a request is sent to the anycast address, only one responds.</p> <p>An anycast address is indistinguishable from a unicast address. A unicast address becomes an anycast address when more than one interface is configured with that address.</p>

Note: There are no broadcast addresses in IPv6. Their function has been superseded by the multicast address.

Autoconfiguration: The primary mechanisms available that enable a node to start up and communicate with other nodes over an IPv4 network are hard-coding, **BOOTP**, and **DHCP**.

IPv6 introduces the concept of *scope* to **IP** addresses, one of which is link-local. This allows a host to construct a valid address from the predefined link-local prefix and its local identifier. This local identifier is typically the medium access control (MAC) address of the interface to be configured. Using this address, the node can multicast to a server, rather than broadcast and, for a fully-isolated subnet, might not need any other address configuration.

Meaningful Addresses: With IPv4, the only generally recognizable meaning in addresses are broadcast (typically all 1s or all 0s), and classes (for example, a class D is multicast). With IPv6, the prefix can be quickly examined to determine *scope* (for example, link-local), multicast versus unicast, and a mechanism of assignment (provider-based or geography-based).

Routing information might be explicitly loaded into the upper bits of addresses as well, but this has not yet been finalized by the IETF (for provider-based addresses, routing information is implicitly present in the address).

Duplicate Address Detection: When an interface is initialized or reinitialized, it uses autoconfiguration to tentatively associate a link-local address with that interface (the address is not yet assigned to that interface in the traditional sense). At this point, the interface joins the all-nodes and solicited-nodes multicast groups, and sends a neighbor discovery message to these groups. By using the multicast address, the node can determine whether that particular link-local address has been previously assigned, and choose an alternate address. This eliminates accidentally assigning the same address to two different interfaces on the same link. (It is still possible to create duplicate global-scope addresses for nodes that are not on the same link.)

Neighbor Discovery/Stateless Address Autoconfiguration: **Neighbor Discovery Protocol (NDP)** for IPv6 is used by nodes (hosts and routers) to determine the link-layer addresses for neighbors known to reside on attached links, and maintain per-destination routing tables for active connections. Hosts also use **ND** to find neighboring routers that are willing to forward packets on their behalf and detect changed link-layer addresses. **NDP** uses the **Internet Control Message Protocol (ICMP)** Version 6 with its own unique message types. In general terms, the IPv6 Neighbor Discovery protocol corresponds to a

combination of the IPv4 **Address Resolution Protocol (ARP)**, ICMP Router Discovery (RDISC), and **ICMP Redirect (ICMPv4)**, but with many improvements over these IPv4 protocols.

IPv6 defines both a stateful and a stateless address autoconfiguration mechanism. *Stateless autoconfiguration* requires no manual configuration of hosts; minimal, if any, configuration of routers; and no additional servers. The stateless mechanism allows a host to generate its own addresses using a combination of locally available information and information advertised by routers. Routers advertise prefixes that identify the subnets associated with a link, while hosts generate an interface token that uniquely identifies an interface on a subnet. An address is formed by combining the two. In the absence of routers, a host can only generate link-local addresses. However, link-local addresses are sufficient for allowing communication among nodes attached to the same link.

Routing Simplification

To simplify routing issues, IPv6 addresses are considered in two parts: a prefix and an ID. This might seem the same as the IPv4 net-host address breakdown, but it has two advantages:

- no class** No fixed number of bits for prefix or ID, which allows for a reduction in loss due to over-allocation
- nesting** An arbitrary number of divisions can be employed by considering different numbers of bits as the prefix.

Case 1

128 bits
node address

Case 2

n bits	$128-n$ bits
Subnet prefix	Interface ID

Case 3:

n bits	$80-n$ bits	48 bits
Subscriber prefix	Subnet ID	Interface ID

Case 4:

s bits	n bits	m bits	$128-s-n-m$ bits
Subscribe prefix	Area ID	Subnet ID	Interface ID

Generally, IPv4 cannot go beyond Case 3, even with Variable Length Subnet Mask (VLSM is a means of allocating IP addressing resources to subnets according to their individual need rather than some general network-wide rule). This is as much an artifact of the shorter address length as the definition of variable length prefixes, but is worth noting nonetheless.

Header Format Simplification

IPv6 simplifies the IP header, by removing entirely or by moving to an extension header, some of the fields found in the IPv4 header. It defines a more flexible format for optional information (the extension headers). Specifically, note the absence of:

- header length (length is constant)
- identification

- flags
- fragment offset (moved into fragmentation extension headers)
- header checksum (upper-layer protocol or security extension header handles data integrity).

IPv4 Header:

Table 1.

Version	IHL	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Time to Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options				Padding

IPv6 Header:

Table 2.

Version	Prio	Flow Label		
Payload Length		Next Header	Hop Limit	
Source Address				
Destination Address				

IPng includes an improved options mechanism over IPv4. IPv6 options are placed in separate extension headers that are located between the IPv6 header and the transport-layer header in a packet. Most extension headers are not examined or processed by any router along a packet delivery path until it arrives at its final destination. This mechanism facilitates a major improvement in router performance for packets containing options. In IPv4 the presence of any options requires the router to examine all options.

Another improvement is that, unlike IPv4 options, IPv6 extension headers can be of arbitrary length and the total amount of options carried in a packet is not limited to 40 bytes. This feature, plus the manner in which it is processed, permits IPv6 options to be used for functions that were not practical in IPv4, such as the IPv6 Authentication and Security Encapsulation options.

To improve the performance when handling subsequent option headers and the transport protocol that follows, IPv6 options are always an integer multiple of eight octets long to retain this alignment for subsequent headers.

By using extension headers instead of a protocol specifier and options fields, newly defined extensions can be integrated more easily.

Current specifications define extension headers in the following ways:

- Hop-by-hop options that apply to each hop (router) along the path
- Routing header for loose/strict source routing (used infrequently)
- A fragment defines the packet as a fragment and contains information about the fragment (IPv6 routers do not fragment)
- Authentication IP Security
- Encryption IP Security
- Destination options for the destination node (ignored by routers).

Improved Quality-of-Service/Traffic Control

While quality of service can be controlled by use of a control protocol such as **RSVP**, IPv6 provides for explicit priority definition for packets by using the priority field in the **IP** header. A node can set this value to indicate the relative priority of a particular packet or set of packets, which can then be used by the node, one or more routers, or the destination to make choices concerning the packet (that is, dropping it or not).

IPv6 specifies two types of priorities, those for congestion-controlled traffic, and those for non-congestion-controlled traffic. No relative ordering is implied between the two types.

Congestion-controlled traffic is defined as traffic that responds to congestion through some sort of "back-off" or other limiting algorithm. Priorities for congestion-controlled traffic are:

0	uncharacterized traffic
1	"filler" traffic (for example, netnews)
2	unattended data transfer (for example, mail)
3	(reserved)
4	attended bulk transfer (for example, FTP)
5	(reserved)
6	interactive traffic (for example, Telnet)
7	control traffic (for example, routing protocols)

Non-congestion-controlled traffic is defined as traffic that responds to congestion by dropping (or simply not resending) packets, such as video, audio, or other real-time traffic. Explicit levels are not defined with examples, but the ordering is similar to that for congestion-controlled traffic:

- The lowest value that the source is most willing to have discarded should be used for traffic.
- The highest value that the source is least willing to have discarded should be used for traffic.

This priority control is only applicable to traffic from a particular source address. Control traffic from one address is not an explicitly higher priority than attended bulk transfer from another address.

Flow Labeling: Outside of basic prioritization of traffic, IPv6 defines a mechanism for specifying a particular flow of packets. In IPv6 terms, a *flow* is defined as a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers.

This flow identification can be used for priority control, but might also be used for any number of other controls.

The flow label is chosen randomly, and does not identify any characteristic of the traffic other than the flow to which it belongs. This means that a router cannot determine that a packet is a particular type by examining the flow label. It can, however, determine that it is part of the same sequence of packets as the last packet containing that label.

Note: Until IPv6 is in general use, the flow label is mostly experimental. Uses and controls involving flow labels have not yet been defined nor standardized.

Jumbograms: An IPv4 packet size is limited to 64K. Using the jumbo payload extension header, an IPv6 packet can be up to 2^{32} octets (slightly over 4 gigabytes).

Tunneling

The key to a successful IPv6 transition is compatibility with the existing installed base of IPv4 hosts and routers. Maintaining compatibility with IPv4 while deploying IPv6 streamlines the task of transitioning the Internet to IPv6.

While the IPv6 infrastructure is being deployed, the existing IPv4 routing infrastructure can remain functional, and can be used to carry IPv6 traffic. Tunneling provides a way to use an existing IPv4 routing infrastructure to carry IPv6 traffic.

IPv6 or IPv4 hosts and routers can tunnel IPv6 datagrams over regions of IPv4 routing topology by encapsulating them within IPv4 packets. Tunneling can be used in a variety of ways:

Router-to-Router	IPv6 or IPv4 routers interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans one segment of the end-to-end path that the IPv6 packet takes.
Host-to-Router	IPv6 or IPv4 hosts can tunnel IPv6 packets to an intermediary IPv6 or IPv4 router that is reachable through an IPv4 infrastructure. This type of tunnel spans the first segment of the packet's end-to-end path.
Host-to-Host	IPv6 or IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans the entire end-to-end path that the packet takes.
Router-to-Host	IPv6/IPv4 routers can tunnel IPv6 packets to their final destination IPv6 or IPv4 host. This tunnel spans only the last segment of the end-to-end path.

Tunneling techniques are usually classified according to the mechanism by which the encapsulating node determines the address of the node at the end of the tunnel. In router-to-router or host-to-router methods, the IPv6 packet is being tunneled to a router. In host-to-host or router-to-host methods, the IPv6 packet is tunneled all the way to its final destination.

The entry node of the tunnel (the encapsulating node) creates an encapsulating IPv4 header and transmits the encapsulated packet. The exit node of the tunnel (the decapsulating node) receives the encapsulated packet, removes the IPv4 header, updates the IPv6 header, and processes the received IPv6 packet. However, the encapsulating node needs to maintain soft state information for each tunnel, such as the maximum transmission unit (MTU) of the tunnel, to process IPv6 packets forwarded into the tunnel.

IPv6 Security

For details about IP Security, versions 4 and 6, see Internet Protocol (IP) Security.

IPv6 Multihomed Link-Local and Site-Local Support

A host can have more than one interface defined. A host with two or more active interfaces is called multihomed. Each interface has a link-local address associated with it. Link-local addresses are sufficient for allowing communication among nodes attached to the same link.

A multihomed host has two or more associated link-local addresses. The AIX IPv6 implementation has 4 options to handle how link-layer address resolution is resolved on multihomed hosts. Option 1 is the default.

Option 0	No multihomed actions are taken. Transmissions will go out on the first link-local interface. When the Neighbor Discovery Protocol (NDP) must perform address resolution, it multicasts a Neighbor Solicitation message out on each interface with a link local address defined. NDP queues the data packet until the first Neighbor Advertisement message is received. The data packet is then sent out on this link.
Option 1	When the NDP must perform address resolution, that is, when sending a data packet to a destination and the the link-layer information for the next hop is not in the Neighbor Cache, it multicasts a Neighbor Solicitation message out on each interface with a link-local address defined. NDP then queues the data packet until it gets the link-layer information. NDP then waits until a response is received for each interface. This guarantees that the data packets are sent on the appropriate outgoing interfaces. If NDP did not wait, but responded to the first Neighbor Advertisement received, it would be possible for a data packet to be sent out on a link not associated with the packet source address. Because NDP must wait, a delay in the first packet being sent occurs. However, the delay occurs anyway in waiting for the first response.

Option 2

Multihomed operation is allowed, but dispatching of a data packet is limited to the interface specified by `main_if6`. When the **NDP** must perform address resolution, it multicasts a Neighbor Solicitation message out on each interface with a link-local address defined. It then waits for a Neighbor Advertisement message from the interface specified by `main_if6` (see the **no** command). Upon receiving a response from this interface, the data packet is sent out on this link.

Option 3

Multihomed operation is allowed, but dispatching of a data packet is limited to the interface specified by `main_if6` and site-local addresses are only routed for the interface specified by `main_site6` (see the **no** command). The NDP operates just as it does for Option 2. For applications that route data packets using site-local addresses on a multihomed host, only the site-local address specified by `main_site6` are used.

Packet Tracing

Packet tracing is the process by which you can verify the path of a packet through the layers to its destination. The **iptrace** command performs network interface level packet tracing. The **ipreport** command issues output on the packet trace in both hexadecimal and ASCII format. The **trpt** command performs transport protocol level packet tracking for the **TCP**. The **trpt** command output is more detailed, including information on time, **TCP** state, and packet sequencing.

Network Interface Packet Headers

At the Network Interface layer, packet headers are attached to outgoing data.

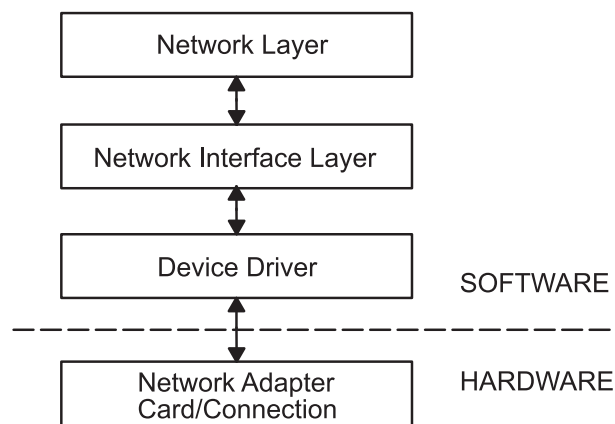


Figure 7. Packet Flow through Network Interface Structure. This illustration shows bi-directional data flow through the layers of the Network Interface Structure. From the top (software) they are the Network Layer, Network Interface Layer, Device Driver, and the (hardware) Network Adapter Card or Connection.

Packets are then sent through the network adapter to the appropriate network. Packets can pass through many gateways before reaching their destinations. At the destination network, the headers are stripped from the packets and the data is sent to the appropriate host.

The following section contains packet header information for several of the more common network interfaces.

Ethernet Adapter Frame Headers

An **Internet Protocol (IP)** or **Address Resolution Protocol (ARP)** frame header for the Ethernet adapter is composed of three fields as shown in the following table..

Ethernet Adapter Frame Header		
Field	Length	Definition
DA	6 bytes	Destination address.
SA	6 bytes	Source address. If bit 0 of this field is set to 1, it indicates that routing information (RI) is present.
Type	2 bytes	Specifies whether the packet is IP or ARP . The type number values are listed below.

Type field numbers:

IP 0800
 ARP 0806

Token-Ring Frame Headers

The medium access control (MAC) header for the token-ring adapter is composed of five fields, as shown in the following table.

Token-Ring MAC Header		
Field	Length	Definition
AC	1 byte	Access control. The value in this field x'00' gives the header priority 0.
FC	1 byte	Field control. The value in this field x'40' specifies the Logical Link Control frame.
DA	6 bytes	Destination address.
SA	6 bytes	Source address. If bit 0 of this field is set to 1, it indicates that routing information (RI) is present.
RI	18 bytes	Routing information. The valid fields are discussed below.

The MAC header consists of two routing information fields of two bytes each: routing control (RC) and segment numbers. A maximum of eight segment numbers can be used to specify recipients of a limited broadcast. RC information is contained in bytes 0 and 1 of the RI field. The settings of the first two bits of the RC field have the following meanings:

- bit (0) = 0** Use the nonbroadcast route specified in the RI field.
- bit (0) = 1** Create the RI field and broadcast to all rings.
- bit (1) = 0** Broadcast through all bridges.
- bit (1) = 1** Broadcast through limited bridges.

The logical link control (LLC) header is composed of five fields, as shown in the following LLC header table.

802.3 LLC Header		
Field	Length	Definition

DSAP	1 byte	Destination service access point. The value in this field is x'aa'.
SSAP	1 byte	Source service access point. The value in this field is x'aa'.
CONTROL	1 byte	Determines the LLC commands and responses. The three possible values for this field are discussed below.
PROT_ID	3 bytes	Protocol ID. This field is reserved. It has a value of x'0'.
TYPE	2 bytes	Specifies whether the packet is IP or ARP .

Control Field Values:

- x'03'** Unnumbered Information (UI) frame. This is the normal, or unsequenced, way in which token-ring adapter data is transmitted through the network. **TCP/IP** sequences the data.
- x'AF'** Exchange identification (XID) frame. This frame conveys the characteristics of the sending host.
- x'E3'** Test frame. This frame supports testing of the transmission path, echoing back the data that is received.

802.3 Frame Headers

The MAC header for the 802.3 adapter is composed of two fields, as shown in the following MAC header table.

802.3 MAC Header		
Field	Length	Definition
DA	6 bytes	Destination address.
SA	6 bytes	Source address. If bit 0 of this field is set to 1, it indicates that routing information (RI) is present.

The LLC header for 802.3 is the same as for Token-Ring MAC header.

Internet Network-Level Protocols

The Internet network-level protocols handle machine-to-machine communication. In other words, this layer implements **TCP/IP** routing. These protocols accept requests to send packets (along with the network address of the destination machine) from the Transport layer, convert the packets to datagram format, and send them down to the Network Interface layer for further processing.

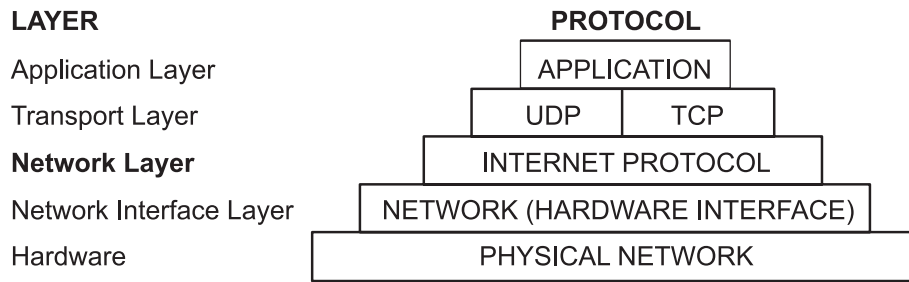


Figure 8. Network Layer of the TCP/IP Suite of Protocols. This illustration shows the various layers of the TCP/IP Suite of Protocols. From the top, the application layer consists of the application. The transport layer contains UDP and TCP. The network layer contains the network (hardware) interface. And finally, the hardware layer contains the physical network.

TCP/IP provides the protocols that are required to comply with RFC 1100, *Official Internet Protocols*, as well as other protocols commonly used by hosts in the Internet community.

Note: The use of Internet network, version, socket, service, and protocol numbers in **TCP/IP** also complies with RFC 1010, *Assigned Numbers*.

Address Resolution Protocol

The first network-level protocol is the **Address Resolution Protocol (ARP)**. **ARP** dynamically translates Internet addresses into the unique hardware addresses on local area networks.

To illustrate how **ARP** works, consider two nodes, X and Y. If node X wishes to communicate with Y, and X and Y are on different local area networks (LANs), X and Y communicate through *bridges, routers, or gateways*, using IP addresses. Within a LAN, nodes communicate using low-level hardware addresses.

Nodes on the same segment of the same LAN use **ARP** to determine the hardware address of other nodes. First, node X broadcasts an **ARP** request for node Y's hardware address. The **ARP** request contains X's **IP** and hardware addresses, and Y's **IP** address. When Y receives the **ARP** request, it places an entry for X in its **ARP** cache (which is used to map quickly from IP address to hardware address), then responds directly to X with an **ARP** response containing Y's **IP** and hardware addresses. When node X receives Y's **ARP** response, it places an entry for Y in its **ARP** cache.

Once an **ARP** cache entry exists at X for Y, node X is able to send packets directly to Y without resorting again to **ARP** (unless the **ARP** cache entry for Y is deleted, in which case **ARP** is reused to contact Y).

Unlike most protocols, **ARP** packets do not have fixed-format headers. Instead, the message is designed to be useful with a variety of network technologies, such as:

- Ethernet LAN adapter (supports both Ethernet and 802.3 protocols)
- Token-ring network adapter
- Fiber Distributed Data Interface (FDDI) network adapter

However, **ARP** does not translate addresses for **Serial Line Interface Protocol (SLIP)** or **Serial Optical Channel Converter (SOC)**, since these are point-to-point connections.

The kernel maintains the translation tables, and the **ARP** is not directly available to users or applications. When an application sends an Internet packet to one of the interface drivers, the driver requests the appropriate address mapping. If the mapping is not in the table, an **ARP** broadcast packet is sent through the requesting interface driver to the hosts on the local area network.

Entries in the **ARP** mapping table are deleted after 20 minutes; incomplete entries are deleted after 3 minutes. To make a permanent entry in the **ARP** mapping tables, use the **arp** command with the *pub* parameter:

```
arp -s 802.3 host2 0:dd:0:a:8s:0 pub
```

When any host that supports **ARP** receives an **ARP** request packet, the host notes the **IP** and hardware addresses of the requesting system and updates its mapping table, if necessary. If the receiving host **IP** address does not match the requested address, the host discards the request packet. If the **IP** address does match, the receiving host sends a response packet to the requesting system. The requesting system stores the new mapping and uses it to transmit any similar pending Internet packets.

Internet Control Message Protocol

The second network-level protocol is the **Internet Control Message Protocol (ICMP)**. **ICMP** is a required part of every **IP** implementation. **ICMP** handles error and control messages for **IP**. This protocol allows gateways and hosts to send problem reports to the machine sending a packet. **ICMP** does the following:

- Tests whether a destination is alive and reachable
- Reports parameter problems with a datagram header
- Performs clock synchronization and transit time estimations
- Obtains Internet addresses and subnet masks

Note: **ICMP** uses the basic support of **IP** as if it were a higher-level protocol. However, **ICMP** is actually an integral part of **IP** and must be implemented by every **IP** module.

ICMP provides feedback about problems in the communications environment, but does not make **IP** reliable. That is, **ICMP** does not guarantee that an **IP** packet is delivered reliably or that an **ICMP** message is returned to the source host when an **IP** packet is not delivered or is incorrectly delivered.

ICMP messages might be sent in any of the following situations:

- When a packet cannot reach its destination
- When a gateway host does not have the buffering capacity to forward a packet
- When a gateway can direct a host to send traffic on a shorter route

TCP/IP sends and receives several **ICMP** message types. **ICMP** is embedded in the kernel, and no application programming interface (API) is provided to this protocol.

Internet Control Message Protocol Message Types

ICMP sends and receives the following message types:

echo request	Sent by hosts and gateways to test whether a destination is alive and reachable.
information request	Sent by hosts and gateways to obtain an Internet address for a network to which they are attached. This message type is sent with the network portion of IP destination address set to a value of 0.
timestamp request	Sent to request that the destination machine return its current value for time of day.
address mask request	Sent by host to learn its subnet mask. The host can either send to a gateway, if it knows the gateway address, or send a broadcast message.
destination unreachable	Sent when a gateway cannot deliver an IP datagram.
source quench	Sent by discarding machine when datagrams arrive too quickly for a gateway or host to process, in order to request that the original source slow down its rate of sending datagrams.

redirect message	Sent when a gateway detects that some host is using a nonoptimum route.
echo reply	Sent by any machine that receives an echo request in reply to the machine which sent the request.
information reply	Sent by gateways in response to requests for network addresses, with both the source and destination fields of the IP datagram specified.
timestamp reply	Sent with current value of time of day.
address mask reply	Sent to machines requesting subnet masks.
parameter problem	Sent when a host or gateway finds a problem with a datagram header.
time exceeded	Sent when the following are true: <ul style="list-style-type: none"> • Each IP datagram contains a time-to-live counter (hop count), which is decremented by each gateway. • A gateway discards a datagram because its hop count has reached a value of 0.
Internet Timestamp	Used to record the time stamps through the route.

Internet Protocol

The third network-level protocol is the **Internet Protocol (IP)**, which provides unreliable, connectionless packet delivery for the Internet. **IP** is connectionless because it treats each packet of information independently. It is unreliable because it does not guarantee delivery, meaning, it does not require acknowledgments from the sending host, the receiving host, or intermediate hosts.

IP provides the interface to the network interface level protocols. The physical connections of a network transfer information in a frame with a header and data. The header contains the source address and the destination address. **IP** uses an Internet datagram that contains information similar to the physical frame. The datagram also has a header containing Internet addresses of both source and destination of the data.

IP defines the format of all the data sent over the Internet.

Bits

0	4	8	16	19	31
Version	Length	Type of Service	Total Length		
Identification			Flags	Fragment Offset	
Time to Live		Protocol	Header Checksum		
Source Address					
Destination Address					
Options					
Data					

Figure 9. Internet Protocol Packet Header. This illustration shows the first 32 bits of a typical IP packet header. Table below lists the various entities.

IP Header Field Definitions

Version	Specifies the version of the IP used. The current version of the IP protocol is 4.
Length	Specifies the datagram header length, measured in 32-bit words.

Type of Service	Contains five subfields that specify the type of precedence, delay, throughput, and reliability desired for that packet. (The Internet does not guarantee this request.) The default settings for these five subfields are routine precedence, normal delay, normal throughput, and normal reliability. This field is not generally used by the Internet at this time. This implementation of IP complies with the requirements of the IP specification, RFC 791, <i>Internet Protocol</i> .
Total Length	Specifies the length of the datagram including both the header and the data measured in octets. Packet fragmentation at gateways, with reassembly at destinations, is provided. The total length of the IP packet can be configured on an interface-by-interface basis with the Web-based System Manager, wsm , the ifconfig command, or the System Management Interface Tool (SMIT) fast path, smit chinet . Use Web-based System Manager or SMIT to set the values permanently in the configuration database; use the ifconfig command to set or change the values in the running system.
Identification Flags	Contains a unique integer that identifies the datagram. Controls datagram fragmentation, along with the Identification field. The Fragment Flags specify whether the datagram can be fragmented and whether the current fragment is the last one.
Fragment Offset	Specifies the offset of this fragment in the original datagram measured in units of 8 octets.
Time to Live	Specifies how long the datagram can remain on the Internet. This keeps misrouted datagrams from remaining on the Internet indefinitely. The default time to live is 255 seconds.
Protocol	Specifies the high-level protocol type.
Header Checksum	Indicates a number computed to ensure the integrity of header values.
Source Address	Specifies the Internet address of the sending host.
Destination Address	Specifies the Internet address of the receiving host.

Options

Provides network testing and debugging. This field is not required for every datagram.

End of Option List

Indicates the end of the option list. It is used at the end of the final option, not at the end of each option individually. This option should be used only if the end of the options would not otherwise coincide with the end of the **IP** header. End of Option List is used if options exceed the length of the datagram.

No Operation

Provides alignment between other options; for example, to align the beginning of a subsequent option on a 32-bit boundary.

Loose Source and Record Route

Provides a means for the source of an Internet datagram to supply routing information used by the gateways in forwarding the datagram to a destination and in recording the route information. This is a *loose* source route: the gateway or host **IP** is allowed to use any route of any number of other intermediate gateways in order to reach the next address in the route.

Strict Source and Record Route

Provides a means for the source of an Internet datagram to supply routing information used by the gateways in forwarding the datagram to a destination and in recording the route information. This is a *strict* source route: In order to reach the next gateway or host specified in the route, the gateway or host **IP** must send the datagram directly to the next address in the source route and only to the directly connected network that is indicated in the next address.

Record Route

Provides a means to record the route of an Internet datagram.

Stream Identifier

Provides a way for a stream identifier to be carried through networks that do not support the stream concept.

Internet Timestamp

Provides a record of the time stamps through the route.

Outgoing packets automatically have an **IP** header prefixed to them. Incoming packets have their **IP** header removed before being sent to the higher-level protocols. The **IP** protocol provides for the universal addressing of hosts in the Internet network.

Internet Transport-Level Protocols

The **TCP/IP** transport-level protocols allow application programs to communicate with other application programs.

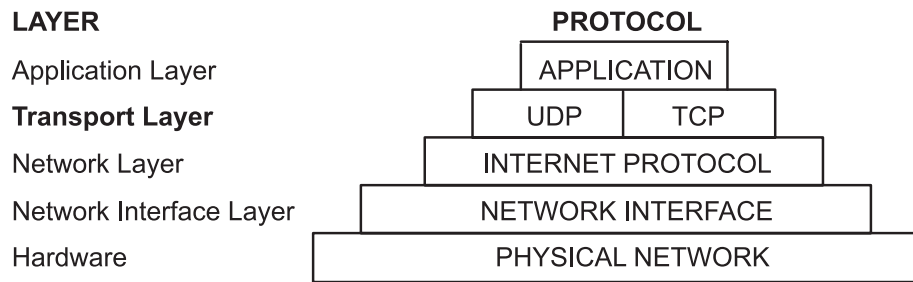


Figure 10. Transport Layer of the TCP/IP Suite of Protocols. This illustration shows the various layers of the TCP/IP Suite of Protocols. From the top, the application layer consists of the application. The transport layer contains UDP and TCP. The network layer contains the network (hardware) interface. And finally, the hardware layer contains the physical network.

User Datagram Protocol (UDP) and the **TCP** are the basic transport-level protocols for making connections between Internet hosts. Both **TCP** and **UDP** allow programs to send messages to and receive messages from applications on other hosts. When an application sends a request to the Transport layer to send a message, **UDP** and **TCP** break the information into packets, add a packet header including the destination address, and send the information to the Network layer for further processing. Both **TCP** and **UDP** use protocol ports on the host to identify the specific destination of the message.

Higher-level protocols and applications use **UDP** to make datagram connections and **TCP** to make stream connections. The operating system sockets interface implements these protocols.

User Datagram Protocol

Sometimes an application on a network needs to send messages to a specific application or process on another network. The **UDP** provides a datagram means of communication between applications on Internet hosts. Because senders do not know which processes are active at any given moment, **UDP** uses destination protocol ports (or abstract destination points within a machine), identified by positive integers, to send messages to one of multiple destinations on a host. The protocol ports receive and hold messages in queues until applications on the receiving network can retrieve them.

Since **UDP** relies on the underlying **IP** to send its datagrams, **UDP** provides the same connectionless message delivery as **IP**. It offers no assurance of datagram delivery or duplication protection. However, **UDP** does allow the sender to specify source and destination port numbers for the message and calculates a checksum of both the data and header. These two features allow the sending and receiving applications to ensure the correct delivery of a message.

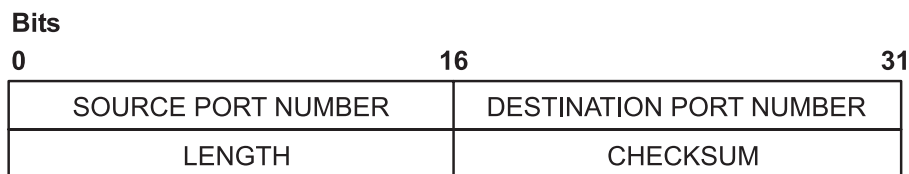


Figure 11. User Datagram Protocol (UDP) Packet Header. This illustration shows the first 32 bits of the UDP packet header. The first 16 bits contain the source port number and the length. The second 16 bits contain the destination port number and the checksum.

Applications that require reliable delivery of datagrams must implement their own reliability checks when using **UDP**. Applications that require reliable delivery of streams of data should use **TCP**.

UDP Header Field Definitions

Source Port Number Address of the protocol port sending the information.

Destination Port Number	Address of the protocol port receiving the information.
Length	Length in octets of the UDP datagram.
Checksum	Provides a check on the UDP datagram using the same algorithm as the IP .

The applications programming interface (API) to **UDP** is a set of library subroutines provided by the sockets interface.

Transmission Control Protocol

TCP provides reliable stream delivery of data between Internet hosts. Like **UDP**, **TCP** uses Internet Protocol, the underlying protocol, to transport datagrams, and supports the block transmission of a continuous stream of datagrams between process ports. Unlike **UDP**, **TCP** provides reliable message delivery. **TCP** ensures that data is not damaged, lost, duplicated, or delivered out of order to a receiving process. This assurance of transport reliability keeps applications programmers from having to build communications safeguards into their software.

The following are operational characteristics of **TCP**:

Basic Data Transfer	TCP can transfer a continuous stream of 8-bit octets in each direction between its users by packaging some number of bytes into segments for transmission through the Internet system. TCP implementation allows a segment size of at least 1024 bytes. In general, TCP decides when to block and forward packets at its own convenience.
Reliability	TCP must recover data that is damaged, lost, duplicated, or delivered out of order by the Internet. TCP achieves this reliability by assigning a sequence number to each octet it transmits and requiring a positive acknowledgment (ACK) from the receiving TCP . If the ACK is not received within the time-out interval, the data is retransmitted. The TCP retransmission time-out value is dynamically determined for each connection, based on round-trip time. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments.
Flow Control	TCP governs the amount of data sent by returning a window with every ACK to indicate a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of octets that the sender may transmit before receiving further permission.
Multiplexing	TCP allows many processes within a single host to use TCP communications facilities simultaneously. TCP receives a set of addresses of ports within each host. TCP combines the port number with the network address and the host address to uniquely identify each socket. A pair of sockets uniquely identifies each connection.
Connections	TCP must initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection. Each connection is uniquely specified by a pair of sockets identifying its two sides.
Precedence and Security	Users of TCP may indicate the security and precedence of their communications. Default values are used when these features are not needed.

The **TCP Packet Header** figure illustrates these characteristics.

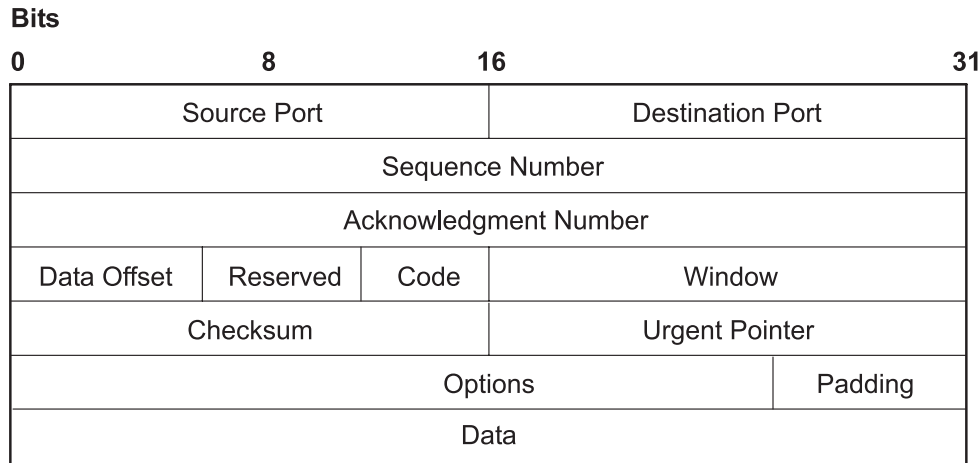


Figure 12. Transmission Control Protocol (TCP) Packet Header. This illustration shows what is contained in the TCP packet header. The individual entities are listed in the text below.

TCP Header Field Definitions

Source Port	Identifies the port number of a source application program.
Destination Port	Identifies the port number of a destination application program.
Sequence Number	Specifies the sequence number of the first byte of data in this segment.
Acknowledgment Number	Identifies the position of the highest byte received.
Data Offset	Specifies the offset of data portion of the segment.
Reserved	Reserved for future use.
Code	Control bits to identify the purpose of the segment:
URG	Urgent pointer field is valid.
ACK	Acknowledgement field is valid.
PSH	Segment requests a PUSH.
RTS	Resets the connection.
SYN	Synchronizes the sequence numbers.
FIN	Sender has reached the end of its byte stream.
Window	Specifies the amount of data the destination is willing to accept.
Checksum	Verifies the integrity of the segment header and data.
Urgent Pointer	Indicates data that is to be delivered as quickly as possible. This pointer specifies the position where urgent data ends.
Options	<p>End of Option List Indicates the end of the option list. It is used at the final option, not at the end of each option individually. This option needs to be used only if the end of the options would not otherwise coincide with the end of the TCP header.</p> <p>No Operation Indicates boundaries between options. Can be used between other options; for example, to align the beginning of a subsequent option on a word boundary. There is no guarantee that senders will use this option, so receivers must be prepared to process options even if they do not begin on a word boundary.</p> <p>Maximum Segment Size Indicates the maximum segment size TCP can receive. This is only sent in the initial connection request.</p>

The applications programming interface to **TCP** consists of a set of library subroutines provided by the sockets interface.

Internet Application-Level Protocols

TCP/IP implements higher-level Internet protocols at the application program level.

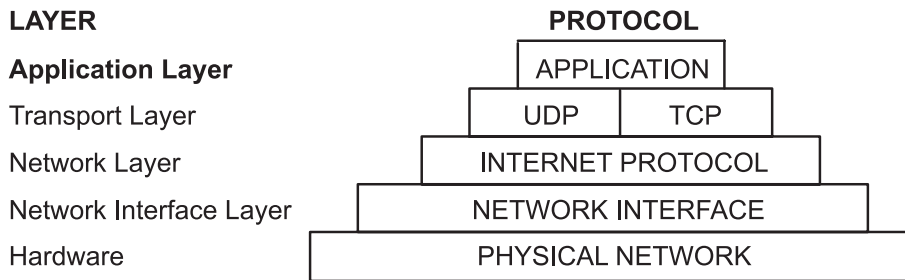


Figure 13. Application Layer of the TCP/IP Suite of Protocols. This illustration shows the various layers of the TCP/IP Suite of Protocols. From the top, the application layer consists of the application. The transport layer contains UDP and TCP. The network layer contains the network (hardware) interface. And finally, the hardware layer contains the physical network.

When an application needs to send data to another application on another host, the applications send the information down to the transport level protocols to prepare the information for transmission.

The official Internet application-level protocols include:

- **Domain Name Protocol (DOMAIN)**
- **Exterior Gateway Protocol (EGP)**
- **File Transfer Protocol (FTP)**
- **Name/Finger Protocol (FINGER)**
- **Telnet Protocol (TELNET)**
- **Trivial File Transfer Protocol (TFTP)**

TCP/IP implements other higher-level protocols that are not official Internet protocols but are commonly used in the Internet community at the application program level. These protocols include:

- Distributed Computer Network (DCN) **Local-Network Protocol (HELLO)**
- **Remote Command Execution Protocol (EXEC)**
- **Remote Login Protocol (LOGIN)**
- **Remote Shell Protocol (SHELL)**
- **Routing Information Protocol (RIP)**
- **Time Server Protocol (TIMED).**

TCP/IP does not provide APIs to any of these application-level protocols.

Domain Name Protocol

The **Domain Name Protocol (DOMAIN)** allows a host in a domain to act as a *name server* for other hosts within the domain. **DOMAIN** uses **UDP** or **TCP** as its underlying protocol and allows a local network to assign host names within its domain independently from other domains. Normally, the **DOMAIN** protocol uses **UDP**. However, if the **UDP** response is truncated, **TCP** can be used. The **DOMAIN** protocol in **TCP/IP** supports both.

In the **DOMAIN** hierarchical naming system, local resolver routines can resolve Internet names and addresses using a local name resolution database maintained by the **named** daemon. If the name requested by the host is not in the local database, the resolver routine queries a remote **DOMAIN** name server. In either case, if the name resolution information is unavailable, the resolver routines attempt to use the **/etc/hosts** file for name resolution.

Note: **TCP/IP** configures local resolver routines for the **DOMAIN** protocol if the local file **/etc/resolv.conf** exists. If this file does not exist, the **TCP/IP** configures the local resolver routines to use the **/etc/hosts** database.

TCP/IP implements the **DOMAIN** protocol in the **named** daemon and in the resolver routines and does not provide an API to this protocol.

Exterior Gateway Protocol

Exterior Gateway Protocol (EGP) is the mechanism that allows the exterior gateway of an *autonomous system* to share routing information with exterior gateways on other autonomous systems.

Autonomous Systems

An autonomous system is a group of networks and gateways for which one administrative authority has responsibility. Gateways are *interior neighbors* if they reside on the same autonomous system and *exterior neighbors* if they reside on different autonomous systems. Gateways that exchange routing information using **EGP** are said to be **EGP peers** or *neighbors*. Autonomous system gateways use **EGP** to provide access information to their **EGP** neighbors.

EGP allows an exterior gateway to ask another exterior gateway to agree to exchange access information, continually checks to ensure that its **EGP** neighbors are responding, and helps **EGP** neighbors to exchange access information by passing routing update messages.

EGP restricts exterior gateways by allowing them to advertise only those destination networks reachable entirely within that gateway's autonomous system. Thus, an exterior gateway using **EGP** passes along information to its **EGP** neighbors but does not advertise access information about its **EGP** neighbors outside its autonomous system.

EGP does not interpret any of the distance metrics that appear in routing update messages from other protocols. **EGP** uses the distance field to specify whether a path exists (a value of 255 means that the network is unreachable). The value cannot be used to compute the shorter of two routes unless those routes are both contained within a single autonomous system. Therefore, **EGP** cannot be used as a routing algorithm. As a result, there will be only one path from the exterior gateway to any network.

In contrast to the **Routing Information Protocol (RIP)**, which can be used within an autonomous system of Internet networks that dynamically reconfigure routes, **EGP** routes are predetermined in the **/etc/gated.conf** file. **EGP** assumes that **IP** is the underlying protocol.

EGP Message Types

Neighbor Acquisition Request

Used by exterior gateways to request to become neighbors of each other.

Neighbor Acquisition Reply

Used by exterior gateways to accept the request to become neighbors.

Neighbor Acquisition Refusal

Used by exterior gateways to deny the request to become neighbors. The refusal message includes reasons for refusal, such as out of table space.

Neighbor Cease

Used by exterior gateways to cease the neighbor relationship. The cease message includes reasons for ceasing, such as going down.

Neighbor Cease Acknowledgment	Used by exterior gateways to acknowledge the request to cease the neighbor relationship.
Neighbor Hello	Used by exterior gateways to determine connectivity. A gateway issues a Hello message and another gateway issues an I Heard You message.
I Heard You	Used by exterior gateways to reply to a Hello message. The I Heard You message includes the address of the answering gateway and, if the gateway is unreachable, a reason for lack of access, such as You are unreachable because of problems with my network interface.
NR Poll	Used by exterior gateways to query neighbor gateways about their ability to reach other gateways.
Network Reachability	Used by exterior gateways to answer the NR Poll message. For each gateway in the message, the Network Reachability message contains information on the addresses that gateway can reach through its neighbors.
EGP Error	Used by exterior gateways to respond to EGP messages that contain bad checksums or have fields containing incorrect values.

TCP/IP implements the **EGP** protocol in the **gated** server command and does not provide an API to this protocol.

File Transfer Protocol

File Transfer Protocol (FTP) allows hosts to transfer data among dissimilar hosts, as well as files between two foreign hosts indirectly. **FTP** provides for such tasks as listing remote directories, changing the current remote directory, creating and removing remote directories, and transferring multiple files in a single request. **FTP** keeps the transport secure by passing user and account passwords to the foreign host. Although **FTP** is designed primarily to be used by applications, it also allows interactive user-oriented sessions.

FTP uses reliable stream delivery (**TCP/IP**) to send the files and uses a Telnet connection to transfer commands and replies. **FTP** also understands several basic file formats including NETASCII, IMAGE, and Local 8.

TCP/IP implements **FTP** in the **ftp** user command and the **ftpd** server command and does not provide an applications programming interface (API) to this protocol.

When creating anonymous ftp users and directories please be sure that the home directory for users ftp and anonymous (for example, **/u/ftp**) is owned by root and does not allow write permissions (for example, dr-xr-xr-x). The script **/usr/samples/tcpip/anon.ftp** can be used to create these accounts, files and directories.

Telnet Protocol

The **Telnet Protocol (TELNET)** provides a standard method for terminal devices and terminal-oriented processes to interface. **TELNET** is commonly used by terminal emulation programs that allow you to log into a remote host. However, **TELNET** can also be used for terminal-to-terminal communication and interprocess communication. **TELNET** is also used by other protocols (for example, **FTP**) for establishing a protocol control channel.

TCP/IP implements **TELNET** in the **tn**, **telnet**, or **tn3270** user commands. The **telnetd** daemon does not provide an API to **TELNET**.

TCP/IP supports the following **TELNET** options which are negotiated between the client and server:

BINARY TRANSMISSION (Used in tn3270 sessions)	Transmits characters as binary data.
SUPPRESS GO_AHEAD (The operating system suppresses GO-AHEAD options.)	Indicates that when in effect on a connection between a sender of data and the receiver of the data, the sender need not transmit a GO_AHEAD option. If the GO_AHEAD option is not desired, the parties in the connection will probably suppress it in both directions. This action must take place in both directions independently.
TIMING MARK (Recognized, but has a negative response)	Makes sure that previously transmitted data has been completely processed.
EXTENDED OPTIONS LIST	Extends the TELNET option list for another 256 options. Without this option, the TELNET option allows only 256 options.
ECHO (User-changeable command)	Transmits echo data characters already received back to the original sender.
TERM TYPE	Enables the server to determine the type of terminal connected to a user TELNET program.
SAK (Secure Attention Key)	Establishes the environment necessary for secure communication between you and the system.
NAWS (Negotiate About Window Size)	Enables client and server to negotiate dynamically for the window size. This is used by applications that support changing the window size.

Note: **TELNET** must allow transmission of eight bit characters when not in binary mode in order to implement ISO 8859 Latin code page. This is necessary for internationalization of the **TCP/IP** commands.

Trivial File Transfer Protocol

The **Trivial File Transfer Protocol (TFTP)** can read and write files to and from a foreign host. Because **TFTP** uses the unreliable User Datagram Protocol to transport files, it is generally quicker than **FTP**. Like **FTP**, **TFTP** can transfer files as either NETASCII characters or as 8-bit binary data. Unlike **FTP**, **TFTP** cannot be used to list or change directories at a foreign host and it has no provisions for security like password protection. Also, data can be written or retrieved only in public directories.

The **TCP/IP** implements **TFTP** in the **tftp** and **utftp** user commands and in the **tftpd** server command. The **utftp** command is a form of the **tftp** command for use in a pipe. **TCP/IP** does not provide an API to this protocol.

Name/Finger Protocol

The **Name/Finger Protocol (FINGER)** is an application-level Internet protocol that provides an interface between the **finger** command and the **fingerd** daemon. The **fingerd** daemon returns information about the users currently logged in to a specified remote host. If you execute the **finger** command specifying a user at a particular host, you will obtain specific information about that user. The **FINGER** Protocol must be present at the remote host and at the requesting host. **FINGER** uses **Transmission Control Protocol (TCP)** as its underlying protocol.

Note: **TCP/IP** does not provide an API to this protocol.

Distributed Computer Network Local-Network Protocol

Local-Network Protocol (HELLO) is an interior gateway protocol designed for use within autonomous systems. (For more information, see Autonomous Systems.) **HELLO** maintains connectivity, routing, and time-keeping information. It allows each machine in the network to determine the shortest path to a destination based on time delay and then dynamically updates the routing information to that destination.

The **gated** daemon provides the Distributed Computer Network (DCN) local network protocol.

Remote Command Execution Protocol

The **rexec** user command and the **rexecd** daemon provide the remote command execution protocol, allowing users to run commands on a compatible remote host.

Remote Login Protocol

The **rlogin** user command and the **rlogind** daemon provide the **remote login protocol**, allowing users to log in to a remote host and use their terminals as if they were directly connected to the remote host.

Remote Shell Protocol

The **rsh** user command and the **rshd** daemon provide the **remote command shell protocol**, allowing users to open a shell on a compatible foreign host for running commands.

Routing Information Protocol

Routing Information Protocol (RIP) and the **routed** and **gated** daemons that implement it keep track of routing information based on gateway hops and maintain kernel-routing table entries.

Time Server Protocol

The **timed** daemon is used to synchronize one host with the time of other hosts. It is based on the client/server concept.

Assigned Numbers

For compatibility with the general network environment, well-known numbers are assigned for the Internet versions, networks, ports, protocols, and protocol options. Additionally, well-known names are also assigned to machines, networks, operating systems, protocols, services, and terminals. **TCP/IP** complies with the assigned numbers and names defined in RFC 1010, *Assigned Numbers*.

The **Internet Protocol (IP)** defines a 4-bit field in the **IP** header that identifies the version of the general Internetwork protocol in use. For **IP**, this version number in decimal is 4. For details on the assigned numbers and names used by **TCP/IP**, see **/etc/protocols** and **/etc/services** files included with **TCP/IP**. For further details on the assigned numbers and names, refer to RFC 1010 and the **/etc/services** file.

TCP/IP Local Area Network Adapter Cards

The topics discussed in this section are:

- Installing a Network Adapter
- Configuring a Token-Ring or Ethernet Adapter
- Configuring and Using Virtual Local Area Networks (VLANs)
- Using ATM Adapters.

The network adapter card is the hardware that is physically attached to the network cabling. It is responsible for receiving and transmitting data at the physical level. The network adapter card is controlled by the network adapter device driver.

A machine must have one network adapter card (or connection) for each network (not network type) to which it connects. For instance, if a host attaches to two token-ring networks, it must have two network adapter cards.

TCP/IP uses the following network adapter cards and connections:

- Standard Ethernet Version 2
- IEEE 802.3
- Token-ring
- Asynchronous adapters and native serial ports (described in *AIX 5L Version 5.1 Asynchronous Communications Guide*)
- Fiber Distributed Data Interface (FDDI)
- Serial Optical Channel Converter (described in *AIX 5L Version 5.1 Kernel Extensions and Device Support Programming Concepts*)
- Asynchronous Transfer Mode (ATM).

The Ethernet and 802.3 network technologies use the same type of adapter.

Each machine provides a limited number of expansion slots, some or all of which you might wish to use for communications adapters. Additionally, each machine supports a limited number of communications adapters of a given type. Each machine supports up to eight Ethernet/802.3 adapters, up to eight token-ring adapters, and one asynchronous adapter card with up to 64 connections. Within these limits (software limitations), you can install any combination of these adapters up to the total number of expansion slots available in your machine (hardware limitations).

Only one Transmission Control Protocol/Internet Protocol (TCP/IP) interface is configurable regardless of the number of Serial Optical Channel Converters supported by the system. The Serial Optical device driver makes use of both channel converters even though only one logical TCP/IP interface is configured.

Installing a Network Adapter

To install a network adapter:

1. Shut down the computer. See the **shutdown** command for information on how to shut down a system.
2. Turn off the computer power.
3. Remove the computer cover.
4. Find a free slot on the Micro Channel bus and insert the network adapter. Be careful to seat the adapter properly in the slot.
5. Replace the computer cover.
6. Restart the computer.

Configuring and Managing Adapters

To configure and manage token-ring or Ethernet adapters, use the tasks in the following table.

Configuring and Managing Adapters Tasks			
Task	SMIT Fast Path	Command or File	Web-based System Manager Management Environment ⁵

Configuring and Managing Adapters Tasks			
Configure an Adapter	smit chgtok (token ring) smit chgenet (Ethernet)	<ol style="list-style-type: none"> Determine adapter name:¹ <pre>lsdev -C -c adapter -t tokenring -H</pre> or <pre>lsdev -C -c adapter -t ethernet -H</pre> Reset ring speed (token ring) or connector type (Ethernet), if necessary. For example: <pre>chdev -l tok0 -a ring_speed=16 -P</pre> or <pre>chdev -l ent0 -a bnc_select=dix -P</pre> 	
Determining a Network Adapter Hardware Address	smit chgtok (token ring) smit chgenet (Ethernet)	<pre>lscfg -l tok0 -v (token ring)²</pre> <pre>lscfg -l ent0 -v (Ethernet)²</pre>	
Setting an Alternate Hardware Address	smit chgtok (token ring) smit chgenet (Ethernet)	<ol style="list-style-type: none"> Define the alternate hardware address. For example, for token ring:^{2,3} <pre>chdev -l tok0 -a alt_addr=0X10005A4F1B7F</pre> For Ethernet:^{2,3} <pre>chdev -l ent0 -a alt_addr=0X10005A4F1B7F -p</pre> Begin using alternate address, for token ring:⁴ <pre>chdev -l tok0 -a use_alt_addr=yes</pre> For Ethernet:⁴ <pre>chdev -l ent0 -a use_alt_addr=yes</pre> 	

Notes:

- The name of a network adapter can change if you move it from one slot to another or remove it from the system. If you ever move the adapter, issue the **diag -a** command to update the configuration database.
- Substitute your adapter name for tok0 and ent0.
- Substitute your hardware address for 0X10005A4F1B7F.
- After performing this procedure, you might experience a disruption of communication with other hosts until they flush their Address Resolution Protocol (ARP) cache and obtain the new hardware address of this host.
- These tasks are not available in Web-based System Manager Management Environment.

Configuring and Using Virtual Local Area Networks (VLANs)

VLANs (Virtual Local Area Networks) can be thought of as logical broadcast domains. A VLAN splits up groups of network users on a real physical network onto segments of logical networks. This implementation supports the IEEE 802.1Q VLAN tagging standard with the capability to support multiple VLAN IDs running on Ethernet adapters. Each VLAN ID is associated with a separate Ethernet interface to the upper layers (IP, etc.) and creates unique logical Ethernet adapter instances per VLAN, for example ent1, ent2 and so on.

The IEEE 802.1Q VLAN support can be configured over any supported Ethernet adapters. The adapters must be connected to a switch that supports IEEE 802.1Q VLAN.

You can configure multiple VLAN logical devices on a single system. Each VLAN logical devices constitutes an additional Ethernet adapter instance. These logical devices can be used to configure the same Ethernet IP interfaces as are used with physical Ethernet adapters. As such, the **no** option, *ifsize* (default 8), needs to be increased to include not only the Ethernet interfaces for each adapter, but also any VLAN logical devices that are configured. See the **no** command documentation.

Each VLAN can have a different maximum transmission unit (MTU) value even if sharing a single physical Ethernet adapter.

VLAN support is managed through SMIT. Type the **smit vlan** fast path from the command line and make your selection from the main VLAN menu. Online help is available.

After you configure VLAN, configure the IP interface, for example, en1 for standard Ethernet or et1 for IEEE 802.3, using Web-based System Manager, SMIT, or commands.

Notes:

- If you try to configure a VLAN ID value that is already in use for the specified adapter, the configuration fails with the following error:

```
Method error (/usr/lib/methods/chgvlan):
  0514-018 The values specified for the following attributes
           are not valid:
           vlan_tag_id  VLAN Tag ID
```

- If a user (for example, IP interface) is currently using the VLAN logical device, any attempt to remove the VLAN logical device fails. A message similar to the following displays:

```
Method error (/usr/lib/methods/ucfgcommo):
  0514-062 Cannot perform the requested function because the
           specified device is busy.
```

To remove the logical VLAN device, first detach the user. For example, if the user is IP interface en1, then you can use the following command:

```
ifconfig en1 detach
```

Then remove the network interface using the SMIT TCP/IP menus.

- If a user (for example, IP interface) is currently using the VLAN logical device, any attempt to change the VLAN characteristic (VLAN tag ID or base adapter) fails. A message similar to the following displays:

```
Method error (/usr/lib/methods/chgvlan):
  0514-062 Cannot perform the requested function because the
           specified device is busy.
```

To change the logical VLAN device, first detach the user. For example, if the user is the IP interface en1, you could use the following command:

```
ifconfig en1 detach
```

Then change the VLAN and add the network interface again using the SMIT TCP/IP menus.

Troubleshooting

tcpdump and **trace** can be used to troubleshoot the VLAN. The trace hook ID for each type of transmit packet follows:

transmit packets	3FD
receive packets	3FE
other events	3FF

The **entstat** command gives the aggregate statistics of the physical adapter for which the VLAN is configured. It does *not* provide the individual statistics for that particular VLAN logical device.

Restrictions

Remote dump is not supported over a VLAN. Also, VLAN logical devices cannot be used to create a Cisco Systems' Etherchannel.

Using ATM Adapters

Asynchronous Transfer Mode (ATM) is an international standard that defines a high-speed networking method to transport any mixture of voice, video, and traditional computer data across local, municipal, and wide-area networks (LANs, MANs, and WANs). ATM adapters provide full-duplex connectivity for RS/6000 servers or clients using permanent virtual circuits (PVCs) and switched virtual circuits (SVCs). The PVC and SVC implementations are designed to be compliant with the ATM Forum specifications. The maximum number of virtual circuits supported depends on the adapter. Most adapters support at least 1024 virtual circuits.

ATM Technology

Asynchronous Transfer Mode (ATM) is a cell-switching, connection-oriented technology. In ATM networks, end stations attach to the network using dedicated full duplex connections. The ATM networks are constructed using switches, and switches are interconnected using dedicated physical connections. Before any data transfers can begin, end-to-end connections must be established. Multiple connections can and do exist on a single physical interface. Sending stations transmit data by segmenting Protocol Data Units (PDUs) into 53-byte cells. Payload stays in the form of cells during network transport. Receiving stations reassemble cells into PDUs. The connections are identified using a virtual path identifier (VPI) and a virtual channel identifier (VCI). The VPI field occupies one byte in the ATM cell five-byte header; whereas, the VCI field occupies two bytes in the ATM cell five-byte header. Basically, a VPI:VCI pair identifies the source of the ATM cell. The function of the ATM switch is to recognize the source of the cell, determine the next hop, and output the cell to a port. The VPI:VCI changes on a hop-by-hop basis. Thus, VPI:VCI values are not universal. Each virtual circuit is described as a concatenation of VPI:VCI values across the network.

ATM Connections

ATM architecture has two kinds of virtual circuits: permanent (PVCs) and switched (SVCs).

Permanent Virtual Circuits

PVCs are statically and manually configured. The switches forming the ATM network must first be set up to recognize the VPI:VCI combination of each endpoint and to route the endpoint ATM cells to the destination endpoint through the ATM network. Once a link connection through the network has been established from one endpoint to another, ATM cells can be transmitted through the ATM network and ATM switches. The network switches translate the VPI:VCI values in the appropriate way so as to route the cell to its destination.

Switched Virtual Circuits

SVCs are dynamically set up on an as needed basis. The ATM end stations are assigned 20-byte addresses. SVCs use a control plane and a data plane.

The control plane uses a signaling channel VPI:VCI 0:5.

SVCs involve on demand call setup, whereby an ATM station sends information elements specifying the destination ATM address (and optionally, the source ATM address). In general, calling station, network, and called station participate in a negotiation. Finally, a call is either accepted or rejected. If a call is accepted, network assigns VPI:VCI values for the data plane to the calling station and called station. In the control plane, the ATM network routes (or switches) signaling packets on the basis of the ATM addresses. While these packets are being routed, the switches set up data plane cell routing tables. In the data plane, ATM networks switch cells on the basis of VPI:VCI much like in the case of PVCs. When data transfer is over, connection is terminated.

The ATM address is constructed by registering with the ATM network and by acquiring the most significant 13 bytes. The next six bytes contain the adapter's factory-assigned, unique MAC address. The least significant byte is the selector. Use of this byte is left to the discretion of the end station. ATM networks do not interpret this byte.

TCP/IP over ATM

The *Internet Engineering Task Force RFC1577: Classical IP and ARP over ATM* standard specifies the mechanism for implementing Internet Protocol (IP) over ATM. Since ATM is connection-oriented technology and IP is a datagram-oriented technology, mapping the IP over ATM is not trivial.

In general, the ATM network is divided into logical IP subnetworks (LISs). Each LIS is comprised of some number of ATM stations. LISs are analogous to traditional LAN segments. LISs are interconnected using routers. A particular adapter (on an ATM station) can be part of multiple LISs. This feature can be very useful for implementing routers.

RFC1577 specifies RFC1483, which specifies logical link control/Sub-Network Access Protocol (LLC/SNAP) encapsulation as the default. In PVC networks for each IP station, all PVCs must be manually defined by configuring VPI:VCI values. If LLC/SNAP encapsulation is not being used, the destination IP address associated with each VPI:VCI must be defined. If LLC/SNAP encapsulation is being used, the IP station can learn the remote IP address by an InARP mechanism.

For SVC networks, RFC1577 specifies an ARP server per LIS. The purpose of the ARP server is to resolve IP addresses into ATM addresses without using broadcasts. Each IP station is configured with the ATM address of the ARP server. IP stations set up SVCs with the ARP server, which in turn, sends InARP requests to the IP stations. Based on InARP reply, an ARP server sets up IP to ATM address maps. IP stations send ARP packets to the ARP server to resolve addresses, which returns ATM addresses. IP stations then set up a SVC to the destination station and data transfer begins. The ARP entries in IP stations and the ARP server age based on a well defined mechanism. For both the PVC and SVC environments, each IP station has at least one virtual circuit per destination address.

The Internet Engineering Task Force RFC2225 adds the support of ATM ARP Request Address list to RFC1577. The ATM ARP Request Address list is a list containing one or more ATM addresses of individual ATM ARP servers located within the LIS. The RFC2225 client eliminates the single point of failure associated with the 1577 clients' ATM ARP services. The 2225 clients have the ability to switch to backup ARP servers when the current ATM ARP server fails.

RS/6000 sets the first entry in the ATM ARP Request Address list as the Primary ATM ARP server and the rest of the entries as Secondary ATM ARP servers.

The client will always try to use the Primary ATM ARP server. If the effort to connect to the Primary ATM ARP server fails, the client tries to connect to the first Secondary server (the position in the ATM ARP Request Address list determines the order of the Secondary ATM ARP server). If the connection to the first Secondary ATM ARP server fails, the client tries to contact the next Secondary ATM ARP server in the list. This process continues until the connection is successful.

If the connection to the Primary ATM ARP server fails, regardless of which Secondary ATM ARP server it is connected to or attempting to connect to, the client continues to retry the Primary ATM ARP server every 15 minutes. If it finally connects to the Primary ATM ARP server, then the connection to the current Secondary ATM ARP server is dropped.

The ATM ARP Request Address list is entered manually either through SMIT or by using the **ifconfig** command. The ATM ARP Request Address list cannot be configured with the Management Information Base (MIB).

PVC Network: Use the "Representative ATM Network" figure as an example to configure your network.

Within the "Representative ATM Network" figure, one logical IP subnet is represented by dashed lines from each host to the switch. The other IP subnet is represented by solid lines from each host to the switch.

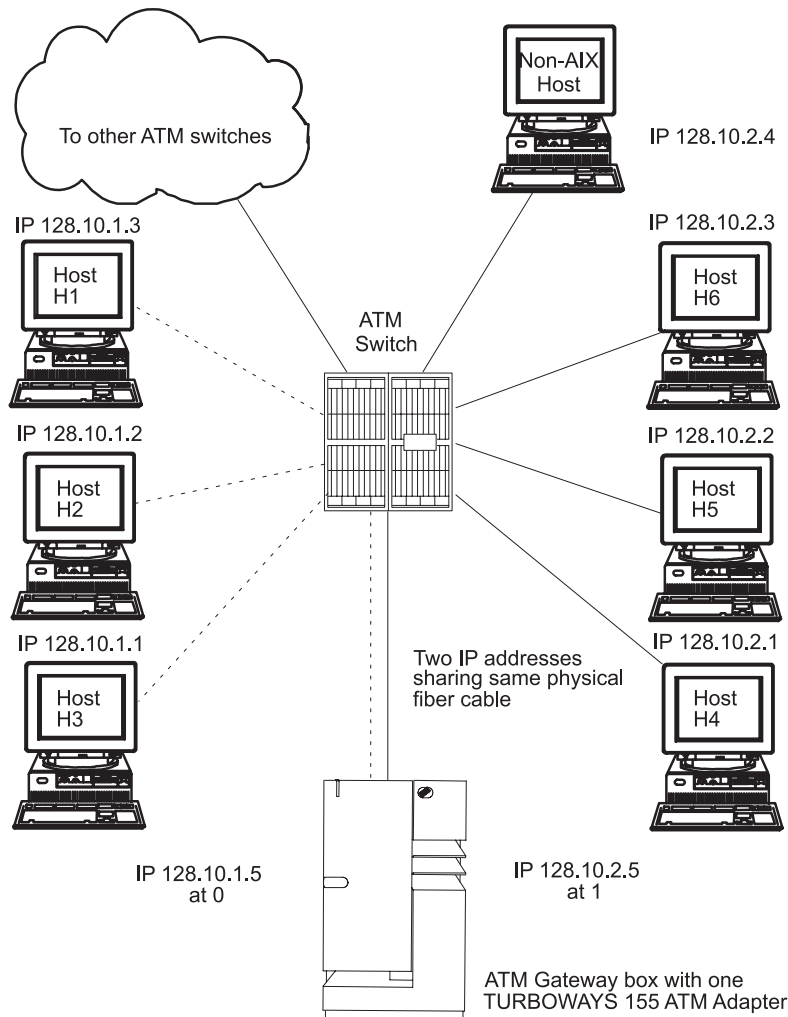


Figure 14. Representative ATM Network. This illustration depicts an ATM network laid out in a typical star topology. In the center of the star is the ATM switch. Numbered IP hosts are branched off of the switch as are links to other ATM switches and one ATM gateway box and adapter.

The following Representative Host Configuration table indicates how hosts H3 and H4 are configured to communicate with a gateway and with each host on its own logical IP subnet.

Representative Host Configuration		
Network Interface Driver	VPI:VCI	Comment
Host H3		
at0	0:40	Connection to 128.10.1.5 (gateway)
at0	0:42	Connection to 128.10.1.2
at0	0:43	Connection to 128.10.1.3
Host H4		
at0	0:50	Connection to 128.10.2.5 (gateway)
at0	0:52	Connection to 128.10.2.2
at0	0:53	Connection to 128.10.2.3
at0	0:54	Connection to 128.10.2.4

To reach hosts on another logical IP subnet, only a VPI:VCI connection to the gateway needs to be created. (The VPI:VCIs are for illustration purposes only.)

The ATM gateway box has one ATM with two IP addresses sharing the same physical cable.

SVC Network: Using the "Representative ATM Network" figure as an example, imagine that host H3 wants to call H4. H1 is the ARP server for subnet 1 and H6 is the ARP server for subnet 2. Assuming a subnet mask of 255.255.255.0, stations with addresses of 128.10.1.X are members of one subnet, whereas stations with addresses of 128.10.2.X are members of a second subnet. See the following list of representative host configurations using SVCs.

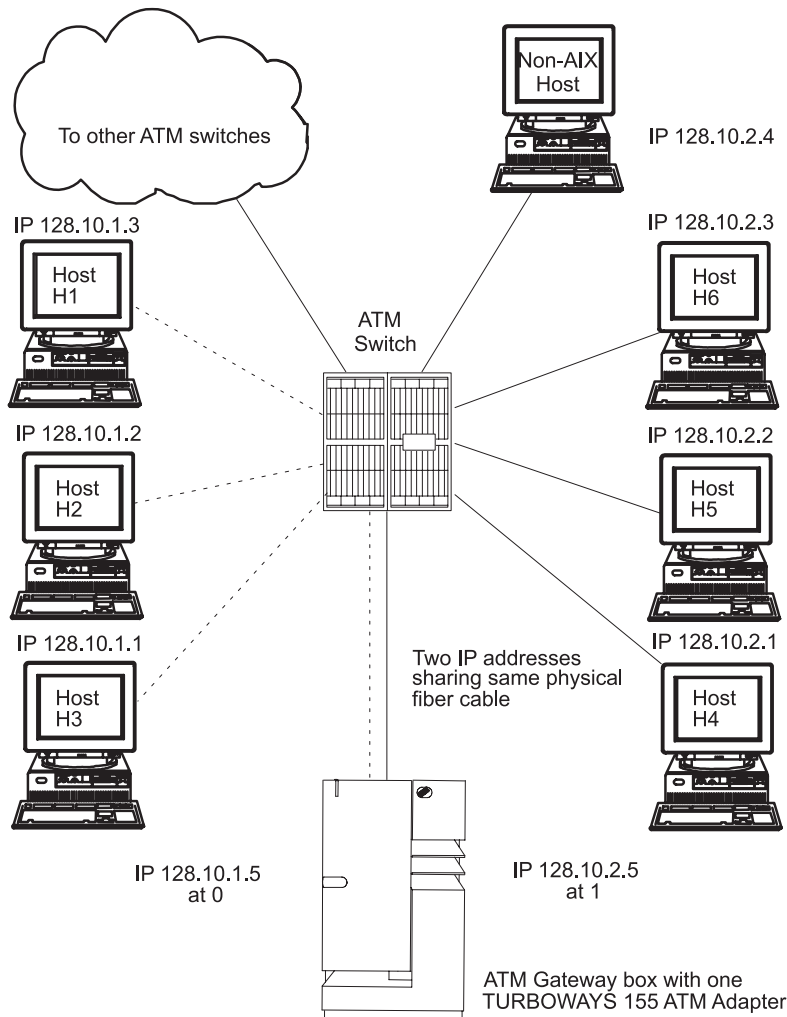


Figure 15. Representative ATM Network. This illustration depicts an ATM network laid out in a typical star topography. In the center of the star is the ATM switch. Numbered IP hosts are branched off of the switch as are links to other ATM switches and one ATM gateway box and adapter.

List of Representative Host Configurations				
Network Interface Driver	IP Address	ARP Server	ARP Server Address	Gateway Address
Host H1				
at0	128.10.1.3	Yes		128.10.1.5
Host H3				

at0	128.10.1.1	No	ATM address of H1	128.10.1.5
Gateway				
at0	128.10.1.5	No	ATM address of H1	
at1	128.10.2.5	No	ATM address of H6	
Host H4				
at0	128.10.2.1	No	ATM address of H6	128.10.2.5
Host H6				
at0	128.10.2.3	Yes		128.10.2.5

Note: Each subnet requires one and only one ARP server.

Because H3 recognizes that address 128.10.2.1 is not on its subnet, H3 consults H1 to resolve the default gateway IP address to an ATM address. H3 then places a call to the gateway. The gateway recognizes that the data is bound for the second subnet and consults H6 to successfully resolve the H4 IP address to an ATM address. Connections are then established between H3 and the gateway and between the gateway and H4.

Configuring an ATM Adapter

To configure your ATM adapter, use the Web-based System Manager, **wsm**, or the SMIT fast path **smit chg_atm**. Select an adapter name, then use the online help and multiple-choice lists to decide which changes to make for your configuration.

ATM Adapter Statistics

The **atmstat** command can be used for getting ATM adapter statistics. Using the **atmstat** command with the **-r** flag resets the statistics. The format of the command is **atmstat DeviceName**. This command returns the following sets of statistics:

Transmit Statistics:

Packets:

This field contains the number of packets (or PDUs) transmitted.

Bytes: This field contains the count of bytes transmitted. These are the user bytes. The ATM overhead, for example, ATM cell header, and AAL 5 PDU trailer, are excluded.

Interrupts:

This field is not used.

Transmit Errors:

This field contains the number of transmit errors for this device.

Packets Dropped:

This field contains the number of Transmit Packets that were dropped, for instance, because of an out of buffers condition.

Max Packets on S/W Transmit Queue:

This field does not apply to ATM.

S/W Transmit Queue Overflow:

This field does not apply to ATM.

Current S/W + H/W Transmit Queue Length:

This field contains the current transmit queue length.

Cells Transmitted:

This field contains the number of cells transmitted by this device.

Out of Xmit Buffers:

This field contains the number of packets dropped because of out of xmit buffers condition.

Current HW Transmit Queue Length:

This field contains the current number of transmit packets on the hardware queue.

Current SW Transmit Queue Length:

This field does not apply to ATM.

Receive Statistics:**Packets:**

This field contains the number of packets (or PDUs) received.

Bytes: This field contains the count of bytes received. These are the user bytes. The ATM overhead, for example, ATM cell header and AAL 5 PDU trailer are excluded.

Interrupts:

This field contains the number of Interrupts taken by the system for the adapter-to-system indications. Some of the events that cause these interrupts are packet received, transmit done indication, and so on.

Receive Errors:

This field contains the number of receive errors for this device.

Packets Dropped:

This field contains the number of received packets dropped, for instance, due to out of buffers condition.

Bad Packets:

This field does not apply to ATM.

Cells Received:

This field contains the number of cells received by this device.

Out of Rcv Buffers:

This field contains the number of packets dropped because of out of receive buffers condition.

CRC Errors:

This field contains the number of received packets that encountered CRC errors.

Packets Too Long:

This field contains the number of received packets that exceeded the maximum PDU size.

Incomplete Packets:

This field contains the number of incomplete received packets.

Cells Dropped:

This field contains the number of dropped cells. Cells could be dropped for a number of reasons, such as bad header error control (HEC), out of buffer condition, and so on.

General Statistics:**No mbuf Errors:**

This field contains the number of mbuf requests that were denied.

Adapter Loss of Signals:

This field contains the number of times the adapter encountered loss of signal.

Adapter Reset Count:

This field contains the number of times the adapter has been reset.

Driver Flags: Up Running Simplex

This field contains the neighborhood discovery daemon (NDD) flags.

Virtual Connections in use:

This field contains the number of virtual connections that are currently allocated or in use.

Max Virtual Connections in use:

This field contains the maximum number of virtual connections allocated since the last reset of the statistics.

Virtual Connections Overflow:

This field contains the number of allocate virtual connections requests that have been denied.

SVC UNI Version:

This field contains the current UNI version of the signaling protocol in use.

Additional Microchannel ATM Statistics

Using the **atmstat** command with the **-d** flag provides detailed statistics.

Packets Dropped - No small direct memory access (DMA) buffer:

This field contains the number of received packets dropped because the adapter did not have small system buffers for DMA.

Packets Dropped - No medium DMA buffer:

This field contains the number of received packets dropped because the adapter did not have medium system buffers for DMA.

Packets Dropped - No large DMA buffer:

This field contains the number of received packets dropped because the adapter did not have large system buffers for DMA.

Receive Aborted - No Adapter Receive buffer:

This field contains the number of received packets aborted because the adapter did not have on-card receive buffers.

Transmit Aborted - No small DMA buffer:

This field contains the number of transmit packets dropped because of the lack of small system buffers for DMA.

Transmit Aborted - No medium DMA buffer:

This field contains the number of transmit packets dropped because of the lack of medium system buffers for DMA.

Transmit Aborted - No large DMA buffer:

This field contains the number of transmit packets dropped because of the lack of large system buffers for DMA.

Transmit Aborted - No MTB DMA buffer:

This field contains the number of transmit packets dropped because of the lack of large system buffers for DMA.

Transmit Aborted - No Adapter Transmit buffer:

This field contains the number of transmit packets dropped because of the lack of adapter on-card transmit buffers.

Max Hardware Transmit Queue Length:

This field contains the maximum number of transmit packets queued in the hardware queue.

Small Mbufs in Use:

This field contains the number of small mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

Medium Mbufs in Use:

This field contains the number of medium mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

Large Mbufs in Use:

This field contains the number of large mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

Huge Mbufs in Use:

This field contains the number of huge mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by the system administrators. This information can be used to tune the configuration information.

MTB Mbufs in Use:

This field contains the number of MTB mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by the system administrators. This information can be used to tune the configuration information.

Max Small Mbufs in Use:

This field contains the maximum number of small mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by the system administrators. This information can be used to tune the configuration information.

Max Medium Mbufs in Use:

This field contains the maximum number of medium mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

Max Large Mbufs in Use:

This field contains the maximum number of large mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

Max Huge Mbufs in Use:

This field contains the maximum number of huge mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

MTB Mbufs in Use:

This field contains the maximum number of MTB mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

Small Mbufs overflow:

This field contains the number of times that a small mbuf could not be allocated. This information can be used to tune the configuration information.

Medium Mbufs overflow:

This field contains the number of times that a medium mbuf could not be allocated. This information can be used to tune the configuration information.

Large Mbufs overflow:

This field contains the number of times that a large mbuf could not be allocated. This information can be used to tune the configuration information.

Huge Mbufs overflow:

This field contains the number of times that a huge mbuf could not be allocated. This information can be used to tune the configuration information.

MTB Mbufs overflow:

This field contains the number of times that an MTB mbuf could not be allocated. This information can be used to tune the configuration information.

PCI ATM Adapter Specific Statistics

Total 4K byte Receive Buffers: 768 Using: 512

This message contains the number of receive buffers allocated as well as the number that are currently in use.

Max 4K byte Receive Buffers limit: 1228 max_used: 514

This message contains the maximum number of receive buffers that can be allocated as well as the number that have been used since the adapter was last configured or opened.

TCP/IP Network Interfaces

The TCP/IP Network Interface layer formats IP datagrams at the Network layer into packets that specific network technologies can understand and transmit. A network interface is the network-specific software that communicates with the network-specific device driver and the IP layer in order to provide the IP layer with a consistent interface to all network adapters that might be present.

The IP layer selects the appropriate network interface based on the destination address of the packet to be transmitted. Each network interface has a network address. The Network Interface layer is responsible for adding or removing any link layer protocol header required to deliver a message to its destination. The **network adapter** device driver controls the network adapter card.

Although not required, a network interface is usually associated with a network adapter. For instance, the loopback interface has no network adapter associated with it. A machine must have one network adapter card for each network (not network type) to which it connects. However, a machine requires only one copy of the network interface software for each network adapter it uses. For instance, if a host attaches to two token-ring networks, it must have two network adapter cards. However, only one copy of the **token-ring** network interface software and one copy of the token-ring device driver is required.

TCP/IP supports types of network interfaces:

- Standard Ethernet Version 2 (en)
- IEEE 802.3 (et)
- Token-ring (tr)
- Serial Line Internet Protocol (SLIP)
- Loopback (lo)
- FDDI
- Serial Optical (so)
- ATM (at)
- **Point-to-Point Protocol** (PPP)
- Virtual IP Address (vi)

The Ethernet, 802.3, and token-ring interfaces are for use with local area networks (LANs). The SLIP interface is for use with serial connections. The loopback interface is used by a host to send messages back to itself. The Serial Optical interface is for use with optical point-to-point networks using the Serial Optical Link device handler. The ATM interface is for use with 100 Mb/s and 155 Mb/s ATM connections. Point to Point protocol is most often used when connecting to another computer or network via a modem. The Virtual IP Address interface (also called *virtual interface*) is not associated with any particular network adapter. Multiple instances of a virtual interface can be configured on a host. When virtual interfaces are configured, the address of the first virtual interface becomes the source address unless an application has chosen a different interface. Processes that use a virtual IP address as their source address can send packets through any network interface that provides the best route for that destination. Incoming packets destined for a virtual IP address are delivered to the process regardless of the interface through which they arrive.

Automatic Configuration of Network Interfaces

When a new network adapter is physically installed in the system, the operating system automatically adds the appropriate network interface for that adapter. For example, if you install a token-ring adapter in your system, the operating system assigns it the name `tok0` and add a token-ring network interface named `tr0`. If you install an Ethernet adapter in your system, the operating system assigns it the name `ent0` and add both an Ethernet Version 2 and an IEEE 802.3 interface, named `en0` and `et0` respectively.

In most cases, there is a one-to-one correspondence between adapter names and network interface names. For example, token-ring adapter `tok0` corresponds to interface `tr0`, adapter `tok1` corresponds to interface `tr1`, and so on. Similarly, Ethernet adapter `ent0` corresponds to interface `en0` (for Ethernet Version 2) and `et0` (for IEEE 802.3), and adapter `ent1` corresponds to interface `en1` (for Ethernet Version 2) and `et1` (for IEEE 802.3).

In the case of ATM, according to RFC1577, it is possible for an ATM station to be part of multiple Logical IP Subnetworks. In this case, multiple interfaces are associated with a device. This requires that an interface be specifically added and a device name be assigned to it.

Note: Under normal circumstances, you do not need to delete or add a network interface manually. However, some problem determination procedures might require you to do so. In this case, use the Web-based System Manager **wsm**, or the SMIT fast path, **smit inet**, to delete and re-add the appropriate interface.

At each system startup, the operating system automatically configures the network interface software based upon the information in the ODM database. Initially, the network interface is configured with default values. In order to communicate through a given network interface, the Internet address must be set. This is the only attribute that you need to set. All other necessary attributes can use the default values. The default values for each network interface follow.

Ethernet Default Configuration Values

The following is a list of valid Ethernet network adapter attributes along with their default values, which can be changed using the Web-based System Manager, **wsm** or the Network Interface Selection menu in SMIT.

Attribute	Default Value	Possible Values
<code>netaddr</code>		
<code>state</code>	<code>down</code>	<code>up</code> , <code>down</code> , <code>detach</code>
<code>arp</code>	<code>yes</code>	<code>yes</code> , <code>no</code>
<code>netmask</code>		
<code>broadcast</code>		

The following valid Ethernet network device driver attribute is shown along with its default values, which can be changed using the Web-based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default Value	Possible Values
<code>mtu</code>	<code>1500</code>	<code>60</code> through <code>1500</code>

802.3 Default Configuration Values

The following is a list of valid 802.3 network adapter attributes along with their default values, which can be changed using the Web-based System Manager, **wsm**, or the Network Interface Selection menu in

SMIT.

Attribute	Default Value	Possible Values
netaddr		
state	down	up, down, detach
arp	yes	yes, no
netmask		
broadcast		

The following valid 802.3 network device driver attribute is shown along with its default values, which can be changed using the Web-based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default Value	Possible Values
mtu	1492	60 through 1492

Token-Ring Default Configuration Values

The following is a list of valid token-ring network adapter attributes along with their default values, which can be changed using the Web-based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	Default Value	Possible Values
netaddr		
netmask		
state	down	up, down, detach
arp	yes	yes, no
hwloop	no	yes, no
netmask		
broadcast		
allcast	no	yes, no

The following valid token-ring network device driver attributes are shown along with its default values, which may be changed using the Web-based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default Value	Possible Values
mtu (4Mbps)	1500	60 through 4056
mtu (16Mbps)	1500	60 through 17960

Note: When operating through a bridge, the default value of 1500 for the maximum transmission unit (MTU) should be changed to a value that is 8 less than the maximum information field (maximum I-frame) advertised by the bridge in the routing control field. For example, if the maximum I-frame value is 1500 in the routing control field, the MTU size should be set to 1492. This is for token-ring network interfaces only. For more information, see Problems with a Token-Ring/Token-Ring Bridge.

When using the IBM 16/4 POWER-based token-ring adapter (ISA), the MTU is restricted to 2000.

SLIP Default Configuration Values

The following is a list of valid SLIP network adapter attributes along with their default values as shown under the Web-based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	Default Value	Possible Values
netaddr		
dest		
state	up	up, down, detach
netmask		

The following valid SLIP network device driver attribute is shown along with its default values as displayed under the Web-based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default Value	Possible Values
mtu	1006	60 through 4096

Serial Optical Default Configuration Values

The following is a list of valid Serial Optical network channel converter attributes along with their default values as shown under the Web-based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	Default Value	Possible Values
netaddr		
state	down	up, down, detach
netmask		

The following valid serial optical network device handler attribute is shown along with its default values as displayed under the Web-based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default Value	Possible Values
mtu	61428	1 through 61428

ATM Default Configuration Values

The following is a list of valid ATM network adapter attributes along with their default values as shown under the Web-based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	Default Value	Possible Values
netaddr		
netmask		
state	up	up, down, detach
Connection Type	svc_s	svc_c, svc_s, pvc
ATM Server Address		
Alternate Device		
idle timer	60	1 through 60
Best Effort Bit Rate (UBR) in kbits/sec	0	1 through 155,000

The following valid ATM network device driver attribute is shown along with its default values as displayed under the Web-based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default Value	Possible Values
mtu	9180	1 through 64K

Note: Network Administrators need to exercise caution while changing MTU size from default to some other value. The MTU size needs to be coordinated with other stations on the network.

If PVCs are to be used on an interface, the VPI:VCIs needs to be defined. This is performed through the Network Interface Selection Menu. The PVCs for IP over ATM Network option on this menu is used to list, add, change, or remove PVCs.

Implications of Multiple Network Interfaces on the Same Network

While it is possible to have more than one interface on the same network, in general this is not recommended for two reasons:

1. Having two interfaces on the same network is a violation of TCP/IP architecture.
In TCP/IP architecture, a host machine with two network adapters is defined as an IP router. Different network adapters must be attached to different physical networks. In the case of token-ring, TCP/IP addresses multiple rings bridged together as a single logical ring (as if it were a single physical ring).
2. Having two interfaces on the same network can cause broadcast storms.
Whenever an IP host sees traffic for a network whose IP address is different from its own network, it generates an Internet Control Message Protocol (ICMP) packet announcing this conflict. Since every host on the network sees the traffic that is misaddressed, every host generates ICMP packets. If the amount of misaddressed traffic is significant, the ICMP traffic can grow to the point that network performance degrades.

It is possible to avoid the broadcast storms created when multiple interfaces are connected to the same network by giving each interface a different IP addresses.

Managing Network Interfaces

To manage network interfaces, use the web-based system manager, WSM Network, FastPath (application) or the tasks in the following table.

Managing Network Interfaces Tasks			
Task	SMIT Fast Path	Command or File	Web-based System Manager Management Environment
List all network devices	smit lsinet	lsdev -C -c if	Software → Devices → All Devices .
Configure a network device	smit chinnet	See the ifconfig command and the rc.net file	Software → Network → TCPIP (IPv4 and IPv6) → Protocol Configuration → Set up basic TCP/IP configuration .

Managing Network Interfaces Tasks			
Changing network interface info with remotely mounted /usr	smit chdev ^{1,2}	chgif ^{1,2}	Software → Network → TCPIP (IPv4 and IPv6) → Network Interfaces →. Right-click and select Properties → Aliases .
Obtaining statistics for a network interface		netstat -v	Software → Network → TCPIP (IPv4 and IPv6) → Network Interfaces → Network Statistics .

Notes:

1. Changes from a remotely mounted **/usr** affect only the Information Database (ODM) until the network is restarted or until the **ifconfig** command is used to make the changes take effect right away.
2. When using a remotely mounted **/usr**, be careful not to modify the interface being used, because that is the location of the libraries, commands, and kernel.

Interface-Specific Network Options

TCP/IP interfaces must be specially tuned to achieve good, high-speed network performance (100 Mb or more). This effort is complicated by the fact that multiple network interfaces and a combination of traditional and high-speed TCP/IP interfaces can be used on a single system. Before AIX 4.3.3 (4330-08) and AIX 5.1, AIX provided a single set of system-wide values for the key IP interface network tuning parameters making it impossible to tune a system that has widely differing network adapter interfaces. Beginning with AIX 4.3.3 (4330-08) and AIX 5.1, Interface Specific Network Options (ISNO) allows system administrators to tune each TCP/IP interface individually for best performance.

There are five ISNO parameters for each supported interface: **rfc1323**, **tcp_nodelay**, **tcp_sendspace**, **tcp_recvspace**, and **tcp_mssdflt**. When set, the values for these parameters override the system-wide parameters of the same names that had been set with the **no** command. When ISNO options are not set for a particular interface, system-wide options are used. When options have been set by an application for a particular socket using the **setsockopt** subroutine, such options override the ISNOs.

The network option **use_isno**, set with the **no** command, must have a value of 1 for the ISNOs to take effect. The default value for **use_isno** is 1.

Some high-speed adapters have ISNO parameters set by default in the ODM database.

Gigabit Ethernet interfaces, when configured to use an MTU of 9000, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330-08) Value	AIX 5.1 (and later) Value
tcp_sendspace	131072	262144	262144
tcp_recvspace	92160	131072	131072
rfc1323	1	1	1

Gigabit Ethernet interfaces, when configured to use an MTU of 1500, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330-08) Value	AIX 5.1 (and later) Value
tcp_sendspace	65536	131072	131072

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330-08) Value	AIX 5.1 (and later) Value
tcp_recvspace	16384	65536	65536
rfc1323	0	not set	not set

ATM interfaces, when configured to use an MTU of 1500, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330-08) Value	AIX 5.1 (and later) Value
tcp_sendspace	16384	not set	not set
tcp_recvspace	16384	not set	not set
rfc1323	0	not set	not set
tcp_nodelay	0	not set	not set
tcp_mssdflt	512	not set	not set

ATM interfaces, when configured to use an MTU of 65527, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330-08) Value	AIX 5.1 (and later) Value
tcp_sendspace	655360	655360	655360
tcp_recvspace	655360	655360	655360
rfc1323	0	1	1
tcp_nodelay	0	not set	not set
tcp_mssdflt	512	not set	not set

ATM interfaces, when configured to use an MTU of 9180, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330-08) Value	AIX 5.1 (and later) Value
tcp_sendspace	65536	65536	65536
tcp_recvspace	65536	65536	65536
rfc1323	0	not set	not set
tcp_nodelay	0	not set	not set
tcp_mssdflt	512	not set	not set

FDDI interfaces, when configured to use an MTU of 4352, use the following ISNO values by default:

Name	Value
tcp_sendspace	45046
tcp_recvspace	45046

The ISNO parameters cannot be displayed or changed using SMIT. They can be set using the **chdev** command or the **ifconfig** command. The **ifconfig** command changes the values only until the next reboot. The **chdev** command changes the values in the ODM database so they are used on subsequent reboots. The **lsattr** or **ifconfig** commands can be used to display the current values.

Example

The following commands can be used first to verify system and interface support and then to set and verify the new values.

1. Verify general system and interface support using the **no** and **lsattr** commands.

- Ensure the **use_isno** option is enabled using a command similar to the the following:

```
$ no -a | grep isno
      use_isno=1
```

- Ensure the interface supports the five new ISNOs using the **lsattr -EI** command, as shown in the following:

```
$ lsattr -E -l en0 -H
      attribute  value  description
      rfc1323           N/A
      tcp_nodelay       N/A
      tcp_sendspace    N/A
      tcp_recvspace    N/A
      tcp_msdf1t       N/A
```

2. Set the interface specific values, using either the **ifconfig** or **chdev** command. The **ifconfig** command sets values temporarily, which is recommended for testing. The **chdev** command alters the ODM, so customized values remain valid after reboot.

- Set the **tcp_recvspace** and **tcp_sendspace** to 64K and enable **tcp_nodelay** by using one of the following:

```
$ ifconfig en0 tcp_recvspace 65536 tcp_sendspace 65536 tcp_nodelay 1
$ chdev -l en0 -a tcp_recvspace=65536 -a tcp_sendspace=65536 -a tcp_nodelay=1
```

- Alternatively, assuming the **no** command reports an **rfc1323=1** global value, the root user can turn **rfc1323** off for all connections over en0 with the following commands:

```
$ ifconfig en0 rfc1323 0
$ chdev -l en0 -a rfc1323=0
```

3. Verify the settings using the **ifconfig** or **lsattr** command, as shown in the following example:

```
$ ifconfig en0 <UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT>
      en0: flags=e080863
            inet 9.19.161.100 netmask 0xfffff00 broadcast 9.19.161.255
            tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1 rfc1323 0
$ lsattr -EI en0
      rfc1323           0           N/A           True
      tcp_nodelay       1           N/A           True
      tcp_sendspace    65536       N/A           True
      tcp_recvspace    65536       N/A           True
      tcp_msdf1t       N/A         N/A           True
```

TCP/IP Addressing

TCP/IP includes an Internet addressing scheme that allows users and applications to identify a specific network or host with which to communicate. An Internet address works like a postal address, allowing data to be routed to the chosen destination. TCP/IP provides standards for assigning addresses to networks, subnetworks, hosts, and sockets, and for using special addresses for broadcasts and local loopback.

Internet addresses are made up of a network address and a host (or local) address. This two-part address allows a sender to specify the network as well as a specific host on the network. A unique, official network address is assigned to each network when it connects to other Internet networks. However, if a local network is not going to connect to other Internet networks, it can be assigned any network address that is convenient for local use.

The Internet addressing scheme consists of Internet Protocol (IP) addresses and two special cases of IP addresses: broadcast addresses and loopback addresses.

Internet Addresses

The Internet Protocol (IP) uses a 32-bit, two-part address field. The 32 bits are divided into four *octets* as in the following:

```
01111101 00001101 01001001 00001111
```

These binary numbers translate into:

125 13 73 15

The two parts of an Internet address are the network address portion and the host address portion. This allows a remote host to specify both the remote network and the host on the remote network when sending information. By convention, a host number of 0 is used to refer to the network itself.


TCP/IP supports three classes of Internet addresses: Class A, Class B, and Class C. The different classes of Internet addresses are designated by how the 32 bits of the address are allocated. The particular address class a network is assigned depends on the size of the network.

Class A Addresses

A Class A address consists of an 8-bit network address and a 24-bit local or host address. The first bit in the network address is dedicated to indicating the network class, leaving 7 bits for the actual network address. Because the highest number that 7 bits can represent in binary is 128, there are 128 possible Class A network addresses. Of the 128 possible network addresses, two are reserved for special cases: the network address 127 is reserved for local loopback addresses, and a network address of all ones indicates a broadcast address.

There are 126 possible Class A network addresses and 16,777,216 possible local host addresses. In a Class A address, the highest order bit is set to 0.

Network Address (8 bits)	Local Host Address (24 bits)		
01111101	00001101	01001001	00001111



Note: The high-order bit (or first bit) will always be 0 in a Class A address.

Figure 16. Class A Address. This illustration shows a typical class A address structure. The first 8 bits contain the network address (always beginning with a zero). The remaining 24 bits contain the local host address.

The first octet of a Class A address is in the range 1 to 126.

Class B Addresses

A Class B address consists of a 16-bit network address and a 16-bit local or host address. The first two bits in the network address are dedicated to indicating the network class, leaving 14 bits for the actual network address. There are 16,384 possible network addresses and 65,536 local host addresses. In a Class B address, the highest order bits are set to 1 and 0.

Network Address (16 bits)		Local Host Address (16 bits)	
10011101	00001101	01001001	00001111



Note: The two highest order bits (or first two bits) will always be 1 and 0 in a Class B address.

Figure 17. Class B Address. This illustration shows a typical class B address structure. The first 16 bits contain the network address. The two highest order bits will always be a one and a zero. The remaining 16 bits contain the local host address.

The first octet of a Class B address is in the range 128 to 191.

Class C Addresses

A Class C address consists of a 24-bit network address and an 8-bit local host address. The first two bits in the network address are dedicated to indicating the network class, leaving 22 bits for the actual network address. Therefore, there are 2,097,152 possible network addresses and 256 possible local host addresses. In a Class C address, the highest order bits are set to 1 and 1.

Network Address (24 bits)			Local Host Address (8 bits)
11011101	00001101	01001001	00001111



Note: The two highest order bits (or first two bits) will always be 1 and 1 in a Class C address.

Figure 18. Class C Address. This illustration shows a typical class C address structure. The first 24 bits contain the network address (the two highest order bits will always be a one and a one). The remaining 8 bits contain the local host address.

In other words, the first octet of a Class C address is in the range 192 to 223.

When deciding which network address class to use, you must consider how many local hosts there will be on the network and how many subnetworks will be in the organization. If the organization is small and the network will have fewer than 256 hosts, a Class C address is probably sufficient. If the organization is large, then a Class B or Class A address might be more appropriate.

Note: Class D (1-1-1-0 in the highest order bits) addresses provide for multicast addresses and are supported by UDP/IP under this operating system.

Machines read addresses in binary code. The conventional notation for Internet host addresses is the *dotted decimal*, which divides the 32-bit address into four 8-bit fields. The following binary value:

0001010 00000010 00000000 00110100

can be expressed as:

010.002.000.052 or 10.2.0.52

where the value of each field is specified as a decimal number and the fields are separated by periods.

Note: The **hostent** command does recognize the following addresses: .08, .008, .09, and .009. Addresses with leading zeros are interpreted as octal, and numerals in octal cannot contain 8s or 9s.

TCP/IP requires a unique Internet address for each network interface (adapter) on a network. These addresses are determined by entries in the configuration database, which must agree with entries in the **/etc/hosts** file or the **named** database if the network is using a name server.

Internet Addresses Using Zeros

When a C class Internet address contains a 0 as the host address portion, (for example, 192.9.200.0), TCP/IP sends a wildcard address on the network. All machines with a Class C address of 192.9.200.X (where X represents a value between 0 and 254) should respond to the request. This results in a network flooded with requests to nonexistent machines.

Similarly, problems occur for Class B addresses such as 129.5.0.0. All machines with a Class B address of 129.5.X.X. (where X represents a value between 0 and 254) are obliged to respond to the request. In this case, because Class B addresses account for bigger networks than Class C addresses, the network is flooded with significantly more requests to nonexistent machines than for a Class C network.

Subnet Addresses

Subnet addressing allows an autonomous system made up of multiple networks to share the same Internet address. The subnetwork capability of TCP/IP also makes it possible to divide a single network into multiple logical networks (subnets). For example, an organization can have a single Internet network address that is known to users outside the organization, yet it can configure its network internally into departmental subnets. In either case, fewer Internet network addresses are required while local routing capabilities are enhanced.

A standard Internet Protocol address field has two parts: a network address and a local address. To make subnets possible, the local address part of an Internet address is divided into a subnet number and a host number. The subnet is identified so that the local autonomous system can route messages reliably.

In the basic Class A Internet address, which consists of an 8-bit network address and 24-bit local address, the local address identifies the specific host machine on the network.

Network Address (8 bits)	Local Host Address (24 bits)		
01111101	00001101	01001001	00001111

Figure 19. Class A Address. This illustration shows a typical class A address structure. The first 8 bits contain the network address (always beginning with a zero). The remaining 24 bits contain the local host address.

To create a subnet address for this Class A Internet address, the local address can be divided into a number identifying the physical network (or subnet) and a number identifying the host on the subnet. Senders route messages to the advertised network address, and the local system takes responsibility for routing messages to its subnets and their hosts. When deciding how to partition the local address into subnet address and host address, you should consider the number of subnets and the number of hosts on those subnets.

In the following figure, the local address is partitioned into a 12-bit subnet address and a 12-bit host address.

Figure 20. Class A Address with Corresponding Subnet Address. This illustration shows a typical class A address structure. The first 8 bits contain the network address (always beginning with a zero). The remaining 24 bits contain the local host address with the subnet address occupying the first 8 bits and the host address occupying the last 8 bits.

Network Address (8 bits)	Local Host Address (24 bits)		
Network Address	Subnet Address		Host Address
01111101	00001101	0100	1001 00001111

Note: The high-order bit (or first bit) will always be 0 in a Class A address.

You have flexibility when assigning subnet addresses and host addresses. The bits of the local address can be divided according to the needs and potential growth of the organization and its network structure. The only restrictions are:

- network_address is the Internet address for the network.
- subnet_address is a field of a constant width for a given network.
- host_address is a field that is at least 1-bit wide.

If the width of the subnet_address field is 0, the network is not organized into subnets, and addressing to the network is performed using the Internet network address.

The bits that identify the subnet are specified by a bit mask and, therefore, are not required to be adjacent in the address. However, it is generally desirable for the subnet bits to be contiguous and located as the most significant bits of the local address.

Subnet Masks

When a host sends a message to a destination, the system must determine whether the destination is on the same network as the source or if the destination can be reached directly through one of the local interfaces. The system compares the destination address to the host address using the *subnet mask*. If the destination is not local, the system sends the message on to a gateway. The gateway performs the same comparison to see if the destination address is on a network it can reach locally.

The subnet mask tells the system what the subnet partitioning scheme is. This bit mask consists of the network address portion and subnet address portion of the Internet address.

Network Address (8 bits)	Local Host Address (24 bits)			
Network Address	Subnet Address		Host Address	
01111101	00001101	0100	1001	00001111

Class A Address with Corresponding Subnet Address

Network Address (8 bits)	Local Host Address (24 bits)			
Network Address	Subnet Address		Host Address	
Subnet Mask			Host Address	
01111101	00001101	0100	1001	00001111

Class A Address with Corresponding Subnet Mask

Figure 21. Class A Address with Corresponding Subnet Address. This illustration shows a typical class A address structure. The first 8 bits contain the network address (always beginning with a zero). The remaining 24 bits contain the local host address with the subnet address occupying the first 8 bits and the host address occupying the last 8 bits.

For example, the subnet mask of the Class A address with the partitioning scheme defined above is shown in this figure.

The subnet mask is a set of 4 bytes, just like the Internetnetwork address. The subnet mask consists of high bits (1's) corresponding to the bit positions of the network and subnetwork address, and low bits (0's) corresponding to the bit positions of the host address. A subnet mask for the previous address looks like the following figure.

Network Address (8 bits)	Local Host Address (24 bits)			
Network Address	Subnet Address		Host Address	
11111111	11111111	1111	0000	00000000

Figure 22. Example Subnet Mask. This illustration shows a an example of a subnet mask structure. The first 8 bits contain the network address. The remaining 24 bits contain the local host address with the subnet address occupying the first 8 bits and the host address occupying the last 8 bits.

Address Comparison

The destination address and the local network address are compared by performing the logical AND and exclusive OR on the subnet mask of the source host.

The comparison process is outlined below:

1. Perform a logical AND of the destination address and the mask of the local subnet address.
2. Perform an exclusive OR on the result of the previous operation and the local net address of the local interface.

If the result is all 0's, the destination is assumed to be reachable directly through one of the local interfaces.

3. If an autonomous system has more than one interface (therefore more than one Internet address), the comparison process is repeated for each local interface.

For example, assume that there are two local interfaces defined for a host network, T125. Their Internet addresses and the binary representations of those addresses are shown in the following example:

Local Network Interface Addresses:

CLASS A 73.1.5.2 = 01001001 00000001 00000101 00000010

CLASS B 145.21.6.3 = 10010001 00010101 00000110 00000011

The corresponding subnet masks for the local network interfaces are shown in the following example:

Local Network Interface Addresses:

CLASS A 73.1.5.2 = 11111111 11111111 11100000 00000000

CLASS B 145.21.6.3 = 11111111 11111111 11111111 11000000

If the source network, T125, is requested to send a message to a destination network with the host address 114.16.23.8 (represented in binary as: 01110010 00010000 00010111 00001000), the system checks whether the destination can be reached through a local interface.

Note: The **subnetmask** keyword must be set in the configuration database of each host that is to support subnets. Before the subnetwork capability can be used, all hosts on the network must support it. Set the subnet mask permanently in the configuration database using the Web-based System Manager Network application or the Network Interface Selection menu in SMIT. The subnet mask can also be set in the running system using the **ifconfig** command. Using the **ifconfig** command to set the subnet mask is not a permanent change.

Broadcast Addresses

The TCP/IP can send data to all hosts on a local network or to all hosts on all directly connected networks. Such transmissions are called *broadcast messages*. For example, the **routed** routing daemon uses broadcast messages to query and respond to routing queries.

For data to be broadcast to all hosts on all directly connected networks, User Datagram Protocol (UDP) and Internet Protocol (IP) are used to send the data, and the host destination address in the IP header has all bits set to 1. For data to be broadcast to all hosts on a specific network, all the bits in the local address part of the IP address are set to 0. There are no user commands that use the broadcast capability, although such commands, or programs, can be developed.

The broadcast address can be changed temporarily by changing the *broadcast* parameter in the **ifconfig** command. Change the broadcast address permanently by using the Web-based System Manager, **wsm**, or the SMIT fast path **smit chinnet**. Changing the broadcast address may be useful if you need to be compatible with older versions of software that use a different broadcast address; for example, the host IDs are all set to 0.

Local Loopback Addresses

The Internet Protocol defines the special network address, 127.0.0.1, as a local loopback address. Hosts use local loopback addresses to send messages to themselves. The local loopback address is set by the configuration manager during the system startup process. Local loopback is implemented in the kernel and can also be set with the **ifconfig** command. Loopback is invoked when the system is started.

TCP/IP Address and Parameter Assignment - Dynamic Host Configuration Protocol (DHCP)

Transmission Control Protocol/Internet Protocol (TCP/IP) enables communication between machines with configured addresses. Part of the burden a network administrator must face is address assignment and parameter distribution for all machines on the network. Commonly, this is a process in which the administrator dictates the configuration to each user, allowing the user to configure his own machine. However, misconfigurations and misunderstandings can generate service calls that the administrator must deal with individually. The Dynamic Host Configuration Protocol (DHCP) gives the network administrator a method to remove the end user from this configuration problem and maintain the network configuration in a centralized location.

DHCP is an application-layer protocol that allows a client machine on the network, to get an IP address and other configuration parameters from the server. It gets information by exchanging packets between a daemon on the client and another on the server. Most operating systems now provide a DHCP client in their base package.

To obtain an address, the DHCP client daemon (**dhcpcd**) broadcasts a DHCP discover message, which is received by the server and processed. (Multiple servers can be configured on the network for redundancy.) If a free address is available for that client, a DHCP offer message is created. This message contains an IP address and other options that are appropriate for that client. The client receives the server DHCP offer and stores it while waiting for other offers. When the client chooses the best offer, it broadcasts a DHCP request that specifies which server offer it wants.

All configured DHCP servers receive the request. Each checks to see if it is the requested server. If not, the server frees the address assigned to that client. The requested server marks the address as assigned and returns a DHCP acknowledgement, at which time, the transaction is complete. The client has an address for the period of time (lease) designated by the server.

When half of the lease time is used, the client sends the server a *renew* packet to extend the lease time. If the server is willing to renew, it sends a DHCP acknowledgement. If the client does not get a response from the server that owns its current address, it broadcasts a DHCP rebind packet to reach the server if, for example, the server has been moved from one network to another. If the client has not renewed its address after the full lease time, the interface is brought down and the process starts over. This cycle prevents multiple clients on a network from being assigned the same address.

The DHCP server assigns addresses based on keys. Four common keys are network, class, vendor, and client ID. The server uses these keys to get an address and a set of configuration options to return to the client.

network

Identifies which network segment the packet came from. The network key allows the server to check its address database and assign an address by network segment.

class Is completely client configurable. It can specify an address and options. This key can be used to denote machine function in the network or to describe how machines are grouped for administrative purposes. For example, the network administrator might want to create a netbios class that contains options for NetBIOS clients or an accounting class that represents Accounting department machines that need access to a specific printer.

vendor

Helps identify the client by its hardware/software platform (for example, a Windows 95 client or an OS/2 Warp client).

client ID

Identifies the client either through the machine host name or its medium access control (MAC)

layer address. The client ID is specified in the configuration file of the **dhcpcd** daemon. Also, the client ID can be used by the server to pass options to a specific client or prohibit a particular client from receiving any parameters.

These keys can be used by the configuration either singularly or in combinations. If multiple keys are provided by the client and multiple addresses can be assigned, only one is chosen, and the option set is derived from the chosen key first. For more detailed information about the selection of keys and addresses, see Configuring DHCP.

A relay agent is needed so initial broadcasts from the client can leave the local network. This agent is called the BOOTP relay agent. The relay agents act as forwarding agents for DHCP and BOOTP packets.

The DHCP Server

Beginning with AIX 4.3.1, the DHCP server has been segmented into three main pieces, a database, a protocol engine, and a set of service threads, each with its own configuration information.

The DHCP Database

The **db_file.dhcpro** database is used to track clients and addresses and for access control (for example, allowing certain clients on some networks but not others, or disabling BOOTP clients on a particular network). Options are also stored in the database for retrieval and delivery to clients. The database is implemented as a dynamically loadable object, which allows for easy server upgrade and maintenance.

Using the information in the configuration file, the database is primed and verified for consistency. A set of checkpoint files handles updates to the database and reduces the overhead of writes to the main storage file. The database also contains the address and option pools, but these are static and are discussed in Configuring DHCP.

The main storage file and its back up are flat ASCII files that can be edited. The format for the database main storage files are:

```
DF01
"CLIENT ID" "0.0.0.0" State LeaseTimeStart LeaseTimeDuration LeaseTimeEnd
  "Server IP Address" "Class ID" "Vendor ID" "Hostname" "Domain Name"
"CLIENT ID" "0.0.0.0" State LeaseTimeStart LeaseTimeDuration LeaseTimeEnd
  "Server IP Address" "Class ID" "Vendor ID" "Host Name" "Domain Name"
...
```

The first line is a version identifier for the file: DF01c. The lines that follow are client record definition lines. The server reads from the second line to the end of the file. (The parameters in quotes must be enclosed in quotes.)

"CLIENT ID"

The ID the client uses to represent itself to the server.

"0.0.0.0"

is the IP address currently assigned to the DHCP server. If no address has been assigned, it is "0.0.0.0".

State The current state of the client. The DHCP protocol engine contains the allowable set, and the states are maintained in the DHCP database. The number next to *State* represents its value. The states can be:

(0) UNKNOWN

Represents clients that have no address assigned. This state never applies to addresses. **dadmin** reports "Unknown" and **lssrc** reports "Corrupt" for this state.

(1) FREE

Represents addresses that are available for use. In general, clients do not have this state unless they have no address assigned. **dadmin** and the output from **lssrc** report this state as "Free".

(2) RESERVED

Indicates client and address are tied, but loosely. The client has issued a DHCP discover message and the DHCP server has responded, but the client has not yet responded with a DHCP request for that address. **dadmin** and the output from **Issrc** report this state as "Reserved".

(3) BOUND

Indicates client and address are tied and that the client has been assigned this address for some amount of time. **dadmin** and the output from **Issrc** report this state as "Leased".

(4) RELEASED

Indicates the client and address are tied for informational purposes only. The DHCP protocol suggests that DHCP servers maintain information about the clients it has served for future reference (mainly to try giving the same address to that client that has been assigned that address in the past). This state indicates that the client has released the address. The address is available for use by other clients, if no other addresses are available. **dadmin** and the output from **Issrc** report this as "Released".

(5) EXPIRED

Indicates the client and address are tied together, but only for informational purposes, in a similar manner to released addresses. The expired state, however, represents clients that let their leases expire. An expired address is available for use and is reassigned after all free addresses are unavailable and before released addresses are reassigned. **dadmin** and the output from **Issrc** report this state as "Expired".

(6) BAD

Represents an address that is in use in the network but has not been handed out by the DHCP server. This state also represents addresses that clients have rejected. This state does not apply to clients.. **dadmin** and the output from **Issrc** report this state as "Used" and "Bad", respectively.

LeaseTimeStart

Is the start of the current lease time (in the number of seconds since January 1, 1970).

LeaseTimeDuration

Represents the duration of the lease (in seconds).

LeaseTimeEnd

Uses the same format as *LeaseTimeStart*, but it represents the end of the lease. Some configuration options use different values for the start and end of a lease and these values can be overridden by configuration file options. See DHCP Server File Syntax for *db_file* Database.

"Server IP Address"

Is the IP address of the DHCP server that owns this record.

"Class ID"

"Vendor ID"

"Host Name"

"Domain Name"

Values that the server uses to determine which options are sent to the server (stored as quoted strings). These parameters increase performance because option lists can be pregenerated for these clients when the DHCP server starts up.

Checkpoint Files: The syntax for the checkpoint files is not specified. If the server crashes or you have to shut down and cannot do a normal closing of the database, the server can process the checkpoint and backup files to reconstruct a valid database. The client that is being written to the checkpoint file when the server crashes is lost. The default files are:

/etc/db_file.cr

normal database operation

/etc/db_file.crbk

backups for the database

/etc/db_file.chkpt and /etc/db_file.chkpt2

rotating checkpoint files

The DHCP server for AIX 4.3.1 and later is threaded. To maintain high throughput, database operations (including save operations) are thread-efficient. When a save is requested, the existing checkpoint file is rotated to the next checkpoint file, the existing database file is copied to the backup file, and the new save file is created. Each client record is then logged and a bit is toggled to indicate that the client should use the new checkpoint file for logging. When all client records are recorded, the save is closed, and the backup and old checkpoint files are deleted. Clients can still be processed and, depending on whether the client record has been saved, database changes go into a new save file or to a new checkpoint file.

The DHCP Protocol Engine

For AIX 4.3.1 and later, the DHCP protocol engine has been updated to RFC 2131, but is still compatible with RFC 1541. (The server can also process options as defined in RFC 2132.) The protocol engine uses the database to determine what information is returned to the client.

The configuration of the address pools have some configuration options that affect the state of each machine. For example, the DHCP server pings addresses before it hands them out. The amount of time the server waits for a response is now configurable for each address pool.

DHCP Threaded Operations

The last piece of the DHCP server is actually a set of operations that are used to keep things running. Since the DHCP server is threaded, these operations are actually set up as threads that occasionally do things to make sure everything is together.

The first thread, the **main** thread, handles the SRC requests (such as **startsrc**, **stopsrc**, **lssrc**, **traceson**, and **refresh**). This thread also coordinates all operations that affect all threads and handles signals. For example,

- A SIGHUP (-1) causes a refresh of all databases in the configuration file.
- A SIGTERM (-15) will cause the server to gracefully stop.

The next thread, the **dadmin** thread, interfaces with **dadmin** client program and the DHCP server. The **dadmin** tool can be used to get status as well as modify the database to avoid editing the database files manually. Previous versions of the DHCP server prevented any clients from getting addresses if a status request was running. With the addition of the **dadmin** and **src** threads, the server can handle service requests and still handle client requests.

The next thread is the **garbage** thread, which runs timers that periodically clean the database, save the database, purge clients that do not have addresses, and remove reserved addresses that have been in reserve state for too long. All these timers are configurable (see Configuring DHCP). The other threads are packet processors. The number of these is configurable; the default is 10. Each of these can handle a request from a DHCP client. The number of packet processors required is somewhat load- and machine-dependent. If the machine is used for other services than DHCP, it is not wise to start up 500 threads.

Planning DHCP

To use this protocol, the network administrator needs to set up a DHCP server and configure BOOTP relay agents on links that do not have a DHCP server. Advance planning can reduce DHCP load on the network. For example, one server can be configured to handle all your clients, but all packets must be passed through it. If you have a single router between two large networks, it is wiser to place two servers in your network, one on each link.

Another aspect to consider is that DHCP implies a pattern of traffic. For example, if you set your default lease time to fewer than two days and your machines are powered off for the weekend, Monday morning becomes a period of high DHCP traffic. Although DHCP traffic does not cause huge overhead for the network, it needs to be considered when deciding where to place DHCP servers on a network and how many to use.

After enabling DHCP to get the client on the network, a client has no requirement to enter anything. The DHCP client, `dhcpcd`, reads the `dhcpcd.ini` file, which contains information on logging and other parameters needed to start running. After installation, decide which method to use for TCP/IP configuration: minimum configuration or DHCP. If DHCP is selected, choose an interface and specify some optional parameters. To choose the interface, select the keyword **any**, which tells `dhcpcd` to find the first interface that works and use it. This method minimizes the amount of input on the client side.

Configuring DHCP

By default, the DHCP server is configured by reading the `/etc/dhcpd.conf` file, which specifies the initial database of options and addresses. The server is started in the `/etc/rc.tcpip` file. It can also be started from Web-based System Manager, from SMIT, or through SRC commands. The DHCP client can be configured by running Web-based System Manager, the System Management Interface Tool (SMIT), or editing a flat ASCII file.

Configuring the DHCP server is usually the hardest part of using DHCP in your network. First, decide what networks you want to have DHCP clients on. Each subnet in your network represents a pool of addresses that the DHCP server must add to its database. For example:

```
database db_file
{
    subnet 9.3.149.0 255.255.255.0
    { option 3 9.3.149.1 # The default gateway clients on this network should use
      option 6 9.3.149.2 # The nameserver clients on this network should use
    }
    ... options or other containers added later
}
```

The example above shows a subnet, 9.3.149.0, with a subnet mask 255.255.255.0. All addresses in this subnet, 9.3.149.1 through 9.3.149.254, are in the pool. Optionally, a range can be specified on the end of the line or a range or exclude statement can be included in the subnet container. See DHCP Server File Known Options for common configuration methods and definitions.

The database clause with `db_file` indicates which database method to use for processing this part of the configuration file. Comments begin with a # (pound sign). Text from the initial #, to the end of the line, is ignored by the DHCP server. Each option line is used by the server to tell the client what to do. DHCP Server File Known Options describes the currently supported and known options. See DHCP Server File Syntax for General Server Operation for ways to specify options that the server does not know about.

If the server does not understand how to parse an option, it uses default methods to send the option to the client. This also allows the DHCP server to send site-specific options that are not RFC defined, but may be used by certain clients or client configurations.

The Configuration File

The configuration file has an address section and an option definition section. These sections use containers to hold options, modifiers, and, potentially, other containers.

A *container* (basically, a method to group options) uses an identifier to classify clients into groups. The container types are `subnet`, `class`, `vendor`, and `client`. Currently, there is not a generic user-definable container. The identifier uniquely defines the client so that the client can be tracked if, for example, it moves between subnets. More than one container type can be used to define client access.

Options are identifiers that are returned to the client, such as default gateway and DNS address.

Modifiers are single statements that modify some aspect of a container, such as lease time default.

Containers: When the DHCP server receives a request, the packet is parsed and identifying keys determine which containers, options, and addresses are extracted.

The previous example shows a subnet container. Its identifying key is the position of the client in the network. If the client is from that network, then it falls into that container.

Each type of container uses a different option to identify a client:

- The subnet container uses the **giaddr** field or the interface address of the receiving interface to determine from which subnet the client came.
- The class container uses the value in option 77 (User Site Class Identifier).
- The vendor uses the value in option 60 (Vendor Class Identifier).
- The client container uses the option 61 (Client Identifier) for DHCP clients and the **chaddr** field in the BOOTP packet for BOOTP clients.

Except for subnets, each container allows the specification of the value that matches it, including regular expression matching.

There is also an implicit container, the *global* container. Options and modifiers are placed in the global container unless overridden or denied. Most containers can be placed inside other containers implying a scope of visibility. Containers may or may not have address ranges associated with them. Subnets, by their nature, have ranges associated with them.

The basic rules for containers and subcontainers are:

- All containers are valid at the global level.
- Subnets can not be placed inside other containers.
- Restricted containers cannot have regular containers of the same type within them. (For example, a container with an option that only allows a class of Accounting cannot include a container with an option that allows all classes that start with the letter "a". This is illegal.)
- Restricted client containers cannot have subcontainers.

Given the above rules, you can generate a hierarchy of containers that segment your options into groups for specific clients or sets of clients.

If a client matches multiple containers, how are options and addresses handed out? The DHCP server receives messages, it passes the request to the database (**db_file** in this case), and a container list is generated. The list is presented in order of depth and priority. Priority is defined as an implicit hierarchy in the containers. Strict containers are higher priority than regular containers. Clients, classes, vendors, and finally subnets are sorted, in that order, and within container type by depth. This generates a list ordered by most specific to least specific. For example:

```
Subnet 1
--Class 1
--Client 1
Subnet 2
--Class 1
----Vendor 1
----Client 1
--Client 1
```

The example shows two subnets, Subnet 1 and Subnet 2. There is one class name, Class 1, one vendor name, Vendor 1, and one client name, Client 1. Class 1 and Client 1 are defined in multiple places. Because they are in different containers, their names can be the same but values inside them can be

different. If Client 1 sends a message to the DHCP server from Subnet 1 with Class 1 specified in its option list, the DHCP server would generate the following container path:

Subnet 1, Class 1, Client 1

The most specific container is listed last. To get an address, the list is examined in reverse hierarchy to find the first available address. Then, the list is examined in forward hierarchy to get the options. Options override previous values unless an option *deny* is present in the container. Also, because Class 1 and Client 1 are in Subnet 1, they are ordered according to the container priority. If the same client is in Subnet 2 and sends the same message, the container list generated is:

Subnet 2, Class 1, Client 1 (at the Subnet 2 level), Client 1 (at the Class 1 level)

Subnet 2 is listed first, then Class 1, then the Client 1 at the Subnet 2 level (because this client statement is only one level down in the hierarchy). The hierarchy implies that a client matching the first client statement is less specific than the client matching Client 1 of Class 1 within Subnet 2.

Priority selected by depth within the hierarchy is not superseded by the priority of the containers themselves. For example, if the same client issues the same message and specifies a vendor identifier, the container list is:

Subnet 2, Class 1, Vendor 1, Client 1 (at Subnet 2 level), Client 1 (at Class 1 level)

Container priority improves search performance because it follows a general concept that client containers are the most specific way to define one or more clients. The class container holds less specific addresses than a client container; vendor is even less specific; and subnet is the least specific.

Addresses and Address Ranges: Any container type can have associated addresses ranges; subnets must have associated address ranges. Each range within a container must be a subset of the range and must not overlap with ranges of other containers. For example, if a class is defined within a subnet and the class has a range, the range must be a subset of the subnet range. Also, the range within that class container cannot overlap with any other ranges at its level.

Ranges can be expressed on the container line and modified by range and exclude statements to allow for disjoint address sets associated with a container. If you have the top ten addresses and the second ten addresses of a subnet available, the subnet can specify these addresses by range in the subnet clause to reduce both memory use and the chance of address collision with other clients not in the specified ranges.

Once an address has been selected, any subsequent container in the list that contains address ranges is removed from the list along with its children. Network-specific options in removed containers are not valid if an address is not used from within that container.

Options: After the list has been culled to determine addresses, a set of options is generated for the client. In this selection process, options overwrite previously selected options unless a *deny* is encountered, in which case, the denied option is removed from the list being sent to the client. This method allows inheritance from parent containers to reduce the amount of data that must be specified.

Modifiers: Modifiers are items that change some aspect of a particular container, such as access or lease time. Define the address and option pools before modifying the container. The most common modifiers are **leasetimedefault**, **supportBootp**, and **supportUnlistedclients**.

leasetimedefault

Defines the amount of time an address is to be leased to a client.

supportBootp

Defines whether or not the server responds to BOOTP clients.

supportUnlistedclients

Indicates whether clients are to be explicitly defined by a client statement to receive addresses.

The value for `supportUnlistedClients` can be **none**, **dhcp**, **bootp**, or **both**. This allows for you to restrict access to bootp client and allow all DHCP clients to get addresses.

Other modifiers are listed in DHCP Server File Syntax for `db_file` Database.

Logging: After selecting modifiers, the next item to set up is logging. Logging parameters are specified in a container like the database, but the container keyword is **logging_info**. When learning to configure DHCP, it is advisable to turn logging to its highest level. Also, it is best to specify the logging configuration before any other configuration file data to ensure that configuration errors are logged after the logging subsystem is initialized. Use the **logitem** keyword to turn on a logging level or remove the **logitem** keyword to disable a logging level. Other keywords for logging allow the specification of the log filename, file size, and the number of rotating log files.

Server-specific Options: The last set of parameters to specify are server-specific options that allow the user to control the number of packet processors, how often the garbage collection threads are run, and so on..

For example, two server-specific options are:

reservedTime

Indicates how long an address stays in the reserved state after sending an OFFER to the DHCP client

reservedTimeInterval

Indicates how often the DHCP server scans through the addresses to see if there are any that have been in the reserved state longer than **reservedTime**.

These options are useful if you have several clients that broadcast DISCOVER messages and, either they do not broadcast their REQUEST message, or their REQUEST message gets lost in the network. Using these parameters keeps addresses from being reserved indefinitely for a noncompliant client.

Another particularly useful option is **SaveInterval**, which indicates how often saves occur. All server-specific options are listed in DHCP Server File Syntax for General Server Operation with the logging keywords.

Performance Considerations: It is important to understand that certain configuration keywords and the structure of the configuration file have an effect on the memory use and performance of the DHCP server.

First, excessive memory use can be avoided by understanding the inheritance model of options from parent to child containers. In an environment that supports no unlisted clients, the administrator must explicitly list each client in the file. When options are listed for any specific client, the server uses more memory storing that configuration tree than when options are inherited from a parent container (for example, the subnet, network, or global containers). Therefore, the administrator should verify whether any options are repeated at the client level within the configuration file and determine whether these options can be specified in the parent container and shared by the set of clients as a whole.

Also, when using the **logitem** entries INFO and TRACE, numerous messages are logged during the processing of every DHCP client message. Appending a line to the log file can be an expensive operation; therefore, limiting the amount of logging improves the performance of the DHCP server. When an error with the DHCP server is suspected, logging can be dynamically re-enabled using either the SRC traceson or dadmin commands.

Finally, selecting a **numprocessors** value depends on the size of the DHCP-supported network, the **pingTime db_file** configuration parameter, and the typical propagation delay on the network. Because each packet processor thread issues an ICMP Echo Request to verify the status of a server-owned address before offering it to a client, the amount of time that any Echo Response is waited for directly affects the amount of processing time for a DISCOVER message. Essentially, the packet processor thread

is able to do nothing more than wait for any response or for the **pingTime** timeout. Lowering the **numprocessors** value improves the response time of the server by lowering the number of client retransmissions, yet still maintaining the ping benefit of the server design.

For best performance, select a **pingTime** based on the propagation delay of any remote networks supported by the DHCP server. Also, select the **numprocessors** value based on this **pingTime** value and the size of the network. Selecting a value that is too small can cause all packet processing threads to be stopped. The server is then caused to wait for any Echo Responses while incoming DHCP client messages are queueing on the server port. This causes the server to handle client messages in batches rather than in a constant stream.

A selected value that is too small can cause all packet processing threads to be stopped waiting for any Echo Responses , which would result in the .

To prevent this situation, set the value for **numprocessors** to a number higher than the estimated number of DISCOVER messages that can be received within one **pingTime** interval during a period of high DHCP client activity. However, do not set the **numprocessors** value so high that it could burden the kernel with thread management.

For example, the values **numprocessors 5** and **pingTime 300** cause poor performance in an environment with a potential 10 DISCOVER messages per second because at peak demand, only 5 messages are handled every 3 seconds. Configure this environment with values similar to **numprocessors 20** and **pingTime 80**.

Customizing a Configuration File

Many networks include multiple client types; for example, a single network may include computers running a variety of operating systems, such as Windows, OS/2, Java OS, and UNIX. Each of these require unique vendor identifiers (the field used to identify the type of machine to the DHCP server). Java OS clients and IBM Thin Client machines can require unique parameters such as bootfiles, and configuration options that need to be tailored specifically for them. Windows 95 computers do not handle Java-specific options well.

Machine-specific options can be encapsulated within vendor containers if the primary use for certain machines is based on the type of user for those machines. For instance, the development staff might use this operating system's clients for programming, the marketing staff might use the OS/2 clients, sales might use Java OS clients and IBM Thin Client machines, and accounting might use Windows 95 machines. Each of these user families might need different configuration options (different printers, nameservers, or default web servers, and so forth). In this case, such options could be included in the vendor container, since each group uses a different machine type.

If the same machine type is used by multiple groups, placing the options within a subordinate class identifier instead, would allow your marketing managers, for example, to use a specific set of printers that other employees could not access.

Note: The following fictional example represents part of a configuration file. Comments are preceded by a pound sign (#) and describe how each line defines the installation.

```
vendor "AIX_CLIENT"
{
# No specific options, handles things based on class
}

vendor "OS/2 Client"
{
# No specific options, handles things based on class
}

vendor "Windows 95"
{ option 44 9.3.150.3          # Default NetBIOS Nameserver
}
```

```

vendor "Java OS"
{
  bootstrapserver 9.3.150.4 # Default TFTP server for the Java OS boxes
  option 67 "javaos.bin" # The bootfile of the Java OS box
}

vendor "IBM Thin Client"
{
  bootstrapserver 9.3.150.5 # Default TFTP server for Thin Client boxes
  option 67 "thin.os.bin" # Default bootfile for the Thin Client boxes
}

subnet 9.3.149.0 255.255.255.0
{
  option 3 9.3.149.1 # The default gateway for the subnet
  option 6 9.3.150.2 # This is the nameserver for the subnet
  class accounting 9.3.149.5-9.3.149.20
  {
    # The accounting class is limited to address range 9.3.149.5-9.3.149.20
    # The printer for this group is also in this range, so it is excluded.
    exclude 9.3.149.15
    option 9 9.3.149.15 # The LPR server (print server)
    vendor "Windows 95"
    {
      option 9 deny # This installation of Windows 95 does not support
                  # this printer, so the option is denied.
    }
  }
}
. . .
}

```

DHCP and the Dynamic Domain Name System (DDNS)

The DHCP server provides options that enable operation in a DDNS environment. To use DHCP in a DDNS environment, you must set and use a Dynamic Zone on a DNS server.

After the DDNS server is configured, decide if the DHCP server is going to do A-record updates, PTR-record updates, updates for both record types, or none at all. This decision depends on whether a client machine can do part or all of this work.

- If the client can share update responsibility, configure the server to do the PTR-record updates and configure the client to do the A-record updates.
- If the client can do both updates, configure the server to do none.
- If the client cannot do updates, configure the server to do both.

The DHCP server has a set of configuration keywords that allow you to specify a command to run when an update is required. These are:

updatedns

(Deprecated.) Represents the command to issue to do any type of update. It is called for both the PTR-record and the A-record update.

updatednsA

Specifies the command to update the A-record.

updatednsP

Specifies the command to update the PTR-record.

These keywords specify executable strings that the DHCP server runs when an update is required. The keyword strings must contain four %s (percent symbol, letter s). The first %s is the hostname; the second is the domain name; the third is the IP address; and the fourth is the lease time. These are used as the first four parameters for the **dhcpcaction** command. The remaining two parameters for the **dhcpcaction** command indicate the record to update (A, PTR, NONE, or BOTH) and whether NIM should be updated (NIM or NONIM). See DHCP and Network Installation Management (NIM) Interactions and Suggestions for more information about NIM and DHCP interaction. For example:


```

updatednsA "/usr/sbin/dhcpaction '%s' '%s' '%s' '%s' A NONIM"
# This does the dhcpaction command only on the A record
updatednsP "/usr/sbin/dhcpaction '%s' '%s' '%s' '%s' PTR NONIM"
# This does the command only on the PTR record
updatedns "/usr/sbin/dhcpaction '%s' '%s' '%s' '%s' BOTH NIM"
# This does the command on both records and updates NIM

```

The DHCP server also has a set of keywords to remove the DNS entries when a lease is released or expires. The keywords are:

releasednsA

Removes the A-record.

releasednsP

Removes the PTR-record.

removedns

Removes both record types.

These keywords specify executable strings that the DHCP server runs when an address is released or expired. The **dhcpremove** command works similarly to **dhcpaction**, but only takes three parameters:

1. The IP address, specified as a %s in the command string
2. Which record to remove (A, PTR, NONE, or BOTH).
3. Whether NIM should be updated (NIM or NONIM).

For example:

```

releasednsA "/usr/sbin/dhcpremove '%s' A NONIM"
# This does the dhcpremove command only the A record
releasednsP "/usr/sbin/dhcpremove '%s' PTR NONIM"
# This does the command only on the PTR record
removedns "/usr/sbin/dhcpremove '%s' BOTH NIM"
# This does the command on both records and updates NIM

```

The **dhcpaction** and **dhcpremove** scripts do some parameter checking, then set up a call to **nsupdate**, which has been updated to work with this operating system's servers and with OS/2 DDNS servers. See the **nsupdate** command description for more information.

If NIM interaction is **NOT** required by the name update, the DHCP server can be configured to use a socket transfer between the **DHCP** daemon and the **nsupdate** command to improve performance and enable DNS updates to be retried upon failure. To configure this option, the **updateDNSA**, **updateDNSP**, **releaseDNSA**, or the **releaseDNSP** keyword must specify "nsupdate_daemon" as the first quoted word. The parameters and flags for this update are identical to those that are accepted by the **nsupdate** command. Additionally, the following variable names can be used for substitution:

\$hostname	Replaced by the host name of the client on DNS update or the host name previously associated with the client for DNS removal.
\$domain	Replaced by the DNS domain for the update or the previously used domain of the client host name for a DNS removal.
\$ipadress	Replaced by the IP address to be associated or disassociated from the DHCP client name.
\$leasetime	Replaced by the lease time (in seconds).
\$clientid	Replaced by the string representation of the DHCP client identifier or the combination hardware type and hardware address for BOOTP clients.

For example:

```

updateDNSA "nsupdate_daemon -p 9.3.149.2 -h $hostname -d $domain
-s"d;a;*;a;$ipadress;s;$leasetime;3110400"

```

```
updateDNSP "nsupdate_daemon -p 9.3.149.2 -r $ipaddress
-s"d;ptr;*;a;ptr;$hostname.$domain.;s;$leasetime;3110400"

releaseDNSA "nsupdate_daemon -p 9.3.149.2 -h $hostname -d $domain -s"d;a;*;s;1;3110400"

releaseDNSP "nsupdate_daemon -p 9.3.149.2 -r $ipaddress -s"d;ptr;*;s;1;3110400"
```

See the **nsupdate** command description for more information.

Also, administrator-defined policies have been added for hostname exchanges between the server and the clients. By default, the hostname that is returned to the client and used for a DDNS update is option 12 (defined in the server configuration file). Alternatively, the default hostname can be the client-suggested hostname, either through option 81 (the DHCPDDNS option) or through option 12 (the HOSTNAME option). However, the administrator can override the default hostname by using the **hostnamepolicy**, **proxyarec**, and **appenddomain** configuration keywords. These options and their parameters are defined in DHCP Server File Syntax for db_file Database.

DHCP Compatibility with Older Versions

The DHCP server for AIX 4.3.1 and later recognizes the previous versions configuration and database files, **dhcps.ar** and **dhcps.cr**. It parses the old configuration files and generates new database files in the old locations. The old databases are converted automatically to the new file. The configuration file itself is not converted.

The DHCP server database module, **db_file**, can read the old format. The DHCP server can recognize when a database container is not in the configuration file and treats the whole file as configuring the server parameters, logging parameters, and the **db_file** database parameters.

Notes: Some old configuration file syntax is deprecated, but is still supported. Other deprecations are:

- The network container is completely deprecated. To specify correctly, either convert the network clause with a range into a valid subnet container with a subnet address, subnet netmask, and the range. If the network container has subnet containers, remove the network container keyword and its braces and then place the subnet mask in the appropriate place on the line. To start using the database container, group everything that pertains to networks and client access into one database container of type **db_file**.
- The **updatedns** and **removedns** keywords are deprecated and replaced in favor of specifying the action for A and PTR records separately.
- The **clientrecorddb** and **addressrecorddb** keywords have been deprecated to **clientrecorddb** and **backupfile**, respectively.
- The **option sa** and **option ga** keywords have been replaced by **bootstrapserver** and **giaddrfield** keywords, respectively. See DHCP Server File Syntax for General Server Operation and DHCP Server File Syntax for db_file Database for more information.

DHCP Server File Known Options

Note: The options that are shown in the following table as not allowed to be specified (No in the Can Specify? column) can be specified in the configuration file, but are overwritten by the correct value. For a better definition of each option, see RFC 2132.

Option Number	Default Data Type	Can Specify?	Description/Use
0	None	No	The server pads the option field, if necessary.
1	Dotted quad	No	The net mask of the subnet from which the address was drawn.

Option Number	Default Data Type	Can Specify?	Description/Use
2	32-bit integer	Yes	Specifies the offset of the client subnet, in seconds from Coordinated Universal Time (UTC).
3	One or more dotted quads	Yes	A list of the default gateways' IP addresses.
4	One or more dotted quads	Yes	A list of time server IP addresses.
5	One or more dotted quads	Yes	A list of name server IP addresses.
6	One or more dotted quads	Yes	A list of DNS IP addresses.
7	One or more dotted quads	Yes	A list of log server IP addresses.
8	One or more dotted quads	Yes	A list of cookie server IP addresses.
9	One or more dotted quads	Yes	A list of LPR server IP addresses.
10	One or more dotted quads	Yes	A list of Impress server IP addresses.
11	One or more dotted quads	Yes	A list of Resource Location server IP addresses.
12	An ASCII string	Yes	A hostname for the client to use.
13	16-bit unsigned integer	Yes	The size of the bootfile.
14	An ASCII string	Yes	The path for Merit Dump file.
15	An ASCII string	Yes	The default DNS domain name.
16	An IP address	Yes	The address of the Swap server.
17	An ASCII string	Yes	The default root path.
18	An ASCII string	Yes	The path to extensions for the client.
19	Yes, No, True, False, 1, 0	Yes	Specify whether IP Forwarding should be turned on.
20	Yes, No, True, False, 1, 0	Yes	Specify whether non-local source routing should be used.
21	One or more pairs of dotted quads, in the form <i>DottedQuad:DottedQuad</i>	Yes	The filter policies for IP addresses.
22	16-bit unsigned integer	Yes	The maximum size to allow for datagram fragments.
23	8-bit unsigned integer	Yes	The IP time-to-live (TTL).
24	32-bit unsigned integer	Yes	The number of seconds to use in the Path MTU aging timeout.
25	List of one or more 16-bit unsigned integers	Yes	The path MTU Plateau table. Specifies a set of values that represent the MTU sizes to use when using Path MTU discovery.
26	16-bit unsigned integer	Yes	Specifies MTU size for the receiving interface.
27	Yes, No, True, False, 1, 0	Yes	Specifies whether all subnets are local.
28	An IP address (dotted quad)	Yes	Specifies broadcast address for the interface.
29	Yes, No, True, False, 1, 0	Yes	Specifies whether ICMP netmask discovery should be used.
30	Yes, No, True, False, 1, 0	Yes	Specifies whether client should become an ICMP netmask supplier.
31	Yes, No, True, False, 1, 0	Yes	Specifies whether ICMP Router Discovery messages should be used.
32	IP address (dotted quad)	Yes	Specifies address to use for router solicitation.
33	One or more IP address pairs, in the form <i>DottedQuad:DottedQuad</i>	Yes	Each address pair represents a static route.
34	Yes/No, True/False, 1/0	Yes	Specifies whether trailer encapsulation should be used.
35	32-bit unsigned integer	Yes	ARP cache timeout value.
36	Yes/No, True/False, 1/0	Yes	Specifies whether Ethernet encapsulation should be used.
37	8-bit unsigned integer	Yes	The TCP time-to-live (TTL).
38	32-bit unsigned integer	Yes	The TCP keep alive interval.

Option Number	Default Data Type	Can Specify?	Description/Use
39	Yes/No, True/False, 1/0	Yes	Specifies whether TCP keep alive should be used.
40	An ASCII string	Yes	The NIS default domain.
41	One or more dotted quads	Yes	Specifies the IP addresses of the NIS servers.
42	One or more dotted quads	Yes	Specifies the IP addresses of the NTP servers.
43	hex string of digits, in the form of hex " <i>digits</i> ", hex " <i>digits</i> ", or <i>0xdigits</i>	Yes, but really only specified with vendor container	Encapsulated option container for the vendor container.
44	One or more dotted quads	Yes	Specifies NetBIOS name server IP addresses.
45	One or more dotted quads	Yes	Specifies NetBIOS datagram distribution server IP addresses.
46	8-bit unsigned integer	Yes	Specifies NetBIOS Node Type.
47	hex string of digits, in form of hex " <i>digits</i> ", hex " <i>digits</i> ", or <i>0xdigits</i>	Yes	NetBIOS Scope.
48	One or more dotted quads	Yes	Specifies X Windows font server IP addresses.
49	One or more dotted quads	Yes	Specifies X Windows Display Manager.
50	None	No	Requested IP Address, used by client to indicate the address it wants.
51	32-bit unsigned integer	Yes	Lease time for the returned address. By default, the DHCP server uses the leasetimedefault keyword, but direct specification of option 51 overrides it.
52	None	No	Option overload. Client uses this to indicate the sname and file fields of the BOOTP packet may have options.
53	None	No	DHCP server or client uses this option to indicate the type of DHCP message.
54	None	No	DHCP server or client uses this option to indicate the server's address or the server to which the message is directed.
55	None	No	DHCP client uses this to indicate desired options.
56	An ASCII string	Yes	A string the DHCP server sends to the client. In general, this can be used by the DHCP server and client to indicate problems.
57	No	No	DHCP client uses this option to tell the DHCP server the maximum DHCP packet size the client can receive.
58	32-bit unsigned integer	Yes	Specifies the number of seconds until the client should send a renew packet.
59	32-bit unsigned integer	Yes	Specifies the number of seconds until the client should send a rebind packet.
60	None	No	DHCP client uses this option to indicate its vendor type. The DHCP server uses this field to match vendor containers.
61	None	No	DHCP client uses this to uniquely identify itself. The DHCP server uses this field to match client containers.
64	An ASCII string	Yes	Specifies the NIS+ domain.
65	One or more dotted quads	Yes	IP Addresses of NIS+ servers.
66	An ASCII string	Yes	Specifies the TFTP server name. This is a hostname and is used instead of the siaddr field if the client understands this option.

Option Number	Default Data Type	Can Specify?	Description/Use
67	An ASCII string	Yes	Specifies the bootfile name. This can be used instead of the bootfile keyword, which places the file in the filename field of the packet.
68	One or more dotted quads, or NONE	Yes	Specifies addresses of home agents.
69	One or more dotted quads	Yes	Specifies default SMTP servers to use.
70	One or more dotted quads	Yes	Specifies default POP3 servers to use.
71	One or more dotted quads	Yes	Specifies default NNTP servers to use.
72	One or more dotted quads	Yes	Specifies default WWW servers to use.
73	One or more dotted quads	Yes	Specifies default Finger servers to use.
74	One or more dotted quads	Yes	Specifies default IRC servers to use.
75	One or more dotted quads	Yes	Specifies default Street Talk servers to use.
76	One or more dotted quads	Yes	Specifies default Street Talk directory assistance servers to use.
77	An ASCII string	Yes	The user site class identifier. The DHCP server uses this field to match class containers.
81	An ASCII string plus other items	No	The DHCP client uses this option to define the policy the DHCP server should use with respect to DDNS.
93	None	No	The DHCP client uses this option to define the client system architecture.
94	None	No	The DHCP client uses this option to define the client network interface identifier.
255	None	No	DHCP server and client use this option to indicate the end of an option list.

Preboot Execution Environment (PXE) Vendor Container Suboption

When supporting a preboot execution environment (PXE) client, the DHCP server passes the following option to the BINLD server, which is used by BINLD to configure itself:

Opt Num	Default Data Type	Can Specify?	Description
7	one dotted quad	Yes	Multicast IP address. Boot server discovery multicast IP address.

The following example shows how this option can be used:

```
pxeservertype    proxy_on_dhcp_server
```

```
Vendor pxeserver
{
    option 7    9.3.4.68
}
```

In the above example, the DHCP server informs the client that the proxy server is running on the same machine but is listening on port 4011 for client requests. The vendor container is required here because the BINLD server broadcasts an INFORM/REQUEST message on port 67 with option 60 set to "PXEServer." In response, the DHCP server sends the Multicast IP address on which the BINLD has to listen for PXEClient's request.

Configuration File Examples Supporting PXE Clients

In the following example, the **dhcpsd** server either gives the bootfile name to the PXEClient or it directs the PXEClient to the BINLD server by sending suboptions. The **pxebootfile** keyword is used to create a list of boot files for a given client architecture and major and minor versions of the client system.

```

pxeservertype    dhcp_pxe_binld

subnet default
{
    vendor pxe
    {
        option 6 2    # Disable Multicast
        option 8 5 4 10.10.10.1 12.1.1.15 12.5.5.5 12.6.6.6\
            2 2 10.1.1.10 9.3.4.5 1 1 10.5.5.9\
            1 1 9.3.149.15\
            4 0
        option 9 5 "WorkSpace On Demand" 2 "Intel"\
            1 "Microsoft WindowsNT" 4 "NEC ESMPRO"
        option 10 2 "Press F8 to View Menu"
    }
    vendor pxeserver
    {
        option 7 239.0.0.239
    }
}

subnet 9.3.149.0 255.255.255.0
{
    option 3 9.3.149.1
    option 6 9.3.149.15

    vendor pxe
    {
        option 6 4 # bootfile is present in the offer packet
        pxebotfile 1 2 1 os2.one
        pxebotfile 2 2 1 aix.one
    }
}

```

Each option line in pxe container is used by the server to tell the client what to do. PXE Vendor Container Suboptions describes the currently supported and known PXE sub-options.

DHCP Server File Syntax for General Server Operation

Note: Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.

Keyword	Form	Subcontainers?	Default Value	Meaning
database	database <i>db_type</i>	Yes	None	The primary container that holds the definitions for the address pools, options, and client access statements. <i>db_type</i> is the name of a module that is loaded to process this part of the file. The only value currently available is db_file .
logging_info	logging_info	Yes	None	The primary logging container that defines the logging parameters.

Keyword	Form	Subcontainers?	Default Value	Meaning
logitem	logitem NONE	No	All default to not enabled.	Enables the logging level. Multiple lines are allowed.
	logitem SYSERR			
	logitem OBJERR			
	logitem PROTOCOL			
	logitem PROTERR			
	logitem WARN			
	logitem WARNING			
	logitem CONFIG			
	logitem EVENT			
	logitem PARSEERR			
	logitem ACTION			
	logitem ACNTING			
	logitem STAT			
	logitem TRACE			
logitem RTRACE				
logitem START				
numLogFiles	numLogFiles <i>n</i>	No	0	Specifies the number of log files to create. The log rotates when the first one fills. <i>n</i> is the number of files to create.
logFileSize	logFileSize <i>n</i>	No	0	Specifies the size of each log file in 1024-byte units.
logFileName	logFileName <i>path</i>	No	None	Specifies the path to the first log file. The original log file is named <i>filename</i> or <i>filename.extension</i> . The <i>filename</i> must be eight or fewer characters. When a file is rotated, it is renamed beginning with the base <i>filename</i> , then either appending a number or replacing the extension with a number. For example, if the original file name is <i>file</i> , the rotated file name becomes <i>file01</i> . If the original file name is <i>file.log</i> , it becomes <i>file.01</i> .

Keyword	Form	Subcontainers?	Default Value	Meaning
CharFlag	charflag yes	No	true	Not applicable to this operating system's DHCP server, but the OS/2 DHCP server uses it to produce debug windows.
	charflag true			
	charflag false			
	charflag no			
StatisticSnapShot	StatisticSnapShot <i>n</i>	No	-1, never	Specifies how often statistics are written to the log file in seconds.
UsedIppAddressExpireInterval	UsedIppAddressExpireInterval <i>n time_units</i>	No	-1, never	Specifies how often addresses placed in the BAD state are recouped and retested for validity.
leaseExpireInterval	leaseExpireInterval <i>n time_units</i>	No	900 seconds	Specifies how often addresses in the BOUND state are checked to see if they have expired. If the address has expired, the state is moved to EXPIRED.
reservedTime	reservedTime <i>n time_units</i>	No	-1, never	Specifies how long addresses should sit in RESERVED state before being recouped into the FREE state.
reservedTimeInterval	reservedTimeInterval <i>n time_units</i>	No	900 seconds	Specifies how often addresses in the RESERVE state are checked to see if they should be recouped into the FREE state.
saveInterval	saveInterval <i>n time_units</i>	No	3600 seconds	Specifies how often the DHCP server should force a save of the open databases. For heavily loaded servers, this should be 60 or 120 seconds.
clientpruneintv	clientpruneintv <i>n time_units</i>	No	3600 seconds	Specifies how often the DHCP server has the databases remove clients are not associated with any address (in the UNKNOWN state). This reduces the memory use of the DHCP server.
numprocessors	numprocessors <i>n</i>	No	10	Specifies the number of packet processors to create. Minimum of one.

Keyword	Form	Subcontainers?	Default Value	Meaning
userObject	userObject <i>obj_name</i>	Yes	None	Indicates that the server should load a user-defined shared object and call routines within this object through each interaction with DHCP clients. The object to be loaded is located in the /usr/sbin directory by the name <i>obj_name</i> .dhcpo. See the DHCP Server User-Defined Extension API for more information.
pxeservertype	pxeservertype <i>server_type</i>	No	dhcp_only	Indicates the type of dhcpcd server that it is. <i>server_type</i> can be one of the following: dhcp_pxe_bindl DHCP performs dhcpsd , pxed , and bindl functions. proxy_on_dhcp_server DHCP refers the PXE client to the proxy server port on the same machine. The default is dhcp_only, meaning the dhcpsd does not support PXE clients in default mode.

DHCP Server File Syntax for db_file Database

Notes: Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.

Also, items that are specified in one container can be overridden inside a subcontainer. For example, you could globally define BOOTP clients, but within a certain subnet allow BOOTP clients by specifying the supportBootp keyword in both containers.

The client, class, and vendor containers allow for regular expression support. For class and vendor, a quoted string with the first character after the quote being an exclamation point (!) indicates that the

rest of the string should be treated as a regular expression. The client container allows for regular expressions on both the hwtype and the hwaddr fields. A single string is used to represent both fields with the following format:

decimal_number-data

If decimal_number is zero, then data is an ASCII string. If any other number, data is hex digits.

Keyword	Form	Subcontainers?	Default Value	Meaning
subnet	subnet default	Yes	None	Specifies a subnet without an associated range. This subnet is used by the server only when responding to a client INFORM/REQUEST packet from the client and the client's address does not have another matching subnet container.
subnet	subnet <i>subnet id netmask</i>	Yes	None	Specifies a subnet and a pool of addresses. All addresses are assumed to be in the pool unless a range is specified on the line or addresses are modified later in the container by a range or exclude statement. The optional range is a pair of IP addresses in dotted quad format separated by a dash. An optional label and priority can be specified. These are used by virtual subnets to identify and order the subnets in the virtual subnet. The label and priority are separated by a colon. These containers are only allowed at the global or database container level.
	subnet <i>subnet id netmask range</i>			
	subnet <i>subnet id netmask label:priority</i>			
	subnet <i>subnet id netmask range label:priority</i>			
subnet	subnet <i>subnet id range</i>	Yes	None	Specifies a subnet that goes within a network container. It defines a range of addresses that is the whole subnet unless the optional range part is specified. The netmask associated with the subnet is taken from the surrounding network container. Note: This method is deprecated in favor of the other subnet forms.

Keyword	Form	Subcontainers?	Default Value	Meaning
option	option <i>number data ...</i>	No	None	Specifies an option to send to a client or, in the case of deny, an option to prevent from being sent to the client. The option * deny clause means all options not specified in the current container are not to be returned to the client. The option <i>numberdeny</i> only denies the specified option. <i>number</i> is an unsigned 8-bit integer. <i>data</i> is specific to the option (see above) or can be specified as a quoted string (indicating ASCII text) or <i>0xhexdigits</i> or hex" <i>hexdigits</i> " or hex " <i>hexdigits</i> ". If the option is in a vendor container, the option will be encapsulated with other options in an option 43.
	option <i>numberdeny</i>			
	option * deny			
exclude	exclude <i>an IP address</i>	No	None	Modifies the range on the container in which the exclude statement is in. The exclude statement is not valid in the global or database container levels. The exclude statement removes the specified address or range from the current range on the container. The exclude statement allows you to create non-contiguous ranges for subnets or other containers.
	exclude <i>dotted_quad-dotted_quad</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
range	range <i>IP_address</i>	No	None	Modifies the range on the container in which the range statement is in. The range statement is not valid in the global or database container levels. If the range is the first in the container that does not specify a range on the container definition line, then the range for the container becomes the range specified by the range statement. Any range statement after the first range or all range statements for a containers that specifies ranges in its definition are added to the current range. With the range statement, a single address or set of addresses can be added to the range. The range must fit inside the subnet container definition.
	range <i>dotted_quad-dotted_quad</i>			
client	client <i>hwtype hwaddr</i> NONE	Yes	None	Specifies a client container that denies the client specified by the <i>hwaddr</i> and <i>hwtype</i> from getting an address. If <i>hwtype</i> is 0, then <i>hwaddr</i> is an ASCII string. Otherwise, <i>hwtype</i> is the hardware type for the client and <i>hwaddr</i> is the hardware address of the client. If the <i>hwaddr</i> is a string, then quotes are accepted around the string. If the <i>hwaddr</i> is a hexstring, then the address may be specified by <i>0xhexdigits</i> or <i>hex digits</i> . <i>range</i> allows the client specified by the <i>hwaddr</i> and <i>hwtype</i> to get an address in the <i>range</i> . Must be regular expressions to match multiple clients.
	client <i>hwtype hwaddr</i> ANY			
	client <i>hwtype hwaddr</i> <i>dotted_quad</i>			
	client <i>hwtype hwaddr</i> <i>range</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
class	class <i>string</i>	Yes	None	Specifies a class container with name <i>string</i> . String can be quoted or not. If quoted, the quotes are removed before comparison. Quotes are required for strings with spaces or tabs. This container is valid at any level. A range can be supplied to indicate a set of addresses to hand out to a client with this class. The range is either a single dotted quad IP address or two dotted quad IP addresses separated by a dash.
	class <i>string range</i>			
network	network <i>network id netmask</i>	Yes	None	Specifies a network ID using class information (for example, 9.3.149.0 with a netmask of 255.255.255.0 would be network 9.0.0.0 255.255.255.0). This version of the network container is used to hold subnets with the same network ID and netmask. When a range is provided, all the addresses in the range are in the pool. The range must be in the network ID's network. This uses class full addressing. This is only valid in the global or database container level. Note: The network keyword is deprecated in favor of the subnet container.
	network <i>network id</i>			
	network <i>network id range</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
vendor	vendor <i>vendor_id</i>	Yes	None	Specifies a vendor container. Vendor containers are used to return option 43 to the client. The vendor id may be specified in a quoted string or a binary string in the form <i>0xhexdigits</i> or <i>hex" digits"</i> . An optional range may be placed after the vendor id. The range is specified as two dotted quads separated by a dash. After the optional range, an optional hexstring or ASCII string can be specified as the first part of the option 43. If options are in the container, they are appended to the option 43 data. After all options are processed an End Of Option List Option is appended to the data. To return options outside of an option 43, use a regular expression client that matches all clients to specify normal options to return based on the vendor ID.
	vendor <i>vendor_id</i> hex""			
	vendor <i>vendor_id</i> hex ""			
	vendor <i>vendor_id</i> 0xdata			
	vendor <i>vendor_id</i> ""			
	vendor <i>vendor_id</i> range			
	vendor <i>vendor_id</i> range hex""			
	vendor <i>vendor_id</i> range hex ""			
	vendor <i>vendor_id</i> range 0xdata			
vendor <i>vendor_id</i> range ""				

Keyword	Form	Subcontainers?	Default Value	Meaning
inoption	inoption <i>number</i> <i>option_data</i>	Yes	None	Specifies a container to be matched against any arbitrary incoming option specified by the client. <i>number</i> specifies the option number. <i>option_data</i> specifies the key to match for this container to be selected during address and option selection for the client. <i>option_data</i> is specified in expected form — quoted string, IP address, integer value — for well known options, or it can be optionally specified as a hexadecimal string of bytes if preceded by the characters 0x. For options that are not well known to the server, a hexadecimal string of bytes can be specified in the same fashion. Additionally, the <i>option_data</i> can indicate a regular expression to be compared against the string representation of the client's option data. Regular expressions are specified in a quoted string beginning "!" (double quote followed by an exclamation mark). The string form of options not well known to the server will be a hexadecimal string of bytes NOT preceded with the characters 0x.
	inoption <i>number</i> <i>option_data range</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
virtual	virtual fill <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy. fill means use all addresses in the container before going to the next container. rotate means select an address from the next pool in the list on each request. sfill and srotate are the same as fill and rotate, but a search is done to see if the client matches containers, vendors, or classes in the subnet. If a match is found that can supply an address, the address is taken from that container instead of following the policy. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet id.
	virtual sfill <i>id id ...</i>			
	virtual rotate <i>id id ...</i>			
	virtual srotate <i>id id ...</i>			
inorder:	inorder: <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy of fill, which means use all addresses in the container before going to the next container. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.
balance:	balance: <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy of rotate, which means use the next address in the next container. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.

Keyword	Form	Subcontainers?	Default Value	Meaning
supportBootp	supportBootp true	No	Yes	Specifies whether the current container and all below it (until overridden) should support BOOTP clients.
	supportBootp 1			
	supportBootp yes			
	supportBootp false			
	supportBootp 0			
	supportBootp no			
supportUnlistedclients	supportUnlistedclients BOTH	No	Both	<p>Specifies whether the current container and all below it (until overridden) should support unlisted clients. The value indicates whether all clients should be allowed access without specific client statements, DHCP clients only, BOOTP clients only, or no one.</p> <p>Note: The true and false values are supported for compatibility with previous versions and are deprecated. The true value corresponds to BOTH and the false value corresponds to NONE.</p>
	supportUnlistedclients DHCP			
	supportUnlistedclients BOOTP			
	supportUnlistedclients NONE			
	supportUnlistedclients true			
	supportUnlistedclients yes			
	supportUnlistedclients 1			
	supportUnlistedclients false			
	supportUnlistedclients no			
	supportUnlistedclients 0			
addressrecrddb	addressrecrddb <i>path</i>	No	None	<p>If specified, it works like the backupfile keyword. Only valid in the global or database container level.</p> <p>Note: This method is deprecated.</p>
backupfile	backupfile <i>path</i>	No	/etc/db_file.crbk	Specifies the file to use for database backups. Only valid in the global or database container level.
checkpointfile	checkpointfile <i>path</i>	No	/etc/db_file.chkpt	Specifies the database checkpoint files. The first checkpoint file is the <i>path</i> . The second checkpoint file is <i>path</i> with the last character replaced with a 2. So, the checkpoint file should not end in 2. Only valid in the global or database container level.

Keyword	Form	Subcontainers?	Default Value	Meaning
clientrecorddb	clientrecorddb <i>path</i>	No	/etc/db_file.cr	Specifies the database save file. The file contains all the client records the DHCP server has serviced. Only valid in the global or database container level.
bootstrapserver	bootstrapserver <i>IP address</i>	No	None	Specifies the server clients should use from which to TFTP files after receiving BOOTP or DHCP packets. This value fills in the siaddr field in the packet. This is valid at any container level.
giaddrfield	giaddrfield <i>IP address</i>	No	None	Specifies the giaddrfield for response packets. Note: This specification is illegal in the BOOTP and DHCP protocols, but some clients require the giaddr field to be the default gateway for the network. Because of this potential conflict, giaddrfield should only be used within a client container, although it can work at any level.
pingTime	pingTime <i>n time_unit</i>	No	3 seconds	Specifies the amount of time to wait for a ping response before handing out an address. The default time unit is hundredths of a second. The time unit value is defined in the note preceding this table. This is valid at any container level. The <i>time_unit</i> parameter is optional.
bootptime	bootptime <i>n time_unit</i>	No	-1, infinite	Specifies the amount of time to lease an address to a BOOTP client. The default is -1, which means infinite. The normal time unit values are available. The <i>time_unit</i> parameter is optional. This is valid at any container level.

Keyword	Form	Subcontainers?	Default Value	Meaning
AllRoutesBroadcast	allroutesbroadcast no	No	0	Specifies whether responses should be broadcast to all routes, if a broadcast response is required. This is valid at any container level. This is ignored by the operating system's DHCP servers, because the actual MAC address of the client, including RIFs, are stored for the return packet. This is valid at any container level.
	allroutesbroadcast false			
	allroutesbroadcast 0			
	allroutesbroadcast yes			
	allroutesbroadcast true			
	allroutesbroadcast 1			
addressassigned	addressassigned " <i>string</i> "	No	None	Specifies a quoted string to execute when an address is assigned to a client. The string should have two %s. The first %s is the client id in the form <i>type-string</i> . The second %s is an IP address in dotted quad format. This is valid at any container level.
addressreleased	addressreleased " <i>string</i> "	No	None	Specifies a quoted string to execute when an address is released by a client. The string should have one %s. The %s is the IP address being released in dotted quad format. This is valid at any container level.
appenddomain	appenddomain 0	No	No	Specifies whether to append the defined option 15 domain name to the client-suggested hostname in the event that the client does not suggest a domain name as well. This is valid at any container level.
	appenddomain no			
	appenddomain false			
	appenddomain 1			
	appenddomain yes			
	appenddomain true			
canonical	canonical 0	No	0	Specifies that the client id is in canonical format. This is valid only in the client container.
	canonical no			
	canonical false			
	canonical 1			
	canonical yes			
	canonical true			

Keyword	Form	Subcontainers?	Default Value	Meaning
leaseTimeDefault	leaseTimeDefault <i>n</i> <i>time_unit</i>	No	86400 seconds	Specifies the default lease time for clients. This is valid at any container level. The <i>time_unit</i> parameter is optional.
proxyarec	proxyarec never	No	usedhcpddnsplus	Specifies what options and methods should be used for A record updates in the DNS. never means never update the A record. usedhcpddns means use option 81 if the client specifies it. usedhcpddnsplus means use option 81 or option 12 and 15, if specified. always means do the A record update for all clients. XXXXprotected modifies the nsupdate command to make sure the client is allowed. standard is a synonym for always. protected is a synonym for alwaysprotected. This is valid at any container level.
	proxyarec usedhcpddns			
	proxyarec usedhcpddnsplus			
	proxyarec always			
	proxyarec usedhcpddnsprotected			
	proxyarec usedhcpddnsplusprotected			
	proxyarec alwaysprotected			
	proxyarec standard			
	proxyarec protected			
releasednsA	releasednsA " <i>string</i> "	No	None	Specifies the execution string to use when an address is released. The string is used to remove the A record associated with the address released. This is valid at any container level.
releasednsP	releasednsP " <i>string</i> "	No	None	Specifies the execution string to use when an address is released. The string is used to remove the PTR record associated with the address released. This is valid at any container level.

Keyword	Form	Subcontainers?	Default Value	Meaning
removedns	removedns <i>"string"</i>	No	None	Specifies the execution string to use when an address is released. The string is used to remove the PTR and A record associated with the address released. This is valid at any container level. Note: This is deprecated in favor of the releasednsA and releasednsP keywords.
updatedns	updatedns <i>"string"</i>	No	None	Specifies the execution string to use when an address is bound. The string is used to update both the A and the PTR record associated with the address. This is valid at any container level. Note: This is deprecated in favor of the updatednsA and updatednsP keywords.
updatednsA	updatednsA <i>"string"</i>	No	None	Specifies the execution string to use when an address is bound. The string is used to update the A record associated with the address. This is valid at any container level.
updatednsP	updatednsP <i>"string"</i>	No	None	Specifies the execution string to use when an address is bound. The string is used to update the PTR record associated with the address. This is valid at any container level.

Keyword	Form	Subcontainers?	Default Value	Meaning
hostnamepolicy	hostnamepolicy suggested	No	default	Specifies which hostname to return to the client. Default policy is to prefer the defined hostname and domain name over suggested names. Other policies imply strict adherence (for example: defined will return the defined name or none if no name is defined in the configuration). Also, policies using the always modifier will dictate the server to return the hostname option regardless of whether the client requested it through the parameter list option. Note that suggesting a hostname also implies requesting it, and hostnames can be suggested through option 81 or through options 12 and 15. This keyword is valid at any container level.
	hostnamepolicy resolved			
	hostnamepolicy always_resolved			
	hostnamepolicy defined			
	hostnamepolicy always_defined			
	hostnamepolicy default			
bootfilepolicy	bootfilepolicy suggested	No	suggested	Specifies a preference for returning the bootfile name to a client. suggested prefers the client-suggested bootfile name to any server-configured name. merge appends the client suggested name to the server-configured home directory. defined prefers the defined name over any suggested bootfile name. always returns the defined name regardless of whether the client requests the bootfile option through the parameter list option.
	bootfilepolicy merge			
	bootfilepolicy defined			
	bootfilepolicy always			

Keyword	Form	Subcontainers?	Default Value	Meaning
stealfromchildren	stealfromchildren true	No	No	Specifies whether the parent container should "steal" from children containers when the parent container runs out of addresses. This means that if you have a subnet with class defined with a range of addresses, those addresses are reserved for those clients that specify that class. If stealfromchildren is true, then addresses will be pulled from a child to try and satisfy the request. The default is to not steal an address.
	stealfromchildren 1			
	stealfromchildren yes			
	stealfromchildren false			
	stealfromchildren 0			
	stealfromchildren no			
homedirectory	homedirectory <i>path</i>	No	None	Specifies the home directory to use in the file section of the response packet. This can be specified at any container level. The bootfile policy defines how items specified in the file section of the incoming packet interact with the bootfile and the home directory statements.
bootfile	bootfile <i>path</i>	No	None	Specifies the bootfile to use in the file section of the response packet. This can be specified at any container level. The bootfile policy defines how items specified in the file section of the incoming packet interact with the bootfile and the home directory statements.

Keyword	Form	Subcontainers?	Default Value	Meaning
pxebootfile	pxebootfile system_architecture major_version minor_version bootfilename	No	None	Specifies the bootfile to be given for a client. This is used only when dhcpsd supports PXE clients (pxeservertype is dhcp_pxe_binld). The configuration file parser generates an error if the number of parameters after <code>pxebootfile</code> is less than four, and it ignores any additional parameters. <code>pxebootfile</code> can only be used within a container.

DHCP and Network Installation Management (NIM) Suggestions

The concept of dynamically assigning Internet Protocol (IP) addresses is fairly new. The following suggestions are provided to help with DHCP and NIM interaction.

1. When configuring objects in the NIM environment, use host names whenever possible. This allows you to use a dynamic name server that updates the IP addresses when the host name is converted to an IP address in the NIM environment.
2. Place the NIM master and the DHCP server on the same system. The DHCP server has an option in the update DNS string that, when set to NIM, attempts to keep the NIM objects out of those states that need static IP addresses when those addresses change.
3. For NIM clients, set the default lease time to twice the time it takes to install a client. This allows a leased IP address to be valid during the installation. After the installation, restart the client. DHCP will be started or will need to be configured, depending on the type of installation.
4. The **dhcpsd** server should be responsible for both the PTR and the A DNS records. When NIM reinstalls the machine, the file containing the RSA is deleted, and the client cannot update its records. The server updates the system records. To do this, change the `updatedns` line in `/etc/dhccpd.ini` to:

```
updatedns "/usr/sbin/dhccpaction '%s' '%s' '%s' '%s' NONE NONIM"
```

In the `/etc/dhccpsd.cnf` file, change the `updatedns` line to:

```
updatedns "/usr/sbin/dhccpaction '%s' '%s' '%s' '%s' BOTH NIM"
```

Note: When a NIM object is placed into the BOS installation-pending state, the **dhcpsd** server might pass arguments that are different from those originally intended. Minimize the time the client is in this pending state to avoid this situation.

These suggestions allow the NIM environment to work with dynamic clients.

For more information on Network Installation Management, see *AIX 5L Version 5.1 Network Installation Management Guide and Reference*.

Preboot Execution Environment Proxy DHCP Daemon (pxed)

The PXE Proxy DHCP server behaves much like a DHCP server by listening for ordinary DHCP client traffic and responding to certain client requests. However, unlike the DHCP server, the PXE Proxy DHCP server does not administer network addresses, and it only responds to clients that identify themselves as PXE clients. The responses given by the PXE Proxy DHCP server contain the mechanism by which the client locate sthe boot servers or the network addresses and descriptions of the supported, compatible boot servers.

Using a PXE Proxy DHCP server in addition to a DHCP server provides three key features. First, you can separate the administration of network addresses from the administration of boot images. Using two different processes on the same system, you can configure the boot information managed by the PXE Proxy DHCP server without disturbing or requiring access to the DHCP server configuration. Second, you can define multiple boot servers and let the PXE client select a particular server during boot time. Each boot server can, for example, offer a different type of operating system or system configuration. Finally, using the proxy server offers the ability to configure the PXE client to use multicast IP addressing to discover the location of the compatible boot servers.

The PXE Proxy DHCP server can be configured to run on the same system that is running the DHCP server or on a different system. Also, it can be configured to run on the same system that is also running the boot server daemon or on a different system.

The PXE Proxy DHCP Server

The PXED server is segmented into three main pieces, a database, a protocol engine, and a set of service threads, each with its own configuration information.

The PXED Database

The **db_file.dhcpo** database is used to generate the options to be sent to the client when the client send an REQUEST packet. The options returned by the database depend on the type of server chosen. This is set using the keyword **pxeservertype** in the **pxed.cnf** file.

Using the information in the configuration file, the database is primed and verified for consistency.

The PXED Protocol Engine

For AIX 4.3.1 and later, the PXED protocol engine is based on Intel's Preboot Execution Environment (PXE) Specification Version 2.1, but is still compatible with Intel's PXE Specification Version 1.1. The protocol engine uses the database to determine what information should be returned to the client.

PXED Threaded Operations

The last piece of the PXED server is actually a set of operations that are used to keep things running. Since the PXED server is threaded, these operations are actually set up as threads that occasionally do things to make sure everything is together.

The first thread, the *main* thread, handles the SRC requests (such as **startsrc**, **stopsrc**, **lssrc**, **traceson**, and **refresh**). This thread also coordinates all operations that affect all threads and handles signals. For example,

- A SIGHUP (-1) causes a refresh of all databases in the configuration file.
- A SIGTERM (-15) causes the server to gracefully stop.

The other thread processes packets. Depending on the server type, there can one or two threads. One thread listens on port 67 and the second one listens to port 4011. Each of these can handle a request from a client.

Configuring the PXED Server

By default, the PXED server is configured by reading the `/etc/pxed.cnf` file, which specifies the server's initial database of options and addresses. The server is started from the Web-based System Manager, from SMIT, or through SRC commands.

Configuring the PXED server is usually the hardest part of using PXED in your network. First, figure out what networks you need to have PXE clients on. The following example configures the `pxed` daemon to run on the same machine as the DHCP server:

```
pxeservertype      proxy_on_dhcp_server

subnet default
{
    vendor pxe
    {
        option      6      2      # Disable Multicast boot server discovery
        option      8      1 2    9.3.4.5 9.3.4.6 2 1 9.3.149.29
                                # The above option gives the list of bootservers
        option      9      0      "PXE bootstrap server" \
                                1      "Microsoft Windows NT Boot Server" \
                                2      "DOS/UNDI Boot Server"
        option      10     20     "seconds left before the first item in the boot menu is auto-selected"
    }
}
```

The suboptions in the vendor container are sent to PXE clients only if the client's IP address is in the subnet's IP address range (for example, 9.3.149.0 through 9.3.149.255).

The following example configures the `pxed` daemon to run on a different machine than the DHCP server:

```
subnet default
{
    vendor pxe
    {
        option      6      10     # The bootfile name is present in the client's initial pxed
                                # offer packet.
        option      8      1 2    9.3.4.5 9.3.4.6 2 1 9.3.149.29
                                # The above option gives the list of bootservers
        option      9      0      "PXE bootstrap server" \
                                1      "Microsoft Windows NT Boot Server" \
                                2      "DOS/UNDI Boot Server"
        option      10     20     "seconds left before the first item in the boot menu is auto-selected"
        bootstrapsrv 9.3.148.65
        pxebootfile 1 2 1 window.one
        pxebootfile 2 2 1 linux.one
        pxebootfile 1 2 1 hello.one
        client 6 10005a8ad14d any
        {
            pxebootfile 1 2 1 aix.one
            pxebootfile 2 2 1 window.one
        }
    }
}

Vendor pxeserver
{
    option      7      224.234.202.202
}
}
```

The `pxeservertype` keyword is not set in the configuration file so the default value is taken, which is `pdhcp_only`, meaning the PXED server is running on a different machine than the DHCP server. Given this configuration, the PXED server listens on two ports (67 and 4011) for clients' BINLD REQUEST/INFORM packets. The option 7 is sent to the BINLD server when the PXED server receives a REQUEST/INFORM packet on port 67 from BINLD and option 60 is set to PXED server.

The `db_file` database clause indicates which database method to use for processing this part of the configuration file. Comments begin with a pound sign (#). From the # to the end of the line are ignored by the PXED server. Each option line is used by the server to tell the client what to do. PXED Vendor Container Suboptions describes the currently supported and known options. See PXED Server File Syntax for General Server Operation for ways to specify options that the server does not know about.

The Configuration File

The configuration file has an address section and an option definition section, which are based on the concept of containers that hold options, modifiers, and, potentially, other containers.

A *container* (basically, a method to group options) uses an identifier to classify clients into groups. The container types are subnet, class, vendor, and client. Currently, there is not a generic user-definable container. The identifier uniquely defines the client so that the client can be tracked if, for example, it moves between subnets. More than one container type can be used to define client access.

Options are identifiers that are returned to the client, such as default gateway and DNS address.

Containers: When the DHCP server receives a request, the packet is parsed and identifying keys determine which containers, options, and addresses are extracted

.

The previous example shows a subnet container. Its identifying key is the client's position in the network. If the client is from that network, then it falls into that container.

Each type of container uses a different option to identify a client:

- The subnet container uses the `giaddr` field or the interface address of the receiving interface to determine which subnet the client came from.
- The class container uses the value in option 77 (User Site Class Identifier).
- The vendor uses the value in option 60 (Vendor Class Identifier).
- The client container uses the option 61 (Client Identifier) for PXE clients and the `chaddr` field in the BOOTP packet for BOOTP clients.

Except for subnets, each container allows the specification of the value that it will match including regular expression matching.

There is also an implicit container, the *global* container. Options and modifiers in the global container apply to all containers unless overridden or denied. Most containers can be placed inside other containers implying a scope of visibility. Containers might or might not have address ranges associated with them. Subnets, by their nature, have ranges associated with them.

The basic rules for containers and subcontainers are as follows:

- All containers are valid at the global level.
- Subnets can never be placed inside other containers.
- Restricted containers cannot have regular containers of the same type within them. (For example, a container with an option that only allows a class of Accounting cannot include a container with an option that allows all classes that start with the letter "a." This is illegal.)
- Restricted client containers cannot have subcontainers.

Given the above rules, you can generate a hierarchy of containers that segment your options into groups for specific clients or sets of clients.

If a client matches multiple containers, how are options and addresses handed out? The DHCP server receives messages, it passes the request to the database (**db_file** in this case), and a container list is

generated. The list is presented in order of depth and priority. Priority is defined as an implicit hierarchy in the containers. Strict containers are higher priority than regular containers. Clients, classes, vendors, and finally subnets are sorted, in that order, and within container type by depth. This generates a list ordered by most specific to least specific. For example:

```
Subnet 1
--Class 1
--Client 1
Subnet 2
--Class 1
----Vendor 1
----Client 1
--Client 1
```

The above example shows two subnets, Subnet 1 and Subnet 2. There is one class name, Class 1, one vendor name, Vendor 1, and one client name, Client 1. Class 1 and Client 1 are defined in multiple places. Because they are in different containers, their names can be the same but values inside them can be different. If Client 1 sends a message to the DHCP server from Subnet 1 with Class 1 specified in its option list, the DHCP server would generate the following container path:

```
Subnet 1, Class 1, Client 1
```

The most specific container is listed last. To get an address, the list is examined in reverse hierarchy to find the first available address. Then, the list is examined in forward hierarchy to get the options. Options override previous values unless an option *deny* is present in the container. Also, since Class 1 and Client 1 are in Subnet 1, they are ordered according to the container priority. If the same client is in Subnet 2 and sends the same message, the container list generated is:

```
Subnet 2, Class 1, Client 1 (at the Subnet 2 level), Client 1 (at the Class 1 level)
```

Subnet 2 is listed first, then Class 1, then the Client 1 at the Subnet 2 level (because this client statement is only one level down in the hierarchy). The hierarchy implies that a client matching the first client statement is less specific than the client matching Client 1 of Class 1 within Subnet 2.

Priority selected by depth within the hierarchy is not superseded by the priority of the containers themselves. For example, if the same client issues the same message and specifies a vendor identifier, the container list is:

```
Subnet 2, Class 1, Vendor 1, Client 1 (at Subnet 2 level), Client 1 (at Class 1 level)
```

Container priority improves search performance because it follows a general concept that client containers are the most specific way to define one or more clients. The class container holds less specific addresses than a client container; vendor is even less specific; and subnet is the least specific.

Addresses and Address Ranges: Any container type can have associated addresses ranges; subnets must have. Each range within a container must be a subset of the parent container's range and must not overlap with other containers' ranges. For example, if a class is defined within a subnet and the class has a range, the range must be a subset of the subnet's range. Also, the range within that class container cannot overlap with any other ranges at its level.

Ranges can be expressed on the container line and modified by range and exclude statements to allow for disjoint address sets associated with a container. So, if you have the top ten addresses and the second ten addresses of a subnet available, the subnet could specify these addresses by range in the subnet clause to reduce both memory use and the chance of address collision with other clients not in the specified ranges.

Once an address has been selected, any subsequent container in the list that contains address ranges is removed from the list along with its children. The reason for this is that network-specific options in removed containers are not valid if an address is not used from within that container.

Options: After the list has been culled to determine addresses, a set of options is generated for the client. In this selection process, options overwrite previously selected options unless a *deny* is encountered, in which case, the denied option is removed from the list being sent to the client. This method allows inheritance from parent containers to reduce the amount of data that must be specified.

Logging: Logging parameters are specified in a container like the database, but the container keyword is **logging_info**. When learning to configure PXED, it is advisable to turn logging to its highest level. Also, it is best to specify the logging configuration prior to any other configuration file data to ensure that configuration errors are logged after the logging subsystem is initialized. Use the **logitem** keyword to turn on a logging level or remove the **logitem** keyword to disable a logging level. Other keywords for logging allow the specification of the log filename, file size, and the number of rotating log files.

Performance Considerations: It is important to understand that certain configuration keywords and the structure of the configuration file have an effect on the memory use and performance of the PXED server.

First, excessive memory use can be avoided by understanding the inheritance model of options from parent to child containers. In an environment that supports no unlisted clients, the administrator must explicitly list each client in the file. When options are listed for any specific client, the server uses more memory storing that configuration tree than when options are inherited from a parent container (for example, the subnet, network, or global containers). Therefore, the administrator should verify whether any options are repeated at the client level within the configuration file and, if so, determine whether these options can be specified in the parent container and shared by the set of clients as a whole.

Also, when using the **logitem** entries INFO and TRACE, numerous messages are logged during the processing of every PXE client's message. Appending a line to the log file can be an expensive operation; therefore, limiting the amount of logging improves the performance of the PXED server. When an error with the PXED server is suspected, logging can be dynamically re-enabled using the SRC traceson command.

PXE Vendor Container Suboptions

When supporting a PXE client, the DHCP server passes the following option to the BINLD server that BINLD uses to configure itself:

Opt Num	Default Data Type	Can Specify?	Description
6	Decimal number	Yes	<p>PXE_DISCOVERY_CONTROL. Limit 0-16. This is a bit field. Bit 0 is the least significant bit.</p> <p>bit 0 If set, disables broadcast discovery.</p> <p>bit 1 If set, disables multicast discovery.</p> <p>bit 2 If set, only uses/accepts servers in PXE_BOOT_SERVERS.</p> <p>bit 3 If set, and a bootfile name is present in the initial PXED offer packet, downloads the bootfile (does not prompt/menu/discover boot server).</p> <p>bit 4-7 Must be 0. If this option is not supplied then client assumes all bits to be equal to 0.</p>
7	One dotted quad	Yes	<p>Multicast IP address. Boot server discovery multicast IP address. Boot servers capable of multicast discovery must listen on this multicast address. This option is required if the multicast discovery disable bit (bit 1) in the PXE_DISCOVERY_CONTROL option is not set.</p>

Opt Num	Default Data Type	Can Specify?	Description
8	<i>Boot server type(0-65535)</i>	Yes	<p>PXE_BOOT_SERVERS <i>IP address count (0-256)</i></p> <p>Type 0 Microsoft Windows <i>IP address...IP address NT Boot Server Boot server type IP address</i></p> <p>Type 1 Intel LCM Boot Server <i>count IP address</i> ...</p> <p>Type 3 DOS/UNDI Boot Server <i>IP address</i></p> <p>Type 4 NEC ESMPRO Boot Server</p> <p>Type 5 IBM WSoD Boot Server</p> <p>Type 6 IBM LCCM Boot Server</p> <p>Type 7 CA Unicenter TNG Boot Server.</p> <p>Type 8 HP OpenView Boot Server.</p> <p>Type 9 through 32767 Reserved</p> <p>Type 32768 through 65534 Vendor use</p> <p>Type 65535 PXE API Test Server.</p> <p>If <i>IP address count</i> is zero for a server type then the client may accept offers from any boot server of that type. Boot Servers do not respond to discovery requests of types they do not support.</p>
9	<i>Boot server type (0-65535)</i>	Yes	<p>PXE_BOOT_MENU "<i>description</i>" Boot server boot "<i>order</i>" is implicit in the type. "<i>description</i>" ...<i>menu order</i>.</p>
10	<i>Timeout in seconds (0-255)</i>	Yes	<p>PXE_MENU_PROMPT "<i>prompt</i>" The timeout is the number of seconds to wait before auto-selecting the first boot menu item. On the client system, the prompt is displayed followed by the number of seconds remaining before the first item in the boot menu is auto-selected. If the F8 key is pressed on the client system, then a menu is displayed. If this option is provided to the client, then the menu is displayed without prompt and timeout. If the timeout is 0, then the first item in the menu is auto-selected. If the timeout is 255, the menu and prompt is displayed without auto-selecting or timeout.</p>

PXED Server File Syntax for General Server Operation

Note: Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.

Keyword	Form	Subcontainers?	Default Value	Meaning
database	database <i>db type</i>	Yes	None	The primary container that holds the definitions for the address pools, options, and client access statements. <i>db type</i> is the name of a module that is loaded to process this part of the file. The only value currently available is db_file .
logging_info	logging_info	Yes	None	The primary logging container that defines the logging parameters.
logitem	logitem NONE	No	All default to not enabled.	Enables the logging level. Multiple lines are allowed.
	logitem SYSERR			
	logitem OBJERR			
	logitem PROTOCOL			
	logitem PROTERR			
	logitem WARN			
	logitem WARNING			
	logitem CONFIG			
	logitem EVENT			
	logitem PARSEERR			
	logitem ACTION			
	logitem ACNTING			
	logitem STAT			
	logitem TRACE			
logitem RTRACE				
logitem START				
numLogFiles	numLogFiles <i>n</i>	No	0	Specifies the number of log files to create. The log rotates when the first one fills. <i>n</i> is the number of files to create.
logFileSize	logFileSize <i>n</i>	No	0	Specifies the size of each log file in 1024-byte units.

Keyword	Form	Subcontainers?	Default Value	Meaning
logFileName	logFileName <i>path</i>	No	None	Specifies the path to the first log file. The original log file is named <i>filename</i> or <i>filename.extension</i> . The <i>filename</i> must be eight or fewer characters. When a file is rotated, it is renamed beginning with the base <i>filename</i> , then either appending a number or replacing the extension with a number. For example, if the original file name is <i>file</i> , the rotated file name becomes <i>file01</i> . If the original file name is <i>file.log</i> , it becomes <i>file.01</i> .
pxeservertype	pxeservertype <i>servertype</i>	No	dhcp_only	Indicates the type of dhcpsd server it is. <i>servertype</i> can be proxy_on_dhcp_server , which means that PXED is running on the same machine as the DHCP server and it is listening for PXE client requests on port 4011 only, or the default value of pdhcp_only , which means the PXED is running on a separate machine and it has to listen for client packets on port 67 and 4011.

PXED Server File Syntax for db_file Database

Notes:

- Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.
- Items that are specified in one container can be overridden inside a subcontainer. For example, you could globally define BOOTP clients, but within a certain subnet allow BOOTP clients by specifying the supportBootp keyword in both containers.
- The client, class, and vendor containers allow for regular expression support. For class and vendor, a quoted string with the first character after the quote being an exclamation point (!) indicates that the rest of the string should be treated as a regular expression. The client container allows for regular expressions on both the hwtype and the hwaddr fields. A single string is used to represent both fields with the following format:

```
decimal_number-data
```

If decimal_number is zero, then data is an ASCII string. If any other number, data is hex digits.

Keyword	Form	Subcontainers?	Default Value	Meaning
subnet	subnet default	Yes	None	Specifies a subnet that does not have any range. The subnet is used by the server only when it is responding to INFORM packet from the client.
subnet	subnet <i>subnet id netmask</i>	Yes	None	Specifies a subnet and a pool of addresses. All addresses are assumed to be in the pool unless a range is specified on the line or addresses are modified later in the container by a range or exclude statement. The optional range is a pair of IP addresses in dotted quad format separated by a dash. An optional label and priority can be specified. These are used by virtual subnets to identify and order the subnets in the virtual subnet. The label and priority are separated by a colon. These containers are only allowed at the global or database container level.
	subnet <i>subnet id netmask range</i>			
	subnet <i>subnet id netmask label:priority</i>			
	subnet <i>subnet id netmask range label:priority</i>			
subnet	subnet <i>subnet id range</i>	Yes	None	Specifies a subnet that goes within a network container. It defines a range of addresses that is the whole subnet unless the optional range part is specified. The netmask associated with the subnet is taken from the surrounding network container. Note: This method is deprecated in favor of the other subnet forms.

Keyword	Form	Subcontainers?	Default Value	Meaning
option	option <i>number data</i> ...	No	None	Specifies an option to send to a client or, in the case of deny, an option to prevent from being sent to the client. The optional * deny clause means all options not specified in the current container are not to be returned to the client. option <i>numberdeny</i> only denies the specified option. <i>number</i> is an unsigned 8-bit integer. <i>data</i> is specific to the option (see above) or can be specified as a quoted string (indicating ASCII text) or <i>0xhexdigits</i> or hex" <i>hexdigits</i> " or hex " <i>hexdigits</i> ". If the option is in a vendor container, the option will be encapsulated with other options in an option 43.
	option <i>numberdeny</i>			
	option * deny			
exclude	exclude <i>an IP address</i>	No	None	Modifies the range on the container in which the exclude statement is in. The exclude statement is not valid in the global or database container levels. The exclude statement removes the specified address or range from the current range on the container. The exclude statement allows you to create non-contiguous ranges for subnets or other containers.
	exclude <i>dotted_quad-dotted_quad</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
range	range <i>IP_address</i>	No	None	Modifies the range on the container in which the range statement is in. The range statement is not valid in the global or database container levels. If the range is the first in the container that does not specify a range on the container definition line, then the range for the container becomes the range specified by the range statement. Any range statement after the first range or all range statements for a containers that specifies ranges in its definition are added to the current range. With the range statement, a single address or set of addresses can be added to the range. The range must fit inside the subnet container definition.
	range <i>dotted_quad-dotted_quad</i>			
client	client <i>hwtype hwaddr</i> NONE	Yes	None	Specifies a client container that denies the client specified by the <i>hwaddr</i> and <i>hwtype</i> from getting an address. If <i>hwtype</i> is 0, then <i>hwaddr</i> is an ASCII string. Otherwise, <i>hwtype</i> is the hardware type for the client and <i>hwaddr</i> is the hardware address of the client. If the <i>hwaddr</i> is a string, then quotes are accepted around the string. If the <i>hwaddr</i> is a hexstring, then the address may be specified by <i>0hexdigits</i> or <i>hex digits</i> . <i>range</i> allows the client specified by the <i>hwaddr</i> and <i>hwtype</i> to get an address in the <i>range</i> . Must be regular expressions to match multiple clients.
	client <i>hwtype hwaddr</i> ANY			
	client <i>hwtype hwaddr</i> <i>dotted_quad</i>			
	client <i>hwtype hwaddr</i> <i>range</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
class	class <i>string</i>	Yes	None	Specifies a class container with name <i>string</i> . String can be quoted or not. If quoted, the quotes are removed before comparison. Quotes are required for strings with spaces or tabs. This container is valid at any level. A range can be supplied to indicate a set of addresses to hand out to a client with this class. The range is either a single dotted quad IP address or two dotted quad IP addresses separated by a dash.
	class <i>string range</i>			
network	network <i>network id netmask</i>	Yes	None	Specifies a network ID using class information (for example, 9.3.149.0 with a netmask of 255.255.255.0 would be network 9.0.0.0 255.255.255.0). This version of the network container is used to hold subnets with the same network ID and netmask. When a range is provided, all the addresses in the range are in the pool. The range must be in the network ID's network. This uses class full addressing. This is only valid in the global or database container level. Note: The network keyword is deprecated in favor of the subnet container.
	network <i>network id</i>			
	network <i>network id range</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
vendor	vendor <i>vendor_id</i>	Yes	None	Specifies a vendor container. Vendor containers are used to return option 43 to the client. The vendor id may be specified in a quoted string or a binary string in the form <i>0xhexdigits</i> or <i>hex" digits"</i> . An optional range may be placed after the vendor id. The range is specified as two dotted quads separated by a dash. After the optional range, an optional hexstring or ASCII string can be specified as the first part of the option 43. If options are in the container, they are appended to the option 43 data. After all options are processed an End Of Option List Option is appended to the data. To return options outside of an option 43, use a regular expression client that matches all clients to specify normal options to return based on the vendor ID.
	vendor <i>vendor_id</i> hex""			
	vendor <i>vendor_id</i> hex ""			
	vendor <i>vendor_id</i> 0xdata			
	vendor <i>vendor_id</i> ""			
	vendor <i>vendor_id</i> range			
	vendor <i>vendor_id</i> range hex""			
	vendor <i>vendor_id</i> range hex ""			
	vendor <i>vendor_id</i> range 0xdata			
	vendor <i>vendor_id</i> range ""			

Keyword	Form	Subcontainers?	Default Value	Meaning
inoption	inoption <i>number</i> <i>option_data</i>	Yes	None	Specifies a container to be matched against any arbitrary incoming option specified by the client. <i>number</i> specifies the option number. <i>option_data</i> specifies the key to match for this container to be selected during address and option selectoin for the client. <i>option_data</i> is specified in expected form — quoted string, IP address, integer value — for well known options, or it can be optionally speicified as a hexadecimal string of bytes if preceded by the characters 0x. For options that are not well known to the server, a hexadecimal string of bytes can be specified in the same fashion. Additionally, the <i>option_data</i> can indicate a regular expression to be compared against the string representation of the client's option data. Regular expressions are specified in a quoted string beginning "!" (double quote followed by an exclamation mark). The string form of options not well known to the server will be a hexadecimal string of bytes NOT preceded with the characters 0x.
	inoption <i>number</i> <i>option_data range</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
virtual	virtual fill <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy. <i>fill</i> means use all addresses in the container before going to the next container. <i>rotate</i> means select an address from the next pool in the list on each request. <i>sfill</i> and <i>srotate</i> are the same as <i>fill</i> and <i>rotate</i> , but a search is done to see if the client matches containers, vendors, or classes in the subnet. If a match is found that can supply an address, the address is taken from that container instead of following the policy. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet id.
	virtual sfill <i>id id ...</i>			
	virtual rotate <i>id id ...</i>			
	virtual srotate <i>id id ...</i>			
inorder:	inorder: <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy of fill, which means use all addresses in the container before going to the next container. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.
balance:	balance: <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy of rotate, which means use the next address in the next container. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.

Keyword	Form	Subcontainers?	Default Value	Meaning
bootstrapserver	bootstrapserver <i>IP address</i>	No	None	Specifies the server clients should use from which to TFTP files after receiving BOOTP or DHCP packets. This value fills in the siaddr field in the packet. This is valid at any container level.
giaddrfield	giaddrfield <i>IP address</i>	No	None	Specifies the giaddrfield for response packets. Note: This specification is illegal in the BOOTP and DHCP protocols, but some clients require the giaddr field to be the default gateway for the network. Because of this potential conflict, giaddrfield should only be used within a client container, although it can work at any level.
bootfile	bootfile <i>path</i>	No	None	Specifies the bootfile to use in the file section of the response packet. This can be specified at any container level. The bootfile policy defines how items specified in the file section of the incoming packet interact with the bootfile and the home directory statements.
pxebootfile	pxebootfile <i>System Arch MajorVer MinorVer Bootfilename</i>	No	None	Specifies the bootfile to be given to a client. The config file parser generates an error if the number of parameters after the keyword is less than 4 and ignore if more than 4. This keyword can be used only in a container.

For details about other options, see DHCP Server File Known Options.

Boot Image Negotiation Layer Daemon (BINLD)

The Boot Image Image Negotiation Layer daemon (BINLD) server is the third stage of contact for preboot execution environment (PXE) clients. After communicating with the DHCP server to obtain an IP address, and after communication with the PXE Proxy DHCP server to obtain the location of the boot server, the boot server is contacted to get the filename and location from which to download the boot image. The PXE client can return to communicate with the boot server multiple times in the course of booting if the client requires multiple files in its boot process.

The final stage in the PXE network boot is to download the boot image given by the boot server. The location of the TFTP server and the filename that is to be downloaded is given by the boot server to the PXE client.

The BINLD Server

Beginning with AIX 4.3.3 update, the BINLD server is segmented into three main pieces: a database, a protocol engine, and a set of service threads, each with its own configuration information.

The BINLD Database

The **db_file.dhcpo** database is used to generate the options that respond to a client's REQUEST packet. The options returned by the database depend on the type of server chosen. Options are set using the keyword **pxeservertype** in the **binld.cnf** file.

Using the information in the configuration file, the database is primed and verified for consistency.

The BINLD Protocol Engine

The PXED protocol engine is based on the Intel's Preboot Execution Environment (PXE) Specification Version 2.1, but is still compatible with Intel's PXE Specification Version 1.1. The protocol engine uses the database to determine what information should be returned to the client.

BINLD Threaded Operations

The last piece of the BINLD server is actually a set of operations that are used to keep things running. Since the BINLD server is threaded, these operations are actually set up as threads that occasionally do things to make sure everything is together.

The first thread, the *main* thread, handles the SRC requests (such as **startsrc**, **stopsrc**, **lssrc**, **traceson**, and **refresh**). This thread also coordinates all operations that affect all threads and handles signals. For example,

- A SIGHUP (-1) causes a refresh of all databases in the configuration file.
- A SIGTERM (-15) causes the server to gracefully stop.

The other thread processes packets. Depending on the server type, there can one or two threads. One thread listens on port 67 and the second to port 4011. Each can handle a request from a client.

Configuring BINLD

By default, the BINLD server is configured by reading the **/etc/binld.cnf** file, which specifies the server's initial database of options and addresses. The server is started from the Web-based System Manager, from SMIT, or through SRC commands.

Configuring the BINLD server is usually the hardest part of using BINLD in your network. First, figure out what networks you need to have PXE clients on. The following example configures a BINLD server to run on the same machine as the DHCP server:

```
pxeservertype      binld_on_dhcp_server

subnet default
{
```

```

vendor pxe
{
    bootstrapserver 9.3.149.6 #TFTP server IP address
    pxebootfile 1 2 1 window.one 1 0
    pxebootfile 2 2 1 linux.one 2 3
    pxebootfile 1 2 1 hello.one 3 4
    client 6 10005a8ad14d any
    {
        pxebootfile 1 2 1 aix.one 5 6
        pxebootfile 2 2 1 window.one 6 7
    }
}
}

```

Given the above configuration, the BINLD server listens for client's unicast packets on port 4011 and Multicast packets on port 4011 if BINLD gets the Multicast Address from the dhcpd/pxed. The BINLD server responds to client REQUEST/INFORM packets with the bootfile name and TFTP server's IP address. If BINLD does not find the bootfile with a matching Layer specified by the client, then it tries to find a bootfile for the next layer. The BINLD does not respond when there is no boot file that matches the client requirements (*Type, SystemArch, MajorVers, MinorVers, and Layer*).

The following example configures BINLD to run on a separate machine (that is, DHCP / PXED is not running on the same machine).

```

subnet 9.3.149.0 255.255.255.0
{
    vendor pxe
    {
        bootstrapserver 9.3.149.6 # TFTP server ip address.
        pxebootfile 1 2 1 window.one 1 0
        pxebootfile 2 2 1 linux.one 2 3
        pxebootfile 1 2 1 hello.one 3 4
        client 6 10005a8ad14d any
        {
            pxebootfile 1 2 1 aix.one 5 6
            pxebootfile 2 2 1 window.one 6 7
        }
    }
}

```

In the above example, the *pxeservertype* is not set, so the default servertype is **binld_only**. The BINLD server listens for client's unicast packets on port 4011, broadcast & unicast packets on port 67, and Multicast packets on port 4011 if BINLD gets the Multicast Address from the dhcpd/pxed. The bootfile name and TFTP server IP address is sent to a PXE client only if the client's IP address is in the subnet's IP address range (9.3.149.0 through 9.3.149.255).

The following example configures BINLD to run on the same machine as the PXED server:

```

pxeservertype binld_on_proxy_server
subnet default
{
    vendor
    {
        bootstrapserver 9.3.149.6 # TFTP server ip address.
        pxebootfile 1 2 1 window.one 1 0
        pxebootfile 2 2 1 linux.one 2 3
        pxebootfile 1 2 1 hello.one 3 4
        client 6 10005a8ad14d any
        {
            pxebootfile 1 2 1 aix.one 5 6
        }
    }
}

```



```

    pxebootfile 2 2 1 window.one 6 7
  }
}

```

In this configuration, the BINLD server only listens on port 4011 for Multicast packets only if BINLD gets Multicast address from the dhcpd/pxed. If it does not receive any multicast address, then BINLD exits and an error message is logged to the log file.

The database `db_file` clause indicates which database method to use for processing this part of the configuration file. Comments begin with a pound sign (#). From the # to the end of the line are ignored by the PXED server. Each option line is used by the server to tell the client what to do. PXE Vendor Container Suboptions describes the currently supported and known suboptions. See BINLD Server File Syntax for General Server Operation for ways to specify options that the server does not know about.

The Configuration File

The configuration file has an address section and an option definition section, which are based on the concept of containers that hold options, modifiers, and, potentially, other containers.

A *container* (basically, a method to group options) uses an identifier to classify clients into groups. The container types are subnet, class, vendor, and client. Currently, there is not a generic user-definable container. The identifier uniquely defines the client so that the client can be tracked if, for example, it moves between subnets. More than one container type can be used to define client access.

Options are identifiers that are returned to the client, such as default gateway and DNS address.

Containers: When the DHCP server receives a request, the packet is parsed and identifying keys determine which containers, options, and addresses are extracted.

The previous example shows a subnet container. Its identifying key is the client's position in the network. If the client is from that network, then it falls into that container.

Each type of container uses a different option to identify a client:

- The subnet container uses the `giaddr` field or the interface address of the receiving interface to determine which subnet the client came from.
- The class container uses the value in option 77 (User Site Class Identifier).
- The vendor uses the value in option 60 (Vendor Class Identifier).
- The client container uses the option 61 (Client Identifier) for PXED clients and the `chaddr` field in the BOOTP packet for BOOTP clients.

Except for subnets, each container allows the specification of the value that it matches, including regular expression matching.

There is also an implicit container, the *global* container. Options and modifiers placed in the global container apply to all containers unless overridden or denied. Most containers can be placed inside other containers implying a scope of visibility. Containers may or may not have address ranges associated with them. Subnets, by their nature, have ranges associated with them.

The basic rules for containers and subcontainers are as follows:

- All containers are valid at the global level.
- Subnets can never be placed inside other containers.
- Restricted containers cannot have regular containers of the same type within them. (For example, a container with an option that only allows a class of `Accounting` cannot include a container with an option that allows all classes that start with the letter "a". This is illegal.)
- Restricted client containers cannot have subcontainers.

Given the above rules, you can generate a hierarchy of containers that segment your options into groups for specific clients or sets of clients.

If a client matches multiple containers, how are options and addresses handed out? The DHCP server receives messages, it passes the request to the database (**db_file** in this case), and a container list is generated. The list is presented in order of depth and priority. Priority is defined as an implicit hierarchy in the containers. Strict containers are higher priority than regular containers. Clients, classes, vendors, and finally subnets are sorted, in that order, and within container type by depth. This generates a list ordered by most specific to least specific. For example:

```
Subnet 1
--Class 1
--Client 1
Subnet 2
--Class 1
---Vendor 1
---Client 1
--Client 1
```

The example shows two subnets, Subnet 1 and Subnet 2. There is one class name, Class 1, one vendor name, Vendor 1, and one client name, Client 1. Class 1 and Client 1 are defined in multiple places. Because they are in different containers, their names can be the same but values inside them may be different. If Client 1 sends a message to the DHCP server from Subnet 1 with Class 1 specified in its option list, the DHCP server would generate the following container path:

```
Subnet 1, Class 1, Client 1
```

The most specific container is listed last. To get an address, the list is examined in reverse hierarchy to find the first available address. Then, the list is examined in forward hierarchy to get the options. Options override previous values unless an option *deny* is present in the container. Also, since Class 1 and Client 1 are in Subnet 1, they are ordered according to the container priority. If the same client is in Subnet 2 and sends the same message, the container list generated is:

```
Subnet 2, Class 1, Client 1 (at the Subnet 2 level), Client 1 (at the Class 1 level)
```

Subnet 2 is listed first, then Class 1, then the Client 1 at the Subnet 2 level (because this client statement is only one level down in the hierarchy). The hierarchy implies that a client matching the first client statement is less specific than the client matching Client 1 of Class 1 within Subnet 2.

Priority selected by depth within the hierarchy is not superseded by the priority of the containers themselves. For example, if the same client issues the same message and specifies a vendor identifier, the container list is:

```
Subnet 2, Class 1, Vendor 1, Client 1 (at Subnet 2 level), Client 1 (at Class 1 level)
```

Container priority improves search performance because it follows a general concept that client containers are the most specific way to define one or more clients. The class container holds less specific addresses than a client container; vendor is even less specific; and subnet is the least specific.

Addresses and Address Ranges: Any container type may have associated addresses ranges; subnets must have. Each range within a container must be a subset of the parent container's range and must not overlap with other containers' ranges. For example, if a class is defined within a subnet and the class has a range, the range must be a subset of the subnet's range. Also, the range within that class container cannot overlap with any other ranges at its level.

Ranges can be expressed on the container line and modified by range and exclude statements to allow for disjoint address sets associated with a container. So, if you have the top ten addresses and the second

ten addresses of a subnet available, the subnet could specify these addresses by range in the subnet clause to reduce both memory use and the chance of address collision with other clients not in the specified ranges.

Once an address has been selected, any subsequent container in the list that contains address ranges is removed from the list along with its children. The reason for this is that network-specific options in removed containers are not valid if an address is not used from within that container.

Options: After the list has been culled to determine addresses, a set of options is generated for the client. In this selection process, options overwrite previously selected options unless a *deny* is encountered, in which case, the denied option is removed from the list being sent to the client. This method allows inheritance from parent containers to reduce the amount of data that must be specified.

Logging: Logging parameters are specified in a container like the database, but the container keyword is **logging_info**. When learning to configure PXED, it is advisable to turn logging to its highest level. Also, it is best to specify the logging configuration prior to any other configuration file data to ensure that configuration errors are logged after the logging subsystem is initialized. Use the **logitem** keyword to turn on a logging level or remove the **logitem** keyword to disable a logging level. Other keywords for logging allow the specification of the log filename, file size, and the number of rotating log files.

Performance Considerations: It is important to understand that certain configuration keywords and the structure of the configuration file have an effect on the memory use and performance of the PXED server.

First, excessive memory use can be avoided by understanding the inheritance model of options from parent to child containers. In an environment that supports no unlisted clients, the administrator must explicitly list each client in the file. When options are listed for any specific client, the server uses more memory storing that configuration tree than when options are inherited from a parent container (for example, the subnet, network, or global containers). Therefore, the administrator should verify whether any options are repeated at the client level within the configuration file and, if so, determine whether these options can be specified in the parent container and shared by the set of clients as a whole.

Also, when using the **logitem** entries INFO and TRACE, numerous messages are logged during the processing of every PXE client's message. Appending a line to the log file can be an expensive operation; therefore, limiting the amount of logging improves the performance of the PXED server. When an error with the PXED server is suspected, logging can be dynamically re-enabled using the SRC traceson command.

BINLD Server File Syntax for General Server Operation

Note: Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.

Keyword	Form	Subcontainers?	Default Value	Meaning
database	database <i>db type</i>	Yes	None	The primary container that holds the definitions for the address pools, options, and client access statements. <i>db type</i> is the name of a module that is loaded to process this part of the file. The only value currently available is db_file .
logging_info	logging_info	Yes	None	The primary logging container that defines the logging parameters.

Keyword	Form	Subcontainers?	Default Value	Meaning
logitem	logitem NONE	No	All default to not enabled.	Enables the logging level. Multiple lines are allowed.
	logitem SYSERR			
	logitem OBJERR			
	logitem PROTOCOL			
	logitem PROTERR			
	logitem WARN			
	logitem WARNING			
	logitem CONFIG			
	logitem EVENT			
	logitem PARSEERR			
	logitem ACTION			
	logitem ACNTING			
	logitem STAT			
	logitem TRACE			
logitem RTRACE				
logitem START				
numLogFiles	numLogFiles <i>n</i>	No	0	Specifies the number of log files to create. The log rotates when the first one fills. <i>n</i> is the number of files to create.
logFileSize	logFileSize <i>n</i>	No	0	Specifies the size of each log file in 1024-byte units.
logFileName	logFileName <i>path</i>	No	None	Specifies the path to the first log file. The original log file is named <i>filename</i> or <i>filename.extension</i> . The <i>filename</i> must be eight or fewer characters. When a file is rotated, it is renamed beginning with the base <i>filename</i> , then either appending a number or replacing the extension with a number. For example, if the original file name is <i>file</i> , the rotated file name becomes <i>file01</i> . If the original file name is <i>file.log</i> , it becomes <i>file.01</i> .

Keyword	Form	Subcontainers?	Default Value	Meaning
pxeservertype	pxeservertype <i>servertype</i>	No	dhcp_only	Indicate the type of dhcpserver it is. <i>servertype</i> can be one of the following binld_on_dhcp_server This means that BINLD is running on the same machine as DHCP server and it is listening for PXE Client request on port 4011 and Multicast address if received from the DHCP / PXED. binld_on_proxy_server This means that BINLD is running on the same machine as PXED server and it is listening for PXE Client's request on Multicast address if received from the DHCP / PXED. The default value is binld_only ie the BINLD is running on a separate machine and it has to listen for client's packets on port 67 , 4011 and Multicast address if received from the DHCP / PXED.
dhcp_or_proxy_address	dhcp_or_proxy_address <i>IP address</i>	No	None	This gives the IP address of dhcp or pxed server to which the BINLD server can send an Unicast packet of type REQUEST/INFORM to receive the Multicast Address. This keyword is defined only when the dhcp or pxed are on a different subnet than BINLD.

BINLD Server File Syntax for db_file Database

Notes:

- Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.
- Items that are specified in one container can be overridden inside a subcontainer. For example, you could globally define BOOTP clients, but within a certain subnet allow BOOTP clients by specifying the supportBootp keyword in both containers.
- The client, class, and vendor containers allow for regular expression support. For class and vendor, a quoted string with the first character after the quote being an exclamation point (!) indicates that the rest of the string should be treated as a regular expression. The client container allows for regular expressions on both the hwtype and the hwaddr fields. A single string is used to represent both fields with the following format:

decimal_number-data

If decimal_number is zero, then data is an ASCII string. If any other number, data is hex digits.

Keyword	Form	Subcontainers?	Default Value	Meaning
subnet	subnet default	Yes	None	Specifies a subnet that does not have any range. The subnet is used by a server only when it is responding to INFORM packet from the client and the client's address does not have another matching subnet container.
subnet	subnet <i>subnet id netmask</i>	Yes	None	Specifies a subnet and a pool of addresses. All addresses are assumed to be in the pool unless a range is specified on the line or addresses are modified later in the container by a range or exclude statement. The optional range is a pair of IP addresses in dotted quad format separated by a dash. An optional label and priority can be specified. These are used by virtual subnets to identify and order the subnets in the virtual subnet. The label and priority are separated by a colon. These containers are only allowed at the global or database container level.
	subnet <i>subnet id netmask range</i>			
	subnet <i>subnet id netmask label:priority</i>			
	subnet <i>subnet id netmask range label:priority</i>			
subnet	subnet <i>subnet id range</i>	Yes	None	Specifies a subnet that goes within a network container. It defines a range of addresses that is the whole subnet unless the optional range part is specified. The netmask associated with the subnet is taken from the surrounding network container. Note: This method is deprecated in favor of the other subnet forms.

Keyword	Form	Subcontainers?	Default Value	Meaning
option	option <i>number data</i> ...	No	None	Specifies an option to send to a client or, in the case of deny, an option to prevent from being sent to the client. The option * deny clause means all options not specified in the current container are not to be returned to the client. option <i>numberdeny</i> only denies the specified option. <i>number</i> is an unsigned 8-bit integer. <i>data</i> is specific to the option (see above) or can be specified as a quoted string (indicating ASCII text) or 0 <i>hexdigits</i> or hex " <i>hexdigits</i> " or hex " <i>hexdigits</i> ". If the option is in a vendor container, the option will be encapsulated with other options in an option 43.
	option <i>numberdeny</i>			
	option * deny			
exclude	exclude <i>an IP address</i>	No	None	Modifies the range on the container in which the exclude statement is in. The exclude statement is not valid in the global or database container levels. The exclude statement removes the specified address or range from the current range on the container. The exclude statement allows you to create non-contiguous ranges for subnets or other containers.
	exclude <i>dotted_quad-dotted_quad</i>			
range	range <i>IP_address</i>	No	None	Modifies the range on the container in which the range statement is in. The range statement is not valid in the global or database container levels. If the range is the first in the container that does not specify a range on the container definition line, then the range for the container becomes the range specified by the range statement. Any range statement after the first range or all range statements for a containers that specifies ranges in its definition are added to the current range. With the range statement, a single address or set of addresses can be added to the range. The range must fit inside the subnet container definition.
	range <i>dotted_quad-dotted_quad</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
client	client <i>hwtype hwaddr</i> NONE	Yes	None	Specifies a client container that denies the client specified by the <i>hwaddr</i> and <i>hwtype</i> from getting an address. If <i>hwtype</i> is 0, then <i>hwaddr</i> is an ASCII string. Otherwise, <i>hwtype</i> is the hardware type for the client and <i>hwaddr</i> is the hardware address of the client. If the <i>hwaddr</i> is a string, then quotes are accepted around the string. If the <i>hwaddr</i> is a hexstring, then the address may be specified by <i>0xhexdigits</i> or hex <i>digits</i> . <i>range</i> allows the client specified by the <i>hwaddr</i> and <i>hwtype</i> to get an address in the <i>range</i> . Must be regular expressions to match multiple clients.
	client <i>hwtype hwaddr</i> ANY			
	client <i>hwtype hwaddr</i> <i>dotted_quad</i>			
	client <i>hwtype hwaddr</i> <i>range</i>			
class	class <i>string</i>	Yes	None	Specifies a class container with name <i>string</i> . String can be quoted or not. If quoted, the quotes are removed before comparison. Quotes are required for strings with spaces or tabs. This container is valid at any level. A range can be supplied to indicate a set of addresses to hand out to a client with this class. The range is either a single dotted quad IP address or two dotted quad IP addresses separated by a dash.
	class <i>string range</i>			
network	network <i>network id</i> <i>netmask</i>	Yes	None	Specifies a network ID using class information (for example, 9.3.149.0 with a netmask of 255.255.255.0 would be network 9.0.0.0 255.255.255.0). This version of the network container is used to hold subnets with the same network ID and netmask. When a range is provided, all the addresses in the range are in the pool. The range must be in the network ID's network. This uses class full addressing. This is only valid in the global or database container level. Note: The network keyword is deprecated in favor of the subnet container.
	network <i>network id</i>			
	network <i>network id</i> <i>range</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
vendor	vendor <i>vendor_id</i>	Yes	None	<p>Specifies a vendor container. Vendor containers are used to return option 43 to the client. The vendor id may be specified in a quoted string or a binary string in the form <i>0xhexdigits</i> or <i>hex" digits"</i>. An optional range may be placed after the vendor id. The range is specified as two dotted quads separated by a dash. After the optional range, an optional hexstring or ASCII string can be specified as the first part of the option 43. If options are in the container, they are appended to the option 43 data. After all options are processed an End Of Option List Option is appended to the data. To return options outside of an option 43, use a regular expression client that matches all clients to specify normal options to return based on the vendor ID.</p> <p>pxe after the keyword vendor will create a vendor container for PXEClient.</p> <p>pxeserver after the keyword vendor will create a vendor container for PXEServer.</p>
	vendor <i>vendor_id</i> hex""			
	vendor <i>vendor_id</i> hex ""			
	vendor <i>vendor_id</i> 0xdata			
	vendor <i>vendor_id</i> ""			
	vendor <i>vendor_id</i> range			
	vendor <i>vendor_id</i> range hex""			
	vendor <i>vendor_id</i> range hex ""			
	vendor <i>vendor_id</i> range 0xdata			
	vendor <i>vendor_id</i> range ""			
	vendor pxe			
	vendor pxeserver			

Keyword	Form	Subcontainers?	Default Value	Meaning
inoption	inoption <i>number</i> <i>option_data</i>	Yes	None	Specifies a container to be matched against any arbitrary incoming option specified by the client. <i>number</i> specifies the option number. <i>option_data</i> specifies the key to match for this container to be selected during address and option selection for the client. <i>option_data</i> is specified in expected form — quoted string, IP address, integer value — for well known options, or it can be optionally specified as a hexadecimal string of bytes if preceded by the characters 0x. For options that are not well known to the server, a hexadecimal string of bytes can be specified in the same fashion. Additionally, the <i>option_data</i> can indicate a regular expression to be compared against the string representation of the client's option data. Regular expressions are specified in a quoted string beginning "!" (double quote followed by an exclamation mark). The string form of options not well known to the server will be a hexadecimal string of bytes NOT preceded with the characters 0x.
	inoption <i>number</i> <i>option_data range</i>			

Keyword	Form	Subcontainers?	Default Value	Meaning
virtual	virtual fill <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy. <i>fill</i> means use all addresses in the container before going to the next container. <i>rotate</i> means select an address from the next pool in the list on each request. <i>sfill</i> and <i>srotate</i> are the same as <i>fill</i> and <i>rotate</i> , but a search is done to see if the client matches containers, vendors, or classes in the subnet. If a match is found that can supply an address, the address is taken from that container instead of following the policy. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet id.
	virtual sfill <i>id id ...</i>			
	virtual rotate <i>id id ...</i>			
	virtual srotate <i>id id ...</i>			
inorder:	inorder: <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy of fill, which means use all addresses in the container before going to the next container. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.
balance:	balance: <i>id id ...</i>	No	None	Specifies a virtual subnet with a policy of rotate, which means use the next address in the next container. There can be as many IDs as needed. <i>id</i> is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.
bootstrapserver	bootstrapserver <i>IP address</i>	No	None	Specifies the server clients should use from which to TFTP files after receiving BOOTP or DHCP packets. This value fills in the siaddr field in the packet. This is valid at any container level.

Keyword	Form	Subcontainers?	Default Value	Meaning
giaddrfield	giaddrfield <i>IP address</i>	No	None	Specifies the giaddrfield for response packets. Note: This specification is illegal in the BOOTP and DHCP protocols, but some clients require the giaddr field to be the default gateway for the network. Because of this potential conflict, giaddrfield should only be used within a client container, although it can work at any level.
bootfile	bootfile <i>path</i>	No	None	Specifies the bootfile to use in the file section of the response packet. This can be specified at any container level. The bootfile policy defines how items specified in the file section of the incoming packet interact with the bootfile and the home directory statements.
pxebootfile	pxebootfile <i>SystemArch MajorVer MinorVer Bootfilename Type Layer</i>	No	None	Specifies the bootfile to be given to a PXEClient. The config file parser generates an error if the number of parameters after the keyword is less than 4 , ignore if more than 7 and if 4 are there then it assume the value for Type = 0 and Layer = 0. This keyword can be used only in a container.

For details about other options, see DHCP Server File Known Options and PXE Vendor Container Suboptions.

Configuring TCP/IP

If you installed the Transmission Control Protocol/Internet Protocol (**TCP/IP**) and Network File System (NFS) software, you can configure your system to communicate over a network.

After you install **TCP/IP** and NFS software, use the Web-based System Manager, **wsm**, or the System Management Interface Tool (SMIT) fast path, **smit tcpip**, to configure your system. The online help guides you through the process.

Prerequisites

The **TCP/IP** software must be installed. If you install TCP/IP software, you will have to install the TCP/IP Optional Support software product.

You must have root authority to configure TCP/IP.

Updating the Hosts List

A name server is a machine on your network that stores the names and addresses of all the network machines. The names are stored in a Hosts List. When one machine wants to communicate with another, it sends that machine name to the name server. The name server refers to the Hosts List and responds with the address of the machine name requested. Having a name server is an advantage because the Hosts List is stored and updated at one location, but is accessible to all machines on the network. This saves time and storage space.

- If you are using a name server for network communications, you do not need to perform this procedure. You have finished configuring TCP/IP.
- If you are *not* using a nameserver for network communications, you must update the hosts list to include the names of the systems on the network. With root authority, use the Web-based System Manager, **wsm**, or the System Management Interface Tool (SMIT) fast path, **smit hostent**.

TCP/IP Daemons

Daemons (also known as *servers*) are processes that run continuously in the background and perform functions required by other processes. Transmission Control Protocol/Internet Protocol (TCP/IP) provides daemons for implementing certain functions in the operating system. These daemons are background processes that run without interrupting other processes (unless that is part of the daemon function).

Daemons are invoked by commands at the system management level, by other daemons, or by shell scripts. You can also control daemons with the **inetd** daemon, the **rc.tcpip** shell script, and the System Resource Controller (SRC).

Subsystems and Subservers

A *subsystem* is a daemon, or server, that is controlled by the SRC. A *subserver* is a daemon that is controlled by a subsystem. (Daemon commands and daemon names are usually denoted by a **d** at the end of the name.) The categories of subsystem and subserver are mutually exclusive. That is, daemons are not listed as both a subsystem and as a subserver. The only TCP/IP subsystem that controls other daemons is the **inetd** daemon. All TCP/IP subservers are also **inetd** subservers.

The following are TCP/IP daemons controlled by the SRC:

Subsystems

gated	Provides gateway routing functions and supports the Routing Information Protocol (RIP), the Routing Information Protocol Next Generation (RIPng), Exterior Gateway Protocol (EGP), the Border Gateway Protocol (BGP) and BGP4+, the Defense Communications Network Local-Network Protocol (HELLO), Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Internet Control Message Protocol (ICMP and ICMPv6)/Router Discovery routing protocols. In addition, the gated daemon supports the Simple Network Management Protocol (SNMP). The gated daemon is one of two routing daemons available for routing to network addresses and is the preferred routing daemon. The gated daemon is preferred over the routed daemon because the gated daemon supports more gateway protocols.
inetd	Invokes and schedules other daemons when requests for the daemon services are received. This daemon can also start other daemons. The inetd daemon is also known as the super daemon.
iptrace	Provides interface-level packet-tracing function for Internet protocols.
named	Provides the naming function for the Domain Name Server protocol (DOMAIN).
routed	Manages the network routing tables and supports the Routing Information Protocol (RIP). The gated daemon is preferred over the routed daemon because the gated daemon supports more gateway protocols.

rwhod	Sends broadcasts to all other hosts every three minutes and stores information about logged-in users and network status. Use the rwhod daemon with extreme care, because it can use significant amounts of machine resources.
timed	Provides the time server function.

Note: Both the **routed** and **gated** daemons are listed as TCP/IP subsystems. Do not run the **startsrc -g tcpip** command, which initiates both of these routing daemons, along with all the other TCP/IP subsystems. Running both daemons simultaneously on one machine can produce unpredictable results.

TCP/IP daemons controlled by the **inetd** subsystem are the following:

inetd Subservers

comsat	Notifies users of incoming mail.
fingerd	Provides a status report on all logged-in users and network status at the specified remote host. This daemon uses the Finger protocol.
ftpd	Provides the file transfer function for a client process using the File Transfer Protocol (FTP).
rexecd	Provides the foreign host server function for the rexec command.
rlogind	Provides the remote login facility function for the rlogin command.
rshd	Provides the remote command execution server function for the rpc and rsh commands.
talkd	Provides the conversation function for the talk command.
syslogd	Reads and logs system messages. This daemon is in the Remote Access Service (RAS) group of subsystems.
telnetd	Provides the server function for the TELNET protocol.
tftpd	Provides the server function for the Trivial File Transfer Protocol (TFTP).
uucpd	Handles communications between the Basic Network Utilities (BNU) and TCP/IP.

System Resource Control (SRC)

Among other functions, SRC allows you to start daemons, stop them, and trace their activity. In addition, SRC provides the ability to group daemons into subsystems and subservers.

System Resource Control is a tool designed to aid the person who manages your system in controlling daemons. SRC allows control beyond the flags and parameters available with each daemon command.

See the System Resource Controller Overview in *AIX 5L Version 5.1 System Management Concepts: Operating System and Devices* for more information concerning the System Resource Controller.

SRC Commands

SRC commands can affect one daemon, a group of daemons, or a daemon and those daemons it controls (subsystem with subservers). In addition, some TCP/IP daemons do not respond to all SRC commands. The following is a list of SRC commands that can be used to control TCP/IP daemons and their exceptions.

startsrc	Starts all TCP/IP subsystems and inetd subservers. The startsrc command works for all TCP/IP subsystems and inetd subservers.
stopsrc	Stops all TCP/IP subsystems and inetd subservers. This command is also called the stop normal command. The stop normal command allows subsystems to process all outstanding work and terminate gracefully. For inetd subservers, all pending connections are allowed to start and all existing connections are allowed to complete. The stop normal command works for all TCP/IP subsystems and inetd subservers.

stopsrc -f	Stops all TCP/IP subsystems and inetd subservers. This command is also called the stop force . The stop force command immediately terminates all subsystems. For inetd subservers, all pending connections and existing connections are terminated immediately.
refresh	Refreshes the following subsystems and subservers: the inetd , syslogd , named , dhcpsd , and gated subsystems.
lssrc	Provides short status for subsystems, which is the state of the specified subsystem (active or inoperative). Also provides short status for inetd subservers. The short status for inetd subservers includes: subserver name, state, subserver description, command name, and the arguments with which it was invoked.
lssrc -l	Provides the short status plus additional information (long status) for the following subsystems: <p>gated State of debug or trace, routing protocols activated, routing tables, signals accepted and their function.</p> <p>inetd State of debug, list of active subservers and their short status; signals accepted and their function.</p> <p>named State of debug, named.conf file information.</p> <p>dhcpsd State of debug, all controlled IP addresses and their current state.</p> <p>routed State of debug and trace, state of supplying routing information, routing tables.</p> <p>syslogd syslogd configuration information.</p> <p>The lssrc -l command also provides long status for inetd subservers. The long status includes short status information and active connection information. Some subservers will provide additional information. The additional information by subserver includes:</p> <p>ftpd State of debug and logging</p> <p>telnetd Type of terminal emulating</p> <p>rlogind State of debug</p> <p>fingerd State of debug and logging</p> <p>The rwhod and timed subservers do not provide long status.</p>
traceson	Turns on socket-level debugging. Use the trpt command to format the output. The timed and iptraced subsystems do not support the traceson command.
tracesoff	Turns off socket-level debugging. Use the trpt command to format the output. The timed and iptraced subsystems do not support the tracesoff command.

For examples of how to use these commands, see the articles on the individual commands. For more information on the System Resource Controller, see System Resource Controller Overview in *AIX 5L Version 5.1 System Management Concepts: Operating System and Devices*.

Configuring the inetd Daemon

To configure the **inetd** daemon:

1. Specify which subservers it will be invoked by adding an **inetd** daemon.
2. Specify the restart characteristics by changing the restart characteristics of the **inetd** daemon.

Configuring the inetd Daemon Tasks			
Task	SMIT Fast Path	Command or File	Web-based System Manager Management Environment

Configuring the <code>inetd</code> Daemon Tasks			
Starting the <code>inetd</code> Daemon	<code>smit mkinetd</code>	<code>startsrc -s inetd</code>	Software → Network → TCP/IP (IPv4 and IPv6) → Subsystems. Right-click on an inactive subsystem, and select Activate .
Changing Restart Characteristics of the <code>inetd</code> Daemon	<code>smit chinetd</code> or <code>smit lsinetd</code>		Software → Network → TCP/IP (IPv4 and IPv6) → Subsystems → Selected → Properties.
Stopping the <code>inetd</code> Daemon	<code>smit rminetd</code>	<code>stopsrc -s inetd</code>	Software → Network → TCP/IP (IPv4 and IPv6) → Subsystems. Right-click on an active subsystem, and select → Deactivate .
Listing All <code>inetd</code> Subservers	<code>smit inetdconf</code>		Software → Network → TCP/IP (IPv4 and IPv6) → Subsystems.
Adding an <code>inetd</code> Subserver ¹	<code>smit mkinetdconf</code>	edit <code>/etc/inetd.conf</code> then run <code>refresh -s inetd</code> or <code>kill -1 inetdPID²</code>	Software → Network → TCP/IP (IPv4 and IPv6) → Subsystems → Subsystems (drop-down menu) → New inetd Subserver .
Change/Show Characteristics of an <code>inetd</code> Subserver	<code>smit inetdconf</code>	edit <code>/etc/inetd.conf</code> then run <code>refresh -s inetd</code> or <code>kill -1 inetdPID²</code>	Software → Network → TCP/IP (IPv4 and IPv6) → Subsystems → Selected → Properties.
Removing an <code>inetd</code> Subserver	<code>smit rminetd</code>	edit <code>/etc/inetd.conf</code> then run <code>refresh -s inetd</code> or <code>kill -1 inetdPID²</code>	Software → Network → TCP/IP (IPv4 and IPv6) → Subsystems → Selected → Deactivate .

Notes:

1. Adding an `inetd` subserver configures the `inetd` daemon so that it invokes the subserver when it is needed.
2. Both the `refresh` and the `kill` commands inform the `inetd` daemon of changes to its configuration file.

Client Network Services

Client Network Services (accessible using the Web-based System Manager `wsm`, or the SMIT fast path, `smit clientnet`) refers to the TCP/IP protocols available for use by this operating system. Each protocol (or service) is known by the port number that it uses on the network, hence the term **well-known port**. As a convenience to programmers, the port numbers can be referred to by names as well as numbers. For example, the TCP/IP mail protocol uses port 25 and is known by the name `smtp`. If a protocol is listed (uncommented) in the `/etc/services` file, then a host can use that protocol.

By default, all the TCP/IP protocols are defined in the `/etc/services` file. You do not have to configure this file. If you write your own client/server programs, you might want to add your service to the `/etc/services` file, and reserve a specific port number and name for your service. If you do decide to add your service to `/etc/services`, note that port numbers 0 through 1024 are reserved for system use.

Client Network Services Tasks			
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>	<i>Web-based System Manager Management Environment</i>
Listing All Services	smit lsservices	view /etc/services	Software → Network → TCPIP (IPv4 and IPv6) → Services .
Adding a Service	smit mksservices	edit /etc/services	Software → Network → TCPIP (IPv4 and IPv6) → Services → New Service .
Change/Show Characteristics of a Service	smit chservices	edit /etc/services	Software → Network → TCPIP (IPv4 and IPv6) → Services . Select a service, then click Selected → Properties .
Removing a Service	smit rmservices	edit /etc/services	Software → Network → TCPIP (IPv4 and IPv6) → Services . Select a service, then click Selected → Delete .

Server Network Services

Server Network Services include controlling remote access, starting or stopping TCP/IP, and managing the **pty** device driver, as shown in the following table.

The **pty** device driver is installed automatically with the system. By default, it is configured to support 16 BSD-style symbolic links, and it is available for use by the system at boot time.

Server Network Services Tasks			
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>	<i>Web-based System Manager Management Environment</i>
Controlling Remote Access	See "Remote Command Execution Access" and "Restricted File Transfer Program Users".		Software → Network → TCPIP (IPv4 and IPv6) → Access Control . Right-click on Remote Access and select Properties .
Start, Restart, or Stop TCP/IP Subsystems	smit otherserv	See "System Resource Control".	Software → Network → TCPIP (IPv4 and IPv6) → Subsystems . Right-click on a subsystem, and select Properties .
Change/Show Characteristics of the pty Device Driver	smit chgpty	chdev -l pty0 -P -a num=X where X ranges from 0 to 64	
Make the pty Device Driver Unavailable for Use	smit pty then select Remove the PTY; Keep Definition		
Make the pty Device Driver Available for Use	smit pty then select Configure the Defined PTY		
Generate an Error Report	smit errpt		

Server Network Services Tasks			
Trace the pty	smit trace		

TCP/IP Name Resolution

Although 32-bit Internet addresses provide machines an efficient means of identifying the source and destination of datagrams sent across an internetwork, users prefer meaningful, easily remembered names. Transmission Control Protocol/Internet Protocol (TCP/IP) provides a naming system that supports both flat and hierarchical network organizations.

The topics discussed in this section are:

- Naming
- Performing Local Name Resolution (*/etc/hosts*)
- Planning for DOMAIN Name Resolution
- Configuring Name Servers
- Configuring a Forwarder
- Configuring a Forward Only Server
- Configuring a Host to Use a Name Server
- Configuring Dynamic Zones on the DNS Name Server
- Planning and Configuration for LDAP Name Resolution

Naming

Naming in flat networks is very simple. Host names consist of a single set of characters and generally are administered locally. In flat TCP/IP networks, each machine on the network has a file (***/etc/hosts***) containing the name-to-Internet-address mapping information for every host on the network. The administrative burden of keeping each machine naming file current grows as the TCP/IP network grows. When TCP/IP networks become very large, as on the Internet, naming is divided hierarchically. Typically, the divisions follow the network organization. In TCP/IP, hierarchical naming is known as the *domain name system* (DNS) and uses the DOMAIN protocol. The DOMAIN protocol is implemented by the **named** daemon in TCP/IP.

As in naming for flat networks, the domain name hierarchy provides for the assignment of symbolic names to networks and hosts that are meaningful and easy for users to remember. However, instead of each machine on the network keeping a file containing the name-to-address mapping for all other hosts on the network, one or more hosts are selected to function as *name servers*. Name servers translate (resolve) symbolic names assigned to networks and hosts into the efficient Internet addresses used by machines. A name server has complete information about some part of the domain, referred to as a *zone*, and it has *authority* for its zone.

Naming Authority

In a flat network, all hosts in the network are administered by one central authority. This form of network requires that all hosts in the network have unique host names. In a large network, this requirement creates a large administrative burden on the central authority.

In a domain network, groups of hosts are administered separately within a tree-structured hierarchy of domains and subdomains. In this case, host names need to be unique only within the local domain, and only the *root domain* is administered by a central authority. This structure allows subdomains to be administered locally and reduces the burden on the central authority. For example, the root domain of the Internet consists of such domains as com (commercial organizations), edu (educational organizations), gov

(governmental organizations), and `mil` (military groups). New top-level domains can only be added by the central authority. Naming at the second level is delegated to designated agents within the respective domains. For example, in the following figure, `com` has naming authority for all commercial organization subdomains beneath it. Likewise, naming at the third level (and so on) is delegated to agents within that level. For example, in the Domain Structure of the Internet figure, Century has naming authority for its subdomains Austin, Hopkins, and Charlotte.

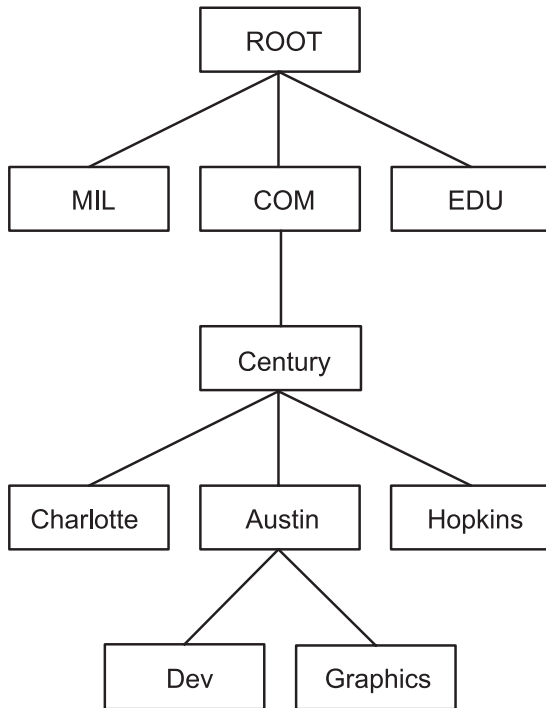


Figure 23. Domain Structure of the Internet. This figure illustrates the hierarchical structure of the internet. It begins at the top with the root and branches to the next level containing the mil, com, and edu domains. Below the com domain is another level containing Charlotte, Austin, and Hopkins. Below Austin is Dev and Graphics.

Century's Austin subdomain might also be divided into zones, for example, Dev and Graphics. In this case, the zone `austin.century.com` has all the data contained in the domain `austin.century.com`, except that which was delegated to Dev and Graphics. The zone `dev.century.com` would contain only the data delegated to Dev; it would know nothing about Graphics, for example. The zone `austin.century.com` (as opposed to the domain of the same name) would contain only that data not delegated to other zones.

Naming Conventions

In the hierarchical domain name system, names consist of a sequence of case-insensitive subnames separated by periods with no embedded blanks. The DOMAIN protocol specifies that a local domain name must be fewer than 64 characters and that a host name must be fewer than 32 characters in length. The host name is given first, followed by a period (`.`), a series of local domain names separated by periods, and finally the root domain. A fully specified domain name for a host, including periods, must be fewer than 255 characters in length and in the following form:

```
host.subdomain1.[subdomain2 . . . subdomain].rootdomain
```

Since host names must be unique within a domain, you can use an abbreviated name when sending messages to a host within the same domain. For example, instead of sending a message to `smith.eng.lsu.edu`, a host in the `eng` domain could send a message to `smith`. Additionally, each host can have several aliases that other hosts can use when sending messages.

Choosing Names for the Hosts on Your Network

The purpose of using names for hosts is to provide a quick, easy, and unambiguous way to refer to the computers in your network. Internet system administrators have discovered that there are good, as well as poor, choices for host names. These suggestions are intended to help you avoid common pitfalls in choosing host names.

The following are some suggestions for choosing unambiguous, easy to remember host names:

- Terms that are rarely used, for example, sphinx or eclipse.
- Theme names, such as colors, elements (for example, helium, argon, or zinc), flowers, fish, and others.
- Real words (as opposed to random strings of characters).

The following are some examples of poor choices. In general, these are poor choices because they are difficult to remember or are confusing (either to humans or computers):

- Terms that are already in common use, for example, up, down, or crash.
- Names containing only numbers.
- Names that contain punctuation marks.
- Names that rely on case distinction, for example, Orange and orange.
- The name or initials of the primary user of the system.
- Names having more than 8 characters.
- Unusual or purposefully incorrect spellings, for example, czek, which could be confused with "check" or "czech."
- Names that are, or resemble, domain names, for example, yale.edu.

Name Servers

In a flat name space, all names must be kept in the `/etc/hosts` file on each host on the network. If the network is very large, this can become a burden on the resources of each machine.

In a hierarchical network, certain hosts designated as *name servers* resolve names into Internet addresses for other hosts. This has two advantages over the flat name space. It keeps the resources of each host on the network from being tied up in resolving names, and it keeps the person who manages the system from having to maintain name resolution files on each machine on the network. The set of names managed by a single name server is known as its *zone of authority*.

Note: Although the host machine that performs the name resolution function for a zone of authority is commonly referred to as a *name server* host, the process controlling the function, the **named** daemon, is the actual name server process.

To further reduce unnecessary network activity, all name servers *cache* (store for a period of time) name-to-address mappings. When a client asks a server to resolve a name, the server checks its cache first to see if the name has been resolved recently. Because domain and host names do change, each item remains in the cache for a limited length of time specified by the TTL of the record. In this way, authorities can specify how long they expect the name resolution to be accurate.

Within any autonomous system there can be multiple name servers. Typically, name servers are organized hierarchically and correspond to the network organization. Referring to the "Domain Structure of the Internet" figure, each domain might have a name server responsible for all subdomains within the domain. Each subdomain name server communicates with the name server of the domain above it (called the *parent* name server), as well as with the name servers of other subdomains.

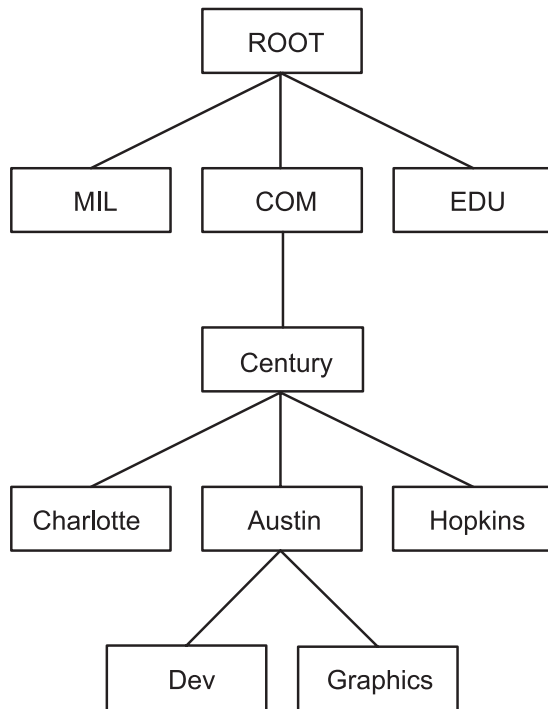


Figure 24. Domain Structure of the Internet. This figure illustrates the hierarchical structure of the internet. It begins at the top with the root and branches to the next level containing the mil, com, and edu domains. Below the com domain is another level containing Charlotte, Austin, and Hopkins. Below Austin is Dev and Graphics.

For example, in the "Domain Structure of the Internet" figure, Austin, Hopkins, and Charlotte are all subdomains of the domain Century. If the tree hierarchy is followed in the network design, the Austin name server communicates with the name servers of Charlotte and Hopkins as well as with the parent Century name server. The Austin name server also communicates with the name servers responsible for its subdomains.

There are several types of name servers:

Master Name Server

Loads its data from a file or disk and can delegate authority to other servers in its domain.

Slave Name Server

Receives its information at system startup time for the given zone of authority from a master name server, and then periodically asks the master server to update its information. On expiration of the refresh value in the start of authority (SOA) Resource Record on a slave name server, or on receipt of a Notify message from the master name server, the slave reloads the database from the master if the serial number of the database on the master is greater than the serial number in the current database on the slave. If it becomes necessary to force a new zone transfer from the master, simply remove the existing slave databases and refresh the **named** daemon on the slave name server.

Stub Name Server

Although its method of database replication is similar to that of the slave name server, the stub name server only replicates the name server records of the master database rather than the whole database.

Hint Server

Indicates a name server that relies only on the hints that it has built from previous queries to other name servers. The hint name server responds to queries by asking other servers that have the authority to provide the information needed if a hint name server does not have a name-to-address mapping in its cache.

Forwarder or Client Server

Forwards queries it cannot satisfy locally to a fixed list of forwarding servers. Forwarding-only servers (a forwarder that obtains information and passes it on to other clients, but that is not actually a server) does not interact with the master name servers for the root domain and other domains. The queries to the forwarding servers are recursive. There can be one or more forwarding servers, which are tried in turn until the list is exhausted. A client and forwarder configuration is typically used when you do not want all the servers at a given site to interact with the rest of the Internet servers, or when you want to build a large cache on a select number of name servers.

Remote Server

Runs all the network programs that use the name server without the name server process running on the local host. All queries are serviced by a name server that is running on another machine on the network.

One name server host can perform in different capacities for different zones of authority. For example, a single name server host can be a master name server for one zone and a slave name server for another zone.

Name Resolution

The process of obtaining an Internet address from a host name is known as name resolution and is done by the **gethostbyname** subroutine. The process of translating an Internet address into a host name is known as reverse name resolution and is done by the **gethostbyaddr** subroutine. These routines are essentially accessors into a library of name translation routines known as *resolvers*.

Resolver routines on hosts running TCP/IP normally attempt to resolve names using the following sources:

1. BIND/DNS (named)
2. Network Information Service (NIS)
3. Local **/etc/hosts** file

When NIS+ is installed, lookup preferences are set using the **irs.conf** file. For more information, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

To resolve a name in a domain network, the resolver routine first queries the domain name server database, which might be local if the host is a domain name server or on a foreign host. Name servers translate domain names into Internet addresses. The group of names for which a name server is responsible is its zone of authority. If the resolver routine is using a remote name server, the routine uses the domain name protocol (DOMAIN) to query for the mapping. To resolve a name in a flat network, the resolver routine checks for an entry in the local **/etc/hosts** file. When NIS or NIS+ is used, the **/etc/hosts** file on the master server is checked.

By default, resolver routines attempt to resolve names using the above resources. BIND/DNS is tried first. If the **/etc/resolv.conf** file does not exist or if BIND/DNS could not find the name, NIS is queried if it is running. NIS is authoritative over the local **/etc/hosts**, so the search ends here if it is running. If NIS is not running, then the local **/etc/hosts** file is searched. If none of these services can find the name, then the resolver routines return with HOST_NOT_FOUND. If all of the services are unavailable, then the resolver routines return with SERVICE_UNAVAILABLE.

The default order described above can be overwritten by creating the **/etc/irs.conf** configuration file and specifying the desired order. Also, both the default and **/etc/irs.conf** orderings can be overwritten with the environment variable, **NSORDER**. If either the **/etc/irs.conf** file or **NSORDER** environment variable are defined, then at least one value must be specified along with the option.

To specify host ordering with the **/etc/irs.conf** file:

```
hosts value [ continue ]
```

The order is specified with each method indicated on a line by itself. The *value* is one of the listed methods and the `continue` keyword indicates that another resolver method follows on the next line.

To specify host ordering with the **NSORDER** environment variable:

```
NSORDER=value,value,value
```

The order is specified on one line with values separated by commas. White spaces are permitted between the commas and the equal sign.

For example, if the local network is organized as a flat network, then only the **/etc/hosts** file is needed. Given this example, the **/etc/irs.conf** file contains the following line:

```
hosts local
```

Alternatively, the **NSORDER** environment variable can be set as:

```
NSORDER=local
```

If the local network is a domain network using a name server for name resolution and an **/etc/hosts** file for backup, then both services should be specified. Given this example, the **/etc/irs.conf** file contains the following lines:

```
hosts dns continue
hosts local
```

The **NSORDER** environment variable is set as:

```
NSORDER=bind,local
```

Note: The values listed must be in lowercase.

When following any defined or default resolver ordering, the search algorithm continues from one resolver to the next only if:

- The current service is not running, therefore, it is unavailable.
- The current service cannot find the name and is not authoritative.

If the **/etc/resolv.conf** file does not exist, then BIND/DNS is considered not set up or running, and therefore it is not available. If the **getdomainname** and **yp_bind** subroutines fail, then the NIS service is considered not set up or running, and therefore it is not available. If the **/etc/hosts** file could not be opened, then a local search is impossible, and therefore the file and service are unavailable.

When a service is listed as *authoritative*, it means that this service is the expert of its successors and has all pertinent names and addresses. Resolver routines do not try successor services, because successors might contain only a subset of the information in the authoritative service. Name resolution ends at service listed as authoritative, even if it does not find the name (in which case, the resolver routine returns `HOST_NOT_FOUND`). If an authoritative service is not available, then the next service specified is queried.

An authoritative source is specified with the string `=auth` directly behind a value. The entire word, *authoritative* can be typed in, but only the `auth` string is used. For example, if the **NSORDER** environment variable contains the following:

```
hosts = nis=auth,dns,local
```

The search ends after the NIS query (if NIS is running), regardless of whether the name was found. If NIS is not running, then the next source is queried, which is DNS.

TCP/IP name servers use caching to reduce the cost of searching for names of hosts on remote networks. Instead of searching for a host name each time a request is made, a name server first looks at its cache to see if the host name has been resolved recently. Since domain and host names do change, each item remains in the cache for a limited length of time specified by the time-to-live (TTL) value of the record. In this way, name servers can specify how long they expect their responses to be considered authoritative.

Potential Host Name Conflict Between *nameserver* and *sendmail*: In a DNS environment, a host name that is set using the **hostname** command from the command line or in the **rc.net** file format must be the official name of the host as returned by the name server. Generally, this name is the full domain name of the host in the form:

```
host.subdomain.subdomain.rootdomain
```

Note: Resolver routines require the default domain to be set. If the default domain is not set in the **hostname** command, then it must be set in the **/etc/resolv.conf** file.

If the host name is not set up as a fully qualified domain name, and if the system is set up to use a domain name server in conjunction with the **sendmail** program, the **sendmail** configuration file (**/etc/sendmail.cf**) must be edited to reflect this official host name. In addition, the domain name macros in this configuration file must be set for the **sendmail** program to operate correctly.

Note: The domain specified in the **/etc/sendmail.cf** file takes precedence over the domain set by the **hostname** command for all **sendmail** functions.

Potential Domain Name Conflict Between *nameserver* and *sendmail*: For a host that is in a DOMAIN network but is not a name server, the local domain name and domain name server are specified in the **/etc/resolv.conf** file. In a DOMAIN name server host, the local domain and other name servers are defined in files read by the **named** daemon when it starts.

Reverse Address Resolution Protocol

The Reverse Address Resolution Protocol (RARP) translates unique hardware addresses into Internet addresses on the Ethernet local area network (LAN) adapter (Ethernet protocol only). Standard Ethernet protocol is supported with the following restrictions:

- The server only replies to RARP requests.
- The server only uses permanent ARP table entries.
- The server does not use dynamic ARP table entries.
- The server does not automatically reply for itself.

The system administrator must manually build and maintain a table of permanent ARP entries using the **arp** command. A specific ARP table entry must be added on the server for each host that requires RARP replies from an authoritative source.

Performing Local Name Resolution (/etc/hosts)

Configure the **/etc/hosts** file if your network is small, and you are using a flat naming scheme. Even if you are using a hierarchical (or domain) naming scheme with name servers, you might want to configure the **/etc/hosts** file to identify hosts that are not known by the name servers.

Configure your system for local host resolution using the Web-based System Manager, the System Management Interface Tool (SMIT), or commands. If you choose the command method, be sure to preserve the format of the **/etc/hosts** file, as described in Hosts File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference*.

Local Name Resolution Tasks			
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>	<i>Web-based System Manager Management Environment</i>
List All the Hosts	smit lshostent	view /etc/hosts	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Hosts File → Contents of /etc/hosts file.
Add a Host	smit mkhostent	edit /etc/hosts	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Hosts File . In Add/Change host entry , complete the following fields: IP Addresses, Host name, Alias(es), and Comment. Click Add/Change Entry → OK.
Change/Show Characteristics of a Host	smit chhostent	edit /etc/hosts	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Hosts File . Select a host in Contents of /etc/hosts/file , and change data in Add/Change host entry . Click Add/Change Entry → OK.
Remove a Host	smit rmhostent	edit /etc/hosts	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Hosts File . Select a host in Contents of /etc/hosts/file , and click Delete Entry → OK.

Planning for DOMAIN Name Resolution

If you are part of a larger internetwork, coordinate setting up your domain and name servers with the central authority.

The following suggestions can help you plan your own DOMAIN name resolution system:

- Because of the vast possibilities in architecture and configuration, become familiar with TCP/IP, DNS, and BIND before you solidify any plans. If you plan to use a network information service, become

familiar with NFS and NIS as well. Books about these topics are widely available. For more information about NIS and NIS+, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

- Plan ahead.
 - >Changing a name is *much* more difficult than setting up the initial one. Obtain consensus from your organization on network, gateway, name server, and host names before you set up your files.
- Set up redundant name servers.
 - If you cannot set up redundant name servers, be sure to set up slave and hint name servers so you have some type of backup.
- In selecting the name servers, keep the following in mind:
 - Choose machines that are physically closest to exterior systems.
 - The name servers should be as independent as possible. Try for different power supplies and independent cabling.
 - Find another network to back up your name resolution service, and do the same for other networks.
- Test the servers.
 - Test both regular and reverse name resolution.
 - Test zone transfer from master to slave name servers.
 - Test each name server after a system crash and reboot.
- Send name resolution requests to forwarder servers before they go to exterior name servers. This allows your name servers to share caches and improve performance by reducing the load on your master name servers.

```
objectclass container
  requires
    objectclass,
    cn
objectclass hosts
  requires
    objectclass,
    hname
  allows
    addr
    halias,
    comment
```

Configuring Name Servers

In a hierarchical network, certain hosts are designated as *name servers*. These hosts resolve names into Internet addresses for other hosts. The **named** daemon controls the name server function and, therefore, must be run on a name server host.

Before you configure a name server, decide which type or types best fit the network it will serve. There are several types of name servers.

A *master name server* actually stores the database containing name-to-address mapping information. It loads its data from a file or disk and can delegate authority to other servers in its domain. A *slave name server* or *stub name server* receives its information at system startup time for a given zone of authority from a master name server, and then periodically asks the master server to update its information. A *hint name server* responds to requests to resolve names by querying other servers that have the authority to provide the information needed.

Note: Previous generations of the **named** name server specified the master name server as the primary name server, the slave name server as the secondary name server, and the hint name server as the caching-only name server. Any reference to the **named.conf** file in this documentation is specific to AIX 4.3.2 and later versions.

Keep in mind that a name server can function in different capacities for different zones of authority. For example, one name server host can be a master name server for one zone and a slave name server for another zone. If your system has NIS or NIS+ installed, these services can also provide name resolution. For more information, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

There are several files involved in configuring name servers.

conf	This file is read when the named daemon starts. The records in the conf file tell the named daemon which type of server it is, which domains it has authority over (its zones of authority), and where to get the data for initially setting up its database. The default name of this file is /etc/named.conf . However, you can change the name of this file by specifying the name and path of the file on the command line when the named daemon is started. If you intend to use the /etc/named.conf as the conf file and it does not exist, a message is generated in syslog file and named terminates. However, if an alternative conf file is specified, and the alternative file does not exist, an error message is not generated and named continues.
cache	Contains information about the local cache. The local cache file contains the names and addresses of the highest authority name servers in the network. The cache file uses the Standard Resource Record Format. The name of the cache file is set in the conf file.
domain data	There are three typical domain data files, also referred to as the named data files. The named local file contains the address resolution information for local loopback. The named data file contains the address resolution data for all machines in the name server zone of authority. The named reverse data file contains the reverse address resolution information for all machines in the name server zone of authority. The domain data files use the Standard Resource Record Format. Their file names are user definable and are set in the conf file. By convention, the names of these files generally include the name of the daemon (named), and the type of file and name of the domain is given in the extension. For example, the name server for the domain abc might have the following files: named.abc.data named.abc.rev named.abc.local When modifying the named data files the serial number in the SOA Resource Record must be incremented for slave name servers to properly realize the new zone changes.
resolv.conf	The presence of this file indicates to a host to go to a name server to resolve a name first. If the resolv.conf file does not exist, the host looks in the /etc/hosts file for name resolution. On a name server, the resolv.conf file must exist and can contain the local host address, the loopback address (127.0.0.1), or be empty.

Note: The resolver routines require the default domain be set. If the default domain is not set in the **/etc/resolv.conf** file, then it must be set in the **hostname**.

Time-to-live (TTL) is specified in resource records. If TTL is not specified in a record, the length of this time period defaults to the minimum field as defined in the start of authority (SOA) record for that zone. TTL is used when data is stored outside a zone (in a cache) to ensure that the data is not retained indefinitely.

Configuring a Master Name Server

To configure a master name server, use the Web-based System Manager, **wsm**, or use the following procedure, which edits a series of files and then uses the System Management Interface Tool (SMIT) or the command line to start the **named** daemon.

1. Edit the **/etc/named.conf** file. If there is no **named.conf** file in the **/etc** directory, copy the **/usr/samples/tcpip/named.conf** sample file into the **/etc** directory and edit it. See **named.conf** File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a configuration file.

This file is read each time the **named** daemon starts. It tells the server which type of server it is, the zone or zones for which it is responsible, and where to get its initial information.

- a. If you want the **named** data files to use paths relative to a specified directory (optional), use the *options* configuration clause to specify the directory in which the **named** data files can be located. For example:

```
options {
    directory "/usr/local/domain";
};
```

- b. If you want to allow record data to be cached outside of defined zones (optional), specify the name of the hint zone file. For example:

```
zone "." IN {
    type hint;
    file "/etc/named.ca";
};
```

- c. Specify a number of zones. To configure a server as the master for a zone, specify each zone and its domain data file. For example, a *master* server for both forward and reverse zones might resemble:

```
zone "abc.aus.century.com" in {
    type master;
    file "/etc/named.abcddata";
};
zone "201.9.192.in-addr.arpa" in {
    type master;
    file "/etc/named.abcrev";
};
```

- d. Define the name of the **named** local file. For example:

```
zone "0.0.127.in-addr.arpa" in {
    type master;
    file "/etc/named.local";
};
```

2. Edit the **/etc/named.ca** file. See the DOMAIN Cache File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a cache file.

This file contains the addresses of the servers that are authoritative, which are the *root* name servers for the domain. For example:

```
; root name servers.
      1          IN      NS      relay.century.com.
relay.century.com. 3600000 IN      A      129.114.1.2
```

Note: All lines in this file must be in Standard Resource Record Format.

3. Edit the **/etc/named.local** file. See the DOMAIN Local Data File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a local data file.

- a. Specify the start of authority (SOA) of the zone and the default time-to-live information. For example:

```
@ IN SOA venus.abc.aus.cntry.com. gail.zeus.abc.aus.cntry.com.
(
    1.1      ;serial
    3600    ;refresh
    600     ;retry
    3600000 ;expire
    86400)  ;minimum
```

- b. Specify the name server (NS) record. For example:

```
IN      NS      venus.abc.aus.century.com.
```

- c. Specify the pointer (PTR) record.

```
1      IN      PTR      localhost.
```

Note: All lines in this file must be in Standard Resource Record Format.

4. Edit the **/etc/named.data** file. The **/usr/samples/tcpip/hosts.awk** file contains directions for creating the **/etc/named.data** file. Use the **/usr/samples/tcpip/named.data** sample file as an example when

creating the **/etc/named.data** file. See the DOMAIN Data File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and an example of a hosts data file.

- a. Specify the start of authority of the zone and the default time-to-live information for the zone. This record designates the start of a zone. Only one start of authority record per zone is allowed. For example:

```
@ IN SOA  venus  bob.robert.abc.aus.century.com.
(
    1.1      ;serial
    3600    ;refresh
    600     ;retry
    3600000 ;expire
    86400   ;minimum)
```

- b. Include name-to-address resolution information on all hosts in the name server zone of authority. For example:

```
venus      IN  A      192.9.201.1
venus      IN  A      128.114.100.1
```

- c. Include name server records for all master name servers in the zone. For example:

```
IN  NS      venus.abc.century.com
IN  NS      kronos.xyz.century.com
```

- d. Include other types of entries, such as canonical name records and mail exchanger records as needed.

Note: All lines in this file must be in Standard Resource Record Format.

5. Edit the **/etc/named.rev** file. The **/usr/samples/tcpip/addrns.awk** file contains directions for creating the **/etc/named.rev** file. See the DOMAIN Reverse Data File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a reverse hosts data file.

- a. Specify the start of authority of the zone and the default time-to-live information. This record designates the start of a zone. Only one start of authority record per zone is allowed. For example:

```
@ IN SOA  venus.abc.aus.century.com. bob.robert.abc.aus.century.com.
(
    1.1      ;serial
    3600    ;refresh
    600     ;retry
    3600000 ;expire
    86400   ;minimum)
```

- b. Include address-to-name resolution information on all hosts to be in the name server's zone of authority.

```
;ABC.AUS.CENTURY.COM Hosts
1          IN  PTR   venus.abc.aus.century.com.
2          IN  PTR   kronos.abc.aus.century.com.
```

- c. Include other types of entries, such as name server records and canonical name records (optional).

Note: All lines in this file must be in Standard Resource Record Format.

6. Create an **/etc/resolv.conf** file by typing the following command:

```
touch /etc/resolv.conf
```

The presence of this file indicates that the host should use a name server, not the **/etc/hosts** file, for name resolution. This file must exist on a name server host and can contain the local host address, contain the loopback address (127.0.0.1), or be empty.

Alternatively, the **/etc/resolv.conf** file can contain the following entry:

```
nameserver 127.0.0.1
```

The 127.0.0.1 address is the loopback address, which causes the host to access itself as the name server. The **/etc/resolv.conf** file can also contain an entry similar to the following:

domain *domainname*

In the previous example, the value for *domainname* is `aus.century.com`.

7. Perform one of the following steps:

- Enable the **named** daemon using the **smit stnamed** SMIT fast path. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.
- Edit the `/etc/rc.tcpip` file. Uncomment the line for the **named** daemon by removing the comment (**#**) symbol from the following line:

```
#start /etc/named "$src_running"
```

This initializes the daemon with each system startup.

8. If you chose not to initialize the **named** daemon through SMIT, start the daemon for this session by running the following command:

```
startsrc -s named
```

Configuring a Slave Name Server

To configure a slave name server, use the Web-based System Manager, **wsm**, or use the following procedure which edits a series of files and then uses SMIT or the command line to start the **named** daemon.

1. Edit the `/etc/named.conf` file. If there is no `named.conf` file in the `/etc` directory, copy the `/usr/samples/tcpip/named.conf` sample file into the `/etc` directory and edit it. See the `named.conf` File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a `conf` file.

The `conf` file is read each time the **named** daemon starts. It tells the server which type of server it is, the zone for which it is responsible, and where to get its initial information.

- a. If you want the **named** data files to use paths relative to this directory (optional), use the `options` configuration clause to specify the directory where the **named** data files can be located. For example:

```
options {
    directory "/usr/local/domain";
};
```

- b. If you want to allow record data to be cached outside the defined zones (optional), specify the name of the hint zone file for the name server. For example:

```
zone "." IN {
    type hint;
    file "/etc/named.ca";
};
```

- c. Specify the slave zone clauses. Each stanza includes the zone type, an optional file name to which the name server can back up its data, and the list of master server Internet addresses. This list of addresses defines the hosts from which the zone is replicated. For example:

```
zone "abc.aus.century.com" IN {
    type slave;
    file "/etc/named.abc.bak";
    masters { 192.9.201.1; 192.9.201.2; };
};
zone "xyz.aus.century.com" IN {
    type slave;
    file "/etc/named.xyz.bak";
    masters { 192.9.201.1; 192.9.201.2; };
};
```

- d. Include slave zone clauses to define the reverse name resolution information for the name server. For example:

```

zone "201.9.192.in-addr.arpa" IN {
    type slave;
    file "named.rev.bak";
    masters { 192.9.201.1; 192.9.201.2; };
};
zone "100.114.128.in-addr.arpa" IN {
    type slave;
    file "named.rev.bak";
    masters { 192.9.201.1; 192.9.201.2; };
};

```

- e. To support resolving the loopback network address, specify a zone of type *master* with a source of **/etc/named.local** as well as the domain for which the name server is responsible.

```

zone "0.0.127.in-addr.arpa" IN {
    type master;
    file "/etc/named.local";
};

```

2. Edit the **/etc/named.ca** file. See the DOMAIN Cache File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a cache file.

This file contains the addresses of the servers that are authoritative name servers (NS) for the root domain of the network. For example:

```

; root name servers.
      1          IN      NS      relay.century.com.
relay.century.com. 3600000 IN      A      129.114.1.2

```

Note: All lines in this file must be in Standard Resource Record Format.

3. Edit the **/etc/named.local** file. See the DOMAIN Local Data File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a local data file.

- a. Specify the start of authority (SOA) of the zone and the default time-to-live information. For example:

```

@ IN SOA venus.abc.aus.cntry.com. gail.zeus.abc.aus.cntry.com.
(
                                1.1      ;serial
                                3600     ;refresh
                                600      ;retry
                                3600000  ;expire
                                86400    ;minimum
)

```

- b. Specify the name server (NS) record. For example:

```

IN      NS      venus.abc.aus.century.com.

```

- c. Specify the pointer (PTR) record.

```

1      IN      PTR      localhost.

```

Note: All lines in this file must be in Standard Resource Record Format.

4. Create an **/etc/resolv.conf** file by issuing the following command:

```
touch /etc/resolv.conf
```

The presence of this file indicates that the host should use a name server, not the **/etc/hosts** file, for name resolution. Optionally, you can want to enter records to specify the name, domain, and address of the name server.

5. >Perform one of the following steps:

- Enable the **named** daemon using the **smit stnamed** SMIT fast path. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.
- Edit the **/etc/rc.tcpip** file. Uncomment the line for the **named** daemon by removing the comment (#) symbol from the following line:

```
#start /etc/named "$src_running"
```


This initializes the daemon with each system startup.

6. If you chose not to initialize the **named** daemon through SMIT, start the daemon for this session by running the following command:

```
startsrc -s named
```

Configuring a Hint Name Server

To configure a hint, or cache-only, name server, use the Web-based System Manager, **wsm**, or use the following procedure, which edits a series of files and then uses SMIT or the command line to start the **named** daemon.

Procedure: Configure a hint name server according to the following steps:

1. Edit the **/etc/named.conf** file. If there is no **named.conf** file in the **/etc** directory, copy the **/usr/samples/tcpip/named.conf** sample file into the **/etc** directory and edit it. Refer to the **named.conf** File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a **conf** file.

- To support resolving the loopback network address, specify a zone of type *master* with a source of **/etc/named.local** as well as the domain for which the name server is responsible. For example:

```
zone "0.0.127.in-addr.arpa" IN {
    type master;
    file "/etc/named.local";
};
```

- Specify the name of the cache zone file. For example:

```
zone "." IN {
    type hint;
    file "/etc/named.ca";
};
```

2. Edit the **/etc/named.ca** file. See the DOMAIN Cache File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a cache file.

This file contains the addresses of the servers that are authoritative name servers for the root domain of the network. For example:

```
; root name servers.
    1          IN      NS      relay.century.com.
relay.century.com. 3600000 IN      A      129.114.1.2
```

Note: All lines in this file must be in Standard Resource Record Format.

3. Edit the **/etc/named.local** file. See the "DOMAIN Local Data File Format for TCP/IP" in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a local data file.

- a. Specify the start of authority (SOA) of the zone and the default time-to-live information. For example:

```
@ IN SOA venus.abc.aus.cntry.com. gail.zeus.abc.aus.cntry.com.
(
    1.1          ;serial
    3600         ;refresh
    600          ;retry
    3600000     ;expire
    86400       ;minimum
```

- b. Specify the name server (NS) record. For example:

```
IN      NS      venus.abc.aus.century.com.
```

- c. Specify the pointer (PTR) record.

```
1       IN      PTR      localhost.
```

Note: All lines in this file must be in Standard Resource Record Format.

4. Create an **/etc/resolv.conf** file by issuing the following command:

```
touch /etc/resolv.conf
```


The presence of this file indicates that the host should use a name server, not the `/etc/hosts` file, for name resolution. You might want to enter records to specify the name, domain, and address of the name server.

5. Perform one of the following steps:

- Enable the **named** daemon using the **smit stnamed** SMIT fast path. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.
- Edit the `/etc/rc.tcpip` file. Uncomment the line for the **named** daemon by removing the comment (`#`) symbol from the following line:

```
#start /etc/named "$src_running"
```

This initializes the daemon with each system startup.

6. If you chose not to initialize the **named** daemon through SMIT, start the daemon for this session by typing the following command:

```
startsrc -s named
```

Configuring a Domain Mail Server

Configuring a domain mail server provides users external to your organization a simple method for addressing mail to your users. That is, without a domain mail server, the mail address must specify a particular host in your organization. For example `sam@orange.widget.com`, where `widget.com` is your organization's domain name, and `orange` is the host that `sam` uses. But with a domain mail server, users outside your organization can simply specify the user name and domain name, without having to know which host the user uses, for example, `sam@widget.com`.

To configure a domain mail server, use the Web-based System Manager, **wsm**, or use one of the following procedures.

To Configure a Domain Mail Server:

1. Create a mail exchanger (MX) record and an address (A) record for the mail server `black.widget.com`:

```
widget.com      IN    MX    10 black.widget.com
widget.com      IN    A     192.10.143.9
black.widget.com IN    A     192.10.143.9
```

2. Edit **sendmail.cf** on the mail server (`black.widget.com`) to add the domain alias (the **w** class):

```
Cw $w $?D$w.$D$. widget.com
```

3. Mail clients must know where to send their non-local mail, so edit **sendmail.cf** on each client to point to the mail server (the **S** macro):

```
DRblack.widget.com
```

4. Use the **NameServOpt** option to configure the **sendmail** daemon so everyone can use the MX records defined in the name server `brown.widget.com`.

5. Add aliases for users in the domain that do not have accounts on the mail server using the aliases file, for example:

```
sam:sam@orange.widget.com
david:david@green.widget.com
judy:judy@red.widget.com
```

Note: Mailbox (MB) records can serve the same function.

6. The serial number in the SOA Resource Record must be incremented because the database has been modified.

7. Refresh the name server database by issuing the **refresh -s named** command.

8. Perform the following steps:

- a. Type the command **sendmail -bz** to recompile the **sendmail.cf** file on the mail server.

- b. Type the command **sendmail -bi** to recompile the aliases database on the mail server.
 - c. Type the **refresh -s sendmail** command to make the changes take effect.
9. On the clients, recompile **sendmail** and run the **refresh -s sendmail** command to make the changes take effect.

There are other methods to configure a domain mail server. The following procedures use mailbox (MB), mail rename (MR), and mail group (MG) records.

To Configure a Domain Mail Server Using mailbox (MB) Records:

1. Define a mailbox (MB) record for each user in the domain. Add entries such as:

```
sam IN MB orange.widget.com.
```

to the **/etc/named.data** file on host `brown.widget.com`. These entries identify to the mail server `black.widget.com` where to send mail for each user in the domain.

2. Configure the **sendmail** daemon on the mail server `black.widget.com` to use the MB records defined in the name server `brown.widget.com`. Use the **NameServOpt** option.
3. Increment the serial number in the SOA Resource Record, because the database has been modified.
4. Refresh the name server database by running the **refresh -s named** command.
5. Type the command **sendmail -bz** to recompile the **sendmail.cf** file on the mail server, and then type the **refresh -s sendmail** command to make the changes take effect.

Defining a Mail Rename (MR) Record for a User:

1. Edit the **/etc/named.data** file on your domain name server.
2. Add a Mail Rename record for each alias. For example, if a user `sam` has an alias `sammy`, the Mail Rename record is:

```
sammy IN MR sam
```

This record causes all mail addressed to `sammy` to be delivered to `sam`. Each MR record should be entered on a line by itself.

3. The serial number in the SOA Resource Record must be incremented, because the database has been modified.
4. Refresh the name server database by typing the **refresh -s named** command.
5. Type the command **sendmail -bz** to recompile the **sendmail.cf** file on the mail server, and then type the **refresh -s sendmail** command to make the changes take effect.

Defining Mail Group (MG) Member Records:

1. Edit the **/etc/named.data** file on your domain name server.
2. Add MG records for each mail group. MG records function like the **/etc/aliases** file, with the aliases maintained on the name server. For example:

```
users IN HINFO users-request widget.com
users IN MG sam
users IN MG david
users IN MG judy
```

This example causes all mail addressed to `users@widget.com` to be delivered to `sam`, `david`, and `judy`. Enter each MG record on a line by itself.

Note: Users `sam`, `david`, and `judy` must have MB records defined.

3. The serial number in the SOA Resource Record must be incremented, because the database has been modified.
4. Refresh the name server database by typing the **refresh -s named** command.

5. Type the command **sendmail -bz** to recompile the **sendmail.cf** file on the mail server, and then type the **refresh -s sendmail** command to make the changes take effect.

Defining Mail Exchanger (MX) Records:

1. Edit the **/etc/named.data** file on your domain name server.
2. Add MX records for each machine not directly connected to your network to which you wish to forward mail. For example, if mail addressed to users on purple.widget.com should be forwarded to post.office.widget, the MX record looks similar to the following:

```
purple.widget.com IN MX 0 post.office.widget.
```

You must specify both host and machine names when using MX records. Enter each MX record on a line by itself. You can use wildcards, for example:

```
*.widget.com IN MX 0 post.office.widget.
```

This example causes mail to an unknown host (a host without an explicit MX record) in the widget.com domain to be forwarded to post.office.widget.

Note: Wildcard MX records are not appropriate for use on the Internet.

3. The serial number in the SOA Resource Record must be incremented because the database has been modified.
4. Refresh the name server database by typing the **refresh -s named** command.
5. Type the command **sendmail -bz** to recompile the **sendmail.cf** file on the mail server, and then type the **refresh -s sendmail** command to make the changes take effect.

Configuring a Forwarder

To configure a forwarder server, use the Web-based System Manager, **wsm**, or use the following procedure, which edits a series of files and then uses SMIT or the command line to start the **named** daemon.

1. Edit the **/etc/named.conf** file. If there is no **named.conf** file in the **/etc** directory, copy the **/usr/samples/tcpip/named.conf** sample file into the **/etc** directory and edit it. See the "named.conf File Format for TCP/IP" in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a conf file.

- Specify a forwarders line in the options stanza of the **/etc/named.conf** file that lists the IP addresses of the name servers that should receive the forwarded requests. For example:

```
options {
    ...
    forwarders { 192.100.61.1; 129.35.128.222; };
    ...
};
```

- Specify the hint zone. For example:

```
zone "." IN {
    type hint;
    file "/etc/named.ca";
};
```

2. Edit the **/etc/named.ca** file. See the "DOMAIN Cache File Format for TCP/IP" in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a cache file.

This file contains the addresses of the servers that are authoritative name servers for the root domain of the network. For example:

```
; root name servers.
    1          IN      NS      relay.century.com.
relay.century.com. 3600000 IN      A       129.114.1.2
```

Note: All lines in this file must be in Standard Resource Record Format.

3. Edit the **/etc/named.local** file. See the DOMAIN Local Data File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a local data file.
 - a. Specify the start of authority (SOA) of the zone and the default time-to-live information. For example:

```
@ IN SOA venus.abc.aus.cntry.com. gail.zeus.abc.aus.cntry.com.
(
    1.1      ;serial
    3600    ;refresh
    600     ;retry
    3600000 ;expire
    86400)  ;minimum
```

- b. Specify the name server (NS) record. For example:

```
IN NS venus.abc.aus.century.com.
```

- c. Specify the pointer (PTR) record.

```
1 IN PTR localhost.
```

Note: All lines in this file must be in Standard Resource Record Format.

4. Create an **/etc/resolv.conf** file by typing the following command:

```
touch /etc/resolv.conf
```

The presence of this file indicates that the host should use a name server, not the **/etc/hosts** file, for name resolution.

Alternatively, the **/etc/resolv.conf** file might contain the following entry:

```
nameserver 127.0.0.1
```

The 127.0.0.1 address is the loopback address, which causes the host to access itself as the name server. The **/etc/resolv.conf** file may also contain an entry like the following:

```
domain domainname
```

In the previous example, the *domainname* value is *austin.century.com*.

5. Perform one of the following steps:
 - Enable the **named** daemon using the **smit stnamed** SMIT fast path. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.
 - Edit the **/etc/rc.tcpip** file. Uncomment the line for the **named** daemon by removing the comment (#) symbol from the following line:

```
#start /etc/named "$src_running"
```

This initializes the daemon with each system startup.

6. If you chose not to initialize the named daemon through SMIT, start the daemon for this session by typing the following command:

```
startsrc -s named
```

Configuring a Forward Only Name Server

To configure a forward only name server, use the Web-based System Manager, **wsm**, or use the following procedure, which edits a series of files and then uses SMIT or the command line to start the **named** daemon.

Note: You can achieve a similar configuration without running a forward only name server. Instead, create an **/etc/resolv.conf** file that contains name server lines that point to the forwarders you wish to use.

1. Edit the **/etc/named.conf** file. If there is no **named.conf** file in the **/etc** directory, copy the **/usr/samples/tcpip/named.conf** sample file into the **/etc** directory and edit it. See the **named.conf File Format for TCP/IP** in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a **conf** file.

- Specify the forwarders and forward only lines in the options stanza of the **/etc/named.conf** file listing the IP addresses of the name servers receiving the forwarded requests. For example:

```
options {
    ...
    forwarders { 192.100.61.1; 129.35.128.222; };
    forward only;
    ...
};
```

2. Edit the **/etc/named.ca** file. See the **DOMAIN Cache File Format for TCP/IP** in *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a cache file. This file contains the addresses of the servers that are authoritative name servers for the root domain of the network. For example:

```
; root name servers.
      1          IN      NS      relay.century.com.
relay.century.com. 3600000 IN      A      129.114.1.2
```

Note: All lines in this file must be in Standard Resource Record Format.

3. Edit the **/etc/named.local** file. See the **DOMAIN Local Data File Format for TCP/IP** in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a local data file.

- a. Specify the start of authority (SOA) of the zone and the default time-to-live information. For example:

```
@ IN SOA venus.abc.aus.cntry.com. gail.zeus.abc.aus.cntry.com.
(
    1.1      ;serial
    3600     ;refresh
    600      ;retry
    3600000  ;expire
    86400    ;minimum
```

- b. Specify the name server (NS) record. For example:

```
IN      NS      venus.abc.aus.century.com.
```

- c. Specify the pointer (PTR) record.

```
1      IN      PTR      localhost.
```

Note: All lines in this file must be in Standard Resource Record Format.

4. Create an **/etc/resolv.conf** file by typing the following command:

```
touch /etc/resolv.conf
```

The presence of this file indicates that the host should use a name server, not the **/etc/hosts** file, for name resolution.

Alternatively, the **/etc/resolv.conf** file might contain the following entry:

```
nameserver 127.0.0.1
```

The 127.0.0.1 address is the loopback address, which causes the host to access itself as the name server. The **/etc/resolv.conf** file can also contain an entry such as:

```
domain domainname
```

In the previous example, the *domainname* value is *austin.century.com*.

5. Perform one of the following steps:

- Enable the **named** daemon using the **smit stnamed** SMIT fast path. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.
- Edit the **/etc/rc.tcpip** file. Uncomment the line for the **named** daemon by removing the comment (#) symbol from the following line:

```
#start /etc/named "$src_running"
```

This initializes the daemon with each system startup.

6. If you chose not to initialize the **named** daemon through SMIT, start the daemon for this session by typing the following command:

```
startsrc -s named
```

Configuring a Host to Use a Name Server

To configure a host to use a name server, use the Web-based System Manager, **wsm**, or use the following procedure.

1. Create an **/etc/resolv.conf** file.
2. If this host is to use more than one name server, add the names of the other name servers.
3. Assuming the name server is operational, you can test the communication between the host and the name server by typing the following command:

```
host hostname
```

Use the name of a host that should be resolved by the name server to see if the process is working. The output you receive should appear similar to the following:

```
brown.abc.aus.century.com is 129.35.145.95
```

Other configuration tasks are shown in the following table.

Configuring a Host to Use Name Server Tasks			
Task	SMIT Fast Path	Command or File	Web-based System Manager Management Environment
Create an /etc/resolv.conf File	smit stnamerslv2	create and edit /etc/resolv.conf ¹	
List All the Name Servers Used by a Host	smit lsnamerslv	view /etc/resolv.conf	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Hosts File → Contents of /etc/hosts file.
Add a Name Server	smit mknamerslv	edit /etc/resolv.conf ²	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → DNS . In the Name Server IP Address field, type the IP Address . Click Add → OK .

Configuring a Host to Use Name Server Tasks			
Remove a Name Server	smit rmnamerslv	edit /etc/resolv.conf	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → DNS . Select a name server in Name server to search . Click Delete → OK .
Start/Restart Using Domain Name Resolution	smit stnamerslv		Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → DNS . Select the Enable domain name resolution using Domain Name Service (DNS) check box. Click OK .
Stop Using Domain Name Resolution	smit spnamerslv		Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → DNS . Clear the Enable domain name resolution using Domain Name Service (DNS) check box. Click OK .
Change/Show the Domain	smit mkdomain	edit /etc/resolv.conf	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → DNS . → Domain name to search . Click Add → OK .
Remove the Domain	smit rmdomain	edit /etc/resolv.conf	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → DNS . Select a domain in the Domain search list . Click Delete → OK .

Notes:

1. On the first line of the **/etc/resolv.conf** file, type the word **domain** followed by the full name of the domain that this host is in. For example:
domain abc.aus.century.com
2. On any blank line below the domain line, type the word **nameserver**, followed by at least one space, followed by the dotted decimal Internet address of the name server that this host is to use

(the name server must serve the domain indicated by the domain statement). You can have up to 16 name server entries. For example, your `/etc/resolv.conf` file might contain the entries:

```
nameserver 192.9.201.1
nameserver 192.9.201.2
```

The system queries the name servers in the order listed.

Configuring Dynamic Zones on the DNS Name Server

The `named` command allows for dynamic updates. The named database and configuration files need to be configured to allow for client machines to issue updates. A zone can be set to dynamic or static. The default zone is static.

To make a zone dynamic, you must add the keyword **allow-update** to that zone's stanza in the `/etc/named.conf` file. The **allow-update** keyword specifies an Internet address match list that defines hosts allowed to submit updates. See the `named.conf` File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a `conf` file. In the following example, all hosts are allowed to update the dynamic zone:

```
zone "aoot.austin.ibm.com" IN {
    type master;
    file "named.data";
    allow-update { any; };
};
```

After a zone is marked dynamic, three modes of security can be initiated:

Unsecured Allows anyone at anytime to update any information in the zone.

Attention: Use of this mode is not recommended. It can lead to data loss, data interception, and user frustration. At the least, an unsecured zone should be limited to updates only from specific Internet addresses.

Controlled Allows for the creation of new information and the replacement of existing information. This is probably the easiest mode to use for a secure transition environment. This mode also requires that all incoming updates be timestamped and have keyed signatures.

Presecured Requires all updates to existing information be replaced with similar information. This mode also requires that all incoming updates be timestamped and have keyed signatures.

A dynamic zone defaults to unsecured mode. To use one of the other modes, type **controlled** or **presecured** after the keyword **update-security** in the zone stanza of the `/etc/named.conf` file. This tells the `named` server the level of security to use with that zone. For example:

```
zone "aoot.austin.ibm.com" IN {
    type master;
    file "named.boot";
    allow-update { any; };
    update-security controlled;
};
```

After a mode is selected, the actual data files must be modified for your level of security. In unsecured mode, the data files are used "as is." For controlled or presecured mode, you must generate a set of master server/hostname key pairs for each name in the zone. This is done with the `nsupdate` command using the `-g` option. This command generates the key pair (a private and a public key). These keys are needed to authentically sign for updates. After generating all the keys for your list of zone names, you need to add them to the data file. The KEY format is as follows:

<i>Index</i>	<i>ttl</i>	<i>Class</i>	<i>Type</i>	<i>KeyFlags</i>	<i>Protocol</i>	<i>Algorithm</i>	<i>KeyData</i>
--------------	------------	--------------	-------------	-----------------	-----------------	------------------	----------------

where:

<i>Index</i>	Specifies the name used to reference the data in the zone.
<i>tll</i>	Specifies the time-to-live (TTL) for this data. This is an optional field.
<i>Class</i>	Specifies the class of the data. This is dependent on the zone, but usually it is IN.
<i>Type</i>	Indicates the type of the record. In this case, it is KEY.
<i>KeyFlags</i>	Gives named information about the key. 0x0000 defines the typical key record used for a host. 0x0100 defines the key record associated with the zone name.
<i>Protocol</i>	Specifies the protocol to use. Currently, there is only one, 0.
<i>Algorithm</i>	Specifies the algorithm of the key. Currently, there is only one, 1. This is the MD5 Private/Public authentication method.
<i>KeyData</i>	Indicates the key in base64 representation. The nsupdate command generates both the public and private keys in base64 representation. The public key is listed last in the output file.

Example

To ensure security over a host name in a dynamic zone, a line similar to the following needs to be added to the zone file for the zone containing the hostname.

```
bears 4660 IN KEY 0x0000 0 1 AQtg.....
```

The above example indicates that bears has a KEY record defined. Someone wanting to update bears would have to sign his update with the private key matching the public key in the database. For the **nsupdate** command to succeed, the private key needs to be placed on the client in a keyfile (defaults to **/etc/keyfile**). It should follow the format:

```
hostname      mastername      base64      key
```

A similar KEY entry is required in the zone definition section. *A zone key is required for both presecured and controlled modes or the mode is considered to be unsecured.* This can be done as shown in the previous bears example, but the private key is left for the administrator to use with the **nsupdate** command's administrative mode.

To generate a key pair using the **nsupdate** command, type:

```
nsupdate -g -h ZoneName -p ServerName -k AdminKeyFile
```

This generates a key for the zone. Place the last key of the pair in the beginning section for the zone as follows:

```
IN KEY 0x0100 0 1 Key
```

The zone is ready to be loaded. The administrator should use the zone key to apply updates and maintenance operations on the zone.

Planning and Configuration for LDAP Name Resolution

The Lightweight Directory Access Protocol (LDAP) is an open industry standard that defines a method for accessing and updating information in a directory. An LDAP schema defines the rules for ordering data. The **ibm-HostTable** object class, part of the IBM SecureWay Directory schema, can be used to store the name-to-Internet-address mapping information for every host on the network.

The **ibm-HostTable** object class is defined as follows:

```
Object Class name:  ibm-HostTable
Description:        Host Table entry which has a collection of hostname to
                   IP address mappings.
OID:                TBD
RDN:                ipAddress
Superior object class: top
Required Attributes: host, ipAddress
Optional Attributes: ibm-hostAlias, ipAddressType, description
```

The attribute definitions follow:

Attribute Name: ipAddress
Description: IP Address of the hostname in the Host Table
OID: TBD
Syntax: caseIgnoreString
Length: 256
Single Valued: Yes
Attribute Name: ibm-hostAlias
Description: Alias of the hostname in the Host Table
OID: TBD
Syntax: caseIgnoreString
Length: 256
Single Valued: Multi-valued
Attribute Name: ipAddressType
Description: Address Family of the IP Address (1=IPv4, 2=IPv6)
OID: TBD
Syntax: Integer
Length: 11
Single Valued: Yes
Attribute Name: host
Description: The hostname of a computer system.
OID: 1.13.18.0.2.4.486
Syntax: caseIgnoreString
Length: 256
Single Valued: Multi-valued
Attribute Name: description
Description: Comments that provide a description of a directory object entry.
OID: 2.5.4.13
Syntax: caseIgnoreString
Length: 1024
Single Valued: Multi-valued

Use the following procedure to configure the LDAP server to store the name-to-Internet-address mapping host information.

1. Add a suffix on the LDAP server. The suffix is the starting point of the hosts database. For example, "cn=hosts". This can be done using the web-based IBM SecureWay Directory Server Administration tool.
2. Create an LDAP Data Interchange Format (LDIF) file. This can be done manually or with the **hosts2ldif** command, which creates a LDIF file from the **/etc/hosts** file. See the **hosts2ldif** Command in the *AIX 5L Version 5.1 Commands Reference* for more information. The following is a sample LDIF file:

```
dn: cn=hosts
objectclass: top
objectclass: container
cn: hosts
dn: ipAddress=1.1.1.1, cn=hosts
host: test
ipAddress: 1.1.1.1
objectclass: ibm-HostTable
ipAddressType: 1
ibm-hostAlias: e-test
ibm-hostAlias: test.austin.ibm.com
description: first ethernet interface
dn: ipAddress=fe80::dead, cn=hosts
host: test
ipAddress: fe80::dead
objectclass: ibm-HostTable
ipAddressType: 2
ibm-hostAlias: test-11
ibm-hostAlias: test-11.austin.ibm.com
description: v6 link level interface
```

3. Import the hosts directory data from the LDIF file on the LDAP server. This can be done with the **ldif2db** command or through the web-based IBM SecureWay Directory Server Administration tool.

To configure the client to access the hosts database on the LDAP server, use the following procedure:

1. Create the **/etc/resolv.lldap** file. See the *resolv.lldap File Format for TCP/IP* in the *AIX 5L Version 5.1 Files Reference* for more information and a detailed example of a **resolv.lldap** file.
2. Change the default name resolution through the **NSORDER** environment variable, the **/etc/netstvc.conf** file, or the **/etc/irs.conf** file. See the *netstvc.conf File Format for TCP/IP* or the *irs.conf File Format for TCP/IP* in the *AIX 5L Version 5.1 Files Reference* for more information.

TCP/IP Routing

The topics discussed in this section are:

- Static and Dynamic Routing
- Gateways
- Planning for Gateways
- Configuring a Gateway
- Restricting Route Use
- Dead Gateway Detection
- Manually Removing Dynamic Routes
- Configuring the routed Daemon
- Getting an Autonomous System Number

A *route* defines a path for sending packets through the Internet network to an address on another network. A route does not define the complete path, only the path segment from one host to a gateway that can forward packets to a destination (or from one gateway to another). There are three types of routes:

host route	Defines a gateway that can forward packets to a specific host on another network.
network route	Defines a gateway that can forward packets to any of the hosts on a specific network.
default route	Defines a gateway to use when a host or network route to a destination is not otherwise defined.

Routes are defined in the kernel *routing table*. The route definitions include information on networks reachable from the local host and on gateways that can be used to reach remote networks. When a gateway receives a datagram, it checks the routing tables to find out where next to send the datagram along the path to its destination.

Beginning with AIX 5.1, you can add multiple routes for the same destination in the kernel routing table. A routing lookup evaluates all routes that match the request then chooses the route with the lowest distance metric. If multiple matching routes have equal distance, a lookup chooses the most specific route. If both criteria are equal for multiple routes, routing lookups alternate choices of matching routes.

Static and Dynamic Routing

In TCP/IP, routing can be one of two types: *static* or *dynamic*. With static routing, you maintain the routing table manually using the **route** command. Static routing is practical for a single network communicating with one or two other networks. However, as your network begins to communicate with more networks, the number of gateways increases, and so does the amount of time and effort required to maintain the routing table manually.

With dynamic routing, daemons update the routing table automatically. Routing daemons continuously receive information broadcast by other routing daemons, and so continuously update the routing table.

TCP/IP provides two daemons for use in dynamic routing, the **routed** and **gated** daemons. The **gated** daemon supports Routing Information Protocol (RIP), Routing Information Protocol Next Generation

(RIPng), Exterior Gateway Protocol (EGP), Border Gateway Protocol (BGP) and BGP4+, Defense Communications Network Local-Network Protocol (HELLO), Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Internet Control Message Protocol (ICMP and ICMPv6)/Router Discovery routing protocols simultaneously. In addition, the **gated** daemon supports the Simple Network Management Protocol (SNMP). The **routed** daemon only supports Routing Information Protocol.

Routing daemons can operate in one of two modes, *passive* or *active*, depending upon the options you use when starting the daemons. In active mode, routing daemons both broadcast routing information periodically about their local network to gateways and hosts, and receive routing information from hosts and gateways. In passive mode, routing daemons receive routing information from hosts and gateways, but do not attempt to keep remote gateways updated (they do not advertise their own routing information).

These two types of routing can be used not only for gateways, but for other hosts on a network as well. Static routing works the same for gateways as for other hosts. Dynamic routing daemons, however, must be run in the passive (quiet) mode when run on a host that is not a gateway.

Gateways

Gateways are a type of router. *Routers* connect two or more networks and provide the routing function. Some routers, for example, route at the network interface level or at the physical level.

Gateways, however, route at the network level. Gateways receive IP datagrams from other gateways or hosts for delivery to hosts on the local network, and route IP datagrams from one network to another. For example, a gateway connecting two Token-Ring networks has two Token-Ring adapter cards, each with its own Token-Ring network interface. To pass on information, the gateway receives datagrams through one network interface and sends them out through the other network interface. Gateways periodically verify their network connections through interface status messages.

Gateways route packets according to the destination network, not according to the destination host. That is, a gateway machine is not required to keep track of every possible host destination for a packet. Instead, a gateway routes packets according to the network of the destination host. The destination network then takes care of sending the packet to the destination host. Thus, a typical gateway machine requires only limited disk storage capacity (if any) and limited main memory capacity.

The distance a message must travel from originating host to destination host depends upon the number of *gateway hops* it must make. A gateway is zero hops from a network to which it is directly attached, one hop from a network that is reachable through one gateway, and so on. Message distance is usually expressed in the number of gateway hops required, or *hop counts* (also called the *metric*).

Interior and Exterior Gateways

Interior gateways are gateways that belong to the same autonomous system. They communicate with each other using the Routing Information Protocol (RIP), Routing Information Protocol Next Generation (RIPng), Intermediate System to Intermediate System protocol, Open Shortest Path First protocol (OSPF), or the HELLO Protocol (HELLO). Exterior gateways belong to different autonomous systems. They use the Exterior Gateway Protocol (EGP), the Border Gateway Protocol (BGP), or BGP4+.

For example, consider two autonomous systems. The first is all the networks administered by the Widget Company. The second is all the networks administered by the Gadget Company. The Widget Company has one machine, called apple, which is Widget's gateway to the Internet. The Gadget Company has one machine, called orange, which is Gadget's gateway to the Internet. Both companies have several different networks internal to the companies. The gateways connecting the internal networks are interior gateways. But apple and orange are exterior gateways.

Each exterior gateway does not communicate with every other exterior gateway. Instead, the exterior gateway acquires a set of neighbors (other exterior gateways) with which it communicates. These neighbors are not defined by geographic proximity, but rather by their established communications with each other. The neighboring gateways, in turn, have other exterior gateway neighbors. In this way, the exterior gateway routing tables are updated and routing information is propagated among the exterior gateways.

The routing information is sent in a pair, (N,D), where N is a network and D is a distance reflecting the cost of reaching the specified network. Each gateway advertises the networks it can reach and the costs of reaching them. The receiving gateway calculates the shortest paths to other networks and passes this information along to its neighbors. Thus, each exterior gateway is continually receiving routing information, updating its routing table and then passing that information to its exterior neighbors.

Gateway Protocols

All gateways, whether interior or exterior, use protocols to communicate with each other. Here are brief descriptions of the more commonly used TCP/IP gateway protocols:

HELLO Protocol (HELLO)

HELLO is one protocol that the interior gateways use to communicate among themselves. HELLO calculates the shortest path to other networks by determining the path that has the least delay time.

Routing Information Protocol (RIP)

Routing Information Protocol is a protocol that the interior gateways use to communicate among themselves. Like the HELLO Protocol, RIP calculates the shortest path to other networks. Unlike HELLO, RIP estimates distance not by delay time, but by hop counts. Because the **gated** daemon stores all metrics internally as time delays, it converts RIP hop counts into time delays.

Routing Information Protocol Next Generation

RIPng is the RIP protocol that is enhanced to support IPv6.

Open Shortest Path First (OSPF)

OSPF is a protocol that the interior gateways use to communicate among themselves. It is a link-state protocol that is better suited than RIP for complex networks with many routers. It provides equal cost multipath routing.

Exterior Gateway Protocol (EGP)

The exterior gateways can use the Exterior Gateway Protocol to communicate among themselves. The EGP does not calculate the shortest path to other networks. Instead, it merely indicates whether a particular network is reachable or not.

Border Gateway Protocol (BGP)

The exterior gateways can use this protocol to communicate among themselves. It exchanges reachability information between autonomous systems, but provides more capabilities than EGP. BGP uses path attributes to provide more information about each route as an aid in selecting the best route.

Border Gateway Protocol 4+

BGP4+ is the BGP protocol version 4, which supports IPv6 and has other enhancements over past versions of the protocol.

Intermediate System to Intermediate System (IS-IS)

Interior gateways use IS-IS protocol to communicate among themselves. It is a link-state protocol that can route IP and ISO/CLNP packets and, like OSPF, uses a "shorter path first" algorithm to determine routes.

Planning for Gateways

Before you configure the gateways for your network, you must first:

1. Consider the number of gateways to use.
2. Decide on the type of routing to use.

Consider the Number of Gateways to Use

The number of gateways you need to configure will depend upon:

- The number of networks you want to connect.
- How you want to connect the networks.
- The level of activity on the connected networks.

For example, suppose users on Network 1, Network 2, and Network 3 all need to communicate with each other.

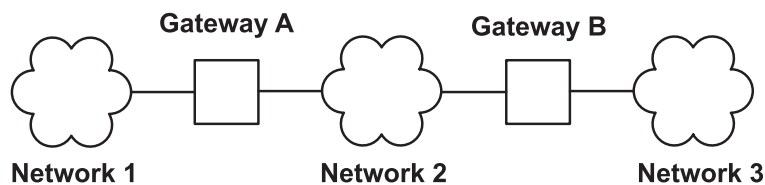


Figure 25. Simple Gateway Configuration. This illustration contains three network clouds numbered one, two, and three. Networks one and two are connected with gateway A. Networks two and three are connected with gateway B.

To connect Network 1 directly to Network 2, you would use a single gateway (Gateway A). To connect Network 2 directly to Network 3, you would use another gateway (Gateway B). Now, assuming the proper routes are defined, all the users on all three networks can communicate.

However, if Network 2 is very busy, communication between Network 1 and Network 3 might suffer unacceptable delays. Furthermore, if most of the inter-network communication occurs between Network 1 and Network 3, you might want to connect Network 1 directly to Network 3. To do this, you could use an additional pair of gateways, Gateway C (on Network 1) and Gateway D (on Network 3), with a direct connection between these two additional gateways. This may be an inefficient solution, however, because one gateway can connect more than two networks.

A more efficient solution would be to connect Gateway A to Gateway B directly, as well as to Network 2. This would require a second network adapter in both Gateway A and Gateway B. In general, the number of networks you connect through a single gateway is limited by the number of network adapter cards the gateway machine can support.

Decide on the Type of Routing to Use

If your network is small, and its configuration rarely changes, you probably want to use static routing. But if you have a large network whose configuration changes frequently, you probably want to use dynamic routing. You might decide to use a combination of static and dynamic routing. That is, you might want to give static definitions to a few specific routes, while allowing other routes to be updated by the daemons. The static routes you create are not advertised to other gateways and are not updated by the routing daemons.

If Using Dynamic Routing: Choose the routing daemon according to the type of gateway you need and the protocols your gateway must support. If the gateway is an interior gateway, and only needs to support RIP, choose the **routed** daemon. If the gateway must support any other protocol, or is an exterior gateway, choose the **gated** daemon.

Note: Unpredictable results can occur if the **gated** and **routed** daemons run on the same host at the same time.

Configuring a Gateway

To configure a machine to act as a gateway, use the following instructions. For clarity, this procedure assumes that the gateway machine connects two networks, and that the gateway machine has already been minimally configured (see Configuring TCP/IP) on one of the networks.

1. Install and configure the second network adapter, if you have not done so already. (See Installing a Network Adapter and Configuring and Managing Adapters.)
2. Choose an IP address for the second network interface, and then configure the network interface by following the instructions in Managing Network Interfaces.
3. Add a route to the second network.
4. To use a machine as an internetwork router over TCP/IP networks, enter:

```
no -o ipforwarding=1
```
5. The gateway machine can now access both of the networks to which it is directly attached.
 - a. If you want to use static routing to communicate with hosts or networks beyond these two networks, add any other routes you want.
 - b. If you want to use dynamic routing, follow the instructions in either Configuring the routed Daemon or Configuring the gated Daemon. If your internetwork is to join the Internet, you should also follow the instructions in Getting an Autonomous System Number.

Configuring Gateway Tasks			
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>	<i>Web-based System Manager Management Environment</i>
Displaying the Routing Table	smit lsroute	netstat -rn ¹	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Static Routes → Statistics.
Adding a Static Route	smit mkroute	route add destination gateway ²	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Static Routes. Complete the following in Add/Change a static route: Destination Type, Gateway address, Network interface name (drop-down menu), Subnet mask, Metric (Cost) , and the Enable active dead gateway detection check box. Click Add/Change Route .

Configuring Gateway Tasks			
Removing a Static Route	smit rmroute	route delete <i>destination gateway²</i>	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Static Routes. Select a route, and click Delete Route .
Flushing the Routing Table	smit fshrttbl	route flush	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Static Routes → Delete All.

Notes:

- The table is divided into columns for destination address, gateway address, flags, reference count (hop count), and network interface. (For a detailed discussion of each of these columns, see the **netstat** command in the *AIX 5L Version 5.1 Commands Reference*.) If frames are not reaching their destination and the routing tables indicate the correct route, one or more of the following conditions might exist:
 - Network is failing.
 - Remote host or gateway is failing.
 - Remote host or gateway is down or not ready to receive frames.
 - Remote host does not have a route back to the source network.
- The *destination* value is the dotted decimal address or symbolic name of the destination host or network, and the *gateway* value is the dotted decimal address or symbolic name of the gateway. (A default route specifies 0 as the destination.)

Restricting Route Use

Routes can be restricted so they can be used only by some users. The restrictions are based on the primary group IDs of users. Using the **route** command, you can specify a list of up to 32 group IDs that are allowed or not allowed to use a route. If the list is of allowed groups, any user that belongs to any group on the list can use the route. If the list is of disallowed groups, only users that do not belong to any of the groups on the list can use the route. The root user can use any route.

Groups can also be associated with an interface using the **ifconfig** command. In this case, a forwardable packet can use any route allowed for the groups associated with its incoming interface.

If there are two or more routes to the same destination, any ICMP redirects that are received for that destination will be ignored and path MTU discovery will not be done on those routes.

Dead Gateway Detection

Beginning with AIX 5.1, a host can be configured to detect whether a gateway it is using is down and adjust its routing table accordingly. If the network option **-passive_dgd** is 1, passive dead gateway detection is enabled for the entire system. If no response is received for **dgd_packets_lost** consecutive ARP requests to a gateway, that gateway is considered to be down and the distance metrics (also known as *hopcount* or *cost*) for all routes using that gateway are raised to the maximum possible value. After **dgd_retry_time** minutes have passed, the routes' costs are restored to their user-configured values. The

host also takes action based on failing TCP connections. If **dgd_packets_lost** consecutive TCP packets are lost, the ARP entry for the gateway in use is deleted and the TCP connection tries the next-best route. The next time the gateway is used, the above actions take place if the gateway is actually down. **passive_dgd**, **dgd_packets_lost**, and **dgd_retry_time** can all be configured using the **no** command.

Hosts can also be configured to use active dead gateway detection on a per-route basis with the **-active_dgd** flag of the **route** command. Active dead gateway detection pings all gateways used by routes for which it is enabled every **dgd_ping_time** seconds. If no response is received from a gateway, it is pinged more rapidly up to **dgd_packets_lost** times. If no response is received, the costs of all routes using that gateway are raised. The gateway continues to be pinged, and if a response is eventually received, the costs on the routes are restored to their user-configured values. **dgd_ping_time** can be configured using the **no** command.

Dead gateway detection is generally most useful for hosts that use static rather than dynamic routing. Passive dead gateway detection has low overhead and is recommended for use on any network that has redundant gateways. However, passive dead gateway detection is done on a best-effort basis only. Some protocols, such as UDP, do not provide any feedback to the host if a data transmission is failing, and in this case no action can be taken by passive dead gateway detection. Active dead gateway detection is most useful when a host must discover immediately when a gateway goes down. Since it queries each gateway for which it is enabled every few seconds, there is some network overhead associated with its use. Active dead gateway detection is recommended only for hosts that provide critical services and on networks with a limited number of hosts.

Manually Removing Dynamic Routes

If you are using the **routed** daemon, a manually deleted route is *not* replaced by incoming RIP information (because *ioctl*'s are used). If you are using the **gated** daemon, and the **-n** flag is not used, the manually deleted route *is* replaced by the route as discovered in incoming RIP information.

Configuring the routed Daemon

To configure the **routed** daemon:

1. Remove the comment symbol (**#**) and modify the **routed** clause in the **/etc/rc.tcpip** shell script. This automatically starts the **routed** daemon with each system startup.
 - Specify whether you want the gateway to run in active (**-s** flag) or passive (**-q** flag) mode.
 - Specify whether you want packet tracing on or off (**-t** flag). Packet tracing can also be turned on after the **routed** daemon is already started by using the **kill** command to send a **SIGUSR1** signal to the daemon. This signal can also be used to increment the level of tracing through four levels. In addition, packet tracing can be turned off while the **routed** daemon is running by using the **kill** command to send a **SIGUSR2** signal to the daemon. For more information, see the **routed** daemon and the **kill** command.
 - Specify whether you want debugging turned on or off (**-d** flag). If you use this flag, specify which log file you want debugging information stored in, or choose for it to be directed to the console display.
 - Specify whether you are running the **routed** daemon on a gateway (**-g** flag).

Note: A host that is not a gateway can run the **routed** daemon, but it must be run in passive mode.

2. Identify any known networks by listing them in the **/etc/networks** file. See Networks File Format for TCP/IP in the *AIX 5L Version 5.1 Files Reference* for more information. A sample **networks** file is located in the **/usr/samples/tcpip** directory.
3. Set up routes in the **/etc/gateways** file to any known gateways that are not directly connected to your network. Refer to Gateways File Format for TCP/IP in *AIX 5L Version 5.1 Files Reference* for detailed examples of entries in the **/etc/gateways** file. A sample **gateways** file is located in the **/usr/samples/tcpip** directory.

Attention: Do not run the **routed** daemon and the **gated** daemon on the same machine. Unpredictable results can occur.

Configuring the gated Daemon

To configure the **gated** daemon:

1. Decide which gateway protocols are most appropriate for your system. The choices for routing protocols are EGP, BGP, RIP, RIPng, HELLO, OSPF, ICMP/Router Discovery, and IS-IS. You can also use SNMP, a protocol allowing you to change or show management information for a network element from a remote host.

Note: Use EGP, BGP, or BGP4+ to advertise addresses of networks in an autonomous system to gateways in other autonomous systems. If you are on the Internet, EGP, BGP, or BGP4+ must be used to advertise network reachability to the core gateway system. Use the interior routing protocols to advertise reachability information within an autonomous system.

2. Identify any known networks by listing them in the **/etc/networks** file. See Networks File Format for TCP/IP in *AIX 5L Version 5.1 Files Reference* for more information. A sample **networks** file is located in the **/usr/samples/tcpip** directory.
3. Edit the **/etc/gated.conf** file to reflect the desired **gated** daemon configuration.

Note: The **gated** version on AIX 4.3.2 and higher is 3.5.9. The syntax of the **/etc/gated.conf** file has changed. The examples given below are for the 3.5.9 version of **gated**. To configure the **/etc/gated.conf** file for versions prior to AIX 4.3.2, use the syntax provided in the **/etc/gated.conf** file itself.

- a. Specify the level of trace output you want. If tracing is needed before the **gated.conf** file is parsed, use the **-t** flag to turn tracing on when the daemon starts. See **gated Daemon** in *AIX 5L Version 5.1 Commands Reference* for more information.
- b. Specify the routing protocols you want to use. Each protocol has its own protocol statement. Remove the comment symbols (**#**) and modify the statements corresponding to the protocols you want to use.

- If using EGP:

- Set up the EGP `autonomoussystem` clause. Obtain an autonomous system number from the Internet authority if you are on the Internet, or if not, assign an autonomous system number considering the autonomous system numbers of other systems on your network.

- Set the EGP statement to `yes`.

- Set up a `group` clause for each autonomous system.

- Set up a `neighbor` clause for each neighbor in that autonomous system. For example:

```
autonomoussystem 283 ;
```

```
egp yes {
    group maxup 1 {
        neighbor nogendefault 192.9.201.1 ;
        neighbor nogendefault 192.9.201.2 ;
    } ;
    group {
        neighbor 192.10.201.1 ;
        neighbor 192.10.201.2 ;
    } ;
} ;
```

- If using RIP or HELLO:

- Set the RIP or HELLO statement to `yes`.

- Specify `nobroadcast` in the RIP or HELLO statement if you want the gateway only to accept routing information, not broadcast information. Or specify `broadcast` in the RIP or HELLO statement if you want the gateway to broadcast routing information as well as accept routing information.
- If you want the gateway to send directly to source gateways, use the `sourcegateways` statement. Specify a gateway name or Internet address in dotted decimal in the `sourcegateways` clause. For example:

```
# Send directly to specific gateways

rip/hello yes {
    sourcegateways
        101.25.32.1
        101.25.32.2 ;
} ;
```

The following example shows the RIP/HELLO stanza in the **gated.conf** file of a machine that does not send RIP packets, and does not receive RIP packets on its `tr0` interface.

```
rip/hello nobroadcast {
    interface tr0 noripin ;
} ;
```

- If using BGP:
 - Set up the `BGP autonomoussystem` clause. Obtain an autonomous system number from the Internet authority if you are on the Internet, or if not, assign an autonomous system number considering the autonomous system numbers of other systems on your network.
 - Set the BGP statement to `yes`.
 - Set up a `peer` clause for each neighbor in that autonomous system. For example:

```
# Perform all BGP operations

bgp yes {
    peer 192.9.201.1 ;
} ;
```

- If using SNMP:
 - Set the SNMP statement to `yes`.

```
snmp yes ;
```

Configuring the **gated** Daemon To Run IPv6

To configure the **gated** daemon to run under Internet Protocol version 6 (IPv6), first ensure that your system has been configured for IPv6 and IPv6 routing:

1. Run **autoconf6** to automatically configure your interfaces for IPv6.
2. Configure site local addresses for each IPv6 interface on which you want to use IPv6 routing using the following command:

```
ifconfig interface inet6 fec0:n::address/64 alias
```

where

interface

Is the name of the interface, such as `tr0` or `en0`.

n

Is any decimal number; for example, 11

address

Is the portion of the IPv6 interface address that follows the double colons; for example, given the IPv6 address `fe80::204:acff:fe86:298d`, the *address* entry would be `204:acff:fe86:298d`.

Note: You can use the command **netstat -i** to see what your IPv6 address is for each configured interface.

If token ring tr0 has an IPv6 address of fe80::204:acff:fe86:298d, you issue the following command:

```
ifconfig tr0 inet6 fec0:13::204:acff:fe86:298d/64 alias
```

3. Turn on IPv6 forwarding with the following command:

```
no -o ip6forwarding=1
```

4. Start **ndpd-router** with the following command:

```
ndpd-router -g
```

See **ndpd-router** to determine which flags to use for your network configuration.

Starting **ndpd-router** allows your system to act as a router for the Neighbor Discovery Protocol. Neighbor Discovery Protocol routers inform Neighbor Discovery hosts with routing information so hosts can route IPv6 packets.

Any hosts on the network that you want to be part of the IPv6 network must run **ndpd-host**. Hosts on the network that run **ndpd-host** will recognize themselves as part of an IPv6 network and use Neighbor Discovery Protocol, which allows them to determine and monitor link-layer addresses both to allow neighbor routing and to find neighboring routers for forwarding packets.

See **ndpd-router**, **ndpd-host**, or read RFC 1970, *Neighbor Discovery*, for more information.

Next, configure the **gated** daemon:

1. Decide which IPv6 gateway protocols are most appropriate for your system. The choices for IPv6 routing protocols are Border Gateway Protocol enhanced for IPv6 (BGP4+) and Routing Information Protocol Next Generation (RIPng).
2. Edit the **etc/gated.conf** file to reflect the desired **gated** daemon configuration.

Note:AIX 4.3.2 and later run **gated** version 3.5.9. The syntax of the **gated.conf** file has changed slightly from earlier versions. Read the **gated.conf** documentation or use the sample file that is shipped in the **/usr/sample/tcpip** directory for correct syntax.

When configuring **BGP4+** or **RIPng**, use IPv6 addresses in which the syntax specifies an IP address.

Note: By default, **RIPng** multicasts its packets.

Once the **/etc/gated.conf** file has been modified, the **gated** daemon can be started.

Getting an Autonomous System Number

If you use **EGP** or **BGP**, you should obtain an official *autonomous system number* for your gateway. To obtain an official autonomous system number, contact the NIC at INFO@INTERNIC.NET.

Path MTU Discovery

For two hosts communicating across a path of multiple networks, a transmitted packet becomes fragmented if its size is greater than the smallest MTU of any network in the path. Because packet fragmentation can result in reduced network performance, it is desirable to avoid fragmentation by transmitting packets with a size no greater than the smallest MTU in the network path. This size is called the path MTU.

The operating system supports a path MTU discovery algorithm as described in RFC 1191. Path MTU discovery can be enabled for TCP and UDP applications by modifying the **tcp_pmtu_discover** and **udp_pmtu_discover** options of the **no** command. When enabled for TCP, path MTU discovery will automatically force the size of all packets transmitted by TCP applications to not exceed the path MTU.

Since UDP applications themselves determine the size of their transmitted packets, UDP applications must be specifically written to utilize path MTU information by using the **IP_FINDPMTU** socket option, even if the **udp_pmtu_discover no** option is enabled. By default, the **tcp_pmtu_discover** and **udp_pmtu_discover** options are disabled on AIX 4.2.1 through AIX 4.3.1, and enabled on AIX 4.3.2 and later.

When the path MTU has been discovered for a network route, a separate host route is cloned for the path. These cloned host routes, as well as the path MTU value for the route, can be displayed using the **netstat -r** command. Accumulation of cloned routes can be avoided by allowing unused routes to expire and be deleted. Route expiration is controlled by the **route_expire** option of the **no** command. Route expiration is disabled by default on AIX 4.2.1 through AIX 4.3.1, and set to 1 minute on AIX 4.3.2 and later.

Since routes can change dynamically, the path MTU value for a path might also change over time. Decreases in the path MTU value will result in packet fragmentation, so discovered path MTU values are periodically checked for decreases. By default, decreases are checked for every 10 minutes, and this value can be changed by modifying the value of the **pmtu_default_age** option of the **no** command.

Increases in the path MTU value can result in a potential increase in network performance, so discovered path MTU values are periodically checked for increases. By default, increases are checked for every 30 minutes, and this value can be changed by modifying the value of the **pmtu_rediscover_interval** option of the **no** command.

If not all of the routers in the network path support RFC 1191, then it might not be possible to determine an exact path MTU value. In these cases, the **mmtu** command can be used to add or delete path MTU values that are attempted.

Notes:

1. Path MTU discovery cannot be used on duplicate routes, including those configured for group routing.
2. Enabling path MTU discovery sets the value of the **arpqsize** option of the **no** command to a minimum value of 5. This value is not decreased if path MTU discovery is subsequently disabled.

SLIP

Configuring SLIP over a Modem

To configure Serial Line Interface Protocol (SLIP) between two systems that communicate through a modem, you can use the Web-based System Manager, **wsm**, or use the following procedure, which alternates between the System Management Interface Tool (SMIT) interface and the command line to complete the configuration. For clarity, the following instructions use the names **bronze** and **gold** for the two hosts.

1. Physically connect the modems to **bronze** and **gold**.
2. To create a tty on **bronze** using SMIT:
 - a. Enter:

```
smit maktty
```
 - b. Select **rs232** as the type of tty you wish to create.
 - c. Select an available serial port, for example **sa0** (system serial port 1).
 - d. Select a port number for this tty from the list.
 - e. Set the BAUD rate to the baud rate of your modem.
 - f. Set Enable LOGIN to disable.
 - g. Exit SMIT.
3. Create a tty on **gold**.

Follow the same procedure as you did for bronze (in step 2), except set Enable LOGIN to **enable**. The rest of these instructions assume that the tty number on both bronze and gold is tty1.

4. Test the physical connection with ATE.
 - a. On bronze, enter:
ate
 - b. At the Unconnected Main Menu, select the **Alter** subcommand. Set the Rate to the baud rate of your modem and the Device to tty1.
 - c. At the Unconnected Main Menu, select the **Connect** subcommand. When ATE prompts you for a phone number, enter the phone number of gold and press Enter.
 - d. At this point, you should receive a login prompt for gold. Login.
 - e. Return to the connected screen, logout from gold, press **Ctrl-v** (to get to the ATE CONNECTED MAIN MENU), press **t** to terminate the connection, and press **q** to exit ATE.

Note: If you do not receive a login prompt, return to step 1 and verify that your configuration is correct. Do not proceed until you can login to gold.

Because the tty configuration for use with ATE is slightly different from the configuration for use with SLIP, you must make the following changes:

- a. On bronze, enter:
smit chgtty
- b. On gold, enter:
smit chgtty-pdisable tty1

Select **tty1**, then select **Change/Show TTY Program**. Set Enable LOGIN to disable, then exit SMIT.

5. Add the following line to the **/usr/lib/uucp/Devices** file on both bronze and gold:
Direct tty1 - 9600 direct

or replace 9600 with whatever your modem speed is.

6. Create a SLIP network interface on bronze.
 - a. Enter:
smit mkinet1s1
 - b. For TTY PORT for SLIP Network Interface, select **tty1**.
 - c. Specify an INTERNET ADDRESS, for example, 130.130.130.1.
 - d. Specify the DESTINATION address (of gold), for example, 130.130.130.2.
 - e. Specify the BAUD RATE of your modem.
 - f. Specify the DIAL STRING, for example:
 - `"" AT OK ATDT555-1234 CONNECT ""`
 - The meaning of this command is: Use tty1 at 9600 baud. Send AT to the modem. The modem should respond with OK. Dial the phone number 555-1234. The modem should respond with CONNECT. The spaces before and after the "" characters are necessary.
 - g. Exit SMIT.
7. Create a SLIP network interface on gold.

Follow the same procedure as you did for bronze (in step 5), except exchange the INTERNET ADDRESS and the DESTINATION address.
8. Add the following two entries to the **/etc/hosts** file on both bronze and gold:
130.130.130.1 bronze
130.130.130.2 gold

The name you assign must be unique. In other words, if the Token-Ring interface on bronze is already assigned the name bronze, assign the SLIP interface a name such as bronze_slip.

Note: For a simplified interface to the **slattach** command, you might use the script `/usr/sbin/slipcall`.

9. Test the SLIP connection.

a. On bronze, enter:

`ping gold`

b. On gold, enter:

`ping bronze`

If both tests succeed, the SLIP connection is ready for use. If not, return to step 5 and verify that the configuration on both bronze and gold is correct.

Configuring SLIP over a Null Modem Cable

To configure SLIP between two systems that are attached using a null modem cable, you can use the Web-based System Manager, **wsm**, or use the following procedure, which alternates between the System Management Interface Tool (SMIT) interface and the command line to complete the configuration. For clarity, these instructions use the names bronze and gold for the two hosts.

1. Physically connect bronze and gold by the null modem cable. The following cables are required.. (The cables are listed in the order they will be connected from bronze to gold.)

a. Cable B (part number 00G0943). Serial Port Jumper Cable; two are provided with each system, except models 220, 340, and 350 do not require them.

b. Cable D (part number 6323741, feature code 2936). Asynchronous Cable EIA-232/V.24.

c. Cable E (part number 59F2861, feature code 2937). Printer/Terminal Interposer EIA-232 (null modem cable).

d. Changer Adapter (both sides of the adapter are sockets).

2. Create a tty on bronze.

a. Enter:

`smit maktty`

b. Select **rs232** as the type of tty you wish to create.

c. Select an available serial port, for example **sa0** (system serial port 1).

d. Select a port number for this tty from the list.

e. Set the BAUD rate to 19200. (Later, you will change this to 38400. But for now, use 19200.)

f. Set Enable LOGIN to disable, and then exit SMIT.

3. Create a tty on gold.

Follow the same steps as you did for bronze (in step 2), except set Enable LOGIN to **enable**.

Note: The rest of these instructions assume that the tty number on both bronze and gold is tty1.

4. Test the physical connection with ATE.

a. On bronze, enter:

`ate`

b. At the Unconnected Main Menu, select the **Alter** subcommand. Set the Rate to 19200, and the Device to tty1.

c. At the Unconnected Main Menu, select the **Connect** subcommand. When ATE prompts you for a phone number, press Enter. You should receive the message:

`ate: 0828-010 The Connect command has made a connection through port tty1`

d. Press Enter. You should receive a login prompt for gold. Login to gold.

- e. Finally, return to the connected screen, logout from gold, press **Ctrl-v** (to get to the ATE CONNECTED MAIN MENU), press **t** to terminate (end) the connection, and press **q** to exit ATE.

Note: If you do not receive a login prompt, return to step 1 and verify that your configuration is correct. Do not proceed until you can login to gold.

Since the tty configuration for use with ATE is slightly different from the configuration for use with SLIP, you must make the following changes:

- a. On bronze, enter:

```
smit chgtty
```

Select **tty1**. Set the BAUD rate to 38400, and then exit SMIT.

- b. On gold, enter:

```
pdisable tty1
```

- c. On gold, enter:

```
smit chgtty
```

Select **tty1**. Set Enable LOGIN to disable, set the BAUD rate to 38400, and then exit SMIT.

5. Add the following line to the **/usr/lib/uucp/Devices** file on both bronze and gold:

```
Direct tty1 - 38400 direct
```

6. Create a SLIP network interface on **bronze**.

- a. Enter:

```
smit mkinet1s1
```

- b. For TTY PORT for SLIP Network Interface, select **tty1**.

- c. Specify an INTERNET ADDRESS, for example 130.130.130.1.

- d. Specify the DESTINATION address (of gold), for example, 130.130.130.2, and then select OK or Enter.

7. Create a SLIP network interface on gold.

Follow the same procedure as you did for bronze (in step 5), except exchange the INTERNET ADDRESS and the DESTINATION address.

8. Add the following two entries to the **/etc/hosts** file on both bronze and gold:

```
130.130.130.1 bronze
130.130.130.2 gold
```

The name you assign must be unique. In other words, if the Token-Ring interface on bronze is already assigned the name bronze, assign the SLIP interface a name such as bronze_slip.

9. Start SLIP on both bronze and gold.

Enter:

```
slattach tty1
```

10. Test the SLIP connection.

- a. On bronze, enter:

```
ping gold
```

- b. On gold, enter:

```
ping bronze
```

If both tests succeed, the SLIP connection is ready for use. If not, return to step 5 and verify that the configuration on both bronze and gold is correct.

Deactivating a SLIP Connection

To deactivate a SLIP connection:

1. Enter:

```
ps -ef | grep slatt
```

Note the process numbers of processes associated with the **slattach** command.

2. For each process number, enter:

```
kill process_number
```

Do not use the **-9** flag of the **kill** command.

If **slattach** is accidentally killed with a **-9** flag, a slip lock might remain in `/etc/locks`. Delete this lock file to clean up after **slattach**.

Removing a TTY

To remove a tty, you can use the Web-based System Manager, **wsm**, or the System Management Interface Tool (SMIT) fast path, **smit rminet**.

Asynchronous Point-to-Point Protocol (PPP) Subsystem

The Asynchronous Point-to-Point Protocol (PPP) subsystem provides an alternative to SLIP. PPP provides a standard method for transporting multiprotocol datagrams over point-to-point media. PPP is comprised of three main layers:

1. A method for encapsulating multiprotocol datagrams. PPP supports the TCP/IP network layer protocols.
2. A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection. PPP implements this through streams kernel extensions.
3. A family of Network Control Protocols (NCPs) for establishing and configuring different network layer protocols. PPP supports Internet Protocol Control Protocol (IPCP) for negotiating a TCP/IP connection.

This implementation of PPP supports the following Request for Comments (RFCs):

- RFC 1661, *The Point-to-Point Protocol, LCP*
- RFC 1332, *The PPP Internet Protocol Control Protocol (IPCP)*
- RFC 1662, *PPP in HDLC-like Framing*
- RFC 1334, *PPP Authentication Protocols*
- RFC 1990, *PPP Multilink*

PPP differentiates between client and server. This operating system can act as both a client and a server. The distinction is made to simplify configuration. PPP servers tend to allocate a pool of IP addresses among the connections that are being made. There is some correlation between the media devices. This implementation of PPP breaks this correlation. All server PPP connections are allocated on a first-available basis. This facilitates the separation of PPP from the media. The attachment process must request to be linked to the proper type of link.

User-Level Processes

The Asynchronous Point-to-Point Protocol on this operating system utilizes three user-level processes:

1. A control daemon (**pppcontrold**) run by root under the System Resource Controller (**startsrc -s pppcontrold**). The control daemon's function encompasses loading and configuring all kernel extensions associated with the subsystem. It remains running as long as PPP function is required by the operating system.
2. An attachment process (**pppattachd**) that binds a TTY stream to an instance of the Link Control Protocol, Network Control Protocol, and a datagram protocol. An instance of **pppattachd** exists for each active PPP connection in the system. Any user of the attachment process must belong to the **uucp** group and contain **/usr/sbin** within their **PATH** environment variable.
3. A dialer process (**pppdial**) that establishes an outgoing connection. The dialer is intended to be executed by **pppattachd** as the connector program. Its purpose is to interact over the asynchronous device prior to PPP negotiation. This interaction is defined similarly to the UUCP chat dialog format. The dialer capability is provided to assist in establishing a connection with a remote system. The actual session establishment is out of the scope of PPP.

Configuring the Asynchronous Point-to-Point Protocol

You can use Web-based System Manager or SMIT to configure the Asynchronous Point-to-Point Protocol. The following table shows all tasks that you may need when configuring your system. You must have root privileges to perform the tasks in this table.

At a minimum, when you initially configure your system, you must choose the following tasks from the table:

- Add a Link Configuration
- Add a Server Interface (if you are setting up the machine as a PPP server)
- Add a Demand Interface (if you want the machine to support demand connections)
- Manipulate PAP or CHAP Users/Passwords (if you want the machine to support PPP authentication)
- Start PPP to effect your changes (or Stop then Start PPP, if PPP is currently running)

Configuring the Asynchronous PPP Tasks		
Task	SMIT Fast Path	Web-based System Manager Management Environment
Create Link Control Configuration	smit ppplcp	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Configure the Point-to-Point Link.
Add a Link Configuration	smit addlcp	
Change/Show a Link Configuration	smit chglcp	
Remove a Link Configuration ¹	smit rmlcp	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Configure the Point-to-Point Link → Link Configuration → Remove Link Configuration.
Create PPP IP Interfaces	smit pppip	
Add a Server Interface	smit addpppserver	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Configure the Point-to-Point Link → Server Interfaces → Add/Change Interface.

Configuring the Asynchronous PPP Tasks		
Change/Show a Server Interface	smit listserver	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Configure the Point-to-Point Link → Server Interfaces → Add/Change Interface.
Remove a Server Interface ¹	smit rmlistserver	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Configure the Point-to-Point Link → Server Interfaces → Delete Interface.
Add a Demand Interface	smit addpppdemand	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Configure the Point-to-Point Link → Demand Interfaces → Add/Change Interface.
Change/Show a Demand Interface	smit listdemand	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Configure the Point-to-Point Link → Demand Interfaces → Add/Change Interface.
Remove a Demand Interface ¹	smit rmlistdemand	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Configure the Point-to-Point Link → Demand Interfaces → Delete Interface.
Manipulate PAP users/passwords	smit ppppap	
Add a PAP User	smit addpapuser	
Change/Show a PAP User	smit listpapuser	
Remove a PAP User	smit rmpapuser	
Manipulate CHAP users/passwords	smit pppchap	
Add a CHAP User	smit addchapuser	
Change/Show a CHAP User	smit listchapuser	
Remove a CHAP User	smit rmchapuser	
Start PPP ²	smit startppp	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Start the PPP Subsystem.
Stop PPP ³	smit stopppp	Software → Network → TCPIP (IPv4 and IPv6) → Point-to Point (PPP) → Stop the PPP Subsystem.

Notes:

1. Selecting this task destroys the existing information.
2. An alternative way to start PPP is to issue the **startsrc -s pppcontrold** command. However, the SMIT interface also allows you to set PPP to start at boot time.

3. An alternative way to stop PPP is to issue the **stopsrc -s pppcontrold** command. However, the SMIT interface also allows you to have PPP not start at boot time.

PPP and SNMP

PPP can interact with the TCP/IP SNMP daemon to report PPP link layer configuration information as well as information about active Link Control Protocol (LCP) interfaces. Providing that both the TCP/IP SNMP and the SNMP management software are configured correctly, PPP SNMP enables:

- retrieval of PPP Link Configuration information (Maximum Receive Unit size, Asynchronous Character Mapping, etc.)
- setting of PPP Link Configuration information
- retrieval of LCP interface information for active LCP links
- changing of the state of active LCP links can be changed to "down" by setting the appropriate **ifAdminStatus** Management Information Base (MIB) object

Not all objects defined by RFC1471 for the PPP MIB are supported. Only the **pppLink** table applies to the PPP subsystem, thus the **pppLqr** and **pppTests** portions are not supported. The **pppLink** portion is supported with the following exceptions:

- The **pppLinkConfigMagicNumber** object is read only. In PPP, magic number negotiation is always performed and cannot be disabled.
- The **pppLinkConfigFcsSize** object is read only. PPP only supports FCS sizes of 16 with this operating system.

Enabling PPP SNMP

By default, SNMP for PPP is disabled. To enable PPP SNMP, you can use the Web-based System Manager, **wsm**, or use the following procedure. You must have root privileges to perform this procedure.

Note: The following procedure assumes that PPP Link Configuration is already set. If not, perform the procedure in "Configuring the Asynchronous Point-to-Point Protocol" before enabling PPP SNMP.

1. Start the SMIT Interface and display the Change/Show a Link Configuration screen by entering:

```
smit chglcp
```
2. Toggle the Enable PPP SNMP subagent field to yes.
3. Accept your changes and exit SMIT.

PPP SNMP is not enabled until PPP is restarted.

- If PPP is currently running,
 1. Stop PPP using the **smit stopppp** fast path (see the table in Configuring the Asynchronous Point-to-Point Protocol).
 2. Periodically check to see if the subsystem has completed shutdown by entering:

```
lssrc -s pppcontrold
```

The amount of time it takes to completely stop the subsystem is dependent on the number of links defined in the PPP configuration. The subsystem is completely shut down when the output of this command shows a status of inoperative.

3. Start PPP using the **smit startppp** fast path (see the table in Configuring the Asynchronous Point-to-Point Protocol).
- If PPP is not currently running, start PPP using the **smit startppp** fast path (see the table in Configuring the Asynchronous Point-to-Point Protocol).

TCP/IP Quality of Service (QoS)

Quality of Service (QoS) is a family of evolving Internet standards that provides ways to give preferential treatment to certain types of IP traffic. With the proper support for QoS along a route, this can ameliorate the effects of variable queueing delays and congestion that contribute to poor network performance. The operating system provides host support for QoS to classify outbound traffic into distinct classes of service and to announce and establish resource reservations as requested by client applications.

QoS can be used by an institution to deploy and enforce network policies governing the use of network bandwidth. With QoS, a host can:

- Regulate the amount of traffic of a certain type injected into the network;
- Mark selected packets according to some policy so that subsequent routers can deliver the indicated service;
- Support services such as the virtual leased line service with proper QoS support along the route; and
- Participate in the resource reservation requests from receivers and announce sender sessions available for resource reservation requests.

The QoS support provides the following functions:

- Differentiated services as defined in RFC 2474
- Traffic policing
- In-profile and out-of-profile packet marking
- Traffic shaping
- Metering
- Integrated services for client and server applications as defined in RFC 1633
- RSVP signaling (RFC 2205)
- Guaranteed service (RFC 2212)
- Controlled-Load service (RFC 2211)
- Policy-based networking
- RAPI shared library for application

The QoS subsystem consists of four components:

QoS kernel extension (/usr/lib/drivers/qos)

The QoS kernel extension resides in **/usr/lib/drivers/qos** and is loaded and unloaded using the **cfgqos** and **ucfgqos** configuration methods. This kernel extension enables QoS support.

Policy agent (/usr/sbin/policyd)

The policy agent is a user-level daemon that resides in **/usr/sbin/policyd**. It provides support for policy management and interfaces with the QoS kernel extension to install, modify, and delete policy rules. Policy rules can be defined in the local configuration file (**/etc/policyd.conf**), retrieved from a central network policy server using LDAP, or both.

RSVP agent (/usr/sbin/rsvpd)

The RSVP agent is a user-level daemon that resides in **/usr/sbin/rsvpd**. It implements the RSVP signaling protocol semantics.

RAPI shared library (/usr/lib/librapi.a)

Applications can use the RSVP API (RAPI) to request enhanced quality of service as defined by the Integrated Services Internet QoS model. This library interacts with the local RSVP agent to propagate the QoS request along the path of the data flow using the RSVP protocol. This API is an open standard.

Note: This implementation of QoS is based on a set of evolving Internet standards and draft standards currently under development by the Internet Engineering Task Force (IETF) and its various

working groups. This technology will become more consistent and well defined as these standardization efforts progress within the IETF. It is also important to note that QoS is an emerging Internet technology that is just beginning to be deployed within the Internet. There are many benefits of QoS at all stages of deployment. However, true end-to-end services can only be realized when QoS support exists all along a particular route.

QoS Models

The QoS models for the Internet are open standards defined by the IETF. There are two Internet QoS models currently being standardized within the IETF: *integrated services* and *differentiated services*. These two Internet QoS models augment the traditional best-effort service model described in RFC 1812.

Integrated Services

Integrated Services (IS) is a dynamic resource reservation model for the Internet described in RFC 1633. Hosts use a signaling protocol called Resource ReSerVation Protocol (RSVP) to dynamically request a specific quality of service from the network. QoS parameters are carried in these RSVP messages and each network node along the path installs the parameters to obtain the requested quality of service. These QoS parameters describe one of two currently defined services, guaranteed service and controlled-load service. An important characteristic of IS is that this signaling is done for each traffic flow and reservations are installed at each hop along the route. Although this model is well-suited for meeting the dynamically changing needs of applications, there exist some significant scaling issues that imply it cannot be deployed in a network in which single routers handle many simultaneous flows.

Differentiated Services

Differentiated Services (DS) removes the per-flow and per-hop scalability issues, replacing them with a simplified mechanism of classifying packets. Rather than a dynamic signaling approach, DS uses bits in the IP type of service (TOS) byte to separate packets into classes. The particular bit pattern in the IP TOS byte is called the DS codepoint and is used by routers to define the quality of service delivered at that particular hop, in much the same way routers do IP forwarding using routing table lookups. The treatment given to a packet with a particular DS codepoint is called a per-hop behavior (PHB) and is administered independently at each network node. When the effects of these individual, independent PHBs are concatenated, this results in an end-to-end service.

Differentiated services is being standardized by an IETF working group, which has defined three PHBs: the Expedited Forwarding (EF) PHB, the Assured Forwarding (AF) PHB group, and the Default (DE) PHB. The EF PHB can be used to implement a low latency, low jitter, low loss, end-to-end service such as a virtual leased line (VLL). AF is a family of PHBs, called a PHB group, that is used to classify packets into various drop precedence levels. The drop precedence assigned to a packet determines the relative importance of a packet within the AF class. It can be used to implement the so-called *Olympic* service, which consists of three classes: bronze, silver, and gold. The DE PHB is the traditional best-effort service model as standardized in RFC 1812.

Supported Standards and Draft Standards

The following RFCs and Internet drafts describe the standards on which this QoS implementation is based.

RFC 2474	Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers
RFC 2475	An Architecture for Differentiated Services
RFC 1633	Integrated Services in the Internet Architecture: an Overview
RFC 2205	Resource ReSerVation Protocol (RSVP)
RFC 2210	The Use of RSVP with IETF Integrated Services
RFC 2211	Specification of the Controlled-Load Network Element Service
RFC 2212	Specification of Guaranteed Quality of Service
RFC 2215	General Characterization Parameters for Integrated Service Network Elements

draft-ietf-diffserv-framework-01.txt, October 1998	A Framework for Differentiated Services
draft-ietf-diffserv-rsvp-01.txt, November 1998	A Framework for Use of RSVP with DIFF-serv Networks
draft-ietf-diffserv-phb-ef-01.txt	An Expedited Forwarding PHB
draft-ietf-diffserv-af-04.txt	Assured Forwarding PHB Group
draft-ietf-diffserv-qos-schema-00.txt, October 1998	Schema for Differentiated Services and Integrated Services in Networks
draft-ietf-rap-framework-01.txt, November 1998	A Framework for Policy-based Admission Control[25]
draft-ietf-rap-rsvp-ext-01.txt, November 1998	RSVP Extensions for Policy Control

Note: QoS is an emerging Internet technology that is just beginning to be deployed within the Internet. There are many benefits of QoS at all stages of deployment. However, true end-to-end services can only be realized when QoS support exists all along a particular route.

QoS Installation

QoS is packaged with **bos.net.tcp.server**. This fileset must be installed in order to use QoS. To use the **RAPI** shared library, **bos.adt.include** must also be installed.

QoS Configuration

Stopping and Starting the QoS Subsystem

QoS can be started or stopped through SMIT with the **smit qos** fast path or with the **mkqos** and **rmqos** commands.

To disable the QoS subsystem now and on the next system restart:

```
/usr/sbin/rmqos -B
```

To enable the QoS subsystem now only:

```
/usr/sbin/mkqos -N
```

See the command descriptions for **mkqos** and **rmqos** for the startup and removal command flags.

The **policyd** and **rsvpd** daemons are configured through the **/etc/policyd.conf** and **/etc/rsvpd.conf** configuration files, respectively. These configuration files *must* be edited to customize the QoS subsystem to the local environment. QoS does not work correctly with the supplied example configurations.

Configuring the RSVP agent

The RSVP agent is required if the host is to support the RSVP protocol. The **/etc/rsvpd.conf** configuration file is used to configure the RSVP agent. The syntax of the configuration file is described in the sample configuration file installed in **/etc/rsvpd.conf**.

Example Configuration:

```
interface 1.2.3.1
interface 1.2.3.2 disabled
interface 1.2.3.3 disabled
interface 1.2.3.4
{
  trafficControl
}

rsvp 1.2.3.1
{
  maxFlows 64
```



```

}

rsvp 1.2.3.4
{
    maxFlows 100
}

```

The example illustrates a possible RSVP configuration in which the host has 4 interfaces (virtual or physical) given by the 4 IP addresses, 1.2.3.1, 1.2.3.2, 1.2.3.3, and 1.2.3.4.

Interface 1.2.3.1 has been enabled for RSVP. However, traffic control has not been specified and incoming RSVP RESV messages do not cause resource reservation within the TCP subsystem. This interface can support a maximum of 64 simultaneous RSVP sessions.

Interfaces 1.2.3.2 and 1.2.3.3 have been disabled. The RSVP agent cannot use this interface to transmit or receive RSVP messages.

Interface 1.2.3.4 has been enabled for RSVP. In addition, it can install resource reservations into the TCP subsystem in response to an RSVP RESV message. This interface can support up to 100 RSVP sessions.

Any other interfaces present on the host but not mentioned explicitly in **/etc/rsvpd.conf** are disabled.

Configuring the Policy Agent

The policy agent is a required component of the QoS subsystem. The **/etc/policyd.conf** configuration file is used to configure the policy agent. The syntax of this configuration file is described in the sample configuration file installed in **/etc/policyd.conf**.

The policy agent can be configured by editing **/etc/policyd.conf**. Additionally, the following commands are provided to assist in configuring policies:

- **qosadd**
- **qosmod**
- **qoslist**
- **qosremove**

Example Configurations: In the following example, a premium service category is created and used in the **tcptraffic** policy rule. This service category has a maximum rate of 110000 Kbps, a token bucket depth of 10000 bits, and an outgoing IP TOS value of 11100000 in binary. The **tcptraffic** policy rule gives this premium service to all traffic with source IP address given by 1.2.3.6, destination address 1.2.3.3, and destination port in the range 0 to 1024.

```

ServiceCategories  premium
{
    PolicyScope      DataTraffic
    MaxRate          110000
    MaxTokenBucket   10000
    OutgoingTOS      11100000
}

ServicePolicyRules  tcptraffic
{
    PolicyScope      DataTraffic
    ProtocolNumber 6 # tcp
    SourceAddressRange  1.2.3.6-1.2.3.6
    DestinationAddressRange 1.2.3.3-1.2.3.3
    DestinationPortRange 0-1024
    ServiceReference   premium
}

```

The following statements set up a default service category and use it to restrict the UDP traffic flowing from interfaces 1.2.3.1 through 1.2.3.4 to IP addresses 1.2.3.6 through 1.2.3.10, port 8000.


```

ServiceCategories default
{
  MaxRate          110000
  MaxTokenBucket   10000
  OutgoingTOS      00000000
}

ServicePolicyRules udptraffic
{
  ProtocolNumber 17 # udp
  SourceAddressRange 1.2.3.1-1.2.3.4
  DestinationAddressRange 1.2.3.6-1.2.3.10
  DestinationPortRange 8000-8000
  ServiceReference default
}

```

The following example configuration can be used to download rules from an LDAP server using the distinguished subtree name, to lookup the policies on the LDAP server host.

```

ReadFromDirectory
{
  LDAP_Server      1.2.3.27
  Base              ou=NetworkPolicies,o=myhost.mydomain.com,c=us
}

```

QoS Problem Determination

The **qosstat** command may be used to display status information about the installed and active policies in the QoS subsystem. This information may be useful to you in determining where a problem exists if you are troubleshooting your QoS configuration. **qosstat** can be used to generate the following report.

Action:

```

Token bucket rate (B/sec): 10240
Token bucket depth (B): 1024
Peak rate (B/sec): 10240
Min policed unit (B): 20
Max packet size (B): 1452
Type: IS-CL
Flags: 0x00001001 (POLICE,SHAPE)

```

Statistics:

```

Compliant packets: 1423 (440538 bytes)

```

Conditions:

Source address	Dest address	Protocol	
192.168.127.39:8000	192.168.256.29:35049	tcp	(1 connection)

Action:

```

Token bucket rate (B/sec): 10240
Token bucket depth (B): 1024
Peak rate (B/sec): 10240
Outgoing TOS (compliant): 0xc0
Outgoing TOS (non-compliant): 0x00
Flags: 0x00001011 (POLICE,MARK)
Type: DS

```

Statistics:

```

Compliant packets: 335172 (20721355 bytes)
Non-compliant packets: 5629 (187719 bytes)

```

Conditions:

Source address	Dest address	Protocol	
192.168.127.39:80	:::	tcp	(1 connection)
192.168.127.40:80	:::	tcp	(5 connections)

Policy Specification

This section describes the object classes and attributes used by the policy agent to specify policies for quality of service (QoS) on outgoing traffic. The object classes and attributes are defined, followed by guidelines to enable marking, policing, and shaping.

These conventions are used in the explanations that follow

```
p : choose one in the allowed parameter set
B : integer value of a byte (i.e., 0 =< B =< 255)
b : bit string starting with left most bit (e.g., 101 is
    equivalent 10100000 in a byte field)
i : integer value
s : a character string
a : IP address format B.B.B.B
(R) : Required parameter
(O) : Optional parameter
```

ReadFromDirectory

This statement specifies parameters for establishing an LDAP session. The ReadFromDirectory statement is used in the **/etc/policyd.conf** file to establish the LDAP session.

```
ReadFromDirectory
{
  LDAP_Server    a    # IP address of directory server running LDAP
  LDAP_Port      i    # Port number LDAP server is listening to
  Base           s    # Distinguished Name for LDAP usage
  LDAP_SelectedTag s # Tag to match SelectorTag in object classes
}
```

where

```
LDAP_Server (R): IP address of LDAP server
LDAP_Port (O): Unique port number, default port is 389
Base (R): Example is o=ibm, c=us where o is your organization and c is country
LDAP_SelectedTag (R): Unique string matching SelectorTag attribute in the object class
```

ServiceCategories

This statement specifies the type of service that a flow of IP packets (for example, from a TCP connection or UDP data) should receive end-to-end as they traverse the network. ServiceCategories can be repeated with each having a different name so that they can be referred to later. A ServiceCategories object requires ServicePolicyRules to complete the policy definition.

```
ServiceCategories s
{
  SelectorTag    s    # Required tag for LDAP Search
  MaxRate        i    # Target rate for traffic in this service class
  MaxTokenBucket i    # The bucket depth
  OutgoingTOS    b    # TOS value of outbound traffic for this service class
  FlowServiceType p # Type of traffic
}
```

where

```
s (R) : is the name of this service category
SelectorTag (R) : Required only for LDAP to Search object classes
MaxRate (O) : in Kbps (K bits per second), default is 0
MaxTokenBucket(O) : in Kb, default is system defined maximum
OutgoingTOS (O) : default is 0
FlowServiceType (O): ControlledLoad | Guaranteed, default is ControlledLoad
```

ServicePolicyRules

This statement specifies characteristics of IP packets that are used to match to a corresponding service category. In other words, it defines a set of IP datagrams that should receive a particular service. ServicePolicyRules are associated with ServiceCategories through the ServiceReference attribute. If two rules refer to the same ServiceCategory, each rule is associated with a unique instance of the ServiceCategory.

```
ServicePolicyRules s
{
  SelectorTag      s # Required tag for LDAP Search
  ProtocolNumber  i # Transport protocol id for the policy rule
  SourceAddressRange a1-a2
  DestinationAddressRange a1-a2
  SourcePortRange i1-i2
  DestinationPortRange i1-i2
  PolicyRulePriority i # Highest value is enforced first
  ServiceReference s # Service category name which for this policy rule
}
```

where

```
s (R): is the name of this policy rule
SelectorTag (R): required only for LDAP to Search object class
ProtocolNumber (R): default is 0 which causes no match, must explicitly specify
SourceAddressRange (0): from a1 to a2 where a2 >= a1, default is 0, any source address
SourcePortRange (0): from i1 to i2 where i2 >= i1, default is 0, any source port
DestinationAddressRange (0): same as SourceAddressRange
DestinationPortRange (0): same as SourcePortRange
PolicyRulePriority (0): Important to specify when overlapping policies exist
ServiceReference (R): service category this rule uses
```

Guidelines for DiffServ Environments

The following are guidelines to specify policies for marking, shaping, and/or policing in a DiffServ environment.

1. Marking Only

```
OutgoingTOS : Desired Type Of Service
FlowServiceType : ControlledLoad
MaxRate : Take default of 0
```

2. Shaping Only

```
OutgoingTOS : Take default of 0
FlowServiceType : Guaranteed
MaxRate : Target rate desired for traffic as a positive integer
```

3. Marking and Policing (See Note)

```
OutgoingTOS : Desired Type of Service
FlowServiceType : ControlledLoad
MaxRate : Target rate desired for traffic as a positive integer
```

4. Marking and Shaping

```
OutgoingTOS : Desired Type of Service
FlowServiceType : Guaranteed
MaxRate : Target rate desired for traffic as a positive integer
```

Note: The type of service set for the out of profile packets is set to zero in the case of policing.

Sample policyd Configuration File

The following is a complete example of the `/etc/policyd.conf` configuration file.

policyd Configuration File

```
#loglevel 511 # Verbose logging
```

```
#####
```

```

#
# Mark rsh traffic on TCP source ports 513 and 514.
ServiceCategories      tcp_513_514_svc
{
    MaxRate              0                # Mark only
    OutgoingTOS          00011100        # binary
    FlowServiceType      ControlledLoad
}

ServicePolicyRules     tcp_513_514flt
{
    ProtocolNumber       6 # TCP
    SourceAddressRange   0.0.0.0-0.0.0.0 # Any IP src addr
    DestinationAddressRange 0.0.0.0-0.0.0.0 # Any IP dst addr
    SourcePortRange      513-514
    DestinationPortRange 0-0             # Any dst port
    ServiceReference      tcp_513_514_svc
}
#
#####
#
# Shape connected UDP traffic on source port 9000.
ServiceCategories      udp_9000_svc
{
    MaxRate              8192            # kilobits
    MaxTokenBucket       64              # kilobits
    FlowServiceType      Guaranteed
}

ServicePolicyRules     udp_9000flt
{
    ProtocolNumber       17 # UDP
    SourceAddressRange   0.0.0.0-0.0.0.0 # Any IP src addr
    DestinationAddressRange 0.0.0.0-0.0.0.0 # Any IP dst addr
    SourcePortRange      9000-9000
    DestinationPortRange 0-0             # Any dst port
    ServiceReference      udp_9000_svc
}
#
#####
#
# Mark and police finger traffic on TCP source port 79.
ServiceCategories      tcp_79_svc
{
    MaxRate              8                # kilobits
    MaxTokenBucket       32              # kilobits
    OutgoingTOS          00011100        # binary
    FlowServiceType      ControlledLoad
}

ServicePolicyRules     tcp_79flt
{
    ProtocolNumber       6 # TCP
    SourceAddressRange   0.0.0.0-0.0.0.0 # Any IP src addr
    DestinationAddressRange 0.0.0.0-0.0.0.0 # Any IP dst addr
    SourcePortRange      79-79
    DestinationPortRange 0-0             # Any dst port
    ServiceReference      tcp_79_svc
}
#
#####
#
# Mark and shape ftp-data traffic on TCP source port 20.
ServiceCategories      tcp_20_svc
{
    MaxRate              81920           # kilobits
    MaxTokenBucket       128            # kilobits
}

```

```

        OutgoingTOS          00011101    # binary
        FlowServiceType     Guaranteed
    }

ServicePolicyRules      tcp_20_flt
{
    ProtocolNumber         6 # TCP
    SourceAddressRange     0.0.0.0-0.0.0.0 # Any IP src addr
    DestinationAddressRange 0.0.0.0-0.0.0.0 # Any IP dst addr
    SourcePortRange        20-20
    DestinationPortRange   0-0          # Any dst port
    ServiceReference       tcp_20_svc
}
#
#####
#
# LDAP server entry.
#ReadFromDirectory
#{
#   LDAP_Server           9.3.33.138 # IP address of LDAP server
#   Base                  o=ibm,c=us # Base distinguished name
#   LDAP_SelectedTag      myhost     # Typically client hostname
#}
#
#####

```

Loading Policies into IBM SecureWay Directory Server

If the policy daemon is used with the IBM SecureWay Directory LDAP Server, use the following schema as a guide to update `/etc/ldapschema/V3.modifiedschema` before starting the LDAP server. Refer to the LDAP server documentation for details.

LDAP Schema

```

objectClasses {
( ServiceCategories-OID NAME 'ServiceCategories' SUP top MUST
( objectClass $ SelectorTag $ serviceName ) MAY
( description $ FlowServiceType $ MaxRate $ MaxTokenBucket $ OutgoingTos ) )
( ServicePolicyRules-OID NAME 'ServicePolicyRules' SUP top MUST
( objectClass $ PolicyName $ SelectorTag ) MAY
( description $ DestinationAddressRange $ DestinationPortRange $
ProtocolNumber $ ServiceReference $ SourceAddressRange $ SourcePortRange ) )
}
attributeTypes {
( DestinationAddressRange-OID NAME 'DestinationAddressRange' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( DestinationPortRange-OID NAME 'DestinationPortRange' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( FlowServiceType-OID NAME 'FlowServiceType'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( MaxRate-OID NAME 'MaxRate' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( MaxTokenBucket-OID NAME 'MaxTokenBucket' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( OutgoingTos-OID NAME 'OutgoingTos' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( PolicyName-OID NAME 'PolicyName' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( ProtocolNumber-OID NAME 'ProtocolNumber' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( SelectorTag-OID NAME 'SelectorTag' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( ServiceReference-OID NAME 'ServiceReference' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( SourceAddressRange-OID NAME 'SourceAddressRange' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( SourcePortRange-OID NAME 'SourcePortRange' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
}

IBMattributeTypes {
( DestinationAddressRange-OID DBNAME ( 'DestinationAddressRange' 'DestinationAddressRange' ) )
( DestinationPortRange-OID DBNAME ( 'DestinationPortRange' 'DestinationPortRange' ) )
( FlowServiceType-OID DBNAME ( 'FlowServiceType' 'FlowServiceType' ) )
}

```

```

( MaxRate-OID DBNAME ( 'MaxRate' 'MaxRate' ) )
( MaxTokenBucket-OID DBNAME ( 'MaxTokenBucket' 'MaxTokenBucket' ) )
( OutgoingTos-OID DBNAME ( 'OutgoingTos' 'OutgoingTos' ) )
( PolicyName-OID DBNAME ( 'PolicyName' 'PolicyName' ) )
( ProtocolNumber-OID DBNAME ( 'ProtocolNumber' 'ProtocolNumber' ) )
( SelectorTag-OID DBNAME ( 'SelectorTag' 'SelectorTag' ) )
( ServiceReference-OID DBNAME ( 'ServiceReference' 'ServiceReference' ) )
( SourceAddressRange-OID DBNAME ( 'SourceAddressRange' 'SourceAddressRange' ) )
( SourcePortRange-OID DBNAME ( 'SourcePortRange' 'SourcePortRange' ) )
}

ldapSyntaxes {
}

matchingRules {
}

```

System Configuration

Overlapping Policies

Policies that overlap are installed in the QoS Manager in a nondeterministic ordering. In the case of overlapping policies the `PolicyRulePriority` attribute of the `ServicePolicyRules` should be specified to determine the ordering of enforcement of policies. The `PolicyRulePriority` attribute takes an integer as a parameter and, in the case of overlapping policies, the rule with the highest integer value is enforced.

UDP usage

Only connected UDP sockets are supported for QoS.

Policy Conflicts with RSVP Reservations

The policy and RSVP agents are mutually independent. Thus, care must be taken not to specify a policy that conflicts with, or is covered by, an existing RSVP reservation. In the presence of such conflicts, the system accepts the first policy or reservation while flagging a violation for the others.

Token Bucket Depth Specification

For correct operation, the `MaxTokenBucket` attribute must be set to at least the maximum MTU of all interfaces configured in the system.

Policy Modification

Policy modifications are handled by the policy agent by automatically deleting the existing policies and installing the new ones. This may result in a short, temporary window of time during which the corresponding traffic receives default (typically best effort) service.

Standards Compliance

This release is compatible with evolving Internet Engineering Task Force (IETF) standards for Differentiated (DiffServ) and Integrated Services (IntServ) on the Internet.

IntServ Model

The following RFCs describe various components of the IntServ model:

- The Use of RSVP with IETF Integrated Services (RFC 2210)
- Specification of the Controlled-Load Network Element Service (RFC 2211)
- Specification of Guaranteed Quality of Service (RFC 2212)

DiffServ Model

The following RFCs describe various components of the DiffServ model:

- Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (RFC 2474)
- An Architecture for Differentiated Services (RFC 2475)

The following RFC outlines the current usage of the IP TOS octet:

- Type of Service in the Internet Protocol Suite (RFC 1349)

The following RFCs outline future practices governing usage of the IP TOS octet:

- Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (RFC 2474)
- Assured Forwarding PHB Group (RFC 2597)
- An Expedited Forwarding PHB (RFC 2598)

IPv6 Support

QoS for AIX 5.1 only supports IPv4; IPv6 is not supported.

Controlling the Policy Daemon

The policy daemon can be controlled using the system resource controller (SRC). For example, the command:

```
startsrc -s policyd -a "-i 60"
```

starts the policy agent with a refresh interval of 60 seconds.

The **refresh** SRC command is not currently supported.

QoS Reference

For important updates to this documentation, consult the README file in `/usr/samples/tcpip/qos`.

Commands

- **qosadd**
- **qoslist**
- **qosmod**
- **qosremove**
- **qosstat**
- **mkqos**
- **rmqos**

Methods

- **cfgqos**
- **ucfgqos**

TCP/IP Security

For any number of reasons, the person who administers your system might have to meet a certain level of security. For instance, the security level might be a matter of corporate policy. Or a system might need access to U.S. government systems and thus be required to communicate at a certain security level. These security standards might be applied to the network, the operating system, application software, even programs written by the person who administers your system.

This section describes the security features provided with Transmission Control Protocol/Internet Protocol (TCP/IP), both in standard mode and as a secure system, and discusses some security considerations that are appropriate in a network environment.

The topics discussed in this section are:

- Operating System-Specific Security
- TCP/IP-Specific Security

- TCP/IP Command Security
- Trusted Processes
- The Network Trusted Computing Base (NTCB)
- Data Security and Information Protection

Operating System-Specific Security

Many of the security features available for TCP/IP are based on those available through the operating system. The following sections outline TCP/IP security.

Access Control

The security policy for networking is an extension of the security policy for the operating system, and it consists of the following major components:

- User authentication
- Connection authentication
- Data import and export security

User authentication is provided at the remote host by a user name and password, the same as when a user logs in to the local system. Trusted TCP/IP commands, such as **ftp**, **rexec**, and **telnet**, have the same requirements and go through the same verification process as trusted commands in the operating system.

Connection authentication is provided to ensure that the remote host has the expected Internet Protocol (IP) address and name. This prevents a remote host from masquerading as another remote host.

Data import and export security permits data at a specified security level to flow to and from network interface adapters at the same security and authority levels. For example, top secret data can flow only between adapters that are set to the top secret security level.

Auditing

Network auditing is provided by TCP/IP, using the **audit** subsystem to audit both kernel network routines and application programs. The purpose of auditing is to record those actions that affect the security of the system and the user responsible for those actions.

The following types of events are audited:

Kernel Events:

- Change configuration
- Change host ID
- Change route
- Connection
- Create socket
- Export object
- Import object

Application Events:

- Access the network
- Change configuration
- Change host ID
- Change static route

- Configure mail
- Connection
- Export data
- Import data
- Write mail to a file

Creation and deletion of objects are audited by the operating system. Application audit records suspend and resume auditing to avoid redundant auditing by the kernel.

Network Trusted Computing Base (NTCB)

The Network Trusted Computing Base consists of hardware and software for ensuring network security. The hardware security features are provided by the network adapters used with TCP/IP. The software portion of the NTCB contains only trusted processes and their associated files.

Trusted Path, Trusted Shell, and Secure Attention Key (SAK)

The operating system provides the *trusted path* to prevent unauthorized programs from reading data from a user terminal. This path is used when a secure communication path with the system is required, such as when you are changing passwords or logging in to the system. The operating system also provides the *trusted shell* feature (**tsh**), which executes only trusted programs that have been tested and verified as secure. TCP/IP supports both of these features, along with the *secure attention key* (SAK), which establishes the environment necessary for secure communication between you and the system. The local SAK is available whenever you are using TCP/IP. A remote SAK is available through the **telnet** command.

The local SAK has the same function in **telnet** that it has in other operating system application programs: it ends the **telnet** process and all other processes associated with the terminal in which **telnet** was running. Inside the **telnet** program, however, you can send a request for a trusted path to the remote system using the **telnet send sak** command (while in **telnet** command mode). You can also define a single key to initiate the SAK request using the **telnet set sak** command.

TCP/IP-Specific Security

Some portions of security are specific to TCP/IP. These features (TCP/IP commands and TCP/IP trusted processes) work together with the operating system security features discussed to provide the security for TCP/IP.

TCP/IP Command Security

Some commands in TCP/IP provide a secure environment during operation. These commands are **ftp**, **rexec**, and **telnet**. The **ftp** function provides security during file transfer. The **rexec** command provides a secure environment for executing commands on a foreign host. The **telnet** (TELNET) function provides security for login to a foreign host.

These commands provide security during their operation only. That is, they do not set up a secure environment for use with other commands. For securing your system for other operations, use the **securetcpip** command. This command gives you the ability to secure your system by disabling the nontrusted daemons and applications, and by giving you the option of securing your IP layer network protocol as well.

The **ftp**, **rexec**, **securetcpip**, and **telnet** commands provide the following forms of system and data security:

ftp The **ftp** command provides a secure environment for transferring files. When a user invokes the **ftp** command to a foreign host, the user is prompted for a login ID. A default login ID is shown: the user's current login ID on the local host. The user is prompted for a password for the remote host.

The automatic login process searches the local user's **\$HOME/.netrc** file for the user's ID and password to use at the foreign host. For security, the permissions on the **\$HOME/.netrc** file must be set to 600 (read and write by owner only). Otherwise, automatic login fails.

Note: Because use of the **.netrc** file requires storage of passwords in a nonencrypted file, the automatic login feature of the **ftp** command is not available when your system has been configured with the **securetcip** command. This feature can be reenabled by removing the **ftp** command from the `tcip:` stanza in the **/etc/security/config** file.

To use the file transfer function, the **ftp** command requires two TCP/IP connections, one for the File Transfer Protocol (FTP) and one for data transfer. The protocol connection is primary and is secure because it is established on reliable communicating ports. The secondary connection is needed for the actual transfer of data, and both the local and remote host verify that the other end of this connection is established with the same host as the primary connection. If the primary and secondary connections are not established with the same host, the **ftp** command first displays an error message stating that the data connection was not authenticated, and then it exits. This verification of the secondary connection prevents a third host from intercepting data intended for another host.

securetcip The **securetcip** command enables TCP/IP security features. Access to commands that are not trusted is removed from the system when this command is issued. Each of the following commands are removed by running the **securetcip** command:

- **rlogin** and **rlogind**
- **rcp**, **rsh**, and **rshd**
- **tftp** and **tftpd**
- **trpt**

The **securetcip** command is used to convert a system from the standard level of security to a higher security level. Once your system has been converted, you do not need to issue the **securetcip** command again unless you reinstall TCP/IP.

rexec The **rexec** command provides a secure environment for executing commands on a foreign host. The user is prompted for both a login ID and a password.

An automatic login feature causes the **rexec** command to search the local user's **\$HOME/.netrc** file for the user's ID and password on a foreign host. For security, the permissions on the **\$HOME/.netrc** file must be set to 600 (read and write by owner only). Otherwise, automatic login fails.

Note: Because use of the **.netrc** file requires storage of passwords in a nonencrypted file, the automatic login feature of **rexec** is not available when your system is operating in secure. This feature can be reenabled by removing the **rexec** entry from the `tcip:` stanza in the **/etc/security/config** file.

telnet or **tn** The **telnet** (TELNET) command provides a secure environment for login to a foreign host. The user is prompted for both a login ID and a password. The user's terminal is treated just like a terminal connected directly to the host. That is, access to the terminal is controlled by permission bits. Other users (group and other) do not have read access to the terminal, but they can write messages to it if the owner gives them write permission. The **telnet** command also provides access to a trusted shell on the remote system through the secure attention key (SAK). This key sequence differs from the sequence that invokes the local trusted path and can be defined within the **telnet** command.

Remote Command Execution Access (/etc/hosts.equiv)

Users on the hosts listed in the **/etc/hosts.equiv** file can run certain commands on your system without supplying a password.

Remote Command Execution Access Tasks			
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>	<i>Web-based System Manager Management Environment</i>
List Remote Hosts That Have Command Execution Access	smit lshostsequiv	view /etc/hosts.equiv	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Hosts File → Contents of /etc/hosts file.
Add a Remote Host for Command Execution Access	smit mkhostsequiv	*edit /etc/hosts.equiv	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Hosts File . In Add/Change host entry , complete the following fields: IP Address , Host name , Alias(es) , and Comment . Click Add/Change Entry , and click OK .
Remove a Remote Host from Command Execution Access	smit rmhostsequiv	*edit /etc/hosts.equiv	Software → Network → TCPIP (IPv4 and IPv6) → TCPIP Protocol Configuration → TCP/IP → Configure TCP/IP → Advanced Methods → Hosts File . Select a host in Contents of /etc/host file . Click Delete Entry → OK .

For more information about file procedures preceded by an asterisk (*), see the "hosts.equiv File Format for TCP/IP" in the *AIX 5L Version 5.1 Files Reference*.

Restricted File Transfer Program Users (/etc/ftpusers)

Users listed in the **/etc/ftpusers** file are protected from remote FTP access. For example, suppose user A is logged into a remote system, and he knows the password of user B on your system. If B is listed in **/etc/ftpusers**, A cannot FTP files to or from B's account, even though A knows B's password.

Remote FTP Users Tasks			
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>	<i>Web-based System Manager Management Environment</i>
List Restricted FTP Users	smit lsftpusers	view /etc/ftpusers	Software → Users → All Users .
Add a Restricted User	smit mkftpusers	*edit /etc/ftpusers	Software → Users → All Users → Selected → Add this User to Group . Select a group, and click OK .

Remote FTP Users Tasks			
Remove a Restricted User	smit rmftusers	*edit /etc/ftusers	Software → Users → All Users → Selected → Delete .

For more information about file procedures preceded by an asterisk (*), see the "ftusers File Format for TCP/IP" in the *AIX 5L Version 5.1 Files Reference*.

Trusted Processes

A trusted program, or trusted process, is a shell script, a daemon, or a program that meets a particular standard of security. These security standards are set and maintained by the U.S. Department of Defense, which also certifies some trusted programs.

Trusted programs are trusted at different levels. Security levels include A1, B1, B2, B3, C1, C2, and D, with level A1 providing the highest security level. Each security level must meet certain requirements. For example, the C2 level of security incorporates the following standards:

program integrity	Ensures that the process does what it is supposed to do, no more and no less.
modularity	Means that the process source code is broken down into modules that cannot be directly affected or accessed by other modules.
principle of least privilege	States that at all times a user is operating at the lowest level of privilege authorized. That is, if a user has access only to view a certain file, then the user does not inadvertently also have access to alter that file.
limitation of object reuse	Keeps a user from, for example, accidentally stumbling across a section of memory that has been flagged for overwriting but not yet cleared, and might contain sensitive material.

TCP/IP contains several trusted daemons and many nontrusted daemons. The trusted daemons have been tested to ensure that they operate within particular security standards.

Examples of trusted daemons are:

- **ftpd**
- **rexecd**
- **telnetd**

Examples of nontrusted daemons are:

- **rshd**
- **rlogind**
- **tftpd**

For a system to be trusted, it must operate with a trusted computing base. This means, for a single host, that the machine must be secure. For a network, this means that all file servers, gateways, and other hosts must be secure.

Network Trusted Computing Base (NTCB)

The network contains both hardware and software mechanisms to implement the networking security features. This section defines the components of the Network Trusted Computing Base as they relate to TCP/IP.

The hardware security features for the network are provided by the network adapters used with TCP/IP. These adapters are programmed to control incoming data by receiving only data destined for the local system and to broadcast data receivable by all systems.

The software component of the NTCB consists of only those programs that are considered trusted. The programs and associated files that are part of a secure system are listed in the following tables on a directory-by-directory basis.

/etc Directory				
Name	Owner	Group	Mode	Permissions
gated.conf	root	system	0664	rw-rw-r—
gateways	root	system	0664	rw-rw-r—
hosts	root	system	0664	rw-rw-r—
hosts.equiv	root	system	0664	rw-rw-r—
inetd.conf	root	system	0644	rw-r—r—
named.conf	root	system	0644	rw-r—r—
named.data	root	system	0664	rw-rw-r—
networks	root	system	0664	rw-rw-r—
protocols	root	system	0644	rw-r—r—
rc.tcpip	root	system	0774	rxwxrwxr—
resolv.conf	root	system	0644	rw-rw-r—
services	root	system	0644	rw-r—r—
3270.keys	root	system	0664	rw-rw-r—
3270keys.rt	root	system	0664	rw-rw-r—

/usr/bin Directory				
Name	Owner	Group	Mode	Permissions
host	root	system	4555	r-sr-xr-x
hostid	bin	bin	0555	r-xr-xr-x
hostname	bin	bin	0555	r-xr-xr-x
finger	root	system	0755	rxwxr-xr-x
ftp	root	system	4555	r-sr-xr-x
netstat	root	bin	4555	r-sr-xr-x
rexec	root	bin	4555	r-sr-xr-x
ruptime	root	system	4555	r-sr-xr-x
rwho	root	system	4555	r-sr-xr-x
talk	bin	bin	0555	r-xr-xr-x
telnet	root	system	4555	r-sr-xr-x

/usr/sbin Directory				
Name	Owner	Group	Mode	Permissions
arp	root	system	4555	r-sr-xr-x
fingerd	root	system	0554	r-xr-xr—
ftpd	root	system	4554	r-sr-xr—

gated	root	system	4554	r-sr-xr—
ifconfig	bin	bin	0555	r-xr-xr-x
inetd	root	system	4554	r-sr-xr—
named	root	system	4554	r-sr-x—
ping	root	system	4555	r-sr-xr-x
rexecd	root	system	4554	r-sr-xr—
route	root	system	4554	r-sr-xr—
routed	root	system	0554	r-xr-x—
rwhod	root	system	4554	r-sr-xr—
securetcpip	root	system	0554	r-xr-xr—
setclock	root	system	4555	r-sr-xr-x
syslogd	root	system	0554	r-xr-xr—
talkd	root	system	4554	r-sr-xr—
telnetd	root	system	4554	r-sr-xr—

/usr/ucb Directory				
Name	Owner	Group	Mode	Permissions
tn	root	system	4555	r-sr-xr-x

/var/spool/rwho Directory				
Name	Owner	Group	Mode	Permissions
rwho (directory)	root	system	0755	drwxr-xr-x

Data Security and Information Protection

The security feature for TCP/IP does not encrypt user data transmitted through the network. Therefore, it is suggested that users identify any risk in communication that could result in the disclosure of passwords and other sensitive information, and based on that risk, apply appropriate countermeasures.

The use of this product in a Department of Defense (DOD) environment might require adherence to DOD 5200.5 and NCSD-11 for communications security.

TCP/IP Problem Determination

This section contains information about diagnosing common problems in a Transmission Control Protocol/Internet Protocol (TCP/IP) network environment.

The **netstat** command is a good tool for determining which area of the network has a problem. Once you have isolated the problem to an area, you can use more sophisticated tools to proceed. For example, you might use the **netstat -i** and **netstat -v** to determine if you have a problem with a particular hardware interface, and then run diagnostics to further isolate the problem. Or, if the **netstat -s** command shows that there are protocol errors, you could then use the **trpt** or **iptrace** commands.

The topics discussed in this section are:

- Communication Problems
- Name Resolution Problems

- Routing Problems
- Problems with System Resource Controller (SRC) Support
- telnet or rlogin Problems
- Configuration Problems
- Common Problems with Network Interfaces
- Problems with Packet Delivery
- Problems with Dynamic Host Configuration Protocol (DHCP)

Communication Problems

If you cannot communicate with a host on your network:

- Try to contact the host, using the **ping** command. Run the **ping** command on the local host to verify that the local interface to the network is up and running.
- Try to resolve the name of the host, using the **host** command. If the name does not resolve, you have a name resolution problem. See "Name Resolution Problems" for more information.

If the name resolves and you are trying to contact a host on another network, you may have a routing problem. See "Routing Problems" for more information.

- If your network is a token-ring network, check to see if the target host is on another ring. If so, the allcast field is probably set incorrectly. Use the Web-based System Manager, **wsm**, or the System Management Interface Tool (SMIT) fast path **smit chinnet** to access the Network Interfaces menu. Then, set the Confine Broadcast to Local Ring field to **no** in the token-ring dialog.
- If there are a large number of Address Resolution Protocol (ARP) packets on your network, verify that your subnet mask is set correctly. This condition is known as a broadcast storm and can affect your system performance.

Name Resolution Problems

Resolver routines on hosts running TCP/IP attempt to resolve names, using the following sources in the order listed:

1. DOMAIN name server (**named**)
2. Network Information Service (NIS)
3. Local **/etc/hosts** file

When NIS+ is installed, lookup preferences are set using the **irs.conf** file. For more information, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

Client Host

If you cannot get a host name resolved, and you are using flat name resolution (using the **/etc/hosts** file), verify that the host name and correct Internet Protocol (IP) address information is in the **/etc/hosts** file.

If you cannot get a host name resolved, and you are using a name server:

1. Verify that you have a **resolv.conf** file specifying the domain name and Internet address of a name server.
2. Verify that the local name server is up by issuing the **ping** command with the IP address of the name server (found in the local **resolv.conf** file).
3. If the local name server is up, verify that the **named** daemon on your local name server is active by issuing the **lssrc -s named** command on the name server.
4. If you are running the **syslogd**, check for logged messages. The output for these messages is defined in the **/etc/syslog.conf** file.

If these steps do not identify the problem, check the name server host.

Name Server Host

If you cannot get a host name resolved:

1. Verify that the **named** daemon is active by issuing the following command:

```
lssrc -s named
```
2. Verify that the address of the target host exists and is correct in the name server database. Send a **SIGINT** signal to the **named** daemon to dump the database and cache to the file **/var/tmp/named_dump.db**. Verify that the address you are trying to resolve is there and is correct. Add or correct name-to-address resolution information in the **named** hosts data file for the master name server of the domain. Then issue the following **SRC** command to reread the data files:

```
refresh -s named
```
3. Verify that the name resolution requests are being processed. To do this, enter the **named** daemon from the command line and specify a debugging level. Valid debug levels are 1 through 9. The higher the level, the more information the debug mechanism logs.

```
startsrc -s named -a "-d DebugLevel"
```
4. Check for configuration problems in the **named** data files. For more information, see "Configuring Name Servers", the "DOMAIN Data File Format," "DOMAIN Reverse Data File Format," "DOMAIN Cache File Format," and the "DOMAIN Local Data File Format" in the *AIX 5L Version 5.1 Files Reference*.

Note: A common error is the incorrect use of the . (period) and the @ (at sign) in the DOMAIN data files.

If external users cannot reach your domains:

- Make sure that all your non-master name servers (slave, hint) have equal time-to-live (TTL) information in the DOMAIN data files.

If external resolvers query your servers constantly:

- Make sure your servers are distributing DOMAIN data files with reasonable TTL values. If the TTL is zero or another small value, the data you transfer times out very quickly. Set the minimum value in your start of authority (SOA) records to a week or more to solve this problem.

Routing Problems

If you cannot reach a destination host, consider the following situations:

- If you receive a Network Unreachable error message, make sure that a route to the gateway host has been defined and is correct. Check this by using the **netstat -r** command to list kernel routing tables.
- If you receive a No route to host error message, verify that the local network interface is up by issuing the **ifconfig interface_name** command. The output indicates whether or not the interface is up. Use the **ping** command to try and reach another host on your network.
- If you receive a Connection timed out error message:
 - Verify that the local gateway is up using the **ping** command with the name or Internet address of the gateway.
 - Make sure that a route to the gateway host has been defined and is correct. Check this by using the **netstat -r** command to list kernel routing tables.
 - Make sure the host you want to communicate with has a routing table entry back to your machine.
- If you are using static routing, make sure that a route to the target host and gateway host has been defined. Check this by using the **netstat -r** command to list kernel routing tables.

Note: Make sure the host you want to communicate with has a routing table entry to your machine.

- If you are using dynamic routing, verify that the gateway is listed and correct in the kernel routing tables by issuing the **netstat -r** command.
- If the gateway host is using the **Routing Information Protocol (RIP)** with the **routed** daemon, make sure that a static route to the target host is set up in the **/etc/gateways** file.

Note: You need to do this only if the routing daemon cannot identify the route to a distant host through queries to other gateways.

- If the gateway host is using the **RIP** with the **gated** daemon, make sure that a static route to the target host is set up in the **gated.conf** file.
- If you are using dynamic routing with the **routed** daemon:
 - If **routed** cannot identify the route through queries (for example, if the target host is not running the **RIP**, check the **/etc/gateways** file to verify that a route to the target host is defined.
 - Make sure that gateways responsible for forwarding packets to the host are up and running the **RIP**. Otherwise, you'll need to define a static route.
 - Run the **routed** daemon using the debug option to log such information as bad packets received. Invoke the daemon from the command line using the following command:


```
startsrc -s routed -a "-d"
```
 - Run the **routed** daemon using the **-t** flag, which causes all packets sent or received to be written to standard output. When **routed** is run in this mode, it remains under the control of the terminal that started it. Therefore, an interrupt from the controlling terminal kills the daemon.
- If you are using dynamic routing with the **gated** daemon:
 - Verify that the **/etc/gated.conf** file is configured correctly and that you are running the correct protocols.
 - Make sure the gateway on the source network is using the same protocol as the gateway on the destination network.
 - Make sure that the machine with which you are trying to communicate has a route back to your host machine.
 - Verify that the gateway names in the **gated.conf** file correspond to the gateway names listed in the **/etc/networks** file.
- If you are using the **RIP** or **HELLO** protocols, and routes to the destination cannot be identified through routing queries, check the **gated.conf** file to verify that a route to the target host is defined. Set static routes under the following conditions:
 - The destination host is not running the same protocol as the source host so cannot exchange routing information.
 - The host must be reached by a distant gateway (a gateway that is on a different autonomous system than the source host). The **RIP** can be used only among hosts on the same autonomous system.

Other Possibilities

If all else fails, you might want to turn on tracing for your routing daemon (either **routed** or **gated**). Use the **SRC traceson** command from the command line, or send a signal to the daemon to specify different levels of tracing. See the **gated** daemon or the **routed** daemon for specifics on sending signals to these daemons.

Problems with SRC Support

- If changes to the **/etc/inetd.conf** file do not take effect:
 - Update the **inetd** daemon by issuing the **refresh -s inetd** command or the **kill -1 InetdPID** command.
- If the **startsrc -s [subsystem name]** returns the following error message:


```
0513-00 The System Resource Controller is not active.
```

The System Resource Controller subsystem has not been activated. Issue the **srcmstr &** command to start SRC, then reissue the **startsrc** command.

You might also want to try starting the daemon from the command line without SRC support.

- If the **refresh -s** [*subsystem name*] or **lssrc -ls** [*subsystem name*] returns the following error message:
[*subsystem name*] does not support this option.

The subsystem does not support the SRC option issued. Check the subsystem documentation to verify options the subsystem supports.

- If the following message is displayed:
SRC was not found, continuing without SRC support.

A daemon was invoked directly from the command line instead of using the **startsrc** command. This is not a problem. However, SRC commands, such as **stopsrc** and **refresh**, will not manipulate a subsystem that is invoked directly.

If the **inetd** daemon is up and running correctly and the appropriate service seems to be correct but you still cannot connect, try running the **inetd** daemon processes through a debugger.

1. Stop the **inetd** daemon temporarily:

```
stopsrc -s inetd
```

The **stopsrc** command stops subsystems like the **inetd** daemon.

2. Edit the **syslog.conf** file to add a debugging line at the bottom. For example:

```
vi /etc/syslog.conf
```

- a. Add the line "`*.debug /tmp/myfile`" at the bottom of the file and exit.
- b. The file that you specify must exist (`/tmp/myfile` in this example). You can use the **touch** command to make your file exists.

3. Refresh the file:

- If you are using SRC, enter:

```
refresh -s syslogd
```
- If you are not using SRC, kill the **syslogd** daemon:

```
kill -1 'ps -e | grep /etc/syslogd | cut -c1-7'
```

4. Start the **inetd** daemon backup with debugging enabled:

```
startsrc -s inetd -a "-d"
```

The **-d** flag enables debugging.

5. Try to make a connection to log errors in the `/tmp/myfile` debugging file. For example:

```
tn bastet
Trying...
connected to bastet
login:>
Connection closed
```

6. See if anything shows up as a problem in the debugging file. For example:

```
tail -f /tmp/myfile
```

telnet or rlogin Problems

The following explanations can be useful in solving problems with the **telnet** or **rlogin** command.

Screen Distortion

If you are having trouble with screen distortion in full-screen applications:

1. Check the **TERM** environment variable by issuing one of the following commands:

```
env
echo $TERM
```

2. Verify that the **TERM** variable is set to a value that matches the type of terminal display you are using.

telnet Debugging

telnet subcommands that can help in debugging problems include:

display	Displays set and toggle values.
toggle	Toggles the display of all network data in hex.
toggle options	Toggles the display of internal telnet process options.

telnetd Daemon Debugging

If the **inetd** daemon could execute the **telnet** service but you still cannot connect using the **telnet** command, there may be something wrong with the **telnet** interface.

1. Verify that **telnet** is using the correct terminal type.
 - a. Check the **\$TERM** variable on your machine:
echo \$TERM
 - b. Log in to the machine to which you are trying to attach and check the **\$TERM** variable:
echo \$TERM
2. Use the **telnet** interface's debugging capabilities by entering the **telnet** command without flags.

```
telnet
tn>
```

- a. Enter open host where *host* is the name of the machine.
 - b. Enter Ctrl-T to get to the tn> prompt.
 - c. At the tn> prompt, enter debug for debugging mode.
3. Try to connect to another machine using the **telnet** interface:

```
telnet bastet
Trying...
Connected to bastet
Escape character is '^T'.
```

Watch the display as the various commands scroll up the screen. For example:

```
SENT do ECHO
SENT do SUPPRESS GO AHEAD
SENT will TERMINAL TYPE (reply)
SENT do SUPPORT SAK
SENT will SUPPORT SAK (reply)
RCVD do TERMINAL TYPE (don't reply)
RCVD will ECHO (don't reply)
RCVD will SUPPRESS GO AHEAD (don't reply)
RCVD wont SUPPORT SAK (reply)
SENT dont SUPPORT SAK (reply)
RCVD do SUPPORT SAK (don't reply)
SENT suboption TELOPT_NAWS Width 80, Height 25
RCVD suboption TELOPT_TTYPE SEND
RCVD suboption TELOPT_TTYPE aixterm
...
```

4. Check **/etc/termcap** or **/usr/lib/terminfo** for the aixterm definition. For example:
ls -a /usr/lib/terminfo
5. If the aixterm definition is missing, add it by building the **ibm.ti** file. For example:
tic ibm.ti

The **tic** command is a terminal information compiler.

Programs Using Extended Curses

Problems with function and arrow keys can arise when using the **rlogin** and **telnet** commands with programs using extended curses. Function and arrow keys generate escape sequences, which are split if too little time is allotted for the entire key sequence. Curses waits a specific amount of time to decide whether an Esc indicates the escape key only or the start of a multibyte escape sequence generated by other keys, such as cursor keys, the action key, and function keys.

If no data, or data that is not valid, follows the Esc in the allotted amount of time, curses decides that the Esc is the escape key, and the key sequence is split. The delay resulting from the **rlogin** or **telnet** command is network dependent. Sometimes arrow and function keys work and sometimes they do not, depending on the speed of the network to which you are connecting. Setting the **ESCDELAY** environment variable to a large value (1000 to 1500) effectively solves this problem.

Configuration Problems

Network interfaces are automatically configured during the first system startup after the adapter card is installed. However, you still need to set some initial values for TCP/IP including the host name, the Internet address, and the subnet mask. To do this, you can use the Web-based System Manager, **wsm**, or you can use the SMIT interface in the following ways:

- Use the **smit mktcPIP** fast path to set the initial values for the host name, the Internet address, and the subnet mask.
- Use the **smit mktcPIP** fast path to specify a name server to provide name resolution service. (Note that **smit mktcPIP** configures one network interface only.)
- Use the **smit chinEt** fast path to set other network attributes.

You may also want to set up any static routes the host needs for sending transmitting information, such as a route to the local gateway. Use the Web-based System Manager, **wsm**, or the SMIT fast path, **smit mkroute**, to set these up permanently in the configuration database.

If you are having other problems with your configuration, see the "Configuring a TCP/IP Network Checklist" for more information.

Common Problems with Network Interfaces

Network interfaces are configured automatically during the first system startup after the adapter card is installed. However, there are certain values that must be set in order for TCP/IP to start. These include the host name and Internet address and can be set using the Web-based System Manager, **wsm**, or the SMIT fast path, **smit mktcPIP**.

If you choose the SMIT method, use the **smit mktcPIP** fast path to set these values permanently in the configuration database. Use the **smit chinEt** and **smit hostname** fast paths to change them in a running system. The **smit mktcPIP** fast path minimally configures TCP/IP. To add adapters, use the Further Configuration menu, which can be reached with the **smit tcPIP** fast path.

If you have already checked these to verify accuracy and you are still having trouble sending and receiving information, check the following:

- Verify that your network adapter has a network interface by executing the **netstat -i** command. The output should list an interface, such as **tr0**, in the Name column. If it does not, create a network interface through Web-based System Manager or by entering the SMIT fast path **smit mkinEt**.
- Verify that IP address for the interface is correct by executing the **netstat -i** command. The output should list the IP address in the Network column. If it is incorrect, set the IP address through Web-based System Manager or by entering the SMIT fast path **smit chinEt**.
- Use the **arp** command to make sure you have the complete IP address for the target machine. For example:

```
arp -a
```

The **arp** command looks for the physical adapter address. This command might show an incomplete address. For example:

```
? (192.100.61.210) at (incomplete)
```

This could be due to an unplugged machine, a stray address with no machine at that particular address, or a hardware problem (such as a machine that connects and receives packets but is not able to send packets back).

- Look for errors on the adapter card. For example:

```
netstat -v
```

The **netstat -v** command shows statistics for the Ethernet, Token Ring, X.25, and 802.3 adapter device drivers. The command also shows network and error logging data for all device drivers active on an interface including: No Mbufs Errors, No Mbuf Extension Errors, and Packets Transmitted and Adapter Errors Detected.

- Check the error log by running the **errpt** command to ensure that there are no adapter problems.
- Verify that the adapter card is good by running diagnostics. Use the Web-based System Manager Devices application, the **smit diag** fast path, or the **diag** command.

If these steps do not identify the problem, see "Problems with a SLIP Network Interface", "Problems with an Ethernet Network Interface", or "Problems with a Token-Ring Network Interface".

Problems with a SLIP Network Interface

In general, the most effective method for debugging problems with a Serial Line Interface Protocol (SLIP) interface is to retrace your configuration, verifying each step. However, you can also:

- Verify that the **slattach** process is running and using the correct tty port by issuing the **ps -ef** command. If it is not, run the **slattach** command. (See "Configuring SLIP over a Modem", or "Configuring SLIP over a Null Modem Cable" for the exact syntax you should use.)
- Verify that the point-to-point addresses are specified correctly by entering the **smit chinnet** fast path. Select the SLIP interface. Make sure that the INTERNET ADDRESS and DESTINATION Address fields are correct.

If the modem is not functioning correctly:

- Make sure that the modem was installed correctly. See the modem installation manual.
- Verify that any flow control the modem does is turned off.

If the tty is not functioning correctly, verify that the tty baud rate and modem characteristics are set correctly in the configuration database by entering the **smit tty** fast path.

Problems with an Ethernet Network Interface

If the network interface has been initialized, the addresses correctly specified, and you have verified that the adapter card is good:

- Verify that you are using a T-connector plugged directly into the inboard/outboard transceiver.
- Make sure you are using an Ethernet cable. (Ethernet cable is 50 OHM.)
- Make sure you are using Ethernet terminators. (Ethernet terminators are 50 OHM.)
- Ethernet adapters can be used with either the transceiver that is on the card or with an external transceiver. There is a jumper on the adapter to specify which you are using. Verify that your jumper is set correctly (see your adapter manual for instructions).
- Verify that you are using the correct Ethernet connector type (thin is BNC; thick is DIX). If you change this connector type, use the Web-based System Manager, **wsm**, or the SMIT fast path, **smit chgenet**,

to set the Apply Change to Database Only field. (Check the field in Web-based System Manager or set to **yes** in SMIT.) Restart the machine to apply the configuration change. (See "Configuring and Managing Adapters".)

Problems with a Token-Ring Network Interface

If you cannot communicate with some of the machines on your network although the network interface has been initialized, the addresses correctly specified, and you have verified that the adapter card is good:

- Check to see if the hosts with whom you cannot communicate are on a different ring. If they are, use the Web-based System Manager, **wsm**, or the SMIT fast path **smit chinnet** to check the Confine BROADCAST to Local Token-Ring field. *Do not* check the field in Web-based System Manager or set to **no** in SMIT.
- Check to see whether the token-ring adapter is configured to run at the correct ring speed. If it is configured incorrectly, use the Web-based System Manager Network application or SMIT to change the adapter ring speed attribute (see "Configuring a High-Performance Token-Ring Adapter"). When TCP/IP is restarted, the token-ring adapter has the same ring speed as the rest of the network.

Problems with a Token-Ring/Ethernet Bridge

If you cannot communicate between a token-ring and an Ethernet network, using a bridge, and you have verified that the bridge is functioning correctly, the Ethernet adapter might be dropping packets. A machine drops packets if the incoming packet (including headers) is greater than the network adapter maximum transmission unit (MTU) value. For instance, a 1500-byte packet sent by a token-ring adapter over the bridge collects an 8-byte logical link control (LLC) header, making the total packet size 1508. If the receiving Ethernet adapter MTU is set to 1500, the packet is dropped.

Check the MTU values of both network adapters. To allow for the eight-byte LLL header, the token-ring adapter attaches to outgoing packets, set the MTU value for the token-ring adapter at least eight bytes lower than the MTU value for the Ethernet adapter. For example, set the MTU for a token-ring adapter to 1492 to communicate with an Ethernet adapter with an MTU of 1500.

Problems with a Token-Ring/Token-Ring Bridge

When operating through a bridge, change the default value of 1500 for the maximum transmission unit (MTU) to a value that is eight less than the maximum information field (maximum I-frame) advertised by the bridge in the routing control field.

To find the routing control field value, use the **iptrace** daemon to look at incoming packets. Bits 1, 2, and 3 of Byte 1 are the Largest Frame Bits, which specify the maximum information field that can be transmitted between two communicating stations on a specific route. See the following for the format of the routing control field:



Figure 26. Routing Control Field. This illustration shows byte 0 and byte 1 of a routing control field. The eight bits of byte one are B, B, B, B, L, L, L, L. The eight bits of byte 1 are D, F, F, F, r, r, r, r.

Values for the Largest Frame Bits are as follows:

- 000** Specifies a maximum of 516 bytes in the information field.
- 001** Specifies a maximum of 1500 bytes in the information field.
- 010** Specifies a maximum of 2052 bytes in the information field.
- 011** Specifies a maximum of 4472 bytes in the information field.
- 100** Specifies a maximum of 8144 bytes in the information field.

- 101 Reserved.
- 110 Reserved.
- 111 Used in all-routes broadcast frames.

For example, if the maximum I-frame value is 2052 in the routing control field, the MTU size should be set to 2044. This is for token-ring network interfaces only.

Note: When using **iptrace**, the output file must *not* be on a Network File System (NFS).

Problems with Packet Delivery

Communicating with a Remote Host

If you cannot communicate with a remote host, try the following:

- Run the **ping** command on the local host to verify that the local interface to the network is up and running.
- Use the **ping** command for hosts and gateways that are progressively more hops from the local host to determine the point at which communication fails.

If you are having trouble with packet loss or are experiencing delays in packet delivery, try the following:

- Use the **trpt** command to trace packets at the socket level.
- Use the **iptrace** command to trace all protocol layers.

If you cannot communicate between a token-ring and an Ethernet network using a bridge, and you have verified that the bridge is good:

- Check the MTU values of both adapters. The MTU values must be compatible to allow communication. A machine drops packets if the incoming packet (including headers) is greater than the adapter's MTU values. For instance, a 1500-byte packet sent over the bridge collects an 8-byte LLC header, making the total packet size 1508. If the receiving machine MTU is set to 1500, a packet of 1508 bytes is dropped.

snmpd Response to Queries

If **snmpd** is not responding to queries and there are no log messages received, the packet might be too large for the kernel User Datagram Protocol (UDP) packet handler. If this is the case, increase the kernel variables, **udp_sendspace** and **udp_recvspace** by issuing the following commands:

```
no -o udp_sendspace=64000
no -o udp_recvspace=64000
```

The maximum size for a UDP packet is 64K. If your query is larger than 64K, it will be rejected. Split the packet into smaller packets to avoid this problem.

Problems with Dynamic Host Configuration Protocol (DHCP)

If you cannot get an IP address or other configuration parameters:

- Check to see that you have specified an interface to be configured. This can be done through the Web-based System Manager Network application, by editing the **/etc/dhcpd.ini** file, or by using the SMIT fast path **smit dhcp**.
- Check to see that there is a server on the local network or a relay agent configured to get your requests off the local network.
- Check to see that the **dhcpd** program is running. If it is not, use the **startsrc -s dhcpd** command.

TCP/IP Reference

Transmission Control Protocol/Internet Protocol (TCP/IP) topics discussed in this section are:

- List of TCP/IP Commands
- List of TCP/IP Daemons
- List of Methods
- List of TCP/IP Files
- List of RFCs

List of TCP/IP Commands

chname	Changes Transmission Control Protocol/Internet Protocol (TCP/IP) based name service configuration on a host.
chprtsv	Changes a print service configuration on a client or server machine.
hostent	Directly manipulates address-mapping entries in the system configuration database.
ifconfig	Configures or displays network interface parameters for a network, using TCP/IP.
mknamsv	Configures TCP/IP-based name service on a host for a client.
mkprtsv	Configures TCP/IP-based print service on a host.
mktcpip	Sets the required values for starting TCP/IP on a host.
no	Configures network options.
rmnamsv	Unconfigures TCP/IP-based name service on a host.
rmprtsv	Unconfigures a print service on a client or server machine.
slattach	Attaches serial lines as network interfaces.
arp	Displays or changes the Internet address to hardware address translation tables used by the Address Resolution Protocol (ARP).
gettable	Gets Network Information Center (NIC) format host tables from a host.
hostid	Sets or displays the identifier of the current local host.
hostname	Sets or displays the name of the current host system.
htable	Converts host files to the format used by network library routines.
ipreport	Generates a packet trace report from the specified packet trace file.
iptrace	Provides interface-level packet tracing for Internet protocols.
lsnamsv	Shows name service information stored in the database.
lsprtsv	Shows print service information stored in the database.
mkhosts	Generates the host table file.
namerslv	Directly manipulates domain name server entries for local resolver routines in the system configuration database.
netstat	Shows network status.
route	Manually manipulates the routing tables.
ruser	Directly manipulates entries in three separate system databases that control foreign host access to programs.
runtime	Displays the status of each host on a network.
securetcpip	Enables the network security feature.
setclock	Sets the time and date for a host on a network.
timedc	Returns information about the timed daemon.
trpt	Performs protocol tracing on Transmission Control Protocol (TCP) sockets.

List of TCP/IP Daemons

fingerd	Provides remote user information.
ftpd	Provides the server function for the Internet File Transfer Protocol (FTP) protocol.
gated	Provides gateway routing functions for the Routing Information Protocol (RIP), Hello Protocol (HELLO), Exterior Gateway Protocol (EGP), Border Gateway Protocol (BGP), and Simple Network Management Protocol (SNMP).
inetd	Provides Internet service management for a network.

named	Provides the server function for the Domain Name Protocol (DOMAIN).
rexecd	Provides the server function for the rexec command.
rlogind	Provides the server function for the rlogin command.
routed	Manages network routing tables.
rshd	Provides the server function for remote command execution.
rwhod	Provides the server function for the rwho and ruptime commands.
syslogd	Reads and logs system messages.
talkd	Provides the server function for the talk command.
telnetd	Provides the server function for the TELNET protocol.
tftpd	Provides the server function for the Trivial File Transfer Protocol (TFTP).
timed	Invokes the timeserver daemon at system startup time.

List of Methods

Device methods are programs associated with a device that perform basic device configuration operations. See List of TCP/IP Programming References in in *AIX 5L Version 5.1 Communications Programming Concepts* for information about TCP/IP methods.

List of TCP/IP Files

/etc/rc.bsdnet

See List of TCP/IP Programming References in in *AIX 5L Version 5.1 Communications Programming Concepts* for information about TCP/IP files and file formats.

List of RFCs

For a list of the RFCs (Request for Comments) supported by this operating system, see the List of TCP/IP Programming References in in *AIX 5L Version 5.1 Communications Programming Concepts*.

- RFC 1359 *Connecting to the Internet: What connecting institutions should anticipate*
- RFC 1325 *FYI on questions and answers: Answers to commonly asked 'new Internet user' questions*
- RFC 1244 *Site Security Handbook*
- RFC 1178 *Choosing a Name for Your Computer*
- RFC 1173 *Responsibilities of host and network managers: A summary of the 'oral tradition' of the Internet*

Chapter 4. Internet Protocol (IP) Security

IP Security enables secure communications over the Internet and within company networks by securing data traffic at the IP layer. This allows individual users or organizations to secure traffic for all applications, without having to make any modifications to the applications. Therefore the transmission of any data, such as e-mail or application-specific company data, can be made secure.

To secure data transmissions, a user creates a virtual tunnel that encapsulates all IP traffic between two hosts, called a *Virtual Private Network (VPN)*. The user's criteria and the tunnel type determine what methods are used for data integrity, privacy, and authentication.

This chapter discusses the following topics:

- IP Security Overview
- IP Security Installation
- Planning IP Security Configuration
- Configuring IKE Tunnels
- Using the IBM Key Manager Tool
- Configuring Manual Tunnels
- Setting Up Filters
- Logging Facilities
- IP Security Problem Determination
- IP Security Reference

IP Security Overview

This section discusses the following topics:

- Benefits of a Virtual Private Network
- IP Security and the Operating System
- IP Security Features
- Security Associations
- Tunnels and Key Management
- Native Filtering Capability
- Digital Certificate Support

Benefits of a Virtual Private Network (VPN)

A virtual private network securely extends a private intranet across a public network such as the Internet. VPNs convey information across what is essentially a private tunnel through the Internet to and from remote users, branch offices, and business partners/suppliers. Companies can opt for Internet access through Internet service providers (ISPs) using direct lines or local telephone numbers and eliminate more expensive leased lines, long-distance calls, and toll-free telephone numbers.

A recommended resource for planning the implementation of a VPN is Chapter 9 of *A Comprehensive Guide to Virtual Private Networks, Volume III*, ISBN SG24-5309-00. This information is also available on the Internet World Wide Web at <http://www.redbooks.ibm.com>. Several additional resources are available when you search the Web for "VPN" or "virtual private network."

IP Security and the Operating System

The operating system uses IP Security, which is an open, standard security technology developed by the Internet Engineering Task Force (IETF), as an integral element of IBM SecureWay VPN solutions. IP Sec provides cryptography-based protection of all data at the IP layer of the communications stack. It provides secure communications transparently. No changes are needed for existing applications. IP Sec is the IETF-chosen industry standard network security framework for both the IP Versions 4 and 6 environments.

IP Security protects your data traffic using the following robust cryptographic techniques:

Authentication

The process by which the identity of a host or end point is verified

Integrity Checking

The process of ensuring that no modifications were made to the data while in-transit across the network

Encryption

The process of ensuring privacy by "hiding" data and private IP addresses while in-transit across the network

Authentication algorithms prove the identity of the sender and data integrity by using a cryptographic hash function to process a packet of data (with the immutable IP header fields included) using a secret key to produce a unique digest. On the receiver side, the data is decapsulated using the same function and key. If either the data has been altered or the sender key is not valid, the datagram is discarded.

Encryption uses a cryptographic algorithm to modify and randomize the data using a certain algorithm and key to produce encrypted data known as *cyphertext*. Encryption makes the data unreadable while in transit. Once received, the data is recovered using the same algorithm and key (with symmetric encryption algorithms). Encryption must occur with authentication to verify the data integrity of the encrypted data.

These basic services are implemented in IP Sec by the use of the Encapsulating Security Payload (ESP) and the Authentication Header (AH). With ESP, confidentiality is provided by encrypting the original IP packet, building an ESP header, and putting the cyphertext in the ESP payload.

The AH can be used alone for authentication and integrity checking if confidentiality is not an issue. With AH, the static fields of the IP header and the data have a hash algorithm applied to compute a keyed digest. The receiver uses its key to compute and compare the digest to make sure the packet is unaltered and the sender's identity is authenticated.

IP Security Features

The IP Security feature of this operating system provides the following functions:

- AH support using RFC 2402, and ESP support using RFC 2406.
- Certificate Revocation List support with retrieval using HTTP or LDAP servers.
- Automatic key refreshment with tunnels using IETF IKE protocol.
- X.509 Digital Certificate and preshared key support in IKE protocol during key negotiation.
- Manual tunnels can be configured to provide interoperability with other systems that do not support the automatic IKE key refreshment method, and for use of IP V6 tunnels.
- Tunnel mode and transport mode of encapsulation for host or gateway tunnels.
- Authentication Algorithms of HMAC MD5 and HMAC SHA.
- Encryption Algorithms include 56 bit DES CBC with 64 bit initial vector (IV), Triple DES, DES CBC 4 (32 bit IV).
- Dual IP Stack Support (IP version 4 and IP version 6).

- Both IP Version 4 and IP Version 6 traffic can be encapsulated and filtered. Because the IP stacks are separate, the IP Security function for each stack can be configured independently.
- IKE tunnels can be created using Linux configuration files (AIX 5.1 and later).
- Filtering of secure and non-secure traffic by a variety of IP characteristics such as source and destination IP addresses, interface, protocol, port numbers, and more.
- Automatic filter rule creation and deletion with most tunnel types.
- Use of host names for the destination address when defining tunnels and filter rules. The host names are converted to IP addresses automatically (as long as DNS is available).
- Logging of IP Security events to **syslog**.
- Extensive use of system traces and statistics for problem determination.
- User defined default action allows the user to specify whether traffic that does not match defined tunnels should be allowed or denied.

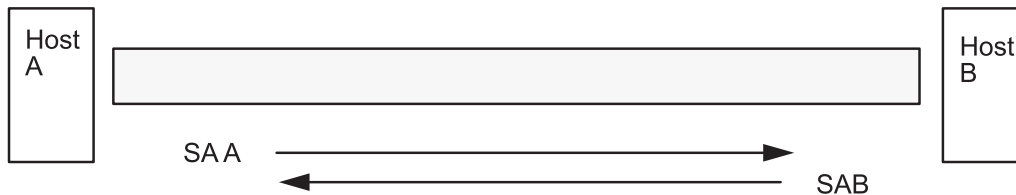
IKE Features

The following features are available with Internet Key Exchange, AIX 4.3.2 and later:

- Authentication with preshared keys and X.509 digital signatures.
- Use of main mode (identity protect mode) and aggressive mode.
- Support for Diffie Hellman groups 1 and 2.
- ESP encryption support for DES, Triple DES, Null encryption; ESP authentication support with HMAC MD5 and HMAC SHA1.
- AH support for HMAC MD5 and HMAC SHA1.
- IP Version 4 and Version 6 support.

Security Associations

The building block on which secure communications is built is a concept known as a *security association*. Security associations (SAs) relate a specific set of security parameters to a type of traffic. With IP Security-protected data, a separate SA exists for each direction and for each header type, AH or ESP. The information contained in the SA includes the IP addresses of the communicating parties, a unique identifier known as the Security Parameters Index (SPI), the algorithms selected for authentication and/or encryption, the authentication and encryption keys, and the key lifetimes figure.



SA = Security Association, consisting of:

- Destination address
- SPI
- Key
- Crypto Algorithm and Format
- Authentication Algorithm
- Key Lifetime

Figure 27. Establishment of a Secure Tunnel Between Hosts A and B. This illustration shows a virtual tunnel running between Host A and Host B. Security Association (SA) A is an arrow directed from Host A to Host B. SA B is an arrow directed from Host B to Host A. A Security Association consists of the Destination Address, SPI, Key, Crypto Algorithm and Format, Authentication Algorithm, and Key Lifetime.

The goal of key management is to negotiate and compute the SAs that protect IP traffic.

Tunnels and Key Management

To set up a secure communication between two hosts, Security Associations must be negotiated and managed during the use of the tunnel. Two types of tunnels are supported, and each uses a different key management technique:

- IKE tunnels (dynamically changing keys, IETF standard)
- Manual tunnels (static, persistent keys, IETF standard)

IKE Tunnel Support

IKE Tunnels are based on the ISAKMP/Oakley standards developed by the IETF. With this protocol, security parameters are negotiated and refreshed, and keys are exchanged securely. Two types of authentication are supported: preshared key and X.509v3 digital certificate signatures.

The negotiation uses a two phase approach. The first phase authenticates the communicating parties, and specifies the algorithms to be used for securely communicating in phase 2. During phase 2, IP Security parameters to be used during data transfer are negotiated, security associations and keys are created and exchanged.

Algorithm	AH IP Version 4 & 6	ESP IP Version 4 & 6
HMAC MD5	X	X
HMAC SHA1	X	X
DES CBC 8		X
Triple DES CBC		X
ESP Null		X

Manual Tunnel Support

Manual tunnels provide backward compatibility and interoperate with machines that do not support IKE key management protocols. The disadvantage of manual tunnels is that the key values are static. The encryption and authentication keys are the same for the life of the tunnel and must be manually updated.

Algorithm	AH IP Version 4	AH IP Version 6	ESP IP Version 4	ESP IP Version 6
HMAC MD5	X	X	X	X
HMAC SHA1	X	X	X	X
Triple DES CBC			X	X
DES CBC 8			X	X
DES CBC 4			X	X

Since IKE tunnels offer more effective security, IKE is the preferred key management method.

Note: Beginning with AIX 4.3.3, CDMF support has been removed because DES is now available world wide. Reconfigure any tunnels that use CDMF to use DES or Triple DES.

Native Filtering Capability

Filtering is a basic function in which incoming and outgoing packets can be accepted or denied based on a variety of characteristics. This allows a user or system administrator to configure the host to control the traffic between this host and other hosts. Filtering is done on a variety of packet properties, such as source and destination addresses, IP version (4 or 6), subnet masks, protocol, port, routing characteristics, fragmentation, interface, and tunnel definition.

Rules, known as *filter rules*, are used to associate certain kinds of traffic with a particular tunnel. In a basic configuration for manual tunnels, when a user defines a host-to-host tunnel, filter rules are autogenerated to direct all traffic from that host through the secure tunnel. If more specific types of traffic are desired (for instance subnet to subnet), the filter rules can be edited or replaced to allow precise control of the traffic using a particular tunnel.

For IKE tunnels, the filter rules are also automatically generated and inserted in the filter table once the tunnel is activated.

Similarly, when the tunnel is modified or deleted, the filter rules for that tunnel are automatically deleted, which greatly simplifies IP Security configuration and helps reduce human error. Tunnel definitions can be propagated and shared among machines and firewalls using import and export utilities, which is especially helpful in the administration of a large number of machines.

Filter rules associate particular types of traffic with a tunnel, but data being filtered does not necessarily need to travel in a tunnel. This aspect of filter rules lets the operating system provide basic firewall functionality to those who want to restrict traffic to or from their machine in an intranet or in a network that does not have the protection of a true firewall. In this scenario, filter rules provide a second barrier of protection around a group of machines.

Once the filter rules are generated, they are stored in a table and loaded into the kernel. When packets are ready to be sent or received from the network, the filter rules are checked in the list from top to bottom to determine whether the packet should be permitted, denied, or sent through a tunnel. The criteria of the rule is compared to the packet characteristics until a match is found or the default rule is reached.

The IP Security function also implements filtering of non-secure packets based on very granular, user-defined criteria, which allows the control of IP traffic between networks and machines that do not require the authentication or encryption properties of IP Security.

Digital Certificate Support

IP Security supports the use of X.509 Version 3 digital certificates. The IBM Key Manager tool manages certificate requests, maintaining the key database, and other administrative functions.

Digital certificates are described in Digital Certificate Configuration. The IBM Key Manager and its functions are described in Using the IBM Key Manager Tool

IP Security Installation

The IP Security feature in AIX is separately installable and loadable. The file sets that need to be installed are:

- **bos.net.ipsec.rte** The run time environment for the kernel IP Security environment and commands
- **bos.msg.LANG.net.ipsec** (where *LANG* is the desired language, such as **en_US**)
- **bos.net.ipsec.keymgt**
- **bos.net.ipsec.websm**
- **bos.crypto-wt** file set for DES encryption
- **bos.crypto-priv** file set for triple DES encryption

The **bos.crypto*** file sets are located on the ExpansionPacks. For IKE digital signature support, also install the **gskit.rte** fileset (AIX Version 4) or **gskkm.rte** (AIX 5.1) from the Expansion Pack.

Once installed, IP Security can be separately loaded for IP Version 4 and IP Version 6, either by using the recommended procedure given in Loading IP Security or by using the **mkdev** command.

Loading IP Security

Attention: Loading IP Security enables the filtering function. Before loading, it is important to ensure the correct filter rules are created, or all outside communication might be blocked.

Use SMIT or Web-based System Manager, to automatically load the IP security modules when IP Security is started. Also, SMIT and Web-based System Manager ensure that the kernel extensions and IKE daemons are loaded in the correct order.

If the loading completes successfully, the **lsdev** command shows the IP Security devices as Available.

```
lsdev -C -c ipsec
```

```
ipsec_v4 Available IP Version 4 Security Extension
ipsec_v6 Available IP Version 6 Security Extension
```

Once the IP Security kernel extension has been loaded, tunnels and filters are ready to be configured.

Planning IP Security Configuration

To configure IP Security, tunnels and filters must be configured. When a simple tunnel is defined for all traffic to use, the filter rules can be automatically generated. If more complex filtering is desired, filter rules can be configured separately.

You can configure IP Security using the Web-based System Manager Network plug-in or the System Management Interface Tool (SMIT). If using SMIT, the following fast paths take you directly to the configuration panels you need:

ips4_basic

Basic configuration for IP version 4

ips6_basic

Basic configuration for IP version 6

Before configuring IP Security for your site, you must decide several what you intend to use; for example, whether you prefer to use tunnels or filters (or both), which type of tunnel best suits your needs, etc. The following sections provide information you must understand before making these decisions:

- Tunnels versus Filters
- Tunnels and Security Associations
- Choosing a Tunnel Type
- Using IKE with DHCP or Dynamically Assigned Addresses

Tunnels versus Filters

There are two related but distinct parts of IP Security: *tunnels* and *filters*. Tunnels require filters, but filters do not require tunnels.

Filtering is a basic function in which incoming and outgoing packets can be accepted or denied based on a variety of characteristics called *rules*. This allows a system administrator to configure the host to control the traffic between this host and other hosts. Filtering is done on a variety of packet properties, such as source and destination addresses, IP Version (4 or 6), subnet masks, protocol, port, routing characteristics, fragmentation, interface, and tunnel definition. This filtering is done at the IP layer, so no changes are required to the applications.

Tunnels define a security association between two hosts. These security associations involve specific security parameters that are shared between end points of the tunnel.

The following illustration indicates how a packet comes in the network adapter to the IP stack. From there, the filter module is called to determine if the packet is permitted or denied. If a tunnel ID is specified, the packet is checked against the existing tunnel definitions. If the decapsulation from the tunnel is successful, the packet is passed to the upper layer protocol. This function occurs in reverse order for outgoing packets. The tunnel relies on a filter rule to associate the packet with a particular tunnel, but the filtering function can occur without passing the packet to the tunnel.

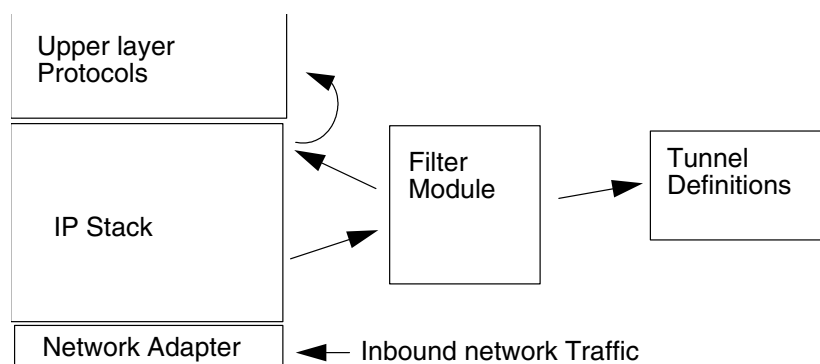
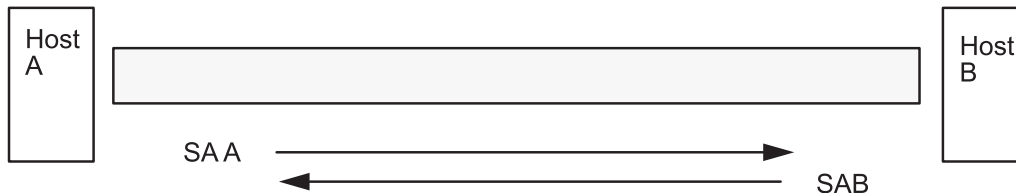


Figure 28. Network Packet Routing. This illustration shows the route a network packet takes. Inbound from the network, the packet enters the network adapter. From there it goes to the IP stack where it is sent to the filter module. From the filter module, the packet is either sent to tunnel definitions or it is returned to the IP stack where it is forwarded to the upper layer protocols.

Tunnels and Security Associations

Tunnels are used whenever it is desired to have data authenticated or authenticated and encrypted. Tunnels are defined by specifying a security association between two hosts. The security association defines the parameters for the encryption and authentication algorithms and characteristics of the tunnel.



SA = Security Association, consisting of:

- Destination address
- SPI
- Key
- Crypto Algorithm and Format
- Authentication Algorithm
- Key Lifetime

Figure 29. Establishment of a Secure Tunnel Between Hosts A and B. This illustration shows a virtual tunnel running between Host A and Host B. Security Association (SA) A is an arrow directed from Host A to Host B. SA B is an arrow directed from Host B to Host A. A Security Association consists of the Destination Address, SPI, Key, Crypto Algorithm and Format, Authentication Algorithm, and Key Lifetime.

The Security Parameter Index (SPI) and the destination address identify a unique security association. These two parameters are required for uniquely specifying a tunnel. Other parameters such as cryptographic algorithm, authentication algorithm, keys, and lifetime can be specified or defaults can be used.

Choosing a Tunnel Type

The decision to use manual tunnels or IKE tunnels depends on the tunnel support of the remote end and the type of key management desired. IKE tunnels are recommended (when available) because they offer secure key negotiation and key refreshment in an industry-standard way. They also take advantage of the IETF ESP and AH header types and support anti-replay protection. You can optionally configure signature mode to allow digital certificates.

If the remote end uses one of the algorithms requiring manual tunnels, manual tunnels should be used. Manual tunnels ensure interoperability with a large number of hosts. Because the keys are static and difficult to change and might be cumbersome to update, they are not as secure. Manual tunnels can be used between a host running this operating system and any other machine running IP Security and having a common set of cryptographic and authentication algorithms. Most vendors offer Keyed MD5 with DES, or HMAC MD5 with DES. This subset works with almost all implementations of IP Security.

When setting up manual tunnels, the procedure depends on whether you are setting up the first host of the tunnel or setting up the second host, which must have parameters matching the first host setup. When setting up the first host, the keys can be autogenerated, and the algorithms can be defaulted. When setting up the second host, it is best to import the tunnel information from the remote end, if possible.

Another important consideration is determining whether the remote system is behind a firewall. If it is, the setup must include information about the intervening firewall.

Using IKE with DHCP or Dynamically Assigned Addresses

One common scenario for using IP Security with an operating system is when remote systems are initiating IKE sessions with a server and their identity cannot be tied to a particular IP address. This case can occur in a Local Area Network (LAN) environment such as using IP Security to connect to a server on a LAN and wanting to encrypt the data. Other common uses involve remote clients dialing into a server, and using either a fully qualified domain name (FQDN) or email address (user@FQDN) to identify the remote ID.

In order to make a policy decision based on explicit information about the remote identity, aggressive mode must be used. In this case, the identity is sent in the first message of the negotiation and can be used to do a policy lookup on the security policy database. This will ensure that only specifically named remote identities will be able to negotiate using the IKE protocol.

For phase 2, when the IP Security associations are being created to encrypt TCP or UDP traffic, a default tunnel can be configured. Therefore, any request that was authenticated during phase 1, will use the default tunnel for defined phase 2 if the IP address is not explicitly configured in the database. This allows any address to match the default tunnel and be used as long as the rigorous public key-based security validation was successful in phase 1.

To define a default phase 2 tunnel, select a Key Management tunnel in the IKE Tunnels container, and select the action to Define a Default Tunnel. The configuration panels are the same as the panels used to define a Data Management tunnel. However, the choices for the ID types are different and the ID fields themselves are disabled. This is because explicit IDs do not need to be specified. The ID types, which are IP v4 or v6 Address Only, IP v4 or v6 Subnet Only, and IP v4 or v6 Address or Subnet, cover all allowable cases of IDs. Set the rest of the information the same way as in a Data Management Tunnel and click OK. Each Key Management tunnel can only have one associated Default Tunnel.

Configuring IKE Tunnels

Internet Key Exchange (IKE) tunnels have more complex policy parameters. In most cases, you must use the Web-based System Manager interface to configure IKE tunnels. The Basic Configuration wizard provides an easy way to define an IKE tunnel with preshared keys. For more advanced options, see Advanced IKE Tunnel Configuration.

Basic Configuration Wizard

The basic configuration wizard is an easy way to define an IKE tunnel through Web-based System Manager using preshared keys or certificates as the authentication method. It adds new key management and data management IKE tunnels to the IP Security subsystem, allows you to input minimal data and choose some options, and makes use of common default values for such parameters as tunnel lifetime. The basic configuration wizard can also be used on the remote endpoint to quickly set up a corresponding IKE tunnel end point.

Once a tunnel is defined using the wizard, the tunnel definition appears in the Web-based System Manager IKE tunnels list and can be activated or modified.

When using the basic configuration wizard, keep the following in mind:

- The wizard can be used only to *define* tunnels. To modify, delete, or activate a tunnel, use the IKE Tunnels plug-in or task bar.
- The tunnel name needs to be unique on your system, but you can use the same name on the remote system. For example, on the local and remote systems, the tunnel name can be hostA_to_hostB, but the Local IP Address and the Remote IP Address fields (endpoints) are switched.
- Phase 1 and phase 2 tunnels are defined with the same encryption and authentication algorithms.
- The preshared key must be entered in hexadecimal form (without a leading 0x) or ASCII text.

- If digital certificates are chosen as the authentication method, you must use the IBM Key Manager tool to create a digital certificate.
- The host ID type can only be IP Address.
- The transforms and proposals you create are assigned names that end with the user-defined tunnel name. You can view the transforms and proposals in Web-based System Manager through **VPN** and the **IKE Tunnel** plug-in.

Use the following procedure to configure a new tunnel using the wizard:

1. Open Web-based System Manager using the **wsm** command.
2. Select the Network plug-in.
3. Select **Virtual Private Networks (IP Security)**.
4. From the Console area, choose the Overview and Tasks folder.
5. Select **Configure a Basic Tunnel Configuration wizard**.
6. Click on Next on the Step 1 Introduction panel then follow the steps to configure an IKE tunnel. There are six configuration dialogs.

Online help is available if you need it.

Advanced IKE Tunnel Configuration

Creating IKE tunnels differs from creating manual tunnels because the setting of security policy is a separate process from defining tunnel endpoints. In IKE, there is a two-step negotiation process, called *phases*, and each phase can have separate security policies.

When the Internet Key negotiation starts, it must set up a secure channel for the negotiations. This is known as the *key management* phase or *phase 1*. During this phase, each party uses preshared keys or digital certificates to authenticate the other and pass ID information. This phase sets up a security association during which the two parties determine how they plan to communicate securely and then which protections are to be used to communicate during the second phase. The result of this phase is an *IKE* or *phase 1* tunnel.

The second phase is known as the *data management* phase or *phase 2* and uses the IKE tunnel to create the IP Sec security associations for AH and ESP that actually protect traffic. The second phase also determines the data that will be using the IP Security tunnel. For example, it can specify:

- a subnet mask,
- an address range,
- or a protocol and port number combination

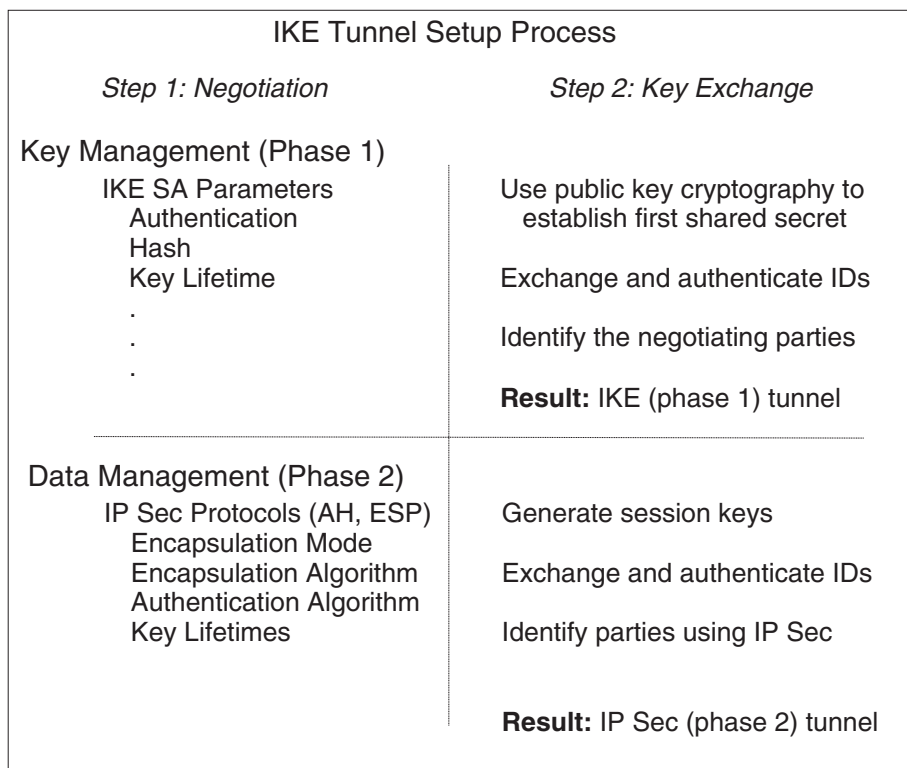


Figure 30. IKE Tunnel Setup Process. This illustration shows the two-step, two-phase process for setting up an IKE tunnel. A description of this process can be found in the document text.

In many cases, the endpoints of the key management (IKE) tunnel will be the same as the endpoints of the data management (IP Security) tunnel. The IKE tunnel endpoints are the IDs of the machines carrying out the negotiation. The IP Security tunnel endpoints describe the type of traffic that will use the IP Security tunnel. For simple host-to-host tunnels, in which all traffic between two tunnels is protected with the same tunnel, the phase 1 and phase 2 tunnel endpoints are the same. When negotiating parties are two gateways, the IKE tunnel endpoints are the two gateways, and the IP Security tunnel endpoints are the machines or subnets (behind the gateways) or the range of addresses (behind the gateways) of the tunnel users.

Key Management Parameters and Policy

The following parameters are set during phase 1 (the key management phase) of an IKE tunnel configuration:

Key Management (Phase 1) Tunnel	The name of this IKE tunnel. For each tunnel, the endpoints of the negotiation must be specified. These are the two machines that plan to send and validate IKE messages. The name of the tunnel may describe the tunnel endpoints such as VPN Boston or VPN Acme.
Host Identity Type	The ID type that will be used in the IKE exchange. The ID type and value must match the value for the preshared key to ensure that proper key lookup is performed. If a separate ID is used to search a preshared key value, the <i>host ID</i> is the key's ID and its <i>type</i> is KEY_ID. The KEY_ID type is useful if a single host has more than one preshared key value
Host Identity	The value of the host ID represented as an IP address, a fully qualified domain name (FQDN), or a user at the fully qualified domain name (<i>user@FQDN</i>). For example: jdoe@studentmail.ut.edu

IP Address	The IP address of the remote host. This value is required when the host ID type is KEY_ID or whenever the host ID type cannot be resolved to an IP address. For example, if the user name cannot be resolved with a local nameserver, the IP address for the remote side must be entered.
-------------------	---

You can customize key management policy by specifying the parameters to be used during IKE negotiation. For example, there are key management policies for preshared key or signature mode authentication. For Phase 1, the user must determine certain key management security properties with which to carry out the exchange. These properties are fully described in the help available with Web-based System Manager and in the documentation for the **ike** command.

Data Management Parameters and Policy

The data management proposal parameters are set during phase 2 of an IKE tunnel configuration. They are the same IP Security parameters used in manual tunnels and describe the type of protection to be used for protecting data traffic in the tunnel. You can start more than one phase 2 tunnel under the same phase 1 tunnel.

The following endpoint ID types describe the type of data that is to use the IP Security Data tunnel:

Host, Subnet, or Range	Describes whether the data traffic traveling in the tunnel will be for a particular host, subnet, or address range.
Host/Subnet ID	Contains the host or subnet identity of the local and remote systems passing traffic over this tunnel. Determines the IDs sent in the phase 2 negotiation and the filter rules that will be built if the negotiation is successful.
Subnet mask	Describes all IP addresses within the subnet (for example, host 9.53.250.96 and mask 255.255.255.0)
Starting IP Address Range	Provides the starting IP address for the range of addresses that will be using the tunnel (for example, 9.53.250.96 of 9.53.250.96 to 9.53.250.93)
Ending IP Address Range	Provides the ending IP address for the range of addresses that will be using the tunnel (for example, 9.53.250.93 of 9.53.250.96 to 9.53.250.93)
Port	Describes data using a specific port number (for example, 21 or 23)
Protocol	Describes data being transported with a specific protocol (for example, TCP or UDP). Determines the protocol sent in the phase 2 negotiation and the filter rules that will be built if the negotiation is successful. The protocol for the local endpoint must match the protocol for the remote end point.

The data management parameters are fully described in the help available with Web-based System Manager and in the documentation for the **ike** command.

Setting up, Key Management

IKE tunnels are configured using the Web-based System Manager tool. Use the following procedure to add an IKE tunnel:

1. Open Web-based System Manager using the **wsm** command.
2. Select the Network plug-in.
3. Select **Virtual Private Networks (IP Security)**.
4. From the Console area, choose the Overview and Tasks folder.
5. Start **IP Security**. This action loads the IP Security kernel extensions and starts the **isakmpd**, **tmd**, and **cpsd** daemons.

A tunnel is created by defining the key management and data management endpoints and their associated security transforms and proposals.

- Key management is the authentication phase, which is computationally expensive and is needed less often. It sets up a secure channel between the negotiating parties needed before the final IP Security parameters and keys are computed.
- Data management describes the type of traffic that will be using a particular tunnel. It can be configured for a single host or group of hosts (with the use of subnets or IP ranges) along with specified protocol and port numbers.

The same key management tunnel can be used to protect multiple data management negotiations and key refreshes, as long as they take place between the same two endpoints, for example between two gateways.

6. To define the key management tunnel endpoints, click on **Internet Key Exchange (IKE) Tunnels** from the Identification tab.
7. Enter information to describe the identities of the systems taking part in the negotiations. In most cases, IP addresses are used, and a policy compatible with the remote side must be created. On the Transforms tab, use matching transforms on both sides, or contact the administrator on the remote end to define a matching transform. A transform containing several choices can be created to allow flexibility when proposing or matching on a transform.
8. If using preshared keys for authentication, enter the preshared key under the **key** tab. This value must match on both the remote and local machines.
9. Create a transform to be associated with this tunnel using the Add button on the Transforms tab. To enable digital certificates and signature mode support, choose an authentication method of **RSA Signature** or **RSA Signature with CRL Checking**. For more information about digital certificates, see Digital Certificate Configuration.

Setting up, Data Management

To set up data management tunnel endpoints and proposals and to complete IKE tunnel setup, launch the Web-based System Manager tool, as described in Setting up, Key Management. A data management tunnel is created by selecting a key management tunnel and defining any unique options. Most data management options can remain as defined by the default.

However, you must specify endpoint types (such as IP address, subnet, or IP address range) under the Endpoints tab. You can select a port number and protocol or accept the default.

On the Proposals panel, you can create a new proposal by selecting the Add button or simply click on OK to create a proposal. If there are multiple proposals, you can use the Move Up or Move Down buttons to change the search order.

Group Support

Beginning with AIX 5.1, IP security supports grouping IKE IDs in a tunnel definition to associate multiple IDs with a single security policy without having to create separate tunnel definitions. Grouping is especially useful when setting up connections to several remote hosts, because you can avoid setting up or managing multiple tunnel definitions. Also, if changes must be made to a security policy, you do not need to change multiple tunnel definitions.

A group is composed of a group name and a list of IKE IDs and ID types. The IDs can all be the same type or a mix of IPv4 addresses, IPv6 addresses, FQDN, user@FQDN, and X500 DN types. During a Security Association negotiation, the IDs in a group are searched linearly for the first match.

A group must be defined before using that group name in tunnel definition. Use the **ikedb** command to define groups. This command accepts XML text as input to create a group definition in the IKE databases. The group's size is limited to 1 Kbyte.

A group name can be used in both key management tunnel and data management tunnel definitions, but can be used only as a remote ID.

Activating a Tunnel

Activate the tunnel using the Web-based System Manager tool (see Setting up, Key Management) or from the command line using the **ike** command. Online help is available for the Web-based System Manager tool. The following instructions describe how to activate a tunnel from the command line.

1. To start a tunnel negotiation (*activate* a tunnel) or to allow the incoming system to act as a responder (depending on the role that is specified), use the **ike** command with a tunnel number, as shown in the following example:

```
ike cmd=activate numlist=1
```

You can also use remote id or IP addresses, as shown in the following examples:

```
ike cmd=activate remid=9.3.97.256
ike cmd=activate ipaddr=9.3.97.100, 9.3.97.256
```

Since it may take several seconds for the commands to complete, the command returns after the negotiation is started.

2. To ensure the command was successful, use the following command to display the tunnel status.

```
ike cmd=list
```

The output looks similar to the following:

```
Phase 1 Tunnel ID      [1]
Phase 2 Tunnel ID      [1]
```

This command shows the phase 1 and phase 2 tunnels that are currently active. To do a verbose listing of the tunnel, use the verbose option, as shown in the following example:

```
ike cmd=list verbose
```

```
Phase 1 Tunnel ID      1
Local ID Type:         Fully_Qualified_Domain_Name
Local ID:               bee.austin.ibm.com
Remote ID Type:         Fully_Qualified_Domain_Name
Remote ID:              ipsec.austin.ibm.com
Mode:                  Aggressive
Security Policy:        BOTH_AGGR_3DES_MD5
Role:                   Initiator
Encryption Alg:         3DES-CBC
Auth Alg:               Preshared Key
Hash Alg:               MD5
Key Lifetime:           28800 Seconds
Key Lifesize:           0 Kbytes
Key Rem Lifetime:       28737 Seconds
Key Rem Lifesize:       0 Kbytes
Key Refresh Overlap:    5%
Tunnel Lifetime:        2592000 Seconds
Tunnel Lifesize:        0 Kbytes
Tun Rem Lifetime:       2591937 Seconds
Status:                 Active

Phase 2 Tunnel ID      1
Local ID Type:         IPv4_Address
Local ID:               10.10.10.1
Local Subnet Mask:      N/A
Local Port:             any
Local Protocol:         all
Remote ID Type:         IPv4_Address
Remote ID:              10.10.10.4
Remote Subnet Mask:     N/A
Remote Port:            any
Remote Portocol:        all
Mode:                   Oakley_quick
Security Policy:        ESP_3DES_MD5_SHA_TUNNEL_NO_PFS
Role:                   Initiator
```



```

Encryption Alg:      ESP_3DES
AH Transform:       N/A
Auth Alg:           HMAC-MD5
PFS:                No
SA Lifetime:        600 Seconds
SA Lifesize:        0 Kbytes
SA Rem Lifetime:    562 Seconds
SA Rem Lifesize:    0 Kbytes
Key Refresh Overlap: 15%
Tunnel Lifetime:    2592000 Seconds
Tunnel Lifesize:    0 Kbytes
Tun Rem Lifetime:   2591962 Seconds
Assoc P1 Tunnel:    0
Encap Mode:         ESP_tunnel
Status:             Active

```

Activating the IKE tunnel causes filter rules for the new tunnel to be inserted into the dynamic filter table. These entries can be viewed using the **lsfilt** command with the **-d** option for dynamic filter rules:

```

1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both both no all
   packets 0 all
2 *** Dynamic filter placement rule *** no
0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both no all
   packets 0 all

*** Dynamic table ***

0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 500 eq 500 local both no all
   packets 0
0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no ah any 0 any 0 both inbound no all
   packets 0
0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no esp any 0 any 0 both inbound no all
   packets 0
1 permit 10.10.10.1 255.255.255.255 10.10.10.4 255.255.255.255 no all any 0 any
   0 both outbound yes all packets 1
1 permit 10.10.10.4 255.255.255.255 10.10.10.1 255.255.255.255 no all any 0 any
   0 both inbound yes all packets 1

```

This example shows a machine that has one IKE tunnel and no other tunnels. The dynamic filter rule (rule #2 in this example of the static table) can be moved by the user to control placement relative to all other user-defined rules. The rules in the dynamic table are constructed automatically as tunnels are negotiated and corresponding rules are inserted into the filter table. These rules can be displayed, but not edited.

- To turn on logging of the dynamic filter rules, set the logging option for rule #2 to yes, as shown in the following example:

```
chfilt -v 4 -n 2 -l y
```

For more details on logging of IKE traffic, see the section on Logging Facilities.

- To deactivate the tunnel, use the remove option, as shown in the following example:

```
ike cmd=remove numlist=1
```

Command Interface to IKE Tunnel Creation

The **ikedb** command, available in AIX 5.1 and later, allows a user to create IKE tunnels using an XML interface. The **ikedb** command can also be used for IKE data retrievals (useful for database backups), updates, deletes, and imports. An example XML file can be found in **/usr/samples/ipsec**. See the **ikedb** command description in the *AIX 5L Version 5.1 Commands Reference* for syntax details.

You can view the tunnel definitions using either the **ikedb** command with the **-g** flag or the Web-based System Manager. To activate the tunnel, use either the **ike cmd=activate** command or the Web-based System Manager.

AIX IKE and Linux affinity

To configure an AIX IKE tunnel using Linux configuration files (AIX 5.1 and later), use the **ikedb** command with the **-c** flag (conversion option), which lets you use the **/etc/ipsec.conf** and **/etc/ipsec.secrets** Linux configuration files as IKE tunnel definitions. The **ikedb** command parses the Linux configuration files, creates an XML file, and optionally adds the XML tunnel definitions to the IKE database. You can then view the tunnel definitions by using either the **ikedb -g** command or the Web-based System Manager.

Examples of IKE Tunnel Configurations

There are typical scenarios that describe the type of situations most customers encounter when trying to set up tunnels. They can be described as the branch office, business partner, and remote access cases.

In the branch office case, the customer has two trusted networks that they want to connect together—the engineering group of one location to the engineering group of another. In this example, there are gateways that connect and all the traffic passing between the gateways use the same tunnel. The traffic at either end of the tunnel is decapsulated and passes in the clear within the company intranet.

In the first phase of the IKE negotiation, the IKE security association is created between the two gateways. The traffic that passes in the IP Security tunnel is the traffic between the two subnets, and the subnet IDs are used in the phase 2 negotiation. After the security policy and tunnel parameters are entered for the tunnel, a tunnel number is created. To start the tunnel, use the **ike** command.

In the business partner scenario, the networks are not trusted, and the network administrator may want to restrict access to a smaller number of hosts behind the security gateway. In this case, the tunnel between the hosts carries traffic protected by IP Security for use between two particular hosts. The protocol of the phase 2 tunnel is AH or ESP. This host-to-host tunnel is secured within a gateway-to-gateway tunnel.

In the remote access case, the tunnels are setup on demand and a high level of security is applied. The IP addresses may not be meaningful, therefore, fully qualified domain names or *user@* fully qualified domain names are preferred. Optionally, you can use KEYID to relate a key to a host ID.

Digital Certificate Configuration

Digital certificates bind an identity to a public key, through which you can verify the sender or the recipient of an encrypted transfer. Beginning with AIX 4.3.3, IP Security uses digital certificates to enable *public-key cryptography*, also known as *asymmetric cryptography*, which encrypts data using a private key known only to the user and decrypts it using an associated public (shared) key from a given public-private key pair. *Key pairs* are long strings of data that act as keys to a user's encryption scheme.

In *public-key cryptography*, the public key is given to anyone with whom the user wants to communicate. The sender digitally signs all secure communications with the corresponding private key for their assigned key pair. The recipient uses the public key to verify the sender's signature. If the message is successfully decrypted with the public key, the receiver can verify that the sender was authenticated.

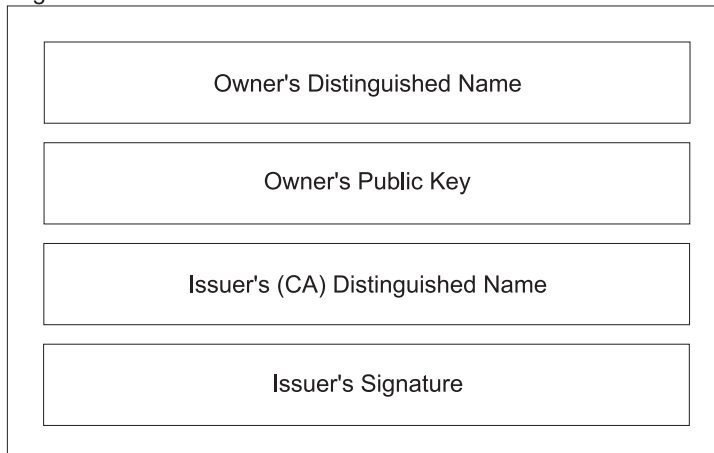
Public-key cryptography relies on trusted, third parties, known as a *certification authorities (CAs)*, to issue reliable digital certificates. The recipient specifies which issuing organizations or authorities are deemed trusted. A certificate is issued for a specific amount of time; when its expiration date has passed, it must be replaced.

AIX 4.3.3 and later versions provide the IBM Key Manager tool, which manages digital certificates. The following sections provide conceptual information about the certificates themselves. Management tasks for these certificates are described in Using the IBM Key Manager Tool.

Format of Digital Certificates

The digital certificate contains specific pieces of information about the identity of the certificate owner and about the certification authority. See the following figure for an illustration of a digital certificate. The following list further describes the contents:

Digital Certificate

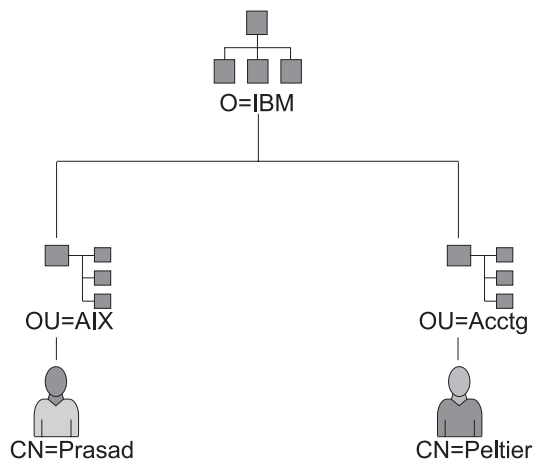


Contents of a Digital Certificate

Figure 31. Contents of a Digital Certificate. This illustration shows the four entities of a digital certificate. From the top they are, Owner's Distinguished Name, Owners Public Key, Issuer's (CA) Distinguished Name, and Issuer's Signature.

Owner's Distinguished Name

Combination of the owner's common name and context (position) in the directory tree. In the following figure of a simple directory tree, for example, Prasad is the owner's common name and the context is country=US, organization=IBM, lower organization=SERV; therefore, the distinguished name is:



Example of Deriving Distinguished Name from Directory Tree

Figure 32. Example of Deriving Distinguished Name from Directory Tree. This illustration is a directory tree with O=IBM at the top level and branching to two entities on the second level. Level two contains OU=AIX and OU=Acctg on separate branches; each has a branch leading to a single entity on the last level.. The last level contains CN=Prasad and CN=Peltier respectively.

/C=US/O=IBM/OU=SERV/CN=prasad.austin.ibm.com

Owner's Public Key

Used by the recipients to decrypt data.

Subject Alternate Name

Can be an identifier such as an IP address, e-mail address, fully qualified domain name, etc.

Issue Date

Date the digital certificate was issued.

Expiration Date

Date the digital certificate expires.

Issuer's Distinguished Name

Distinguished name of the Certification Authority.

Issuer's Digital Signature

Digital signature used to validate a certificate.

Security Considerations for Digital Certificates

A digital certificate alone cannot prove identity. The digital certificate only allows you to verify the identity of the digital certificate owner by providing the public key that is needed to check the owner's digital signature. You can safely send your public key to another because your data cannot be decrypted without the other part of the key pair, your private key. Therefore, the owner must protect the private key that belongs to the public key in the digital certificate. All communications of the owner of a digital certificate can be deciphered if the private key is known. Without the private key, a digital certificate cannot be misused.

Certification Authorities and Trust Hierarchies: A digital certificate is only as trustworthy as the certification authority (CA) that issued it. As part of this trust, the policies under which certificates are issued should be understood. Each organization or user must determine which certification authorities can be accepted as trustworthy.

Certification Authority organizations such as Verisign, Entrust, or Thawte can issue digital certificates. The IBM Key Manager tool also allows organizations to create self-signed certificates, which can be useful for testing or in environments with a small number of users or machines.

As a user of a security service, you need to know its public key to obtain and validate any digital certificates. Also, simply receiving a digital certificate does not assure its authenticity. To verify its authenticity, you need the public key of the certification authority that issued that digital certificate. If you do not already hold an assured copy of the CA's public key, then you might need an additional digital certificate to obtain the CA's public key.

Certificate Revocation Lists (CRLs)

A digital certificate is expected to be used for its entire validity period. If needed, however, a certificate can be invalidated before its actual date of expiration. Invalidating the certificate may be necessary, for example, if an employee leaves the company or if the certificate's private key has been compromised. To invalidate a certificate, you must notify the appropriate Certificate Authority (CA) of the circumstances. When a CA revokes a certificate, it adds the invalid certificate serial number to a Certificate Revocation List (CRL).

CRLs are signed data structures that are issued periodically and made available in a public repository. CRLs can be retrieved from HTTP or LDAP servers. Each CRL contains a current timestamp and a **nextUpdate** timestamp. Each revoked certificate in the list is identified by its certificate serial number.

When configuring an IKE tunnel and using digital certificates as your authentication method, you can confirm the certificate has not been revoked by selecting RSA Signature with CRL Checking. If CRL Checking is enabled, the list is located and checked during the negotiation process to establish the key management tunnel.

Note: To use this feature of IP Security, your system must be configured to use a SOCKS server (version 4 for HTTP servers), an LDAP server, or both. If you know which SOCKS or LDAP server

you are using to obtain CRLs, you can make the necessary configuration selections by choosing CRL Configuration from the Digital Certificates menu within the Web-based System Manager.

Uses for Digital Certificates in Internet Applications

Internet applications that use public-key cryptography systems must use digital certificates to obtain the public keys. There are many applications that use public-key cryptography, including the following ones:

Virtual Private Networks (VPN)

Virtual Private Networks, also called *secure tunnels*, can be set up between systems such as firewalls to enable protected connections between secure networks over unsecured communication links. All traffic destined to these networks is encrypted between the participating systems.

The protocols used in tunneling follow the IP Security and IKE standards, which allow for a secure, encrypted connection between a remote client (for example, an employee working from home) and a secure host or network.

Secure Sockets Layer (SSL)

SSL is a protocol that provides privacy and integrity for communications. It is used by Web servers for secure connections between Web servers and Web browsers, by the Lightweight Directory Access Protocol (LDAP) for secure connections between LDAP clients and LDAP servers, and by Host-on-Demand V.2 for connections between the client and the host system. SSL uses digital certificates for key exchange, server authentication, and, optionally, client authentication.

Secure Electronic Mail

Many electronic mail systems, using standards such as PEM or S/MIME for secure electronic mail, use digital certificates for digital signatures and for the exchange of keys to encrypt and decrypt mail messages.

Digital Certificates and Certificate Requests

Simplified, a signed digital certificate contains fields for the owner's distinguished name, the owner's public key, the CA's distinguished name and the CA's signature. A self-signed digital certificate contains its owner's distinguished name, public key, and signature.

A *certificate request* must be created and sent to a CA to request a digital certificate. The certificate request contains fields for the requestor's distinguished name, public key, and signature. The CA verifies the requestor's signature with the public key in the digital certificate to ensure that:

- The certificate request was not modified in transit between the requestor and the CA.
- The requestor possesses the corresponding private key for the public key that is in the certificate request.

The CA is also responsible for verifying to some level the identity of the requestor. Requirements for this verification can range from very little proof to absolute assurance of the owner's identity.

Using the IBM Key Manager Tool

In AIX 4.3.3 and later, the IBM Key Manager tool manages digital certificates. The IBM Key Manager tool is installed when you install the **gskit.rte** file set from the installation Bonus Pack.

This section describes how to use IBM Key Manager to do the following tasks:

1. Creating a Key Database
2. Adding a CA Root Digital Certificate
3. Establishing Trust Settings
4. Deleting a CA Root Digital Certificate
5. Requesting a Digital Certificate
6. Adding (Receiving) a New Digital Certificate
7. Deleting a Digital Certificate

8. Changing a Database Password
9. Creating IKE Tunnels using Digital Certificates

To set up digital certificates and signature support, at minimum you must do steps 1, 2, 3, 4, 6, and 7. Then, use Web-based System Manager to create an IKE tunnel and associate a policy with the tunnel that uses RSA Signature as the authentication method.

Creating a Key Database

A key database enables VPN endpoints to connect using valid digital certificates. The key database (*.kdb) format is used with IP Security VPNs.

The following types of CA digital certificates are provided with IBM Key Manager:

- RSA Secure Server Certification Authority
- Thawte Personal Premium Certification Authority
- Thawte Personal Freemail Certification Authority
- Thawte Personal Basic Certification Authority
- Thawte Personal Server Certification Authority
- Thawte Server Certification Authority
- Verisign Class 1 Public Primary Certification Authority
- Verisign Class 2 Public Primary Certification Authority
- Verisign Class 3 Public Primary Certification Authority
- Verisign Class 4 Public Primary Certification Authority

These signature digital certificates enable clients to attach to servers that have valid digital certificates from these signers. After you create a key database, you can use it as-is to attach to a server that has a valid digital certificate from one of the signers.

If you need to use a signature digital certificate that is not on this list, you must request it from the CA and add it to your key database (see Adding a CA Root Digital Certificate).

Use the following procedure to create a key database:

1. Start the IBM Key Manager tool by typing:
`certmgr`
2. Select **New** from the Key Database File pull down menu.
3. Accept the default value, **CMS key database file**, for the Key database type field.
4. Enter the following file name in the File Name field:
`ikekey.kdb`
5. Enter the following location of the database in the Location field:
`/etc/security`

Attention: The key database must be named **ikekey.kdb** and it must be placed in the **/etc/security** directory or IP Security cannot function correctly.

6. Click OK. The **Password Prompt** screen is displayed.
7. Enter a password in the Password field, and enter it again in the Confirm Password field.
8. If you want to change the number of days until the password expires, enter the desired number of days in the Set expiration time? field. The default value for this field is 60 days. If you do not want the password to expire, clear the Set expiration time? field.
9. To save an encrypted version of the password in a stash file, select the Stash the password to a file? field and enter **yes**.

- Note:** You must stash the password to enable the use of digital certificates with IP Security.
- Click OK. A confirmation screen is displayed, verifying that you have created a key database.
 - Click OK again and you return to the IBM Key Management screen. You can either perform other tasks or exit the tool.

Adding a CA Root Digital Certificate

After you have requested and received a root digital certificate from a CA, you can add it to your database. Most root digital certificates are of the form *.arm, such as the following:

```
cert.arm
```

Use the following procedure to add a CA root digital certificate to a database:

- Unless you are already using IBM Key Manager, start the tool by typing:

```
certmgr
```
- From the main screen, select **Open** from the Key Database File pull down menu.
- Highlight the key database file to which you want to add a CA root digital certificate and click **Open**.
- Enter the password and click **OK**. When your password is accepted, you are returned to the IBM Key Management screen. The title bar now shows the name of the key database file you selected, indicating that the file is now open and ready to be worked with.
- Select **Signer Certificates** from the Personal/Signer Certificates pull down menu.
- Click on **Add**.
- Select a data type from the Data type pull down menu, such as:

```
Base64-encoded ASCII data
```
- Enter a certificate file name and location for the CA root digital certificate, or click **Browse** to select the name and location.
- Click **OK**.
- Enter a label for the CA root digital certificate, such as Verisign Test CA Root Certificate, and click **OK**. You are returned to the IBM Key Management screen. The Signer Certificates field now shows the label of the CA root digital certificate you just added. You can either perform more tasks or exit the tool.

Establishing Trust Settings

Installed CA certificates are set to **trusted** by default. The procedure to change the trust setting follows.

- Unless you are already using IBM Key Manager, start the tool by typing:

```
certmgr
```
- From the main screen, select **Open** from the Key Database File pull down menu.
- Highlight the key database file in which you want to change the default digital certificate and click **Open**.
- Enter the password and click OK. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open.
- Select **Signer Certificates** from the Personal/Signer Certificates pull down menu.
- Highlight the certificate you want to change and click **View/Edit**, or double-click on the entry. The Key Information screen is displayed for the certificate entry.
- To make this certificate a trusted root certificate, check the box next to Set the certificate as a trusted root and click **OK**. If the certificate is not trusted, clear the check box instead and click **OK**.
- Click **OK** from the Signer Certificates screen. You are returned to the IBM Key Management screen. You can either perform other tasks or exit the tool.

Deleting a CA Root Digital Certificate

If you no longer want to support one of the CAs in your signature digital certificate list, you need to delete the CA root digital certificate.

Note: Before deleting a CA root digital certificate, create a backup copy in case you later want to recreate the CA root.

Use the following procedure to delete a CA root digital certificate from a database:

1. Unless you are already using IBM Key Manager, start the tool by typing:
certmgr
2. From the main screen, select **Open** from the Key Database File pull down menu.
3. Highlight the key database file from which you want to delete a CA root digital certificate and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the **IBM Key Management** screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Signer Certificates** from the Personal/Signer Certificates pull down menu.
6. Highlight the certificate you want to delete and click **Delete**. The Confirm screen is displayed.
7. Click **Yes**. You are returned to the IBM Key Management screen. The label of the CA root digital certificate no longer appears in the Signer Certificates field. You can either perform other tasks or exit the tool.

Requesting a Digital Certificate

To acquire a digital certificate, generate a request using IBM Key Manager and submit the request to a CA. The request file you generate is in the PKCS#10 format. The CA then verifies your identity and sends you a digital certificate.

Use the following procedure to request a digital certificate:

1. Unless you are already using IBM Key Manager, start the tool by typing:
certmgr
2. From the main screen, select **Open** from the Key Database File pull down menu.
3. Highlight the key database file **/etc/security/ikekey.kdb** from which you want to generate the request and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Personal Certificate Requests** from the Personal/Signer Certificates pull down menu (AIX Version 4) or select **Create** —> **New Certificate Request** (SWsym.Version500;).
6. Click **New**.
7. From the following screen, enter a **Key Label** for the self-signed digital certificate, such as:
keytest
8. Enter a **Common Name** (the default is the host name) and **Organization**, and then select a **Country**. For the remaining fields, either accept the default values, or choose new values.
9. Define the **Subject Alternate** name. There are three optional fields associated with **Subject Alternate**: email address, IP address, and DNS name. For a tunnel type of IP address, type the same IP address that is configured in the IKE tunnel into the IP address field. For a tunnel ID type of user@FQDN, complete the email address field. For a tunnel ID type of FQDN, type a fully qualified domain name (for example, *hostname.companyname.com*) in the DNS name field.
10. At the bottom of the screen, enter a name for the file, such as:

certreq.arm

11. Click **OK**. A confirmation screen is displayed, verifying that you have created a request for a new digital certificate.
12. Click **OK**. You are returned to the IBM Key Management screen. The **Personal Certificate Requests** field now shows the key label of the new digital certificate request (PKCS#10) created.
13. Send the file to a CA to request a new digital certificate.
14. At this point, you can either perform other tasks or exit the tool.

Adding (Receiving) a New Digital Certificate

After you receive a new digital certificate from a CA, you must add it to the key database from which you generated the request.

Use the following procedure to add (receive) a new digital certificate:

1. Unless you are already using IBM Key Manager, start the tool by typing:
certmgr
2. From the main screen, select **Open** from the Key Database File pull down menu.
3. Highlight the key database file from which you generated the certificate request and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Personal Certificate Requests** from the Personal/Signer Certificates pull down menu.
6. Click **Receive** (to add the newly received digital certificate to your database).
7. Select the data type of the new digital certificate from the **Data type** pull down menu. The default is **Base64-encoded ASCII data**.
8. Enter the certificate file name and location for the new digital certificate, or click **Browse** to select the name and location.
9. Click **OK**.
10. Enter a descriptive label for the new digital certificate, such as:
VPN Branch Certificate
11. Click **OK**. You are returned to the **IBM Key Management** screen. The **Personal Certificates** field now shows the label of the new digital certificate you just added.

If the procedure is successful, you can either perform other tasks or exit the tool.

If there is an error loading the certificate, check that the certificate file begins with the text `-----BEGIN CERTIFICATE-----` and ends with the text `-----END CERTIFICATE-----`.

For example:

```
-----BEGIN CERTIFICATE-----  
ajdkfjaldfwwwwwwwwadafdw  
kajf;kdsajkf1asasfkjafdaff  
akdjf;ldasjkf;safdfdasfdas  
kaj;fdljk98dafdas43adfadfa  
-----END CERTIFICATE-----
```

If the text does not match, edit the certificate file so it starts and ends appropriately.

Deleting a Digital Certificate

If you no longer need one of your digital certificates, use the following procedure to delete it from your database.

Note: Before deleting a digital certificate, create a backup copy in case you later want to recreate it.

1. Unless you are already using IBM Key Manager, start the tool by typing:
certmgr
 2. From the main screen, select **Open** from the Key Database File pull down menu.
 3. Highlight the key database file from which you want to delete the digital certificate and click **Open**.
 4. Enter the password and click **OK**. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
 5. Select **Personal Certificate Requests** from the Personal/Signer Certificates pull down menu.
 6. Highlight the digital certificate you want to delete and click **Delete**. The Confirm screen is displayed.
 7. Click **Yes**. You are returned to the IBM Key Management screen. The label of the digital certificate you just deleted no longer appears in the **Personal Certificates** field.
- You can either perform other tasks or exit the tool.

Changing a Database Password

To change the key database, follow these steps:

1. Unless you are already using IBM Key Manager, start the tool by typing:
certmgr
2. From the main screen, select **Change Password** from the Key Database File pull down menu.
3. Enter a new password in the **Password** field, and enter it again in the **Confirm Password** field.
4. If you want to change the number of days until the password expires, enter the desired number of days in the Set expiration time? field. The default value for this field is 60 days. If you do not want the password to expire, clear the Set expiration time? field.
5. To save an encrypted version of the password in a stash file, select the Stash the password to a file? field and enter **yes**.

Note: You must stash the password to enable the use of digital certificates with IP Security.

6. Click OK. A message in the status bar indicates that the request completed successfully.
7. Click OK again and you return to the IBM Key Management screen. You can either perform other tasks or exit the tool.

Creating IKE Tunnels using Digital Certificates

To create IKE tunnels that use digital certificates, you must use Web-based System Manager and the IBM Key Manager tool.

To enable the use of digital certificates when defining the key management IKE tunnel policies, you must configure a transform that uses signature mode. *Signature mode* uses an RSA signature algorithm for authentication. IP Security provides the Web-based System Manager dialog "Add/Change a Transform" to allow you to select an authentication method of RSA Signature or RSA Signature with CRL Checking.

At least one endpoint of the tunnel must have a policy defined that uses a signature mode transform. You can also define other transforms using signature mode through Web-based System Manager.

The IKE key management tunnel types (the **Host Identity Type** field on the Identification tab) supported by IP Security are:

- IP address
- Fully Qualified Domain Name (FQDN)
- *user@FQDN*
- X.500 Distinguished Name

- Key identifier

Host identity types are selectable from the **Web-based System Manager** Key Management Tunnel Properties - Identification tab. If you select **IP Address**, **FQDN**, or **user@FQDN**, you must enter values in Web-based System Manager and you must give these values to the CA. This information is used as the Subject Alternate Name in the personal digital certificate.

For example, if you choose a host identity type of **X.500 Distinguished Name** from the Web-based System Manager pull-down list on the **Identification** tab, and you enter the **Host identity** as **/C=US/O=IBM/OU=SERV/CN=name.austin.ibm.com**, the following are the exact values that you must enter in IBM Key Manager when creating a digital certificate request:

- Common name: **name.austin.ibm.com**
- Organization: **IBM**
- Organizational unit: **SERV**
- Country : **US**

The **X.500 Distinguished Name** entered should be the name set up by your system/LDAP administrator. Entering an organizational unit value is optional. The CA then uses this information when creating the digital certificate.

For another example, if you choose a host identity type of **IP Address** from the pull-down list, and you enter the host identity as **10.10.10.1**, the following are the exact values you must enter in the digital certificate request:

- Common name: **name.austin.ibm.com**
- Organization: **IBM**
- Organizational unit: **SERV**
- Country : **US**
- Subject alternate IP address field: **10.10.10.1**

After you create the digital certificate request with this information, the CA uses this information to create the personal digital certificate.

When requesting a personal digital certificate, the CA needs the following information:

- That you are requesting a X.509 certificate.
- That the signature format is MD5 with RSA encryption.
- Whether you are specifying Subject Alternate Name. Alternate name types are:
 - IP address
 - Fully qualified domain name (FQDN)
 - *user@FQDN*

The subject alternate name information is included in the certificate request file.

- Your planned key use (the digital signature bit has to be selected).
- The IBM Key Manager digital certificate request file (in PKCS#10 format).

See Requesting a Digital Certificate for specific steps using the IBM Key Manager tool to create a certificate request.

Before activating the IKE tunnel, you must add the personal digital certificate you received from the CA into the IBM Key Manager database, **ikekey.kdb**. See Adding (Receiving) a New Digital Certificate for more information.

The types of personal digital certificate that IP Security supports are:

Subject DN

The Subject Distinguished Name must be in the following format and order:

```
/C=US/O=IBM/OU=SERV/CN=name.austin.ibm.com
```

The IBM Key Manager tool allows only one **OU** value.

Subject DN and Subject Alternate Name as an IP address

The Subject Distinguished Name and Subject Alternate Name can be designated as an IP address, as shown in the following:

```
/C=US/O=IBM/OU=SERV/CN=name.austin.ibm.com and 10.10.10.1
```

Subject DN and Subject Alternate Name as FQDN

The Subject Distinguished Name and Subject Alternate Name can be designated as a fully qualified domain name, as shown in the following:

```
/C=US/O=IBM/OU=SERV/CN=name.austin.ibm.com and bell.austin.ibm.com.
```

Subject DN and Subject Alternate Name as *user@FQDN*

The Subject Distinguished Name and Subject Alternate Name can be designated as a user address (*user_ID@fully_qualified_domain_name*), as shown in the following:

```
/C=US/O=IBM/OU=SERV/CN=name.austin.ibm.com and name@austin.ibm.com.
```

Subject DN and multiple Subject Alternate Names

The Subject Distinguished Name can be associated with multiple Subject Alternate Names, as shown in the following:

```
/C=US/O=IBM/OU=SERV/CN=name.austin.ibm.com and bell.austin.ibm.com, 10.10.10.1, and user@name.austin.ibm.com.
```

Configuring Manual Tunnels

The following procedures configure IP Security to use manual tunnels.

Setting Up Tunnels and Filters

To set up a manual tunnel, it is not necessary to separately configure the filter rules. As long as all traffic between two hosts goes through the tunnel, the necessary filter rules are automatically generated. The process of setting up a tunnel is to define the tunnel on one end, import the definition on the other end, and activate the tunnel and filter rules on both ends. Then the tunnel is ready to use.

Information about the tunnel must be made to match on both sides if it is not explicitly supplied.

For instance, the encryption and authentication algorithms specified for the source will be used for the destination if the destination values are not specified. This makes creating the tunnel much simpler.

Creating a Manual Tunnel on Host A

You can configure a tunnel using the Web-based System Manager Network application, the SMIT fast path **ips4_basic** (for IP Version 4) or **ips6_basic** (for IP version 6), or you can use the following procedure.

The following is a sample of the **gentun** command used to create a manual tunnel:

```
gentun -v 4 -t manual -s 5.5.5.19 -d 5.5.5.8 \  
-a HMAC_MD5 -e DES_CBC_8 -N 23567
```

You can use the **lstun -v 4** command to list the characteristics of the manual tunnel created by the example above. The output looks similar to the following:

```
Tunnel ID      : 1  
IP Version     : IP Version 4  
Source        : 5.5.5.19
```

```

Destination      : 5.5.5.8
Policy           : auth/encr
Tunnel Mode     : Tunnel
Send AH Algo    : HMAC_MD5
Send ESP Algo   : DES_CBC_8
Receive AH Algo : HMAC_MD5
Receive ESP Algo : DES_CBC_8
Source AH SPI   : 300
Source ESP SPI  : 300
Dest AH SPI     : 23576
Dest ESP SPI    : 23576
Tunnel Life Time : 480
Status          : Inactive
Target          : -
Target Mask     : -
Replay         : No
New Header      : Yes
Snd ENC-MAC Algo : -
Rcv ENC-MAC Algo : -

```

The tunnel is activated when the following command is used:

```
mktun -v 4 -t1
```

The filter rules associated with the tunnel are automatically generated and output (using **lsfilt -v 4**) looks similar to the following:

```

Rule 4:
Rule action      : permit
Source Address   : 5.5.5.19
Source Mask      : 255.255.255.255
Destination Address : 5.5.5.8
Destination Mask : 255.255.255.255
Source Routing   : yes
Protocol         : all
Source Port      : any 0
Destination Port : any 0
Scope           : both
Direction       : outbound
Logging control  : no
Fragment control : all packets
Tunnel ID number : 1
Interface       : all
Auto-Generated  : yes

```

```

Rule 5:
Rule action      : permit
Source Address   : 5.5.5.8
Source Mask      : 255.255.255.255
Destination Address : 5.5.5.19
Destination Mask : 255.255.255.255
Source Routing   : yes
Protocol         : all
Source Port      : any 0
Destination Port : any 0
Scope           : both
Direction       : inbound
Logging control  : no
Fragment control : all packets
Tunnel ID number : 1
Interface       : all
Auto-Generated  : yes

```

These filter rules, in addition to the default filter rules, are activated by the **mktun -v 4 -t 1** command.

To set up the other side (when it is another machine using this operating system), the tunnel definition can be exported on host A then imported to host B.

The following command:

```
exptun -v 4 -t 1 -f /tmp
```

exports the tunnel definition into a file named **ipsec_tun_manu.exp** and any associated filter rules to the file **ipsec_filtr_rule.exp** in the directory indicated by the **-f** flag.

Creating a Manual Tunnel on Host B

To create the matching end of the tunnel, the export files are copied and imported into the remote machine by using the following command:

```
imptun -v 4 -t 1 -f /tmp
```

where

1 Is the tunnel to be imported

/tmp Is the directory where the import files reside

The tunnel number is system generated. You can obtain it from the output of the **gentun** command or by using the **lstun** command to list the tunnels and determine the correct tunnel number to import. If there is only one tunnel in the import file, or if all the tunnels are to be imported, then the **-t** option is not needed.

If the remote machine is not running this operating system, the export file can be used as a reference for setting up the algorithm, keys, and security parameters index (SPI) values for the other end of the tunnel.

Export files from an IBM firewall product can be imported to create tunnels. To do this, use the **-n** option when importing the file, as shown below:

```
imptun -v 4 -f /tmp -n
```

Setting Up Filters

Filtering can be set up to be simple, using mostly autogenerated filter rules, or can be customized by defining very specific filter functions based on the properties of the IP packets. Matches on incoming packets are done by comparing the source address and SPI value to those listed in the filter table. Therefore, this pair must be unique.

Each line in a filter table is known as a *rule*. A collection of rules determine what packets are accepted in and out of the machine and how they are directed. Filter rules can be control many aspects of communications, including source and destination addresses and masks, protocol, port number, direction, fragment control, source routing, tunnel, and interface type.

There are different types of filter rules:

- *Static filter rules* are created in the filter table to be used for the general filtering of traffic or for associating with manual tunnels. They can be added, deleted, modified, and moved around.
- *Dynamic filter rules* (also called *autogenerated* filter rules) are a specific set of rules created for use of IKE tunnels. Both static and dynamic filter rules are created based on data management tunnel information and on data management tunnel negotiation.
- *Predefined filter rules* are generic filter rules that cannot be modified, moved, or deleted, such as the all traffic rule, the ah rule, and the esp rule. They pertain to all traffic.

Associated with these filter rules are *Subnet masks*, which group IDs that are associated with a filter rule, and the host-firewall-host configuration option. The following sections describe the different types of filter rules and their associated features.

Static Filter Rules and Examples

The following example of static filter rules is further explained in the paragraphs that follow it. Within each rule, fields are shown in the following order (an example of each field from rule 1 is shown in parentheses): Rule_number (1), Action (permit), Source_addr (0.0.0.0), Source_mask (0.0.0.0), Dest_addr (0.0.0.0), Dest_mask (0.0.0.0), Source_routing (no), Protocol (udp), Src_prt_operator (eq), Src_prt_value (4001), Dst_prt_operator (eq), Dst_prt_value (4001), Scope (both), Direction (both), Logging (no), Fragment (all packets), Tunnel (0), and Interface (all).

```
1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both both no all
   packets 0 all

2 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no ah any 0 any 0 both both no all packets
   0 all

3 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no esp any 0 any 0 both both no all packets
   0 all

4 permit 10.0.0.1 255.255.255.255 10.0.0.2 255.255.255.255 no all any 0 any 0 both
   outbound no all packets 1 all

5 permit 10.0.0.2 255.255.255.255 10.0.0.1 255.255.255.255 no all any 0 any 0 both
   inbound no all packets 1 all

6 permit 10.0.0.1 255.255.255.255 10.0.0.3 255.255.255.255 no tcp lt 1024 eq 514 local
   outbound yes all packets 2 all

7 permit 10.0.0.3 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack eq 514 lt 1024
   local inbound yes all packets 2 all

8 permit 10.0.0.1 255.255.255.255 10.0.0.3 255.255.255.255 no tcp/ack lt 1024 lt 1024
   local outbound yes all packets 2 all

9 permit 10.0.0.3 255.255.255.255 10.0.0.1 255.255.255.255 no tcp lt 1024 lt 1024 local
   inbound yes all packets 2 all

10 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 0 any 0 local
   outbound yes all packets 3 all

11 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 0 any 0 local
   inbound yes all packets 3 all

12 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp gt 1023 eq 21 local
   outbound yes all packets 4 all

13 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack eq 21 gt 1023 local
   inbound yes all packets 4 all

14 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp eq 20 gt 1023 local
   inbound yes all packets 4 all

15 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp/ack gt 1023 eq 20 local
   outbound yes all packets 4 all

16 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp gt 1023 gt 1023 local
   outbound yes all packets 4 all
```

```
17 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack gt 1023 gt 1023 local
inbound yes all packets 4 all
```

```
18 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no all any 0 any 0 both both yes all
packets
```

The following paragraphs explain each rule in the example.

Rule 1

For the IBM Session Key daemon. This rule only appears in IP Version 4 filter tables. It uses port number 4001 to control packets for refreshing the session key. It is an example of how the port number can be used for a specific purpose.

Note:This filter rule should not be modified except for logging purposes.

Rules 2 and 3

Allow processing of Authentication Headers (AH) and Encapsulating Security Payload (ESP) headers.

Note:Rules 2 and 3 should not be modified except for logging purposes.

Rules 4 and 5

Set of autogenerated rules that filter traffic between addresses 10.0.0.1 and 10.0.0.2 through tunnel #1. Rule 4 is for outbound traffic and rule 5 is for inbound traffic.

Rules 6 through 9

Set of user-defined rules that filter outbound **rsh**, **rnp**, **rdump**, **rrestore**, and **rdist** services between addresses 10.0.0.1 and 10.0.0.3 through tunnel #2. Note that in this example, logging is set to yes so the administrator can monitor this type of traffic.

Rules 10 and 11

Set of user-defined rules that filter both inbound and outbound **icmp** services of any type between addresses 10.0.0.1 and 10.0.0.4 through tunnel #3.

Rules 12 through 17

User-defined filter rules that filter outbound file transfer protocol (FTP) service from 10.0.0.1 and 10.0.0.5 through tunnel #4.

Rule 18

Autogenerated rule always placed at the end of the table. In this example, it permits all packets that do not match the other filter rules. It can be set to deny all traffic not matching the other filter rules.

Each rule can be viewed separately (using **lsfilt**) to list each field with its value. For example:

```
Rule 1:
Rule action      : permit
Source Address   : 0.0.0.0
Source Mask      : 0.0.0.0
Destination Address : 0.0.0.0
Destination Mask : 0.0.0.0
Source Routing   : yes
Protocol         : udp
Source Port      : eq 4001
Destination Port : eq 4001
Scope           : both
Direction       : both
Logging control  : no
```



```
Fragment control      : all packets
Tunnel ID number     : 0
Interface             : all
Auto-Generated       : yes
```

The following list contains all the parameters that can be specified in a filter rule:

```
-v      IP version: 4 or 6.
-a      Action:
        d      Deny
        p      Permit

-s      Source address. Can be an IP address or hostname.
-m      Source subnet mask.
-d      Destination address. Can be an IP address or hostname.
-M      Destination subnet mask.
-g      Source routing control: y or n.
-c      Protocol. Values can be udp, icmp, tcp, tcp/ack, ospf, pip, esp, ah and all.
-o      Source port or ICMP type operation.
-p      Source port or ICMP type value.
-O      Destination port or ICMP code operation.
-P      Destination port or ICMP code value.
-r      Routing:
        r      Forwarded packets
        l      Local destined/originated packets
        b      Both

-l      Log control.
        y      Include in log
        n      Do not include in log.

-f      Fragmentation.
        y      Applies to fragments headers, fragments, and non-fragments
        o      Applies only to fragments and fragment headers
        n      Applies only to non-fragments
        h      Applies only to non-fragments and fragment headers

-t      Tunnel ID.
-i      Interface, such as tr0 or en0.
```

See the **genfilt** and **chfilt** command descriptions for more information.

Autogenerated Filter Rules and User Specified Filter Rules

Certain rules are autogenerated for the use of the IP Security filter and tunnel code. Autogenerated rules include:

- Rules for the session key daemon that refresh the IP version 4 keys in IKE (AIX 4.3.2 and later)
- Rules for the processing of AH and ESP packets.

Filter rules are also autogenerated when defining tunnels. For manual tunnels, autogenerated rules specify the source and destination addresses and the mask values, as well as the tunnel ID. All traffic between those addresses will flow through the tunnel.

For IKE tunnels, autogenerated filter rules determine protocol and port numbers during IKE negotiation. The IKE filter rules are kept in a separate table, which is searched after the static filter rules and before the autogenerated rules. IKE filter rules are inserted in a default position within the static filter table, but they can be moved by the user.

Autogenerated rules permit all traffic over the tunnel. User-defined rules can place restrictions on certain types of traffic. These user-defined rules should be placed before the autogenerated rules, because IP Security uses the first rule it finds that applies to the packet. Below is an example of user-defined filter rules that filter traffic based on ICMP operation.

```
1 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 8 any 0
   local outbound no all packets 3 all
2 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 0 any 0 local
   inbound no all packets 3 all
3 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 8 any 0 local
   inbound no all packets 3 all
4 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 0 any 0 local
   outbound no all packets 3 all
```

To simplify the configuration of a single tunnel, filter rules are autogenerated when tunnels are defined. This function can be suppressed by specifying the **-g** flag in the **gentun**. You can find a sample filter file with **genfilt** commands to generate filter rules for different TCP/IP services in **/usr/samples/ipsec/filter.sample**.

Predefined Filter Rules

There are several predefined filter rules that are autogenerated with certain events. When the `ipsec_v4` or `ipsec_v6` device is loaded, a predefined rule is inserted into the filter table and then activated. By default, this predefined rule is to permit all packets, but it is user configurable and you can set it to deny all packets.

Note: When configuring remotely, ensure that the deny rule is not enabled before the configuration is complete. This will keep your session from getting shut out of the machine. The situation can be avoided either by setting the default action to permit or by configuring a tunnel to the remote machine before activating IP Security.

There is a predefined rule for both IPv4 and IPv6 filter tables. Either may be independently changed to deny all. This will keep traffic from passing unless that traffic is specifically defined by additional filter rules. The only other option to change on the predefined rules is **chfilt** with the **-l** option, which allows packets matching that rule to be logged.

To support IKE tunnels, a dynamic filter rule is placed in the IPv4 filter table. This is the position at which dynamic filter rules are inserted into the filter table. This position can be controlled by the user by moving its position up and down the filter table. Once the tunnel manager daemon and **isakmpd** daemon are initialized that will allow IKE tunnels to be negotiated, rules are automatically created in the dynamic filter table to handle IKE messages as well as AH and ESP packets.

Subnet Masks

Subnet masks are used to group a set of IDs that are associated with a filter rule. The mask value is ANDed with the ID in the filter rules and compared to the ID specified in the packet. For example, a filter rule with a source IP address of 10.10.10.4 and a subnet mask of 255.255.255.255 specified that an exact match must occur of the decimal IP address, as shown in the following:

	Binary	Decimal
Source IP address	1010.1010.1010.0100	10.10.10.4
Subnet mask	1111.1111.1111.1111	255.255.255.255

A 10.10.10.x subnet is specified as 1111.1111.1111.0 or 255.255.255.0. An incoming address would have the subnet mask appended to it, then the combination would be compared to the ID in the filter rule. For example, an address of 10.10.10.100 becomes 10.10.10.0 after the subnet mask is applied, which matches the filter rule.

A subnet mask of 255.255.255.240 allows any value for the last four bits in the address.

Host-Firewall-Host

The host-firewall-host configuration option for tunnels allows you to create a tunnel between your host and a firewall, then automatically generate the necessary filter rules for correct communication between your host and a host behind the firewall. The autogenerated filter rules permit all rules between the two non-firewall hosts over the tunnel specified. The default rules—for user datagram protocol (UDP), Authentication Headers (AH), and Encapsulating Security Payload (ESP) headers—should already handle the host to firewall communication. The firewall will have to be configured appropriately to complete the setup. You should use the export file from the tunnel you created to enter the SPI values and keys that the firewall needs.

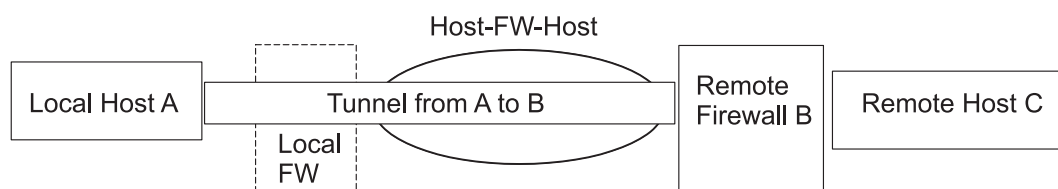


Figure 33. Host — Firewall — Host. This illustration shows a Host — Firewall — Host configuration. Host A has a tunnel running through a local firewall and out to the internet. Then it goes to Remote Firewall B, and then on to Remote Host C.

Logging Facilities

This section describes the configuration and format of system logs relating to IP Security. As hosts communicate with each other, the transferred packets may be logged to the system log daemon, **syslogd**. Other important messages about IP Security will appear as well. An administrator may choose to monitor this logging information for traffic analysis and debugging assistance. The following are the steps for setting up the logging facilities.

1. Edit the **/etc/syslog.conf** file to add the following entry:

```
local4.debug var/adm/ipsec.log
```

Use the `local4` facility to record traffic and IP Security events. Standard operating system priority levels apply. You should set the priority level of debug until traffic through IP Security tunnels and filters show stability and proper movement.

Note: The logging of filter events can create significant activity at the IP Security host and can consume large amounts of storage.

2. Save **/etc/syslog.conf**.
3. Go to the directory you specified for the log file and create an empty file with the same name. In the case above, you would change to `/var/adm` directory and issue the command:

```
touch ipsec.log
```

4. Issue a **refresh** command to the **syslogd** subsystem:

```
refresh -s syslogd
```

5. If using IKE tunnels, ensure the **/etc/isakmpd.conf** file specifies the desired **isakmpd** logging level. (See “IP Security Problem Determination” on page 255 for more information on IKE logging.)

6. While creating filter rules for your host, if you would like packets matching a specific rule to be logged, set the **-l** parameter for the rule to **Y** (yes) using the **genfilt** or the **chfilt** commands.
7. Finally, turn on packet logging and start the **ipsec_logd** daemon using the following command:

```
mkfilt -g start
```

You can stop packet logging by issuing the following command:

```
mkfilt -g stop
```

The sample log file below contains traffic entries and other IP Security log entries:

1. Aug 27 08:08:40 host1 : Filter logging daemon ipsec_logd (level 2.20) initialized at 08:08:40 on 08/27/97A
2. Aug 27 08:08:46 host1 : mkfilt: Status of packet logging set to Start at 08:08:46 on 08/27/97
3. Aug 27 08:08:47 host1 : mktun: Manual tunnel 2 for IPv4, 9.3.97.244, 9.3.97.130 activated.
4. Aug 27 08:08:47 host1 : mkfilt: #:1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp eq 4001 eq 4001 both both l=n f=y t=0 e= a=
5. Aug 27 08:08:47 host1 : mkfilt: #:2 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 ah any 0 any 0 both both l=n f=y t=0 e= a=
6. Aug 27 08:08:47 host1 : mkfilt: #:3 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 esp any 0 any 0 both both l=n f=y t=0 e= a=
7. Aug 27 08:08:47 host1 : mkfilt: #:4 permit 10.0.0.1 255.255.255.255 10.0.0.2 255.255.255.255 icmp any 0 any 0 local outbound l=y f=y t=1 e= a=
8. Aug 27 08:08:47 host1 : mkfilt: #:4 permit 10.0.0.2 255.255.255.255 10.0.0.1 255.255.255.255 icmp any 0 any 0 local inbound l=y f=y t=1 e= a=
9. Aug 27 08:08:47 host1 : mkfilt: #:6 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 any 0 both both l=y f=y t=0 e= a=
10. Aug 27 08:08:47 host1 : mkfilt: Filter support (level 1.00) initialized at 08:08:47 on 08/27/97
11. Aug 27 08:08:48 host1 : #:6 R:p o:10.0.0.1 s:10.0.0.1 d:10.0.0.20 p:udp sp:3327 dp:53 r:l a:n f:n T:0 e:n l:67
12. Aug 27 08:08:48 host1 : #:6 R:p i:10.0.0.1 s:10.0.0.20 d:10.0.0.1 p:udp sp:53 dp:3327 r:l a:n f:n T:0 e:n l:133
13. Aug 27 08:08:48 host1 : #:6 R:p i:10.0.0.1 s:10.0.0.15 d:10.0.0.1 p:tcp sp:4649 dp:23 r:l a:n f:n T:0 e:n l:43
14. Aug 27 08:08:48 host1 : #:6 R:p o:10.0.0.1 s:10.0.0.1 d:10.0.0.15 p:tcp sp:23 dp:4649 r:l a:n f:n T:0 e:n l:41
15. Aug 27 08:08:48 host1 : #:6 R:p i:10.0.0.1 s:10.0.0.15 d:10.0.0.1 p:tcp sp:4649 dp:23 r:l a:n f:n T:0 e:n l:40
16. Aug 27 08:08:51 host1 : #:4 R:p o:10.0.0.1 s:10.0.0.1 d:10.0.0.2 p:icmp t:8 c:0 r:l a:n f:n T:1 e:n l:84
17. Aug 27 08:08:51 host1 : #:5 R:p i:10.0.0.1 s:10.0.0.2 d:10.0.0.1 p:icmp t:0 c:0 r:l a:n f:n T:1 e:n l:84
18. Aug 27 08:08:52 host1 : #:4 R:p o:10.0.0.1 s:10.0.0.1 d:10.0.0.2 p:icmp t:8 c:0 r:l a:n f:n T:1 e:n l:84
19. Aug 27 08:08:52 host1 : #:5 R:p i:10.0.0.1 s:10.0.0.2 d:10.0.0.1 p:icmp t:0 c:0 r:l a:n f:n T:1 e:n l:84
20. Aug 27 08:32:27 host1 : Filter logging daemon terminating at 08:32:27 on 08/27/971

The following paragraphs explain the log entries.

- 1 Filter logging daemon activated.
- 2 Filter packet logging set to on with **mkfilt -g start**.
- 3 Tunnel activation, showing tunnel ID, source address, destination address, and time stamp.
- 4-9 Filters have been activated. Logging shows all loaded filter rules.
- 10 Message showing activation of filters.
- 11-12 These entries show a DNS lookup for a host.

13-15 These entries show a partial Telnet connection (the others have been removed from this example for space reasons).

16-19 These entries show two pings.

20 Filter logging daemon shutting down.

The following example shows two hosts negotiating a phase 1 and a phase 2 tunnel from the initiating host's point of view. (The **isakmpd** logging level has been specified as **isakmp_events**.)

```
1. Dec 6 14:34:42 host1 Tunnel Manager: 0: TM is processing a
   Connection_request_msg
2. Dec 6 14:34:42 host1 Tunnel Manager: 1: Creating new P1 tunnel object (tid)
3. Dec 6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( SA PROPOSAL
   TRANSFORM )
4. Dec 6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 ( SA
   PROPOSAL TRANSFORM )
5. Dec 6 14:34:42 host1 isakmpd: Phase I SA Negotiated
6. Dec 6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( KE NONCE )
7. Dec 6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 ( KE
   NONCE )
8. Dec 6 14:34:42 host1 isakmpd: Encrypting the following msg to send: ( ID HASH
   )
9. Dec 6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
   Payloads )
10. Dec 6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 (
   Encrypted Payloads )
11. Dec 6 14:34:42 host1 Tunnel Manager: 1: TM is processing a P1_sa_created_msg
   (tid)
12. Dec 6 14:34:42 host1 Tunnel Manager: 1: Received good P1 SA, updating P1
   tunnel (tid)
13. Dec 6 14:34:42 host1 Tunnel Manager: 0: Checking to see if any P2 tunnels need
   to start
14. Dec 6 14:34:42 host1 isakmpd: Decrypted the following received msg: ( ID HASH
   )
15. Dec 6 14:34:42 host1 isakmpd: Phase I Done !!!
16. Dec 6 14:34:42 host1 isakmpd: Phase I negotiation authenticated
17. Dec 6 14:34:44 host1 Tunnel Manager: 0: TM is processing a
   Connection_request_msg
18. Dec 6 14:34:44 host1 Tunnel Manager: 0: Received a connection object for an
   active P1 tunnel
19. Dec 6 14:34:44 host1 Tunnel Manager: 1: Created blank P2 tunnel (tid)
20. Dec 6 14:34:44 host1 Tunnel Manager: 0: Checking to see if any P2 tunnels need
   to start
21. Dec 6 14:34:44 host1 Tunnel Manager: 1: Starting negotiations for P2 (P2 tid)
22. Dec 6 14:34:45 host1 isakmpd: Encrypting the following msg to send: ( HASH SA
   PROPOSAL TRANSFORM NONCE ID ID )
23. Dec 6 14:34:45 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
   Payloads )
24. Dec 6 14:34:45 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 (
   Encrypted Payloads )
25. Dec 6 14:34:45 host1 isakmpd: Decrypted the following received msg: ( HASH SA
   PROPOSAL TRANSFORM NONCE ID ID )
26. Dec 6 14:34:45 host1 isakmpd: Encrypting the following msg to send: ( HASH )
27. Dec 6 14:34:45 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
   Payloads )
28. Dec 6 14:34:45 host1 isakmpd: Phase II SA Negotiated
29. Dec 6 14:34:45 host1 isakmpd: PhaseII negotiation complete.
30. Dec 6 14:34:45 host1 Tunnel Manager: 0: TM is processing a P2_sa_created_msg
31. Dec 6 14:34:45 host1 Tunnel Manager: 1: received p2_sa_created for an existing
   tunnel as initiator (tid)
32. Dec 6 14:34:45 host1 Tunnel Manager: 1: Filter::AddFilterRules: Created filter
   rules for tunnel
33. Dec 6 14:34:45 host1 Tunnel Manager: 0: TM is processing a List_tunnels_msg
```

The following paragraphs explain the log entries.

- 1-2** The **ike cmd=activate phase=1** command initiates a connection.
- 3-10** The **isakmpd** daemon negotiates a phase 1 tunnel.
- 11-12** The Tunnel Manager receives a valid phase 1 security association from the responder.
- 13** The Tunnel Manager checks whether **ike cmd=activate** has a phase 2 value for more work. It does not.
- 14-16** The **isakmpd** daemon finishes the phase 1 negotiation.
- 17-21** The **ike cmd=activate phase=2** command initiates a phase 2 tunnel.
- 22-29** The **isakmpd** daemon negotiates a phase 2 tunnel.
- 30-31** The Tunnel Manager receives a valid phase 2 security association from responder.
- 32** The Tunnel Manager writes the dynamic filter rules.
- 33** The **ike cmd=list** command views the IKE tunnels.

Labels in Field Entries

The fields in the log entries are abbreviated to reduce DASD space requirements:

- #** The rule number that caused this packet to be logged.
- R** Rule Type.
 - p** Permit.
 - d** Deny.
- i/o** Direction the packet was traveling when it was intercepted by the filter support code. Identifies IP address of the adapter associated with the packet:
 - For inbound (i) packets, this is the adapter that the packet arrived on.
 - For outbound (o) packets, this is the adapter that the IP layer has determined should handle the transmission of the packet.
- s** Specifies the IP address of the sender of the packet (extracted from the IP header).
- d** Specifies the IP address of the intended recipient of the packet (extracted from the IP header).
- p** Specifies the high-level protocol that was used to create the message in the data portion of the packet. May be a number or name, for example: udp, icmp, tcp, tcp/ack, ospf, pip, esp, ah, or all.
- sp/t** Specifies the protocol port number associated with the sender of the packet (extracted from the TCP/UDP header). When the protocol is ICMP or OSPF, this field is replaced with **t**, which specifies the IP type.
- dp/c** Specifies the protocol port number associated with the intended recipient of the packet (extracted from the TCP/UDP header). When the protocol is ICMP, this field is replaced with **c** which specifies the IP code.
- Specifies that no information is available
- r** Indicates whether the packet had any local affiliation.
 - f** Forwarded packets
 - l** Local packets
 - o** Outgoing
 - b** Both
- l** Specifies the length of a particular packet in bytes.
- f** Identifies if the packet is a fragment.
- T** Indicates the tunnel ID.
- i** Specifies what interface the packet came in on.

IP Security Problem Determination

This section includes some hints and tips that may assist you when you encounter a problem. We recommend that you set up logging from the start. Logs are very useful in determining what is going on with the filters and tunnels. (See Logging Facilities for detailed log information.)

Troubleshooting Manual Tunnel Errors

Error: Issuing **mktun** command results in the following error:

```
insert_tun_man4(): write failed : The requested resource is busy.
```

Problem: The tunnel you requested to activate is already active or you have colliding SPI values.

To fix: Issue the **rmtun** command to deactivate, then issue the **mktun** command to activate. Check to see if the SPI values for the failing tunnel match any other active tunnel. Each tunnel should have its own unique SPI values.

Error: Issuing **mktun** command results in the following error:

```
Device ipsec_v4 is in Defined status.
```

```
Tunnel activation for IP Version 4 not performed.
```

Problem: You have not made the IP Security device available.

To fix: Issue the following command:

```
mkdev -l ipsec -t 4
```

You may have to change **-t** option to 6 if you are getting the same error for Version 6 tunnel activation. The devices must be in available state. To check the IP Security device state, issue the following command:

```
lsdev -Cc ipsec
```

Error: Issuing a **gentun** command results in the following error:

```
Invalid Source IP address
```

Problem: You have not entered a valid IP address for the source address.

To fix: For IP Version 4 tunnels, check to see that you have entered an available IP Version 4 address for the local machine. You cannot use host names for the source when generating tunnels, you may only use host names for the destination.

For IP Version 6 tunnels, check to see that you entered an available IP Version 6 address. If you type `netstat -in` and no IP Version 6 addresses exist, run **/usr/sbin/autoconf6** (interface) for a link local auto-generated address (using MAC address) or use **ifconfig** to manually assign an address.

Error: Issuing a **gentun** command results in the following error:

```
Invalid Source IP address
```

Problem: You have not entered a valid IP address for the source address.

To fix: For IP Version 4 tunnels, check to see that you have entered an available IP Version 4 address for the local machine. You cannot use host names for the source when generating tunnels, you may only use host names for the destination.

For IP Version 6 tunnels, check to see that you entered an available IP Version 6 address. If you type `netstat -in` and no IP Version 6 addresses exist, run **/usr/sbin/autoconf6** (interface) for a link local auto-generated address (using MAC address) or use **ifconfig** to manually assign an address.

Error: Issuing **mktun** command results in the following error:
`insert_tun_man4(): write failed : A system call received a parameter that is not valid.`

Problem: Tunnel generation occurred with invalid ESP and AH combination or without the use of the new header format when necessary.

To fix: Check to see what authentication algorithms are in use by the particular tunnel in question. Remember that the HMAC_MD5 and HMAC_SHA algorithms require the new header format. The new header format can be changed using the SMIT fast path **ips4_basic** or the **-z** parameter with the **chtun** command. Also remember that DES_CBC_4 cannot be used with the new header format.

Error: Starting IP Security from Web-based System Manager results in a `Failure` message.

Problem: The IP Security daemons are not running.

To fix: View which daemons are running by entering the `ps -ef` command. The following daemons are associated with IP Security:

- **tmd**
- **isakmpd**
- **cpsd**

The **cpsd** daemon is active only if the digital certificate code is installed (the fileset named **gskit.rte** or **gskkm.rte**) and you have configured the IBM Key Manager tool to contain digital certificates.

If the daemons are not active, stop IP Security using Web-based System Manager and then restart it, which automatically starts the appropriate daemons.

Error: Trying to use IP Security results in the following error:
 The installed `bos.crypto` is back level and must be updated.

Problem: The **bos.net.ipsec.*** files have been updated to a newer version, but the corresponding **bos.crypto.*** files have not.

To fix: Update the **bos.crypto.*** files to the version that corresponds with the updated **bos.net.ipsec.*** files.

Troubleshooting IKE Tunnel Errors

The following sections described errors that can occur when using IKE tunnels.

IKE Tunnel Process Flow

The IKE tunnels are set up by the communication of the **ike** command or the Web-based System Manager VPN panels with three daemons:

Table 3.

tmd	The Tunnel Manager daemon
isakmpd	The IKE daemon
cpsd	The certificate proxy daemon

For IKE tunnels to be properly set up, the **tmd** and **isakmpd** daemons must be running. If IP Security is set to start at reboot, these daemons start automatically. Otherwise, they must be started using Web-based System Manager.

The Tunnel Manager gives requests to **isakmpd** to start a tunnel. If the tunnel already exists or is not valid (for instance, has an invalid remote address), it reports an error. If negotiation has started, it may take some time, depending on network latency, for the negotiation to complete. The **ike cmd=list** command can list the state of the tunnel to determine if the negotiation was successful. Also, the Tunnel Manager logs events to **syslog** to the levels of **debug**, **event**, and **information**, which can be used to monitor the progress of the negotiation.

The sequence is:

1. Use Web-based System Manager or the **ike** command to initiate a tunnel.
2. The **tmd** daemon gives the **isakmpd** daemon a connection request for key management (phase 1).
3. The **isakmpd** daemon responds with SA created or an error.
4. The **tmd** daemon gives the **isakmpd** daemon a connection request for a data management tunnel (phase 2).
5. The **isakmpd** daemon responds with SA created or an error.
6. Tunnel parameters are inserted into the kernel tunnel cache.
7. Filter rules are added to the kernel dynamic filter table.

When the machine is acting as a responder, the **isakmpd** daemon notifies the Tunnel Manager **tmd** daemon that a tunnel has been negotiated successfully and a new tunnel is inserted into the kernel. In such cases, the process starts with step 3 and continues until step 7, without the **tmd** daemon issuing connection requests.

IKE Logging

The **isakmpd**, **tmd** and **cpsd** daemons log events to **syslog**. For **isakmpd**, logging is enabled using the **ike cmd=log** command. The **/etc/isakmpd.conf** configuration file can be set up to specify the logging level. The level can be set to **none**, **errors**, **isakmp_events**, or **information**.

Note: In versions earlier than AIX 5.1, **isakmpd** logged to a separate file, which was also specified in **/etc/isakmpd.conf**.

The configuration file parameter that can be set for logging is **log_level**. The IKE daemons use the following levels of logging:

none No logging (the default)

error Only logging protocol and API errors

isakmp_events

Only logging IKE protocol events and errors

Information

Logging protocol and implementation information (recommended for debugging)

The syntax for this option is simply:

```
log_level
```

The **isakmpd** daemon code either initiates by sending a proposal or responds by evaluating a proposal. If the proposal is accepted, a security association is created and the tunnel is set up. If the proposal is not accepted or the connection times out before the negotiation completes, **isakmpd** indicates an error. The entries in **syslog** from **tmd** indicate whether the negotiation succeeded. A failure caused by an invalid certificate logs to **syslog**. To find out the exact cause of a failed negotiation, check the logging file specified in **/etc/syslog.conf**.

The **syslog** facility adds a prefix to each log line, noting the date and time, the machine, and the program. The following example uses **googly** as the machine name and **isakmpd** as the program name:

```
Nov 20 09:53:50 googly isakmpd: ISAKMP_MSG_HEADER
Nov 20 09:53:50 googly isakmpd: Icookie : 0xef06a77488f25315, Rcookie :0x0000000000000000
Nov 20 09:53:51 googly isakmpd: Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
Nov 20 09:53:51 googly isakmpd: Xchg Type : 2 (ID protected), Flag= 0, Encr : No,COMMIT : No
Nov 20 09:53:51 googly isakmpd: Msg ID : 0x00000000
```

To improve clarity, the **grep** command can be used to extract log lines of interest (such as all **isakmpd** logging) and the **cut** command can be used to remove the prefix from each line. The **isakmpd** log examples in the rest of this section have been tailored in a similar way.

Parse Payload Logging Function

The security association (SA) between two end points is established by exchanging IKE messages. The Parse Payload function parses the messages in a human-readable format. The logging can be turned on by editing the `/etc/isakmpd.conf` file. The logging entry in the `/etc/isakmpd.conf` file looks similar to the following:

information

The type of IKE payloads that Parse Payload logs depends on the content of the IKE message. Examples include SA Payload, Key Exchange Payload, Certificate Request Payload, Certificate Payload, and Signature Payload. The following is an example of a Parse Payload log in which an ISAKMP_MSG_HEADER is followed by five payloads:

```
ISAKMP_MSG_HEADER
  Icookie : 0x9e539a6fd4540990, Rcookie : 0x0000000000000000
  Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
  Xchg Type : 4 (Aggressive), Flag= 0, Encr : No, COMMIT : No
  Msg ID : 0x00000000
  len : 0x10e(270)
```

```
SA Payload:
  Next Payload : 4(Key Exchange), Payload len : 0x34(52)
  DOI : 0x1(INTERNET)
  bitmask : 1(SIT_IDENTITY_ONLY)
```

```
Proposal Payload:
  Next Payload : 0(NONE), Payload len : 0x28(40)
  Proposal # : 0x1(1), Protocol-ID : 1(ISAKMP)
  SPI size : 0x0(0), # of Trans : 0x1(1)
```

```
Transform Payload:
  Next Payload : 0(NONE), Payload len : 0x20(32)
  Trans # : 0x1(1), Trans.ID : 1(KEY_IKE)
  Attr : 1(Encr.Alg ), len=0x2(2)
  Value=0x1(1),(DES-cbc)
  Attr : 2(Hash Alg ), len=0x2(2)
  Value=0x1(1),(MD5)
  Attr : 3(Auth Method ), len=0x2(2)
  Value=0x3(3),(RSA Signature)
  Attr : 4(Group Desc ), len=0x2(2)
  Value=0x1(1),(default 768-bit MODP group)
  Attr : 11(Life Type ), len=0x2(2)
  Value=0x1(1),(seconds)
  Attr : 12(Life Duration), len=0x2(2)
  Value=0x7080(28800)
```

```
Key Payload:
  Next Payload : 10(Nonce), Payload len : 0x64(100)
```

```
Key Data :
33 17 68 10 91 1f ea da 38 a0 22 2d 84 a3 5d 5d
a0 e1 1f 42 c2 10 aa 8d 9d 14 0f 58 3e c4 ec a3
9f 13 62 aa 27 d8 e5 52 8d 5c c3 cf d5 45 1a 79
8a 59 97 1f 3b 1c 08 3e 2a 55 9b 3c 50 cc 82 2c
d9 8b 39 d1 cb 39 c2 a4 05 8d 2d a1 98 74 7d 95
ab d3 5a 39 7d 67 5b a6 2e 37 d3 07 e6 98 1a 6b
```

```
Nonce Payload:
  Next Payload : 5(ID), Payload len : 0xc(12)
```

```
Nonce Data:
6d 21 73 1d dc 60 49 93
```

```
ID Payload:
  Next Payload : 7(Cert.Req), Payload len : 0x49(73)
  ID type : 9(DER_DN), Protocol : 0, Port = 0x0(0)
```

```
Certificate Request Payload:
  Next Payload : 0(NONE), Payload len : 0x5(5)
  Certificate Encoding Type: 4(X.509 Certificate - Signature)
```

Within each payload, there is a Next Payload field that points to the payload following the current payload. If the current payload is the last one in the IKE message, the Next Payload field has the value of zero (None).

Each Payload in the example has information pertaining to the negotiations that are going on. For example, the SA payload has the Proposal and Transform Payloads, which in turn show the encryption algorithm, authentication mode, hash algorithm, SA life type, and SA duration that the initiator is proposing to the responder.

Also, the SA Payload consists of one or more Proposal Payloads and one or more Transform Payloads. The Next Payload field for Proposal Payload has a value of either zero if it is the only Proposal Payload or a value of two if it is followed by one more Proposal Payloads. Similarly the Next Payload field for a Transform Payload has a value of zero if it is the only Transform Payload or a value of 3 if it is followed by one more Transform Payloads, as shown in the following example:

```
ISAKMP_MSG_HEADER
  Icookie : 0xa764fab442b463c6, Rcookie : 0x0000000000000000
  Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
  Xchg Type : 2 (ID protected), Flag= 0, Encr : No, COMMIT : No
  Msg ID : 0x00000000
  len : 0x70(112)
SA Payload:
  Next Payload : 0(NONE), Payload len : 0x54(84)
  DOI : 0x1(INTERNET)
  bitmask : 1(SIT_IDENTITY_ONLY)
Proposal Payload:
  Next Payload : 0(NONE), Payload len : 0x48(72)
  Proposal # : 0x1(1), Protocol-ID : 1(ISAKMP)
  SPI size : 0x0(0), # of Trans : 0x2(2)
Transform Payload:
  Next Payload : 3(Transform), Payload len : 0x20(32)
  Trans # : 0x1(1), Trans.ID : 1(KEY_IKE)
  Attr : 1(Encr.Alg ), len=0x2(2)
  Value=0x5(5), (3DES-cbc)
  Attr : 2(Hash Alg ), len=0x2(2)
  Value=0x1(1), (MD5)
  Attr : 3(Auth Method ), len=0x2(2)
  Value=0x1(1), (Pre-shared Key)
  Attr : 4(Group Desc ), len=0x2(2)
  Value=0x1(1), (default 768-bit MODP group)
  Attr : 11(Life Type ), len=0x2(2)
  Value=0x1(1), (seconds)
  Attr : 12(Life Duration), len=0x2(2)
  Value=0x7080(28800)
Transform Payload:
  Next Payload : 0(NONE), Payload len : 0x20(32)
  Trans # : 0x2(2), Trans.ID : 1(KEY_IKE)
  Attr : 1(Encr.Alg ), len=0x2(2)
  Value=0x1(1), (DES-cbc)
  Attr : 2(Hash Alg ), len=0x2(2)
  Value=0x1(1), (MD5)
  Attr : 3(Auth Method ), len=0x2(2)
  Value=0x1(1), (Pre-shared Key)
  Attr : 4(Group Desc ), len=0x2(2)
  Value=0x1(1), (default 768-bit MODP group)
  Attr : 11(Life Type ), len=0x2(2)
  Value=0x1(1), (seconds)
  Attr : 12(Life Duration), len=0x2(2)
  Value=0x7080(28800)
```

The IKE Message Header of a Parse Payload log shows the exchange type (Main Mode or Aggressive Mode), the length of the entire message, the message identifier, etc.

The Certificate Request Payload requests a certificate from the responder. The responder sends the certificate in a separate message. The following example shows the Certificate Payload and Signature Payload that are sent to a peer as a part of an SA negotiation. The certificate data and the signature data are printed in hex format.

ISAKMP_MSG_HEADER

```
Icookie : 0x9e539a6fd4540990, Rcookie : 0xc7e0a8d937a8f13e
Next Payload : 6(Certificate), Maj Ver : 1, Min Ver : 0
Xchg Type : 4 (Aggressive), Flag= 0, Encr : No,COMMIT : No
Msg ID : 0x00000000
len : 0x2cd(717)
```

Certificate Payload:

```
Next Payload : 9(Signature), Payload len : 0x22d(557)
Certificate Encoding Type: 4(X.509 Certificate - Signature)
Certificate: (len 0x227(551) in bytes
82 02 24 30 82 01 8d a0 03 02 01 02 02 05 05 8e
fb 3e ce 30 0d 06 09 2a 86 48 86 f7 0d 01 01 04
05 00 30 5c 31 0b 30 09 06 03 55 04 06 13 02 46
49 31 24 30 22 06 03 55 04 0a 13 1b 53 53 48 20
43 6f 6d 6d 75 6e 69 63 61 74 69 6f 6e 73 20 53
65 63 75 72 69 74 79 31 11 30 0f 06 03 55 04 0b
13 08 57 65 62 20 74 65 73 74 31 14 30 12 06 03
55 04 03 13 0b 54 65 73 74 20 52 53 41 20 43 41
30 1e 17 0d 39 39 30 39 32 31 30 30 30 30 30 30
5a 17 0d 39 39 31 30 32 31 32 33 35 39 35 39 5a
30 3f 31 0b 30 09 06 03 55 04 06 13 02 55 53 31
10 30 0e 06 03 55 04 0a 13 07 49 42 4d 2f 41 49
58 31 1e 30 1c 06 03 55 04 03 13 15 62 61 72 6e
65 79 2e 61 75 73 74 69 6e 2e 69 62 6d 2e 63 6f
6d 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d 01 01
01 05 00 03 81 8d 00 30 81 89 02 81 81 00 b2 ef
48 16 86 04 7e ed ba 4c 14 d7 83 cb 18 40 0a 3f
55 e9 ad 8f 0f be c5 b6 6d 19 ec de 9b f5 01 a6
b9 dd 64 52 34 ad 3d cd 0d 8e 82 6a 85 a3 a8 1c
37 e4 00 59 ce aa 62 24 b5 a2 ea 8d 82 a3 0c 6f
b4 07 ad 8a 02 3b 19 92 51 88 fb 2c 44 29 da 72
41 ef 35 72 79 d3 e9 67 02 b2 71 fa 1b 78 13 be
f3 05 6d 10 4a c7 d5 fc fe f4 c0 b8 b8 fb 23 70
a6 4e 16 5f d4 b1 9e 21 18 82 64 6d 17 3b 02 03
01 00 01 a3 0f 30 0d 30 0b 06 03 55 1d 0f 04 04
03 02 07 80 30 0d 06 09 2a 86 48 86 f7 0d 01 01
04 05 00 03 81 81 00 75 a4 ee 9c 3a 18 f2 de 5d
67 d4 1c e4 04 b4 e5 b8 5e 9f 56 e4 ea f0 76 4a
d0 e4 ee 20 42 3f 20 19 d4 25 57 25 70 0a ea 41
81 3b 0b 50 79 b5 fd 1e b6 0f bc 2f 3f 73 7d dd
90 d4 08 17 85 d6 da e7 c5 a4 d6 9a 2e 8a e8 51
7e 59 68 21 55 4c 96 4d 5a 70 7a 50 c1 68 b0 cf
5f 1f 85 d0 12 a4 c2 d3 97 bf a5 42 59 37 be fe
9e 75 23 84 19 14 28 ae c4 c0 63 22 89 47 b1 b6
f4 c7 5d 79 9d ca d0
```

Signature Payload:

```
Next Payload : 0(NONE), Payload len : 0x84(132)

Signature: len 0x80(128) in bytes
9d 1b 0d 90 be aa dc 43 95 ba 65 09 b9 00 6d 67
b4 ca a2 85 0f 15 9e 3e 8d 5f e1 f0 43 98 69 d8
5c b6 9c e2 a5 64 f4 ef 0b 31 c3 cb 48 7c d8 30
e3 a2 87 f4 7c 9d 20 49 b2 39 00 fa 8e bf d9 b0
7d b4 8c 4e 19 3a b8 70 90 88 2c cf 89 69 5d 07
f0 5a 81 58 2e 15 40 37 b7 c8 d6 8c 5c e2 50 c3
4d 19 7e e0 e7 c7 c2 93 42 89 46 6b 5f f8 8b 7d
5b cb 07 ea 36 e5 82 9d 70 79 9a fe bd 6c 86 36
```

Digital Certificate and Signature Mode Problems

Error: The **cpsd** (Certificate Proxy Server daemon) does not start. An entry similar to the following appears in the log file:

```
Sep 21 16:02:00 ripple CPS[19950]: Init():LoadCaCerts() failed, rc=-12
```

Problem: The certificate database has not opened or has not been found.

To Fix: Ensure that the IBM Key Manager certificate databases are present in **/etc/security**. The following files make up the database: **ikekey.crl**, **ikekey.kdb**, **ikekey.rdb**, **ikekey.sth**.

If only the **ikekey.sth** file is missing, the stash password option was not selected when the IBM Key Manager database was created. The password must be stashed to enable using digital certificates with IP Security. (See *Creating a Key Database* for more information.)

Error: IBM Key Manager gives the following error when receiving a certificate:

```
Invalid Base64-encoded data was found
```

Problem: Superfluous data has been found in the certificate file or else data was lost or corrupted.

To Fix: The 'DER' Encoded Certificate should be contained within the following strings (shown below). No other characters should precede or follow other than the BEGIN and END CERTIFICATE strings.

```
-----BEGIN CERTIFICATE-----
MIICMTCAZqgAwIBAgIFFKZtANowDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMC
RkkxJDAiBgNVBAoTGINTSCBDb21tdW5pY2F0aW9ucyBTZW1cm10eTERMA8GA1UE
CxMIY2ViIHRlc3QxZDASBgNVBAMTC1Rlc3QgU1NBIEENBMB4XDTk5MDkyMTAwMDAw
MFoXDTk5MTAyMTIzNTk1OVowOzELMAkGA1UEBhMCVVMxDDAKBgNVBAoTA01CTTEe
MBWGA1UEAxMVcm1wcGx1LmF1c3Rpbj5pYm0uY29tMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQC5EZqo6n7tZrpAL6X4L7mf4yXQSm+m/NsJLhp6afbFpPvXgYWC
wq4pv0tvxgum+FHrE0gysNjbKkE4Y6ixC9PGGAKHnhM3vrmvFjn11G6KtyEz58Lz
BWW39QS6Nj1LqqP1nT+y3+Xzvfv8Eonqzno8mg1CWMX09SguLmWoU1PcZQIDAQAB
oyAwHjALBgNVHQ8EBAMCBaAwDwYDVR0RBAgwBocECQNhHzANBgkqhkiG9w0BAQUF
A0BgQA6b9p4Zay34/fyA1yCkNNAYJRrN3Vc4NHN7IGjUziN6jK5UyB5zL37FERW
hT9ArPLzK7yEZs+MDNvB0bosyGWEDYPZr7EZHHYcoBP4/cd0V5rBFmA8Y2gUthPi
Ioxpi4+KZGHYyLqTrm+8Is/DVJaQmCGRPynHK35xjT6WuQt iYg==
-----END CERTIFICATE-----
```

The following options can help you diagnose and solve this problem.

- If data was lost or corrupted, recreate the Certificate
- Use an ASN.1 parser (available on the Internet World Wide Web) to check whether the certificate is valid by parsing the certificate successfully.

Error: IBM Key Manager gives the following error when receiving a personal certificate:

```
No request key was found for the certificate
```

Problem: A Personal Certificate Request does not exist for the personal certificate being received.

To Fix: Create the Personal Certificate Request again and request a new certificate.

Error: Web-based System Manager gives the following error when you configure an IKE tunnel:

```
Error 171 in the Key Management (Phase 1) Tunnel operation:
PUT_IRL_FAILED
```

Problem: One cause for this error is the host identity type, which is configured on the IKE dialog (Identification tab), is invalid. This can happen when the host identity type selected from the pull-down list does not logically match the type entered in the Host Identity field. For example, if you select a host identity type of **X500 Distinguished Name**, you need to enter a properly formatted distinguished name in the Host Identity field.

To Fix: Ensure the distinguished name you enter is correct for the type selected in the host identity pull-down list.

Error: An IKE negotiation fails and an entry similar to the following appears in the log file:

```
inet_cert_service::channelOpen():clientInitIPC():error,rc =2  
(No such file or directory)
```

Problem: The **cpsd** is not running or has died

To Fix: Start IP Security using Web-based System Manager. This action also starts the appropriate daemons.

Error: An IKE negotiation fails and an entry similar to the following appears in the log file:

```
CertRepo::GetCertObj: DN Does Not Match: ("/C=US/O=IBM/CN=ripple.austin.ibm.com")
```

Problem: The X.500 Distinguished Name (DN) entered while defining the IKE tunnel does not match the X.500 DN in the personal certificate.

To Fix: Change the IKE tunnel definition in Web-based System Manager to match the distinguished name in the certificate.

Error: While defining IKE tunnels in Web-based System Manager, the Digital certificate check box is disabled under the Authentication Method tab.

Problem: The policy associated with this tunnel does not use RSA signature mode authentication.

To Fix: Change the transform of the associated policy to use the RSA signature authentication method. For example, you can choose **IBM_low_CertSig** as a key management policy when defining a IKE tunnel.

Tracing Facilities

Tracing is a debug facility for tracing kernel events. Traces can be used to get more specific information about events or errors occurring in the kernel filter and tunnel code.

SMIT has an IP Security trace facility available through the Advanced IP Security Configuration menu. The information captured by this trace facility includes information on Error, Filter, Filter Information, Tunnel, Tunnel Information, Capsulation/Decapsulation, Capsulation Information, Crypto, and Crypto Information. By design, the error trace hook provides the most critical information. The info trace hook can generate a lot of information and may have an impact on system performance. This tracing will provide clues to you as to what a problem may be. Tracing information will also be required when speaking with an IBM IP Security Technician. To access the tracing facility, use the SMIT fast path **smit ips4_tracing** (for IP Version 4) or **smit ips6_tracing** (for IP Version 6).

ipsecstat

You can issue the **ipsecstat** command to generate the following sample report. This sample report shows that the IP Security devices are in the available state, that there are three authentication algorithms installed, three encryption algorithms installed, and that there is a current report of packet activity. This information may be useful to you in determining where a problem exists if you are troubleshooting your IP Security traffic.

IP Security Devices:

```
ipsec_v4 Available  
ipsec_v6 Available
```

Authentication Algorithm:

```
HMAC_MD5 -- Hashed MAC MD5 Authentication Module  
HMAC_SHA -- Hashed MAC SHA Hash Authentication Module  
KEYED_MD5 -- Keyed MD5 Hash Authentication Module
```

Encryption Algorithm:

```
CDMF -- CDMF Encryption Module  
DES_CBC_4 -- DES CBC 4 Encryption Module  
DES_CBC_8 -- DES CBC 8 Encryption Module  
3DES_CBC -- Triple DES CBC Encryption Module
```

```

IP Security Statistics -
Total incoming packets: 1106
Incoming AH packets:326
Incoming ESP packets: 326
Srcrte packets allowed: 0
Total outgoing packets:844
Outgoing AH packets:527
Outgoing ESP packets: 527
Total incoming packets dropped: 12
  Filter denies on input: 12
  AH did not compute: 0
  ESP did not compute:0
  AH replay violation:0
  ESP replay violation: 0
Total outgoing packets dropped:0
  Filter denies on input:0
Tunnel cache entries added: 7
Tunnel cache entries expired: 0
Tunnel cache entries deleted: 6

```

Note: Beginning with AIX 4.3.3, CDMF support has been removed because DES is now available world wide. Reconfigure any tunnels that use CDMF to use DES or Triple DES.

IP Security Reference

List of Commands

ike cmd=activate[phase=1|2]
[numlist=tunnel_num_list]
[remid=remote_id]
[ipaddr=src_addr,dst_addr] [autostart]

Starts an Internet Key Exchange (IKE) negotiation (AIX 4.3.2 and later). Flag **remid** is used to start phase1 or phase2 tunnel(s) from local ID to the specified remote ID. **remid** could be phase1 ID (such as IP address, FQDN, user@FQDN and X500DN), phase2 ID (such as IP address, subnet, and IP address range) or group ID. The , (comma) is used to delimit subnet ID/subnet mask, and starting or ending IP address. If **remid** is a group name, a tunnel is initiated for each group member. **remid** is an optional flag. It can only be used with the activate subcommand and cannot be used in conjunction with the **ipaddr** or **numlist** flags.

Examples:

1. To activate a phase1 tunnel to remote IP address 9.3.97.100:

```
ike cmd=activate phase=1 remid=9.3.97.100
```
2. To activate a phase2 tunnel to remote subnet ID 9.3.97.100, 255.255.255.0:

```
ike cmd=activate phase=2 \  
remid=9.3.97.100,255.255.255.0
```

ike cmd=remove

ike cmd=list

ikedb

gentun

mktun

chtun

rmtun

lstun

exptun

imptun

genfilt

mkfilt

mvfilt

Deactivates IKE tunnels (AIX 4.3.2 and later)

Lists IKE tunnels (AIX 4.3.2 and later)

Provides the interface to the IKE tunnel database(AIX 5.1 and later)

Creates a tunnel definition

Activates tunnel definition(s)

Changes a tunnel definition

Removes a tunnel definition

Lists tunnel definition(s)

Exports tunnel definition(s)

Imports tunnel definition(s)

Creates a filter definition

Activates filter definition(s)

Moves a filter rule

chfilt	Changes a filter definition
rmfilt	Removes a filter definition
lsfilt	Lists filter definition(s)
expfilt	Exports filter definition(s)
impfilt	Imports filter definition(s)
ipsec_convert	Lists status of IP security
ipsecstat	Lists status of IP security
ipsectrcbuf	Lists the contents of IP security tracing buffer
unloadipsec	Unloads a crypto module

List of Methods

defipsec	Defines an instance of IP Security for IP Version 4 or IP Version 6
cfgipsec	Configures and loads ipsec_v4 or ipsec_v6
ucfgipsec	Unconfigures ipsec_v4 or ipsec_v6

Chapter 5. TTY Devices and Serial Communications

This chapter contains information on managing tty terminal devices. Topics discussed are:

- TTY Overview
- Managing TTY Devices
- Dynamic Screen Utility
- Modems
- ATE Overview
- Setting Up ATE
- TTY Troubleshooting

TTY Overview

A tty terminal device is a character device that performs input and output on a character-by-character basis. The communication between terminal devices and the programs that read and write to them is controlled by the tty interface. Examples of tty devices are:

- Modems
- ASCII terminals
- System console (LFT)
- **aixterm** under AIXwindows

The tty devices can be added, deleted, listed, and changed like any other device on your system by using the Web-based System Manager Devices application, the SMIT tool, or device-specific commands.

TERM Values for Different Displays and Terminals

Information about terminal capabilities is stored in the **terminfo** database. The value of the **TERM** environment variable identifies the specific terminal description in the **terminfo** database. This provides all information that a program needs for communicating effectively with the current tty device.

Display/Terminal	Value
3161 ASCII Terminal	ibm3161
3163 ASCII Terminal	ibm3161
DEC VT100 (terminal)	vt100
DECVT220	vt220
3151 ASCII Display Station with Cartridge or 3161 ASCII Display Station with Cartridge	ibm3161-C
3162 ASCII Display Station	ibm3161
3162 ASCII Display Station with Cartridge	ibm3162
6091 Display	lft
AIXwindows	aixterm

For information about the entries in the **terminfo** database, see the **terminfo** file format. To convert **termcap** entries into **terminfo** entries, see the **captoinfo** command. (The **termcap** file contains the terminal descriptions for older Berkeley systems.)

Setting TTY Characteristics

The *line discipline* provides the hardware-independent user interface for communicating between the computer and an asynchronous device. For example, a user is able to erase a single line or to interrupt a currently running process by typing a particular sequence of characters. You can define the meaning of these character sequences as well as set other terminal characteristics, such as the communication speed, by using the Web-based System Manager Devices application, the **chdev** command, the System Management Interface Tool (SMIT), or the **stty** command.

Most applications (including shells and editors) are designed to interface with terminals using the POSIX line discipline (default). You can change to the Berkeley line discipline by using the **stty** command.

Setting Attributes on the Attached TTY Device

For correct communication between the host and an attached tty device, the following are required:

- A properly wired communications cable
- Matching communications values (line speed, word length, parity, stop bit, and interface) between the host and the attached tty device.

Managing TTY Devices

To do any task in the following table except List, a tty device must be installed.

Managing TTY Devices Tasks			
Task	SMIT Fast Path	Command or File	Web-based System Manager Management Environment
List Defined TTY Devices	smit lsdtty	lsdev -C -c tty -H	Software → Devices → All Devices .
Add a TTY	smit mktty	mkdev -t tty ^{1,2}	Software → Devices → Overview and Tasks .
Move a TTY to Another Port ³	smit movtty	chdev -l Name -p ParentName -w ConnectionLocation ^{2,4}	
Change/Show Characteristics of a TTY	smit chtty	lsattr -l Name -E (to show); chdev -l Name (to change) ^{4,5}	Software → Devices → All Devices → Selected → Properties .
Remove a TTY ³	smit rmtty	rmdev -l Name	Software → Devices → All Devices → Selected → Delete .
Configure a Defined TTY (Make Available for Use)	smit mktty	mkdev -l Name	Software → Devices → Overview and Tasks .

Notes:

1. Other flags may be used to further specify the new tty device. For example, to define and configure an RS-232 tty device connected to port 0 on the 8-port asynchronous adapter sa3 with the speed attribute set to 19200 and other attributes set to values retrieved from the foo file:

```
mkdev -t tty -s rs232 -p sa3 -w 0 -a speed=19200 -f foo
```
2. The **mkdev** and **chdev** commands support options that are not possible with Web-based System Manager or SMIT.
3. Disable the tty before doing this task. See the **pdisable** command.
4. Use flags to change specific characteristics about a tty from the command line.

5. You can select a Posix baud rate from the List function, or you can type in non-Posix baud rate directly into the field. If the selected baud rate cannot be supported by the modem's hardware, the system displays an error message.

If adding or changing a tty from the command line, consult the following list to find out the *Attribute* name to specify in the **-a Attribute=Value** flag for the characteristic you want to set. For example, specify **-a speed=Value** to set the baud rate of a tty device.

Characteristic	Attribute Name
Enable LOGIN	login
BAUD rate speed	speed
PARITY	parity
BITS per character	bpc
Number of STOP BITS	stops
TIME before advancing to next port setting	timeout
XON-XOFF handshaking	xon
TERMINAL type	term
FLOW CONTROL to be used	flow_disp
OPEN DISCIPLINE to be used	open_disp
STTY attributes for RUN time	runmodes
STTY attributes for LOGIN	logmodes
RUN shell activity manager	shell
LOGGER name	logger
STATUS of device at BOOT time	autoconfig
TRANSMIT buffer count	tbc
RECEIVE trigger level	rtrig
STREAMS modules to be pushed at open time	modules
INPUT map file	imap
OUTPUT map file	omap
CODESET map file	csmmap
INTERRUPT character	intr
QUIT character	quit
ERASE character	erase
KILL character	kill
END OF FILE character	eof
END OF LINE character	eol
2nd END OF LINE character	eol2
DELAY SUSPEND PROCESS character	dsusp
SUSPEND PROCESS character	susp
LITERAL NEXT character	lnext
START character	start
STOP character	stop
WORD ERASE character	werase
REPRINT LINE character	reprint

DISCARD character	discard
-------------------	---------

Dynamic Screen Utility

The dynamic screen utility, or **dscreen** command, is a utility that allows a single physical terminal to be connected to several virtual terminal sessions (screens) at one time. It is mainly intended for use with terminals that have two or more pages of screen memory (for example, the IBM 3151 Models 310 or 410 display with the Cartridge for Expansion). With such terminals, switching between virtual screens also switches between physical terminal screen pages allowing each virtual screen's image to be saved and restored. On terminals without multiple pages of screen memory, the **dscreen** command can still be used to switch among virtual screen sessions although the appearance of the screen will not be maintained.

Note: For full support of **dscreen** utility, the terminal must be able to switch internal screen pages on command and must remember the cursor position for each page. While the **dscreen** utility will work on both smart and dumb terminals, screen images are not saved during screen changes on dumb terminals.

dscreen Terminal Configuration Information File

The **dscreen** utility terminal configuration information file (or **dsinfo** file) is used to define a different set of keys to be used with the **dscreen** utility. This might be done, for example, when the originally defined **dscreen** utility keys conflict with a software application in use on the system.

The **dsinfo** file terminal type assumes a single page of screen memory. Therefore, if a terminal supports additional pages of screen memory, the **dsinfo** file must be customized to use the appropriate sequence for page memory control. Consult the appropriate terminal reference guide for the specific control sequence.

The default **dsinfo** file is **/usr/lbin/tty/dsinfo**. Use the **-i** flag to specify a different **dsinfo** file. This remainder of this section refers to the default file. However, the same information applies to any customized **dsinfo** file you create.

For more information concerning the **dsinfo** file, see "Dynamic Screen Assignment".

Key Action Assignments

When the **dscreen** command is executed, it starts a virtual screen. Some of the keys on the terminal keyboard are not passed through to the virtual screen; instead, the **dscreen** utility intercepts these keys and performs certain actions when they are pressed. The actions include:

Select	Selects a specified screen.
Block	Blocks all input and output.
New	Starts a new screen session.
End	Ends the dscreen utility.
Quit	Quits the dscreen utility.
Previous	Switches to previous screen.
List	Lists the dscreen assigned keys and their actions.

The function of each key is dependent on the terminal and the terminal description in the **/usr/lbin/tty/dsinfo** file.

Select Keys

When a new virtual screen is created, it is assigned a select key. Pressing the select key causes the following actions:

- A switch from the physical terminal to the video page associated with the particular virtual screen.

- Input and output is directed appropriately between the physical terminal and the virtual screen.

After all of the select keys defined in the **dsinfo** file have virtual screens assigned to them, no more screens can be created. Individual screen sessions end when the original shell process exits. This frees the associated select key for use with another virtual screen. The **dscreen** utility is ended when there are no more active screens.

Block Keys

Block keys are used to stop output in a fashion similar to Ctrl-S key when using IXON flow control. The purpose of these keys is to allow for transparently setting up terminal sessions on two computers using a terminal that has two serial ports.

New Keys

Pressing a new screen key creates a new logical screen and assigns it to one of the select keys. Each new screen requires:

- A select key as defined in the dsinfo file
- A **dscreen** pseudo-terminal device
- Enough memory for the various structures used in screen tracking
- A process to run the shell from.

If any of these are not available, the new screen operation fails with a message indicating the reason for the failure.

End and Quit Keys

Pressing an end key causes the following to occur:

- Send a **SIGHUP** signal to all the screen sessions
- Clean up
- Exit with a status of 0.

Pressing a quit key performs the same actions but exits with status of 1.

Previous Key

Pressing a previous key switches the terminal to the screen that was last displayed.

Notes:

1. Do not switch screens when the current screen is being written to; an escape sequence might be truncated and leave the terminal in an unknown state.
2. Some terminal displays can save the cursor position for individual screens but might not save other states such as insert mode, inverse video, and so forth. If this is the case, users should avoid these modes while switching screens.

List Key

Pressing a list key displays a list of keys and their actions on the terminal display. Only those keys recognized by the **dscreen** utility will be shown. When a new screen is created using the **dscreen** utility, the message Press *KEY* for help, where *KEY* is the name of the list key displayed on the terminal. Note that the message is displayed *only* if there is a list key defined.

Dynamic Screen Assignment

The terminal description entry in the **/usr/lib/tty/dsinfo** file has the same number of screen selection keys as the terminal has physical screen pages. If more screen selection keys are defined than the number of physical screen pages, the **dscreen** utility will dynamically assign physical screen pages to virtual screens.

When a virtual screen is selected that does not have an associated page of screen memory, the **dscreen** utility assigns the least recently used physical screen to the virtual screen. Depending on the specifications maintained in the `/usr/lib/tty/dsinfo` description file, an indication that the physical screen is connected to a different virtual screen might be noticeable; for example, the screen is cleared.

dsinfo File

The **dsinfo** file is a database of terminal descriptions used by the **dscreen** multiple screen utility. The file contains the following information:

- The **dscreen** utility keys and the functions they perform
- Number of screen memory pages for the terminal
- Code sequences sent or received to use the above features.

The terminal type entries in the default **dsinfo** file resemble the following 3151 ASCII terminal values:

```
# The Cartridge for Expansion (pn: 64F9314) needed for this entry
ibm3151|3151|IBM 3151,
dsk=^M|Shift-F1|, # Selects first screen
dsk=^B^M|Shift-F2|, # Selects second screen
dsk=^C^M|Shift-F3|, # Selects third screen
dsk=^D^M|Shift-F4|, # Selects fourth screen
dsk=^E^M|Shift-F5|, # Creates a new screen
dsk=^F^M|Shift-F6| \E pA\EH\EJ, # Go to screen 1 and end
dsk=^G^M|Shift-F7|, # Lists function keys (help)
dsk=^H^M|Shift-F8|, # Go to previous screen
dsk=^I^M|Shift-F9| \E pA\EH\EJ, # Go to screen 1 and quit
dsp=\E pA|\EH\EJ, # Terminal sequence for screen 1
dsp=\E pB|\EH\EJ, # Terminal sequence for screen 2
dsp=\E pC|\EH\EJ, # Terminal sequence for screen 3
dsp=\E pD|\EH\EJ, # Terminal sequence for screen 4
dst=10, # Allow 1 second timeout buffer
```

Entry Format for dsinfo

Entries in the **dsinfo** file consist of comma-separated fields. The first field is a list of alternate names for the terminal, each name is separated by a pipe (|) character. Any text preceded by a pound (#) character is regarded as a comment and ignored by **dscreen**. The remaining fields are strings describing the capabilities of the terminal to the **dscreen** utility. Within these strings, the following escape codes are recognized:

Escape Sequence	Description
\E,\e	escape character
\n,\l	newline (or linefeed) character
\r	carriage return
\t	tab character
\b	backspace character
\f	formfeed character
\s	space character
\nnn	character with octal value <i>nnn</i>
^x	Ctrl-x for any appropriate x value

Any other character preceded by a backslash will yield the character itself. The strings are entered as *type=string*, where *type* is the type of string as listed below, and *string* is the string value.

It is important that the entry fields in the **dsinfo** file be separated by commas. If a comma is omitted or truncated from the end of a **dsinfo** file entry, the file will become unreadable by the **dscreen** utility and an error will be returned to the display.

String Types

The string types are as follows:

dskx A string type that starts with dsk describes a key. The type must be four letters long, and the fourth letter x indicates what action is taken when the key is received. The key types are:

Type	Action
dsks	Switch Screens
dskb	Block Input and Output
dske	End dscreen
dskq	Quit dscreen (exit status=1)
dskc	Create New Screen
dskp	Switch to Previous Screen
dskl	List Keys and Actions

Any other key type (that is, a string type dskx that does not end in s, b, e, q, p, or l) will cause no internal **dscreen** action, but will show up in the key listing and will be recognized and acted on. A type of dskn (n for No Operation) should be used when no internal **dscreen** action is desired.

The value string for each key has three substrings, which are separated by pipe (|) characters.

Note: Use \ | to include the | character in one of the substrings.

The first substring is the sequence of characters that the terminal sends when the key is pressed. The second substring is a label for the key that is printed when a list of keys is displayed. The third substring is a sequence of characters that **dscreen** sends to the terminal when this key is pressed before performing the action this key requests.

dsp A string type of dsp describes a physical screen in the terminal. One dsp string should be present for each physical screen in the terminal. The value string for each physical screen has two substrings, which are separated by a pipe (|) character.

The first substring is the sequence of characters to send to the terminal to display and output to the physical page on the terminal.

The second substring is sent to the terminal when the page is used for something new. This second substring is often set to the clear screen sequence. It is sent under the following two conditions:

1. When a new virtual terminal session is being created.
2. When there are more virtual terminals than there are physical screens. If a virtual terminal is selected that requires **dscreen** to reuse one of the physical screens, it will send this sequence to the screen to indicate that the screen contents do not match the output of the virtual terminal connected.

Note: Running with more virtual terminals than physical screens can be confusing and is not recommended; it can be avoided by defining no more screen selection keys (dsks=) than physical screens (dsp=) in the dsinfo entry.

dst A String with a type of dst adjusts **dscreen**'s input timeout. The value of the string is a decimal number. The timeout value is in tenths of seconds and has a maximum value of 255 (default=1 [or 0.1 seconds]).

When **dscreen** recognizes a prefix of an input key sequence but does not have all the characters of the sequence, it will wait for more characters to be sent until it is recognizable. If the timeout occurs before more characters are received, the characters are sent on to the virtual screen and **dscreen** will not consider these characters as part of an input key sequence.

It may be necessary to raise this value if one or more of the keys **dscreen** is to trigger on is actually a number of keystrokes (that is assigning Ctrl-Z 1, Ctrl-Z 2, Ctrl-Z 3, etc., for screen selection and Ctrl-Z N for new screen and so on).

Example 1

The following example `/usr/lib/tty/dsinfo` entry is for a Wyse-60 with three screen sessions:

```
wy60|wyse60|wyse model 60,  
dskb=A^M|Shift-F1|,  
dskb=Aa^M|Shift-F2|,  
dskb=Ab^M|Shift-F3|,  
dskc=\200|Ctrl-F1|,  
dske=\201|Ctrl-F2|\Ew0\E+,  
dskl=\202|Ctrl-F3|,  
dsp=\Ew0|\E+,  
dsp=\Ew1|\E+,  
dsp=\Ew2|\E+,
```

With this entry:

- Shift-F1 through Shift-F3 are used for selecting screens 1 through 3.
- Ctrl-F1 creates a new screen.
- Ctrl-F2 sends: Esc w 0 Esc + to the screen (switching to window 0 and clearing the screen) and then ends **dscreen**.
- Ctrl-F3 lists the keys and their functions.

Each time a physical screen is used for a new screen, the sequence Esc + will be sent to the terminal, which will clear the screen.

Example 2

This example is for a Wyse-60 with three screen sessions, but one of the screens is on a second computer communicating through the second serial port on the terminal:

```
wy60-1|wyse60-1|wyse model 60 - first serial port  
dskb=A^M|Shift-F1|,  
dskb=Aa^M|Shift-F2|,  
dskb=Ab^M|Shift-F3|\Ed#Ab\r^T\Ee9,  
dskc=\200|Ctrl-F1|,  
dske=\201|Ctrl-F2|\Ed#\201^T\Ew0\E+,  
dskl=\202|Ctrl-F3|,  
dsp=\Ew0|\E+,dsp=\Ew1|\E+,  
wy60-2|wyse60-2|wyse model 60 - second serial port  
dskb=A^M|Shift-F1|\Ed#A^r^T\Ee8,  
dskb=Aa^M|Shift-F2|\Ed#Aa^r^T\Ee8,  
dskb=Ab^M|Shift-F3|,  
dskc=\200|Ctrl-F1|,  
dske=\201|Ctrl-F2|\Ed#\201^T\Ew0\E+,  
dskl=\202|Ctrl-F3|,  
dsp=\Ew2|\E+,
```

dscreen must be run on both computers, with terminal type wy60-1 on the first computer and terminal type wy60-2 on the second computer (using the **-t** option to **dscreen**). The wy60-1 entry will be examined first.

The first two key entries are unchanged from the original wy60 entry. The third key, however, has type dskb, which means block both input and output. When this key is pressed, the sequence:

```
Esc d # Ctrl-A b CR Ctrl-T Esc e 9
```

is sent to the terminal; after this output is blocked and **dscreen** continues scanning input for key sequences but discards all other input.

The sequence Esc d # puts the terminal in transparent print mode, which echoes all characters up to a Ctrl-T out through the other serial port.

The characters Ctrl-A b CR are sent out the other serial port, informing the **dscreen** process on the other computer that it should activate the window associated with the Shift-F3 key.

The Ctrl-T key sequence exits the transparent print mode. The Esc 9 key sequence causes the terminal to switch to the other AUX serial port for data communications.

At this point, the other computer takes over, sends an Esc w 2 to switch to the third physical screen, and then resumes normal communication.

The wy60-2 entry follows the same general pattern for keys Shift-F1 and Shift-F2:

- Switch to transparent print mode
- Send function key string to other computer
- Switch transparent print off
- Switch to the other serial port

The end key, Ctrl-F2, works the same for both computers; it sends the end key sequence to the other computer through the transparent print mechanism, switches the terminal to window 0, clears the screen, then exits.

Modems

Modems provide serial communications across ordinary telephone lines. This section discusses modem standards, general modem setup, and specific configuration tips for popular modems.

Modem Overview

A *modem* is a device that allows you to connect one computer to another across ordinary telephone lines. The current telephone system is incapable of carrying the voltage changes required for a direct digital connection. A modem overcomes this limitation by modulating digital information into audio tones for transmission across the phone line, and by demodulating those tones back into digital information on reception. Modems are commonly used with Basic Network Utilities (BNU) or other implementations of the UNIX-to-UNIX Copy Program (UUCP). A high-speed (14,400 bps or greater) modem can be used with Serial Line Interface Protocol (SLIP) to provide Transmission Control Protocol/Internet Protocol (TCP/IP) connectivity as well.

Often, the term *baud* is used to refer to modem speed instead of bps. Baud is actually a measurement of the modulation rate. In older modems, only 1 bit was encoded in each signal change, so modem baud rate was equal to modem speed. Modems that operate at higher speeds, however, still generally operate at 2,400 (or even 1,200) baud, and encode two or more bits per signal change. A modem's bps rate is calculated by multiplying the number of data bits per signal with the baud (for example, 2,400 baud x 6 bits per signal change = 14,400 bits per second). Most modern modems can communicate at a variety of speeds (for example, 28,800, 14,400, 9,600, 7,800, 4,800, and 2,400 bps).

Telecommunications Standards

The older speeds of 300, 1,200, and 2,400 bps were well defined. However, as modem manufacturers began to devise methods for gaining higher speeds, each modem manufacturer started to use a proprietary method incompatible with modems from other manufacturers. Today, the ITU-TSS (formerly the United Nations Consultative Committee for International Telephony and Telegraphy, abbreviated CCITT) defines standards for most high-speed communications.

Even high-speed modems are much slower than other methods of computer communication. A high-speed modem can operate at 28,800 bps, but an Ethernet connection operates at 10,000,000 bps. To boost data throughput, high-speed modems typically offer one or more data compression algorithms. These algorithms can boost the throughput of a high-speed modem to speeds of 57,600 bps (if the data rate is 14,400 bps) or 115,200 bps (if the data rate is 28,800 bps). Note that these compression algorithms are sensitive to the data being transmitted. If the data has already been compressed (for example, with the

compress command), the data compression methods of high-speed modems offer little or no benefit, and might even reduce data throughput. When using a modem with data compression technology, the speed of the data terminal equipment/data circuit-terminating equipment (DTE/DCE) connection between the computer and the modem is equal or greater than the nominal data rate of the connection between modems. For example, with a V.32*bis* modem with V.42*bis* data compression, the data rate of the modem (the speed at which the modem communicates across telephone lines) is 14,400 bps. When the V.42*bis* compression is active, actual data throughput can reach 57,600 bps. To accommodate the greater throughput offered by data compression, the speed of the DTE/DCE between the computer and the modem should be set to 57,600 bps.

Attention: Modems implementing data compression and modern modulation schemes may yield a higher data throughput than some systems and asynchronous adapters can accommodate.

The ITU-TSS defines standards for high-speed communications, including data compression algorithms. ITU-TSS standards are usually named V.*nn*, where *nn* is a number. Another, slightly less common standard is the Microcom Networking Protocol (MNP). Available in versions (called classes) 1-9, MNP is a high-performance, high-speed protocol that was available relatively early, and became something of a de facto standard before the advent of the CCITT standards.

ITU-TSS Communications Standards

Following is a list of some common communications standards defined by the ITU-TSS. Note that this only a partial list. For a complete list, refer to the Internet website for the International Telecommunication Union.

V.29	ITU-TSS standard for half-duplex 9600 bps communications.
V.32	ITU-TSS standard for full-duplex 9600 bps communications.
V.32<i>bis</i>	ITU-TSS standard for 14,400 communications. V.32 <i>bis</i> is a revision to the V.32 standard.
V.34	ITU-TSS standard for 33,600 bps communications. Note that this standard achieves 33,600 bps data rates using multiple bit encoding, instead of the data compression scheme used by MNP Class 9. This standard was previously referred to as V. <i>fast</i> .
V.42	ITU-TSS error correcting procedures for DCEs using asynchronous to synchronous conversion.
V.42<i>bis</i>	Revised ITU-TSS data compression standard.

MNP Communications Standards

MNP Class 1	An asynchronous, half-duplex, byte-oriented method of transferring data realizing about 70% efficiency. Uncommon in modern modems.
MNP Class 2	A full-duplex counterpart to MNP Class 1. Uncommon in modern modems.
MNP Class 3	A synchronous, bit-oriented full-duplex method of transferring data realizing about 108% efficiency. (Efficiency greater than 100% is realized because the start/stop bits required for an asynchronous connection are eliminated. The DTE/DCE between the modem and the system are still asynchronous).
MNP Class 4	An enhancement to MNP Class 3 including a mechanism for varying the packet size (adaptive packet assembly) and a means of eliminating redundant administrative overhead (data phase optimization). An MNP Class 4 modem offers approximately 120% efficiency.
MNP Class 5	Class 5 includes data compression along with Class 4 features. An MNP Class 5 modem offers 200% efficiency.
MNP Class 6	MNP Class 6 allows incorporation of multiple, incompatible modulation techniques into one modem (universal link negotiation). This allows MNP Class 6 modems to begin communication at a slower speed and negotiate a transition to a higher speed. Class 6 also includes a statistical duplexing scheme that dynamically allocates utilization of half-duplex modulation to simulate full-duplex service. All features of MNP Class 5 are supported.
MNP Class 7	Incorporates enhanced data compression. Combined with Class 4, efficiencies of 300% can be realized.
MNP Class 8	N/A
MNP Class 9	Combines enhanced data compression with V.32 technology to allow data rates up to 28,800 bps.

Generic Modem Setup

To set up a modem:

1. Attach the modem with the appropriate cables.
2. Add a tty for the modem.
3. Configure the modem.

Attaching the Modem with Appropriate Cables

The first step in setting up a modem is to attach the modem with the appropriate cables. Part numbers and their descriptions are listed below.

6323741

Async Cable, EIA-232; used to attach all asynchronous devices; sometimes used with other cable assemblies.

59F3740

10 to 25-pin D-shell connector used to attach asynchronous cable 6323741 to native serial ports S1 and S2 as shown in the following figure.



Figure 34. 10 to 25-Pin Connector. This illustration shows a 10 to 25-pin connector.

59F3432

Cable P used to connect to 16-port concentrator. Part number includes four RJ45-to-DB25 converter cables.

Following are some examples of cable connections:

1. To attach a modem to native serial port S1, use the following cables:



Figure 35. Modem to Native Serial Port Cable Assembly. This illustration shows a 59F3740 cable on the serial port end and a 6323741 on the modem end.

2. To attach a modem to an 8-port async adapter (EIA-232) interface cable assembly, use the following cables:

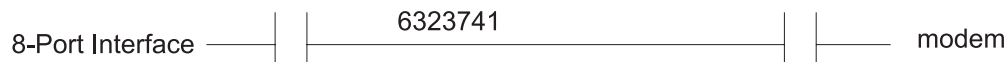


Figure 36. 8-Port Interface to Modem Cable Assembly. This illustration shows an 8-port interface connected to a modem with a 6323741 cable.

3. To attach a modem to a 16-port concentrator on a 64-port adapter, use the following cables:

:

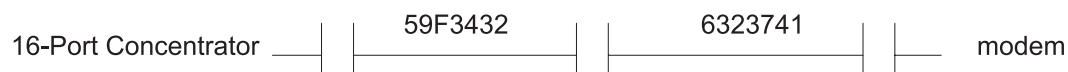


Figure 37. 16-Port Interface to Modem Cable Assembly. This illustration shows a 59F3432 cable on the serial port end and a 6323741 on the modem end.

Adding a TTY for the Modem

First, ensure that the system is turned on and that the modem is turned off. Use the Web-based System Manager, **wsm**, or the SMIT fast path **smit mktty**.

Configuring the Modem

Use only one of the two methods presented in this section for configuring the modem .

Sending AT Commands with the cu Command

If you have the Basic Network Utilities (BNU) installed, use the **cu** command to configure a modem as follows:

1. Add the following line to your **/usr/lib/uucp/Devices** file. Do not add the line if it is already in the file. This should work with most Hayes-compatible modems. (Replace # with the number for your port.)
Direct tty# - Any direct
2. Enter the following commands (read the notes about each command before performing the command itself):

```
pdisable tty#
```

```
cu -ml tty#
```

AT&F Restore the factory default configuration.

ATE1 In command state, echo characters from the keyboard to the screen. (Make sure carrier is not ON on the port or modem.)

Note: You need issue only one of the following commands.

AT&D2 Monitor the data terminal ready (DTR) signal. When an on-to-off transition of the DTR signal occurs, the modem hangs up and enters the command state.

AT&D3 Monitor the DTR signal. When an on-to-off transition of the DTR signal occurs, the modem hangs up and resets.

AT&W Write the storable parameters of the current configuration to modem memory.

AT&C1 Track the status of the carrier detect signal. (The modem might disconnect.)

Note: Issue the following command only if the previous command used the modem to disconnect.

```
cu -l tty#
```

ATS0=1 Put the modem in autoanswer mode.

ATS9=12

Set the carrier-detect response time. Default is 6. Possible values are 1 to 255.

AT&W Write the storable parameters of the current configuration to modem memory.

~. Terminate the **cu** session.

3. Enter *one* of the following commands:

```
penable tty#  
pshare tty#  
pdelay tty#  
pdisable tty#
```

The modem now has the basic configuration needed for most system communications. If you have problems, invoke **cu** with the **cu -dl** command to start a diagnostic trace on the connection.

Sending AT Commands Using a C Program

If the previous method failed, or if you do not have BNU installed, try running the following C program. Create a file called **motalk.c** containing the following code. Save the file. Compile and run it according to the instructions in the program comments.

```

/*****
/* MoTalk - A "C" program for modem setup.          */
/*      This program is meant as an aid only and is */
/*      not supported by IBM.                       */
/*      compile: cc -o motalk motalk.c              */
/*      Usage: motalk /dev/tty? [speed]             */
*****/
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <fcntl.h>
#include <termio.h>
FILE *fdr, *fdw;
int fd;
struct termio term_save, stdin_save;
void Exit(int sig)
{
    if (fdr) fclose(fdr);
    if (fdw) fclose(fdw);
    ioctl(fd, TCSETA, &term_save);
    close(fd);
    ioctl(fileno(stdin), TCSETA, &stdin_save);
    exit(sig);
}
main(int argc, char *argv[])
{
    char *b, buffer[80];
    int baud=0, num;
    struct termio term, tstdin;
    if (argc < 2 || !strcmp(argv[1], "-?"))
    {
        fprintf(stderr, "Usage: motalk /dev/tty? [speed]\n");
        exit(1);
    }
    if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0)
    {
        perror(argv[1]);
        exit(errno);
    }
    if (argc > 2)
    {
        switch(atoi(argv[2]))
        {
            case 300: baud = B300;
                    break;
            case 1200: baud = B1200;
                    break;
            case 2400: baud = B2400;
                    break;
            case 4800: baud = B4800;
                    break;
            case 9600: baud = B9600;
                    break;
            case 19200: baud = B19200;
                    break;
            case 38400: baud = B38400;
                    break;
            default: baud = 0;
        }
    }
}

```

```

        fprintf(stderr, "%s: %s is an unsupported baud\n", argv[0], argv[2]);
        exit(1);
    }
}
/* Save stdin and tty state and trap some signals */
ioctl(fd, TCGETA, &term_save);
ioctl(fileno(stdin), TCGETA, &stdin_save);
signal(SIGHUP, Exit);
signal(SIGINT, Exit);
signal(SIGQUIT, Exit);
signal(SIGTERM, Exit);
/* Set stdin to raw mode, no echo */
ioctl(fileno(stdin), TCGETA, &tstdin);
tstdin.c_iflag = 0;
tstdin.c_lflag &= ~(ICANON | ECHO);
tstdin.c_cc[VMIN] = 0;
tstdin.c_cc[VTIME] = 0;
ioctl(fileno(stdin), TCSETA, &tstdin);
/* Set tty state */
ioctl(fd, TCGETA, &term);
term.c_cflag |= CLOCAL|HUPCL;
if (baud > 0)
{
    term.c_cflag &= ~CBAUD;
    term.c_cflag |= baud;
}
term.c_lflag &= ~(ICANON | ECHO); /* to force raw mode */
term.c_iflag &= ~ICRNL; /* to avoid non-needed blank lines */
term.c_cc[VMIN] = 0;
term.c_cc[VTIME] = 10;
ioctl(fd, TCSETA, &term);
fcntl(fd, F_SETFL, fcntl(fd, F_GETFL, 0) & ~O_NDELAY);
/* Open tty for read and write */
if ((fdr = fopen(argv[1], "r")) == NULL )
{
    perror(argv[1]);
    exit(errno);
}
if ((fdw = fopen(argv[1], "w")) == NULL )
{
    perror(argv[1]);
    exit(errno);
}
/* Talk to the modem */
puts("Ready... ^C to exit");
while (1)
{
    if ((num = read(fileno(stdin), buffer, 80)) > 0)
        write(fileno(fdw), buffer, num);
    if ((num = read(fileno(fdr), buffer, 80)) > 0)
        write(fileno(stdout), buffer, num);
    Exit (0);
}
}

```

Hayes and Hayes-Compatible Modems

1. Change the tty settings, if necessary, using the Web-based System Manager, **wsm**, or the SMIT fast path, **smitt chtty**. For example, you might want to change the Enable LOGIN field to **Share** or **Enable**.
2. Add the following line to **/usr/lib/uucp/Systems** file:

```
hayes Nvr HAYESPROG 2400
```
3. Add this to **/usr/lib/uucp/Devices** file:

```
# For programming the hayes modem only:
HAYESPROG tty0 - 2400 HayesProgrm2400
#regular ACU entry:
ACU tty0 - Any hayes
```

4. Add this to `/usr/lib/uucp/Dialers` file:

```
# This Entry is used to PROGRAM the modem ONLY:
# the next 3 lines should be made into one:
HayesProgrm2400    =,-,    "" \d\dAT\r\c OK AT&F\r\c OK ATM1\r\c OK
AT&D3\r\c OK AT&K3&C1\r\c OK ATLOEQ2\r\c OK ATS0=1\r\c OK AT&W\r\c
OK
hayes    =,-,    "" \dAT\r\c OK ATDT\T\d\r\c CONNECT
```

5. To program the modem, enter the command `cu -d hayes`. This command uses the `cu` command to program the modem. Because no connection is made to another system, the command will fail. The modem is programmed if sendthem AT&W and then OK got it appear in the output.

If you are not doing binary file transfers or using BNU, leave out the `&K3` command, and set XON as the flow control to be used. However, it is more efficient to use hardware flow control (as opposed to XON-XOFF handshaking). To do that, use the settings and the `Dialers` entries from the next step.

6. After the modem is programmed, you can configure the system device driver to use hardware flow control. Using Web-based System Manager, `wsm` or SMIT (`smit chtty` fast path), change the flow control to RTS. Check your modem manuals to find out whether your modem supports hardware flow control.

Troubleshooting Modem Problems

This section attempts to identify common problems when using a modem with your computer.

Symptom	Cause	Solution
The modem (or other device attached to the serial port) causes the system to gradually slow down and eventually hang. Turning off the device usually lets the system function normally again.	An intelligent modem has CD always ON. The system senses this and sends a login herald, which the modem tries to interpret as a command. The modem fails to recognize the login herald as a valid command, and echoes back to the tty port on the system. This cycle repeats continuously.	Set the tty port to delay on the system so no login herald will be sent. With this setting, only a valid carriage return character from the host logging in will cause a login herald to be sent. You can also change the modem's AT set profile to set CD to ON only when a valid carrier is sensed on the telephone line.

Software Services Modem Questionnaire

Before calling for assistance with modem problems, please collect the following information:

- System Xtra?
- Level of the operating system. How long have you been at this level of the operating system?
- Has the modem ever worked before?
- What type of modem are you using? What type of modem is on the other end of the telephone connection?
- To what adapter type (64-port, 128-port, S1,...) is the modem attached?
- To which port number is the modem attached?
- To which tty number is the modem attached?
- What type of cabling are you using?
- What is the login setting (share, delay, enable)?
- Can the modem connect to other modems?
- Can other modems connect to your modem?

- What are the following values in Web-based System Manager, SMIT, modem, or port?
 - XON/XOFF?
 - RTS/CTS?
 - BPS rate?
- Include the following in your problem description:
 - Does the port lock intermittently?
 - Can you dial out? Can others dial in?
 - Any other specific and descriptive error conditions.
- Are there errors on the console? What are they?
- Are there errors in the error report? (**errpt** or **errpt -a**)
- What command are you using to dial out?
- What software is involved on the system?

AT Command Summary

The following is a summary of the Hayes Smartmodem command set. These commands comprise the AT command set used by many popular modems. This information comes from the Hayes Smartmodem 2400 *Quick Reference Card*, published by Hayes Microcomputer Products, Inc. Consult the modem documentation for a list of relevant AT commands.

AT	Command prefix - precedes command line.
<CR>	Carriage return (newline) character - terminated the command line.
A	Go off-hook, remain in command mode.
A/	Repeat previous command line. This command is not preceded with AT or followed by <CR>/ .
B0	Select CCITT V.22 standard for 1200 bps communications.
B1	Select Bell 212A standard for 1200 bps communications.
D	Enter originate mode, dial the number that follows, and attempt to go online. D is usually followed by T for tone, P for pulse may also be used.
DS=<i>n</i>	Dial the number stored in location <i>n</i> .
E0	Disable character echo in the command state.
E1	Enable character echo in the command state.
H0	Go on-hook (hang up the phone).
H1	Operate switch-hook and auxiliary relay.
I0	Return product identification code.
I1	Perform checksum on firmware ROM; return checksum.
I2	Perform checksum on firmware ROM; returns OK or ERROR as the result.
L0	Speaker off.
L1	Low speaker volume.
L2	Medium speaker volume.
L3	High speaker volume.
M0	Speaker off.
M1	Speaker on until carrier detected.
M2	Speaker always on.
M3	Speaker on until carrier detected, except during dialing.
O0	Enter online state.
O1	Enter online state and initiate equalizer retrain.
Q0	Modem returns result codes.
Q1	Modem does not return result codes.
Sr	Set pointer to register r.
Sr=<i>n</i>	Set register r to value <i>n</i> .
V0	Display result codes in numeric form.
V1	Display result codes in verbose form (as words).
X0	Enable features represented by result codes 0-4.

X1	Enable features represented by result codes 0-5, 10.
X2	Enable features represented by result codes 0-6, 10.
X3	Enable features represented by result codes 0-5, 7, 10.
X4	Enable features represented by result codes 0-7, 10.
Y0	Disable long space disconnect.
Y1	Enable long space disconnect.
Z	Reset modem
&C0	Assume data carrier always present.
&C1	Track presence of data carrier.
&D0	Ignore DTR signal.
&D1	Assume command state when an on-to-off transition of DTR occurs.
&D2	Hang up and assume command state when an on-to-off transition of DTR occurs.
&D3	Reset when an on-to-off transition of DTR occurs.
&F	Recall the factory settings as the active configuration.
&G0	No guard tone.
&G1	500 Hz guard tone.
&G2	1800 Hz guard tone.
&J0	RJ-11/RJ41/RJ45S telco jack.
&J1	RJ-11/RJ-13 telco jack.
&P0	Pulse dial with make/break ratio 39/61.
&P1	Pulse dial with make/break ratio 33/67.
&Q0	Operate in asynchronous mode.
&Qn	Operate in synchronous mode <i>n</i> .
&R0	Track CTS according to RTS.
&R1	Ignore RTS; always assume presence of CTS.
&S0	Assume presence of DSR signal.
&S1	Track presence of DSR signal.
&T0	Terminate test in progress.
&T1	Initiate local analog loopback.
&T3	Initiate digital loopback.
&T4	Grant request from remote modem for remote data link (RDL).
&T5	Deny request from remote modem for RDL.
&T6	Initiate remote digital loopback.
&T7	Initiate remote digital loopback with self-test.
&T8	Initiate local analog loopback with self-test.
&V	View active configuration, user profiles, and stored numbers.
&Wn	Save storable parameters of active configuration as user profile <i>n</i> .
&X0	Modem provides transmit clock signal.
&X1	Data terminal provides transmit clock signal.
&X2	Receive carrier provides transmit clock signal.
&Yn	Recall user profile <i>n</i> .
&Zn=x	Store phone number <i>x</i> in location <i>n</i> .

S-Register Summary

Register	Range	Description
S0	0-255	Select number of rings before answer.
S1	0-255	Ring count (incremented with each ring).
S2	0-127	Define escape sequence character (ASCII).
S3	0-127	Define carriage return character (ASCII).
S4	0-127	Define line feed character (ASCII).

S5	0-32, 127	Define backspace character (ASCII).
S6	2-255	Select wait-time in seconds before blind dialing.
S7	1-55	Select wait-time in seconds for carrier/dial tone.
S8	0-255	Select duration in seconds of comma.
S9	1-255	Carrier detect response time in .1 second increments (10 = 1 second).
S10	1-255	Delay between carrier loss and hangup in .1 second increments.
S11	50-255	Duration/spacing of tones in milliseconds.
S12	50-255	Escape sequence guard time in .02 second intervals.
S13	—	Reserved.
S14	—	Reserved.
S15	—	Reserved.
S16	—	Reserved - functions for this register are controlled by the &T commands).
S17	—	Reserved.
S18	0-255	Test timer duration in seconds.
S19	—	Reserved.
S20	—	Reserved.
S21	—	Reserved.
S22	—	Reserved.
S23	—	Reserved.
S24	—	Reserved.
S25	0-255	Select DTR change detect time in .01 second intervals.
S26	0-255	RTS to CTS delay in .01 second intervals.
S27	—	Reserved.

Result Codes Summary

Number	Word	Description
0	OK	Command executed.
1	CONNECT	Connection established at 0-300 bps.
2	RING	Ring signal detected.
3	NO CARRIER	Carrier signal lost or not detected.
4	ERROR	Invalid command, checksum, error in command line, or command line too long.
5	CONNECT 1200	Connection established at 1200 bps.
6	NO DIALTONE	No dial tone detected.
7	BUSY	Busy signal detected.

8	NO ANSWER	No response when dialing a system.
9	CONNECT 2400	Connection established at 2400 bps.

Dial modifiers

The following lists and describes dial modifiers:

0-9 # * A-D	Digits and characters for dialing.
P	Pulse dial.
T	Tone dial.
,	Delay processing of next character.
!	Hookflash.
@	Wait for silence.
W	Wait for dial tone.
;	Return to command state after dialing.
R	Reverse mode.
S=n	Dial number stored at location <i>n</i> .

ATE Overview

The Asynchronous Terminal Emulation (ATE) program, an optional software product, enables a system to emulate a terminal on a remote system. Using ATE, you can log in to most systems that support asynchronous terminals, including any system that supports RS-232C or RS-422A connections. You can set ATE parameter values so the remote system recognizes your terminal as either an attached workstation terminal or a DEC VT100 terminal.

Setting Up ATE Overview

Before you run ATE, you must install the software and set up the ports and connections. ATE uses both direct (cabled) connections and modem connections. Local RS-232C connections allow a maximum distance of 15 meters (50 feet) between machines, and RS-422A connections allow up to 1200 meters (4000 feet) between machines.

Before you use ATE to call a remote system, be sure that the remote system tty device is ready to accept a call. If another user on the remote system uses ATE to call your terminal, be sure your tty device is ready to accept the call.

See Setting Up ATE for more information about installing and setting up ATE.

Note: You must be a member of a UNIX-to-UNIX Copy Program (UUCP) group to use ATE. A user with root authority can use Web-based System Manager or the System Management Interface Tool (SMIT) to set up a UUCP group.

Customizing ATE

The first time you run ATE, the program creates an **ate.def** default file in the current directory. The **ate.def** file contains parameters the ATE program uses for:

- Data transmission characteristics
- Local system features
- Dialing directory file
- Control keys.

To change the defaults, edit the **ate.def** file.

If you need to run ATE with different settings, you can maintain **ate.def** files in different directories. You can then run ATE from the appropriate directory depending on the settings needed for specific sessions. However, running ATE from many directories requires multiple copies of the **ate.def** file, which uses system storage.

See *How to Edit the ATE Default File in AIX 5L Version 5.1 System User's Guide: Communications and Networks* for details about editing the **ate.def** file.

You can temporarily change settings without modifying the default file. To do this, use the **alter** and **modify** subcommands. Settings you change with the **alter** or **modify** subcommand remain in effect until you exit the program with the **quit** subcommand. When you exit ATE, the settings return to the defaults set in the **ate.def** file.

When installed, ATE uses the **/usr/lib/dir** system-wide dialing directory file. You can temporarily change settings in the dialing directory file for a specific modem connection. Settings changed in this way revert to the default when the connection ends, rather than when you exit ATE. A user with root authority can modify the **/usr/lib/dir** file to include numbers for modems used by everyone on the system. Individual users can also create their own dialing directory files and modify their copies of the **ate.def** file to make ATE use those directories.

How to Set up an ATE Dialing Directory in AIX 5L Version 5.1 System User's Guide: Communications and Networks explains how to set up ATE to use a customized dialing directory.

You can edit the dialing directory file to include frequently used phone numbers. Additionally, you can change the baud rate, data character length, stop bits, parity, echoing, and line-feeds for a phone number if these characteristics differ from the defaults. If a number is not in the directory file, you can complete the connection by using the **connect** subcommand.

Note: A dialing directory file can contain up to 20 lines (one entry per line). ATE ignores subsequent lines.

Changing ATE Characteristics

The following table identifies the ATE characteristics that the user can change and the appropriate methods for changing each characteristic.

Note: All ATE characteristics can be changed in the **ate.def** file.

Changing ATE Characteristics	
Characteristic	Change with
Control keys	ate.def file
Data character length	alter subcommand or dialing directory entry
Dialing directory file name	directory subcommand
Echoing (on or off)	modify subcommand or dialing directory entry
File name for incoming data (capture file)	modify subcommand
Final dial suffix for the modem	alter subcommand
Initial dial prefix for the modem	alter subcommand
Line feeds	modify subcommand or dialing directory entry
Number of redialing attempts	alter subcommand
Number of stop bits	alter subcommand or dialing directory entry

Parity (even or odd)	alter subcommand or dialing directory entry
Port name (device)	alter subcommand
Rate (bits per second)	alter subcommand or dialing directory entry
Telephone number	dialing directory entry
Transfer protocol (pacing or xmodem)	alter subcommand
Type of pacing (character or interval)	alter subcommand
VT100 emulation (on or off)	modify subcommand
Wait time between redialing attempts	alter subcommand
Write (capture) incoming data to a file	modify subcommand
Xon/Xoff protocol (on or off)	modify subcommand

Setting Up ATE

This article provides information on setting up Asynchronous Terminal Emulation (ATE).

Prerequisites

- The ATE program must be installed on the system. ATE is an optional program product.
- The user must have root user authority to set up the port for the communications device.

Procedure

To prepare ATE to run on the system:

1. Install an asynchronous adapter card in an appropriate slot in the system unit, unless the system has a built-in serial port.
2. Plug the RS-232C or RS-422A cable into the adapter card or the built-in serial port.
3. Add a tty device for the communications port. To do this, use the Web-based System Manager, **wsm**, or enter:

```
smit mktty
```
4. Select Add a TTY.
5. Select the tty type.
6. Select a parent adapter
7. Select a port.
8. Set the Enable LOGIN field to **disable**.
9. Set Terminal Type to **HFT** or **dumb**.
10. Make the necessary 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 National Language Support (NLS) is required.
11. Set up the port for the device.
 - To set up a port to call out with ATE, use the **pdisable** command. For example, to set up port `tty1`, enter:

```
pdisable tty1
```
 - To set up a port so that others can call in, use the **penable** command. For example, to let other systems call in to the `tty2` port, enter:

```
penable tty2
```
12. Ensure the device has previously been defined to the remote system. Once the device is defined, the ATE program must be customized to reflect the device settings on the remote system. Customize the

default settings with the **alter** and **modify** subcommands or by editing the **ate.def** default file. To change the default settings for a telephone connection, use a dialing directory file entry.

TTY Troubleshooting

This section discusses troubleshooting the tty subsystem:

- Respawning Too Rapidly Errors
- Error Log Information and tty Log Identifiers

Respawning Too Rapidly Errors

The system records the number of getty processes spawned for a particular tty in a short time period. If the number of getty processes spawned in this time frame exceeds five, then the Respawning Too Rapidly error is displayed on the console and the port is disabled by the system.

The tty stays disabled for about 19 minutes or until the system administrator enables the port again. At the end of the 19 minutes, the system automatically enables the port, resulting in the spawning of a new **getty** process.

Possible Causes

- Incorrect modem configuration
- A port is defined and enabled but no cable or device is attached to it
- Bad cabling or loose connection
- Noise on communication line
- Corruption of, or tampering with, **/etc/environment** or **/etc/inittab** files
- tty configuration is corrupted
- Hardware is defective

Procedures for Recovery

- Correct modem configuration:

Ensure that the modem carrier detect is *not* forced high.

Note: The following applies to Hayes-compatible modems.

1. Connect to the modem and examine the active profile.
2. Set the modem carrier detect to **&C1** rather than **&C0** (forced high). Use the following AT modem commands to set and change the carrier attribute:

```
AT&C1
AT&W
```

Notes:

- a. See Sending AT Commands with the cu Command
 - b. See your modem documentation for further information.
- Disable the tty, remove the tty definition, or attach a device to the port:
 - To disable the tty definition use the **chdev** command as follows:
`chdev -l ttyName -a Login=disable`

After running this command, the tty does *not* become enabled after a system restart.

- To remove the tty definition:
 1. Disable the tty port, use the **pdisable** command, enter:
`pdisable ttyName`
 2. Remove the tty definition from the system. See Managing TTY Devices for further information.

- Check for bad cables or loose connections:
 1. Check cabling. Tighten loose connections and replace damaged or inappropriate connectors.
 2. Verify that the suspected cabling is IBM serial cable P/N 6323741 or that the cable meets the same standard. Replace damaged or inappropriate cables.
- Eliminate noise on communication line:
 1. Verify that cabling is correct length and impedance.
 2. Ensure that toroid rings are in place where needed on longer cables.
 3. Check routing of cables; they should not be close to fluorescent lights or motors.
- Check for corruption of, or tampering with, the **/etc/environment** or the **/etc/inittab** files:
 1. If possible, compare these files against known good copies.
 2. Copy the files as a backup and make changes as needed.
 3. In the **/etc/environment** file, remove any lines that are *not*:
 - blank lines
 - comment lines
 - *variable=value*
 4. In the **/etc/inittab** file, examine the tty devices lines. If the tty is set to off, it is likely that the tty port is not being used. If it is not being used, remove the tty definition or attach a device to the port.
- Remove corrupted tty configuration:
 1. Remove the tty definition. Use the Web-based System Manager Devices application or see Managing TTY Devices for further information.
 2. If you want a hard copy record of the tty definition before removing it, press the Image key (F8 or Esc+8). This will capture the current screen image and copy it to the **smit.log** file in your **\$HOME** directory.
 3. Read the tty definition. See the instructions for Adding a TTY under Managing TTY Devices.
- Locate defective hardware:
 1. Run diagnostics using the **diag** command.
 2. If any hardware problems are detected, follow local problem solving procedures.

Error Log Information and TTY Log Identifiers

The following sections discuss important error logging files and commands and common error report messages relating to ttys.

Important Error Logging Files and Commands

Command: **errclear**

This command deletes entries from the error log. The entire log can be erased with `errclear 0` or entries with specified error ID numbers, classes, or types can be removed.

Command: **errpt**

This command generates an error report from entries in the system error log. The most used format for this command is `errpt -a | pg`, which generates a detailed report starting with the most current errors.

File: **/var/adm/ras/errlog**

This file stores instances of errors and failures encountered by system. The **errlog** file tends to become quite lengthy. If not cleared on a regular basis, it can occupy quite a bit of space on your hard disk. Use the **errclear** command mentioned previously to clean out this file.

File: **/usr/include/sys/errids.h**

The **errids.h** header file correlates error IDs with error labels.

Common Error Report Messages

Message	Description	Comments
Core Dump	Software program abnormally terminated	This error is logged when a software program abnormally ends and causes a core dump. Users might not be exiting applications correctly, the system might have been shut down while users were working in application, or the user's terminal might have locked up and the application stopped.
Errlog On	Errdaemon turned on	This error is logged by the error daemon when the error logging is started. The system automatically turns off error logging during shutdown.
Lion Box Died	Lost communication with 64-port concentrator	This error is logged by the 64-port concentrator driver if communications with the concentrator are lost. If you receive this error, check the date and time stamp to see if user might have caused this message to occur. A series of these errors can indicate a problem with the 64-port adapter or its associated hardware.
Lion Buffero	Buffer overrun: 64-port concentrator	This error occurs when the hardware buffer in a 64-port concentrator is overrun. If device and cabling allow, try adding request to send (RTS) handshaking to the port and device. Also try lowering the baud rate.
Lion Chunknumc	Bad chunk count: 64-port controller	This error occurs when the value for the number of characters in a chunk does not match the actual values in the buffer. This error may indicate a problem with the hardware; try running diagnostics on devices.
Lion Hrdwre	Cannot access memory on 64-port controller	This error is logged by the 64-port concentrator driver if it is unable to access memory on the 64-port controller.
Lion Mem ADAP	Cannot allocate memory: ADAP structure	This error is logged by the 64-port concentrator driver if the malloc routine for the adap structure fails.
Lion Mem List	Cannot allocate memory: TYP_T List	This error is logged by the 64-port concentrator driver if the malloc routine for the <i>typ_t</i> list structure fails
Lion Pin ADAP	Cannot pin memory: ADAP structure	This error is logged by the 64-port concentrator driver if the pin routine for the adap structure fails.

SRC	Software program error	This error is logged by the System Resource Controller (SRC) daemon in the event of some abnormal condition. Abnormal conditions are divided in three areas: failing subsystems, communication failures, and other failures.
Lion Unkchunk	Unknown error code from the 64-port concentrator	Error Code: Number of characters in the chunk received.
TTY Badinput	Bad ttyinput return	Error Code: System error code (see sys/errno.h). This is logged by the tty driver if ttyinput routine returns an error.
TTY Overrun	Receiver overrun on input	The sending device is ignoring flow control and overloading the hardware buffer on the adapter. This occurs before the driver accesses the hardware FIFO. Try adding rts handshaking to the port and to device (if possible).
TTY TTYHOG	TTYHOG overrun	The sending device is ignoring flow control. This error occurs after the hardware first in/first out (FIFO) has been accessed and has written to the software buffer. In other words, the tty hog error is logged because the buffer in which incoming characters are placed is overflowing. When the buffer is about three-fourths full, the device driver notifies the hardware (in this case, tty) to send the XOFF char to the sender to stop sending data. If the sender continues to send data <i>and</i> the buffer is not emptied, the device driver flushes the buffer and logs the error message in the error log.
TTY Parerr	Parity/Framing error on input	This error indicates parity errors on incoming data to asynchronous ports on a character-by-character basis.
TTY Prog PTR	Software error: T_HPTR field invalid	This error is logged by the tty driver if <i>t_hptr</i> pointer is null.

Chapter 6. Micro Channel, ISA, and PCI Adapters

This chapter provides installation and configuration information for Micro Channel, ISA, and PCI adapters. Topics discussed are support and configuration for the following:

- Micro Channel adapters (Multiport/2 and Portmaster)
- ISA/PCI Wide Area Network (WAN) Adapters (Multiport Model 2 (ISA) , 2-Port Multiprotocol (PCI) , and ARTIC960HX(PCI)).

Micro Channel Wide Area Network (WAN) Adapters

This section explains the configuration requirements for the following Micro Channel adapters:

- Supported Multiport/2 Adapters
- Supported Portmaster Adapters
- Configuring Multiport/2 and Portmaster Adapters
- Device Driver Support

Supported Multiport/2 Adapters

Realtime Interface Co-Processor Multiport/2 adapters:

- 4-Port EIA 232D Multiport/2 adapter
- 4-Port EIA 232D/4-Port EIA 422A Multiport/2 adapter
- 8-Port EIA 232D Multiport/2 adapter
- 8-Port EIA 422A Multiport/2 adapter
- 6-Port Synchronous EIA 232D Multiport/2 adapter

Supported Portmaster Adapters

In Web-based System Manager, SMIT, and in the following information, these adapters are referred to generically as *Portmaster Adapter/A*:

- 4-Port Multiprotocol communications controller
- Realtime Interface Co-Processor Portmaster adapters:
 - 8-Port EIA 232D Portmaster Adapter/A
 - 8-Port EIA 422A Portmaster Adapter/A
 - 6-Port V.35 Portmaster Adapter/A
 - 6-Port X.21 Portmaster Adapter/A

Device Driver Support

The 4-Port Multiprotocol communications controller device driver is supplied with the operating system.

To use a Realtime Interface Co-Processor Multiport/2 adapter or a Realtime Interface Co-Processor Portmaster adapter, you will need additional software to provide device driver configuration support. One of the following is required:

- Realtime Interface Co-Processor licensed program
- A device driver from a business partner or a customer-written device driver

Configuring Multiport/2 and Portmaster Adapters

The following procedures explain how to configure device drivers and ports for a Multiport/2 or Portmaster communications controller (adapter).

Prerequisites

Install the communications controller (adapter).

Configuring Multiport/2 and Portmaster Adapters Tasks		
Task	SMIT Fast Path	Web-based System Manager Management Environment ³
Add a Device Driver	smit commodev ^{1,2}	
Add Ports on the Adapter	smit commodev , select adapter, then Manage Ports , then Add a Multiprotocol Port	
Reconfigure Ports on the Adapter	smit commodev , select adapter, then Manage Ports , then Change/Show Characteristics of a Port	
Remove a Port on the Adapter	smit commodev , select adapter, then Manage Ports , then Remove a Port	
Make a Defined Port Available	smit commodev , select adapter, then Manage Ports , then Configure a Defined Port	
Transfer Adapter Device Driver from Defined to Available	smit commodev , select adapter, then Manage Ports , then Configure a Defined Device Driver	
Remove Adapter Device Driver	smit commodev , select adapter, then Manage Ports , then Remove a Device Driver	

Notes:

1. This menu varies depending on the software you have installed. The device drivers for the 4-Port Multiprotocol communications controller are included with the basic operating system. To use another type of Multiport/2 adapter or Portmaster Adapter/2, you must have installed the Realtime Interface Co-Processor licensed program, or a device driver from a business partner or a customer-written device driver.
2. If you are using a device driver other than the 4-Port Multiprotocol communications controller device driver, consult the documentation for that device driver.
3. These tasks are not available in Web-based System Manager Management Environment.

ISA/PCI Wide Area Network (WAN) Adapters

This section explains the installation and configuration requirements for the Multiport Model 2 adapter (ISA), the 2-Port Multiprotocol adapter (PCI), and the ARTIC960HX PCI adapter.

Multiport Model 2 Overview

The Multiport Model 2 Adapter (MM2) device driver is a component of the communication I/O subsystem. This device driver provides support for the Multiport Model 2 at a maximum speed of 64K bps. The modems used must provide the clocking, since only external clocking is supported.

The following options provide access to the Multiport Model 2 device driver:

- Systems Network Architecture (SNA)
- The generic data link control (GDLC) Programming Interface
- User-written applications compatible with the MPQP-API (Multiprotocol Quad Port-Application Programming Interface).

These options require use of the **mpqi** special file, which allows access to the Multiport Model 2 adapter through the Multiport Model 2 device driver. The **mpqi** special file resides in the **/dev** directory.

Note: The *i* in **mpqi** specifies the instance of the device driver; for example, **mpq0**.

The Multiport Model 2 device driver allows connectivity to remote host systems using the Multiport Model 2 adapter, either directly over a leased line or over switched circuits, as shown in the "Host Connectivity over Switched Circuits" figure. The device driver can provide a gateway between work group environments and remote data processing facilities.

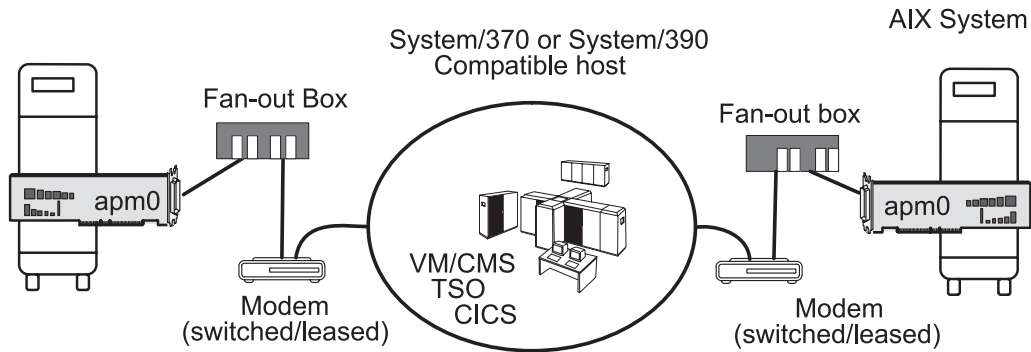


Figure 38. Host Connectivity Over Switched Circuits. This illustration depicts a system connected to a host via switched modems.

Direct connectivity to a system using the Multiport Model 2 adapter is also possible over a wide area network using switched modems (see "Switched Connectivity Using Switched Modems" figure), through a modem eliminator (see "Dedicated Connectivity Using Modem Eliminator" figure), or over leased lines (see "Leased Connectivity Using Leased Line" figure).

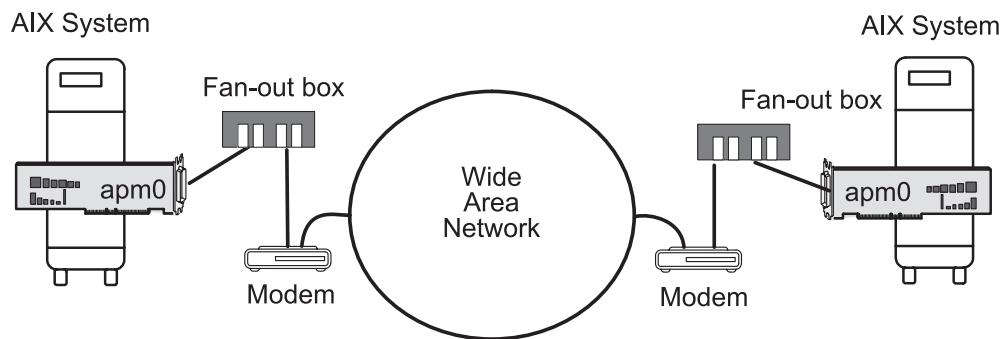


Figure 39. Switched Connectivity Using Switched Modems. This illustration depicts a system connected to a wide area network via switched modems.

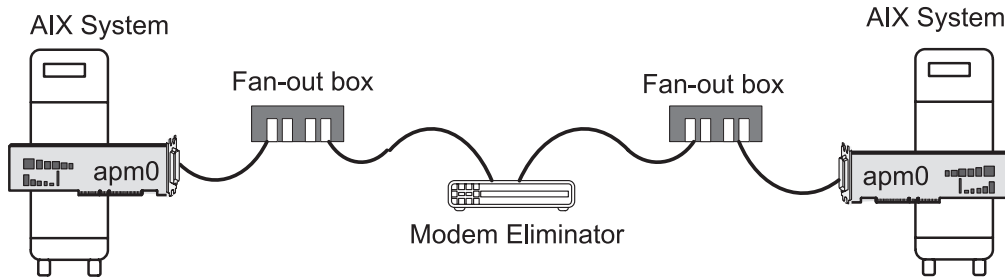


Figure 40. Dedicated Connectivity Using a Modem Eliminator. This illustration depicts a dedicated connection using a modem eliminator.

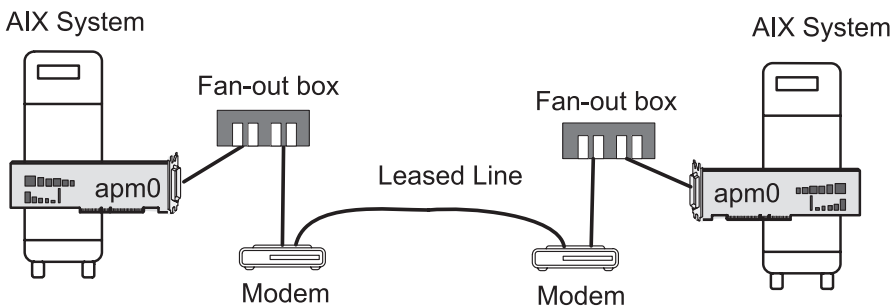


Figure 41. Leased Connectivity Using a Leased Line. This illustration depicts a direct connection using a leased line between two modems.

Configuring the Multiport Model 2 Adapter

The following procedures explain how to configure a Multiport Model 2 adapter.

Configuring the Multiport Model 2 Adapter Tasks			
Task	SMIT Fast Path	Command or File	Web-based System Manager Management Environment ³
Add a Multiport Model 2 Adapter ¹	smit makmm2	mkdev²	
List All Multiport Model 2 Adapters	smit lsmm2	lsdev²	
Change/Show a Multiport Model 2 Adapter	smit chgmm2	chdev²	
Configure a Defined Multiport Model 2 Adapter	smit cfgmm2	mkdev²	
Manage Device Drivers for Multiport Model 2 Adapters	smit mm2isa_dd		
Generate an Error Report	smit errpt		
Trace a Multiport Model 2 Adapter	smit mm2trace		
Remove a Multiport Model 2 Adapter	smit rmvmm2	rmdev²	
Manage DLC Services	smit mpaserv		

Configuring the Multiport Model 2 Adapter Tasks			
Add User-Written Applications	smit mm2apps		

Notes:

1. For information on adding ports, see "Configuring Multiport/2 and Portmaster Adapters".
2. Details about command line options are provided in the command descriptions for **mkdev**, **lsdev**, **chdev**, or **rmdev** in *AIX 5L Version 5.1 Commands Reference*.
3. These tasks are not available in Web-based System Manager Management Environment.

Multiport Model 2 Adapter Object Information and Attributes

The following sections provide information on Multiport Model 2 adapter object classes and attributes.

Predefined Device Object Information

The **Predefined Devices (PdDv)** object class contains entries for all known device types supported by the system. This information is modified only when a new device is added. The following tables contain information for the Multiport Model 2 adapter.

Device type:	portmaster
Device class:	adapter
Device subclass:	isa
Unique Type:	/adapter/isa/portmaster
Prefix name:	apm

Flag	Setting
Base Device	no
Vital Product Data (VPD)	no
Detectable	yes
Change Status	set to NEW when defining device
Bus Extender	no
Field Replaceable Unit (FRU)	yes

Predefined Connection Object Information

The **Predefined Connection (PdCn)** object class contains connection and dependency information for devices. The following table contains predefined connection information for the Multiport Model 2 adapter.

Connection Location:	0
Unique Type:	/adapter/isa/portmaster
Connection Key:	portmaster

Predefined Attribute Object Information

The **Predefined Attribute (PdAt)** object class contains an entry for each existing attribute for a device, beyond what is defined in the **Predefined Device** object class. The **PdAt** information for the Multiport Model 2 adapter includes the following definitions:

Attribute	Description	Possible Values
<i>bus_intr_lvl</i>	ISA Interrupt Level	3,4,7,9,10,11,12
<i>bus_io_addr</i>	Bus IO Address	0x2A0-0x3EA0, skip by 0x400
<i>bus_mem_addr</i>	Bus Memory Address	0xC0000-0xDE0000, skip by 0x200
<i>window_size</i>	Window Size	0x2000 (nonchangeable)
<i>intr_priority</i>	Interrupt Priority	2 (nonchangeable)

Subtypes:

- 4-Port Selectable Multiport Model 2 Adapter
- Multiport Model 2 Adapter
- 6-Port Multiport Model 2 Adapter (V.35)
- 6-Port Multiport Model 2 Adapter (X.21)
- 8-Port Multiport Model 2 Adapter (RS-232)
- 8-Port Multiport Model 2 Adapter (RS-422)

Note: Currently, only the 4-Port Selectable Multiport Model 2 Adapter is supported.

Multiport Model 2 Power Management

The device driver for the Multiport Model 2 adapter provides for Power Management (PM) support. PM is a technique that enables hardware and software to minimize system power consumption. PM is generally only important to low-end models, such as Notebooks and systems operating on battery.

When Power Management is enabled, the system enters a power-saving mode under one of the following conditions:

- Expiration of the idle timer
- A direct command from the user
- A low battery
- Closing of the Notebook lid

PM state transitions include the following:

- **enable**
- **standby**
- **suspend**
- **hibernation**
- **shutdown**

Each transition implies a further decrease in the power supply to the various system components.

Refer to "Using Power Management" in *AIX 5L Version 5.1 System Management Guide: Operating System and Devices* for detailed information on Power Management and its configuration and function.

Impact to External Connection - Network Provider

Bringing a system to the **suspend**, **hibernation**, or **shutdown** states results in loss of power to the adapters. All network connections are lost. Externally, this can be viewed as pulling the physical connection (cable), since it results in loss of signal at the physical layer. A data set ready (DSR) error is generated. Any application that has the device driver opened should be stopped before suspending, hibernating, or shutting down the system.

Once power is restored, all previously configured ports are restored to an **available** state. At that time, any applications previously using these connections can be restarted.

2-Port Multiprotocol HDLC Network Device Driver Overview

The 2-Port Multiprotocol Adapter high-level data link control (HDLC) device driver is a component of the communication I/O subsystem. This device driver provides support for the HDLC operation over the 2-Port Multiprotocol adapter at speeds up to 64K bps.

The following options provide access to the 2-Port Multiprotocol HDLC Network device driver:

- Systems Network Architecture (SNA)
- The synchronous data link control (SDLC) version of the GDLC Programming Interface
- User-written applications compatible with the SDLC MPQP-API (Multiprotocol Quad Port-Application Programming Interface)

Note: The above options require use of the **mpqn** special file, which allows access to the 2-Port Multiprotocol adapter's HDLC device driver through the SDLC COMIO device driver emulation subsystem. This subsystem must be installed and configured for each HDLC network device.

- User-written applications compatible with the HDLC Common Data Link Interface (CDLI) API

Configuring the 2-Port Multiprotocol Adapter

The following table explains how to configure a 2-Port Multiprotocol adapter.

Configuring the 2-Port Multiprotocol Adapter Tasks		
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Web-based System Manager Management Environment¹</i>
Add a Device Driver to the Adapter	smit mkhdlcdmpdd	
Reconfigure the Device Driver on the Adapter	smit chhdlcdmpdd	
Remove a Device Driver on the Adapter	smit rmhdlcdmpdd	
Make a Defined Device Driver Available	smit cfghdlcdmpdd	
Add an SDLC COMIO Emulator on the Adapter	smit mksdlcsciedd	
Reconfigure the SDLC COMIO Emulator on the Adapter	smit chsdlcsciedd	
Remove an SDLC COMIO Emulator on the Adapter	smit rmsdlcsciedd	
Make a Defined SDLC COMIO Emulator Available	smit cfgsdlcsciedd	

Note:

1. These tasks are not available in Web-based System Manager Management Environment.

ARTIC960HX PCI Adapter Overview

The ARTIC960HX PCI Adapter MPQP COMIO device driver emulator is a component of the communication I/O subsystem. This device driver provides support for the ARTIC960HX PCI adapter at a maximum speed of 2M bps. The modems used must provide the clocking, since only external clocking is supported.

The following options provide access to the Multiprot Model 2 device driver:

- Systems Network Architecture (SNA)
- The generic data link control (GDLC) Programming Interface
- User-written applications compatible with the MPQP-API (Multiprotocol Quad Port-Application Programming Interface), such as bisync applications.

These options require use of the **mpqx** special file, which allows access to the ARTIC960HX PCI adapter through the MPQP COMIO emulation device driver. This device driver must be installed and configured for each port on the ARTIC960HX PCI adapter. The **mpqx** special file resides in the **/dev** directory.

Note: The x in **mpqx** specifies the instance of the device driver; for example, mpq0.

The MPQP COMIO emulation device driver allows connectivity to remote host systems using the ARTIC960HX PCI adapter, either directly over a leased line. The device driver can provide a gateway between work group environments and remote data processing facilities.

Configuring the MPQP COMIO Emulation Driver over the ARTIC960HX PCI Adapter

The following table explains how to configure the MPQP COMIO Emulation Driver over the ARTIC960HX PCI Adapter.

Configuring the MPQP COMIO Emulation Driver Tasks		
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Web-based System Manager Management Environment¹</i>
Add a Device Driver	smit mktsdd	
Reconfigure the MPQP COMIO Emulation Driver	smit chtsdd	
Remove a Device Driver	smit rmtsdd	
Configure a Defined Device Driver	smit cfghtsdd	
Add a Port	smit mktsdports	
Reconfigure an MPQP COMIO Emulation Port	smit chtsdports	
Remove a Port	smit rmtsdports	
Configure a Defined Port	smit cfghtsdports	
Trace the MPQP COMIO Emulation Driver	smit trace_link	

Note:

1. These tasks are not available in Web-based System Manager Management Environment.

Chapter 7. Data Link Control

Generic data link control (GDLC) is a generic interface definition that allows application and kernel users a common set of commands to control data link control (DLC) device managers within the operating system. For problem determination, see GDLC Problem Determination in *AIX 5L Version 5.1 Communications Programming Concepts*. Topics discussed in this chapter are:

- Generic Data Link Control Environment Overview
- Implementing the GDLC Interface
- Installing GDLC Data Link Controls
- GDLC Interface ioctl Entry Point Operations
- GDLC Special Kernel Services
- Managing DLC Device Drivers

Generic Data Link Control Environment Overview

Generic data link control (GDLC) is a generic interface definition that provides application and kernel users a common set of commands to control DLC device managers within the operating system.

For more information about the GDLC environment, see:

- Implementing GDLC Interface
- Installing GDLC Data Link Controls
- GDLC Interface ioctl Entry Point Operations
- GDLC Special Kernel Services

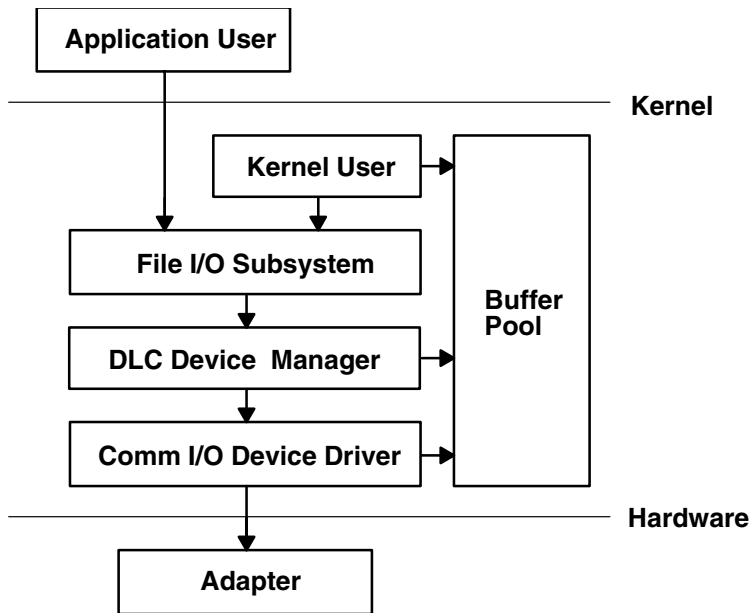
The GDLC interface specifies requirements for entry point definitions, functions provided, and data structures for all DLC device managers. DLCs that conform to the GDLC interface include:

- 8023 (IEEE 802.3 for Ethernet)
- ETHER (Standard Ethernet)
- SDLC (Synchronous Data Link Control)
- TOKEN (Token-Ring)
- FDDI (Fiber Distributed Data Interface)

DLC device managers perform higher-layer protocols and functions beyond the scope of a kernel device driver. However, the managers reside within the kernel for maximum performance and use a kernel device driver for their I/O requests to the adapter. A DLC user is located above or within the kernel.

Synchronous data link control (SDLC) and IEEE 802.2 Data Link Control are examples of DLC device managers. Each DLC device manager operates with a specific device driver or set of device drivers. SDLC, for example, operates with the multiprotocol device driver for the system's product and its associated adapter.

The basic structure of a DLC environment is shown in the "DLC Device Manager Environment" figure. Users within the kernel have access to the communications memory buffers (mbufs) and call the add entry points through the **fp** kernel services. Users above the kernel access the standard interface-to-kernel device drivers, and the file system calls the **dd** entry points. Data transfers require a move of data between user and kernel space.



DLC Device Manager Environment

Figure 42. DLC Device Manager Environment. This illustration shows the link between the application user and the adapter (hardware level). The areas in between are Kernel User, File I/O Subsystem, DLC Device Manager, Comm I/O Device Driver, and the Buffer Pool. These "in-between" entities are at the Kernel Level.

The components of the DLC device manager environment are:

Application User	Resides above the kernel as an application or access method.
Kernel User	Resides within the kernel as a kernel process or device manager.
File I/O Subsystem	Converts the file-descriptor and file-pointer subroutines to file pointer accesses of the switch table.
Buffer Pool	Provides data-buffer services for the communications subsystem.
Comm I/O Device Driver	Controls hardware adapter I/O and direct memory access (DMA) registers, and routes receive packets to multiple DLCs.
Adapter	Attaches to the communications media.

A device manager written in accordance with GDLC specifications runs on all the operating system hardware configurations containing a communications device driver and its target adapter. Each device manager supports multiple users above as well as multiple device drivers and adapters below. In general, users operate concurrently over a single adapter, or each user operates over multiple adapters. DLC device managers vary based on their protocol constraints.

The "Multiple User and Multiple Adapter Configuration" figure illustrates a multiple user configuration:

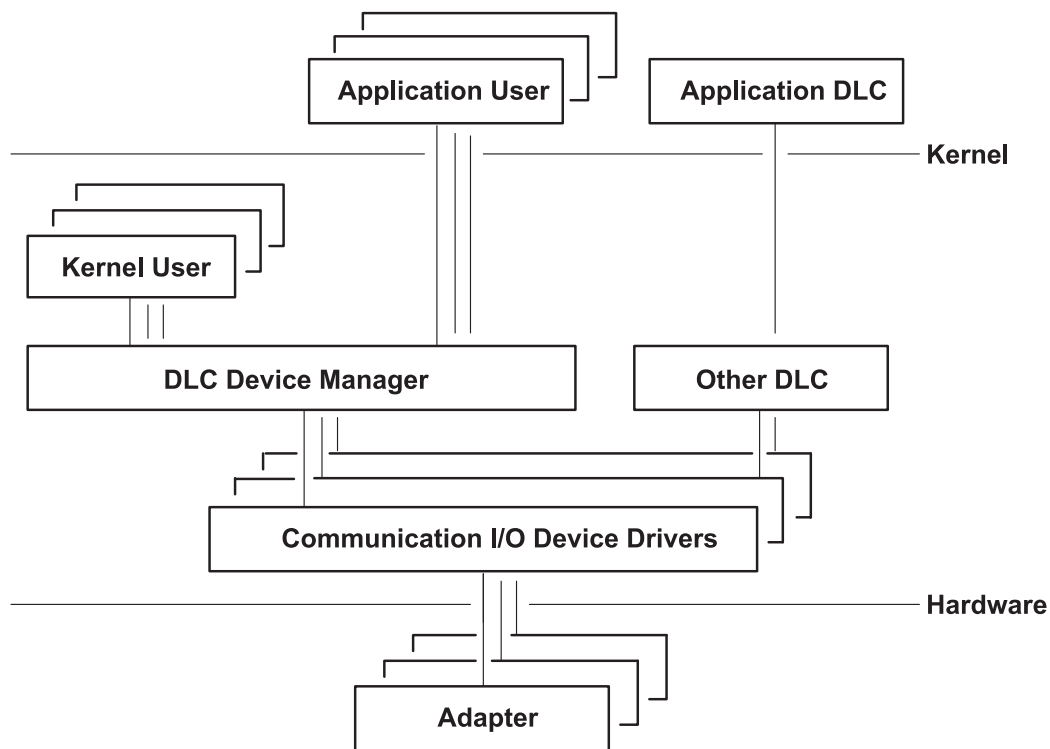


Figure 43. Multiple User and Multiple Adapter Configuration. This illustration is another view of the kernel level between the application user and the adapter. It shows multiple entities representing multiple users.

Meeting the GDLC Criteria

A GDLC interface must meet the following criteria:

- Be flexible and accessible to both application and kernel users.
- >Have multiple user and multiple adapter capability, allowing protocols to take advantage of multiple sessions and ports.
- Support both connection-oriented and connectionless services where possible.
- Allow transparent data transfer for special requirements beyond the scope of the DLC device manager in use.

Implementing the GDLC Interface

Each DLC device manager is a standard `/dev` entry operating in the kernel as a multiplexed device manager for a specified protocol. For an adapter not in use by DLC, each **open** subroutine to a DLC device manager creates a kernel process. An **open** subroutine is also issued to the target adapter device handler. If needed, issue additional **open** subroutines for multiple DLC adapter ports of the same protocol. Any **open** subroutine targeting the same port does not create additional kernel processes, but links the **open** subroutine with the existing process. There is always one kernel process for each port in use.

The internal structure of a DLC device manager has the same basic structure as a kernel device handler, except that a kernel process replaces the interrupt handler in asynchronous events. The read, write, I/O control, and select blocks function as shown in the "Standard Kernel Device Manager" figure.

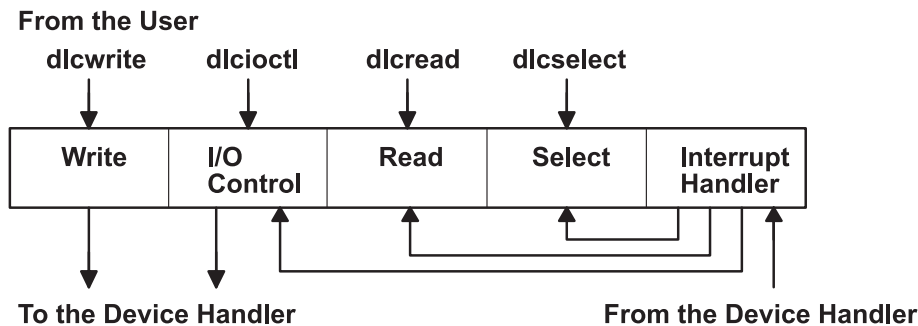


Figure 44. Standard Kernel Device Manager. This illustration shows the internal structure of a DLC device manager. This structure consists of a Write, I/O Control, Read, Select, and Interrupt Handler. The Device Manager receives information from the user where it is passed to the various areas before it is passed on to the Device Handler.

Installing GDLC Data Link Controls

You can install DLCs separately or in a group. A DLC device manager is automatically added into the kernel and set to the "Available" state for each type of DLC installed. Installation can be verified by issuing the **lspp** command as follows:

```
lspp -h dlctype
```

where *dlctype* is one of the following:

bos.dlc.8023	IEEE Ethernet (802.3) Data Link Control
bos.dlc.ether	Standard Ethernet Data Link Control
bos.dlc.fddi	FDDI Data Link Control
bos.dlc.sdLC	SDLC Data Link Control
bos.dlc.token	Token-Ring Data Link Control

Information about an installed DLC can be displayed through Web-based System Manager, the System Management Interface Tool (SMIT), or the command line. On heavily used systems or communications ports, it might be necessary to change the DLC attributes to fine-tune the DLC performance. If receive performance is slow, and the system error log indicates that ring queue overflow is occurring between the DLC and its device handler, increase the DLC queue depth for incoming data. Finally, it is advisable to remove an installed DLC from the kernel when it is not needed for a lengthy period of time. This removal does not deinstall the DLC from the system, but allows kernel resources to be freed for other tasks until the DLC is needed again. Instructions for all of these tasks are in *Managing DLC Device Drivers* .

GDLC Interface ioctl Entry Point Operations

The generic data link control (GDLC) interface supports the following **ioctl** subroutine operations:

DLC_ENABLE_SAP	Enables a service access point (SAP).
DLC_DISABLE_SAP	Disables a SAP.
DLC_START_LS	Starts a link station on a particular SAP as a caller or listener.
DLC_HALT_LS	Halts a link station.
DLC_TRACE	Traces a link station's activity for short or long activities.
DLC_CONTACT	Contacts a remote station for a particular local link station.
DLC_TEST	Tests the link to a remote for a particular local link station.
DLC_ALTER	Alters a link station's configuration parameters.
DLC_QUERY_SAP	Queries statistics of a particular SAP.
DLC_QUERY_LS	Queries statistics of a particular link station.
DLC_ENTER_LBUSY	Enters local-busy mode on a particular link station.

DLC_EXIT_LBUSY	Exits local-busy mode on a particular link station.
DLC_ENTER_SHOLD	Enters short-hold mode on a particular link station.
DLC_EXIT_SHOLD	Exits short-hold mode on a particular link station.
DLC_GET_EXCEP	Returns asynchronous exception notifications to the application user.

Note: This **ioctl** subroutine operation is not used by the kernel user since all exception conditions are passed to the kernel user through their exception handler.

DLC_ADD_GRP	Adds a group or multicast receive address to a port.
DLC_DEL_GRP	Removes a group or multicast receive address from a port.
DLC_ADD_FUNC_ADDR	Adds a group or multicast receive functional address to a port.
DLC_DEL_FUNC_ADDR	Removes a group or multicast receive functional address from a port.
IOCINFO	Returns a structure that describes the GDLC device manager. See the <code>/usr/include/sys/devinfo.h</code> file format for more information.

Service Access Point

A SAP identifies a particular user service that sends and receives a specific class of data. This allows different classes of data to be routed separately to their corresponding service handlers. Those DLCs that support multiple concurrent SAPs have addresses known as Destination SAP and Source SAP imbedded in their packet headers. DLCs that can only support a single SAP do not need or use SAP addressing, but still have the concept of enabling the one SAP. In general, there is a SAP enabled for each DLC user on each port.

Most SAP address values are defined by IEEE standardized network-management entities or user-defined values as specified in the *Token-Ring Network Architecture Reference*. Some of the common SAP addresses are:

Null SAP (0x00)	Provides some ability to respond to remote nodes even when no SAP has been enabled. This SAP supports only connectionless service and responds only to exchange identification (XID) and TEST Link Protocol Data Units (LPDUs).
SNA Path Control (0x04)	Denotes the default individual SAP address used by Systems Network Architecture (SNA) nodes.
PC Network NETBIOS (0xF0)	Used for all DLC communication that is driven by Network Basic Input/Output System (NetBIOS) emulation.
Discovery SAP (0xFC)	Used by the local area network (LAN) name-discovery services.
Global SAP (0xFF)	Identifies all active SAPs.

Link Station

A link station (LS) identifies an attachment between two nodes for a particular SAP pair. This attachment can operate as a connectionless service (datagram) or connection-oriented service (fully sequenced data transfer with error recovery). In general, there is one LS started for each remote attachment.

Local-Busy Mode

When an LS is operating in a connection-oriented mode, it needs to stop the remote station's sending of information packets for reasons such as resource outage. Notification can then be sent to the remote station to cause the local station to enter local-busy mode. Once resources are available, the local station notifies the remote that it is no longer busy and that information packets can flow again. Only sequenced information packets are halted with local-busy mode. All other types of data are unaffected.

Short-Hold Mode

Use the short-hold mode of operation when operating over data networks with the following characteristics:

- Short call-setup time
- Tariff structure that specifies a relatively small fee for the call setup compared to the charge for connect time.

During short-hold mode, an attachment between two stations is maintained only while there is data available for transfer between the two stations. When there is no data to send, the attachment is cleared after a specified time-out period and only reestablished when there is new data to transfer.

Testing and Tracing a Link

To test an attachment between two stations, instruct an LS to send a test packet from the local station. This packet is echoed back from the remote station if the attachment is operating correctly.

Some data links are limited in their support of this function due to protocol constraints. SDLC, for example, only generates the test packet from the host or primary station. Most other protocols, however, allow test packets to be initiated from either station.

To trace a link, line data, and special events (such as station activation, termination, and time outs), obtain a generic trace channel and instruct an LS to write its trace logs into the generic trace facility for each LS. This function helps determine the cause of certain communications attachment problems. Both short and long trace entries are supported.

Statistics

Both SAP and LS statistics can be queried by a GDLC user. The statistics for a SAP consist of the current SAP state and information about the device handler. LS statistics consist of the current station states and various reliability, availability, and serviceability counters that monitor the activity of the station from the time it is started.

GDLC Special Kernel Services

Generic data link control (GDLC) provides special services for a kernel user. However, a trusted environment must exist within the kernel. Instead of the DLC device manager copying asynchronous event data into user space, the kernel user must specify function pointers to special routines called function handlers. Function handlers are called by the DLC at the time of execution. This allows maximum performance between the kernel user and the DLC layers. Each kernel user is required to restrict the number of function handlers to a minimum path length and use the communications memory buffer (mbuf) scheme.

A function handler must never call another DLC entry directly. This is because direct calls are made under lock, causing a fatal sleep. The only exception to this rule is that a kernel user might call the **dlcwrightex** entry point during its service of any of the four receive data functions. Calling the **dlcwrightex** entry point allows immediate responses to be generated without an intermediate task switch. Special logic is required within the DLC device manager to check the process identification of the user calling a write operation. If it is a DLC process and the internal queuing capability of the DLC has been exceeded, the write is sent back with a bad return code (**EAGAIN** return value) instead of putting the calling process (DLC) to sleep. It is then up to the calling user subroutine to return a special notification to the DLC from its receive data function to ensure a retry of the receive buffer at a later time.

The user-provided function handlers are:

Datagram Data Received Routine	Called any time a datagram packet is received for the kernel user.
Exception Condition Routine	Called any time an asynchronous event occurs that must notify the kernel user, such as SAP Closed or Station Contacted.
I-Frame Data Received Routine	Called each time a normal sequenced data packet is received for the kernel user.
Network Data Received Routine	Called any time network-specific data is received for the kernel user.
XID Data Received Routine	Called any time an exchange identification (XID) packet is received for the kernel user.

The **dlcread** and **dlcselect** entry points for DLC are not called by the kernel user because the asynchronous functional entries are called directly by the DLC device manager. Generally, any queuing of these events must occur in the user's function handler. If, however, the kernel user cannot handle a particular receive packet, the DLC device manager may hold the last receive buffer and enter one of two special user-busy modes:

User-Terminated Busy Mode (I-frame only)

If the kernel user cannot handle a received I-frame (due to problems such as queue blockage), a **DLC_FUNC_BUSY** return code is given back, and DLC holds the buffer pointer and enters local-busy mode to stop the remote station's I-frame transmissions. The kernel user must call the Exit Local Busy function to reset local-busy mode and start the reception of I-frames again. Only normal sequenced I-frames can be stopped. XID, datagram, and network data are not affected by local-busy mode.

Timer-Terminated Busy Mode (all frame types)

If the kernel user cannot handle a particular receive packet and wants DLC to hold the receive buffer for a short period and then re-call the user receive function, a **DLC_FUNC_RETRY** return code is given back to DLC. If the receive packet is a sequenced I-frame, the station enters local-busy mode for that period. In all cases, a timer is started; once the timer expires, the receive data functional entry is called again.

Managing DLC Device Drivers

A DLC must be added to the system prior to use. Each installed DLC is automatically added after installation and at each system restart. If a DLC has been removed without a subsequent restart, it can be re-added.

Managing DLC Device Drivers Tasks			
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>	<i>Web-based System Manager Management Environment⁷</i>
Add an Installed DLC	Choose one (by device driver name): smit cmddlc_sdlic smit cmddlc_token smit cmddlc_qllc smit cmddlc_ether¹ smit cmddlc_fddi then select Add	mkdev²	

Managing DLC Device Drivers Tasks			
Change DLC Attributes ^{3,4}	Choose one (by device driver name): smit cmddlc_sdlic_ls smit cmddlc_token_ls smit cmddlc_qllc_ls smit cmddlc_ether_ls¹ smit cmddlc_fddi_ls	chdev²	
Start DLC Local Area Network Monitor Trace ⁵	smit trace	trace -j nnn where the value <i>nnn</i> is the hook ID to be traced	
Stop DLC Local Area Network Monitor Trace	smit trcstop	trcstop²	
Generate DLC Local Area Network Monitor Trace Report	smit trcrpt	trcrpt -d nnn where the value <i>nnn</i> is the hook ID to be reported	
List Current DLC Information ³	Choose one (by device driver name): smit cmddlc_sdlic_ls smit cmddlc_token_ls smit cmddlc_qllc_ls smit cmddlc_ether_ls¹ smit cmddlc_fddi_ls	lsdev² or lsattr²	
Remove a DLC ^{3,6}	Choose one (by device driver name): smit cmddlc_sdlic_rm smit cmddlc_token_rm smit cmddlc_qllc_rm smit cmddlc_ether_rm¹ smit cmddlc_fddi_rm	rmdev²	

Notes:

1. The SMIT fast path for an Ethernet device manager includes both Standard Ethernet and IEEE 802.3 Ethernet device managers.
2. Details about command line options are provided in the command descriptions for **mkdev**, **chdev**, **trace**, **trcstop**, **trcrpt**, **lsdev**, **lsattr**, or **rmdev** in *AIX 5L Version 5.1 Commands Reference*.
3. A DLC must be installed and added before you can list, show, change, or remove its attributes. An attribute change is only successful if there are no active opens against the target DLC. Before issuing the change action, the user might have to stop services such as SNA, OSI, or NetBIOS from using the DLC.
4. Changing the receive-queue size directly affects system resources. Make this change only if the DLC is having receive-queue problems, such as sluggish performance or overflows between the DLC and its device handler.
5. Exercise caution when enabling the monitor trace since it directly affects the performance of the DLCs and their associates.
6. Removing a DLC is only successful if there are no active opens against the target DLC. Before issuing the remove action, the user may have to stop services such as SNA, OSI, or NetBIOS from using the DLC.
7. These tasks are not available in Web-based System Manager Management Environment.

Chapter 8. Basic Networking Utilities

This chapter presents information on installing, configuring, and maintaining Basic Network Utilities (BNU). The following topics are discussed:

- BNU Overview
- Configuring BNU
- Maintaining BNU
- BNU Configuration Files
- BNU Files, Commands, and Directories Reference.

BNU Overview

The Basic Networking Utilities (BNU) are a group of programs, directories, and files that can be used to communicate with any UNIX system on which a version of the UNIX-to-UNIX Copy Program (UUCP) is running. BNU is one of the Extended Services programs that can be installed with the base operating system.

A group of commands related to UUCP, a UNIX-to-UNIX communication program developed by AT&T and modified as part of the Berkeley Software Distribution (BSD) are contained in BNU. BNU provides commands, processes, and a supporting database for connections to local and remote systems. Communication networks such as Token-Ring and Ethernet are used to connect systems on local networks. A local network can be connected to a remote system by hardwire or telephone modem. Commands and files can then be exchanged between the local network and the remote system.

Topics discussed in this section are:

- How BNU Works
- BNU File and Directory Structure
- BNU Security
- BNU Daemons

Before users on your system can run BNU programs, BNU must be installed and configured.

BNU is controlled by a set of configuration files that determine whether remote systems can log in to the local system and what they can do after they log in. These configuration files must be set up according to the requirements and resources of your system.

To maintain BNU, you must read and remove log files periodically and check the BNU queues to ensure jobs are correctly transferring to remote systems. You must also periodically update the configuration files to reflect changes in your system or remote systems.

For more information, see:

- Configuring BNU
 - Information to Collect Before Configuring BNU
- Maintaining BNU
 - Working with BNU Log Files
 - Using BNU Maintenance Commands

How BNU Works

BNU uses a set of hardware connections and software programs to communicate between systems. A structure of directories and files tracks BNU activities. This structure includes a set of public directories, a group of administrative directories and files, configuration files, and lock files. Most of the directories for BNU are created during the installation process. Some of the administrative directories and files are created by various BNU programs.

With the exception of the remote login commands, BNU works as a batch system. When a user requests a job sent to a remote system, BNU stores the information needed to complete the job. This is known as *queuing* the job. At scheduled times, or when a user instructs it to do so, BNU contacts various remote systems, transfers queued work, and accepts jobs. These transfers are controlled by the configuration files on your system and those of the remote system.

National Language Support for BNU Commands

All BNU commands, except **uucpadmin**, are available for National Language Support. User names need not be in ASCII characters. However, all system names must be in ASCII characters. If a user attempts to schedule a transfer or a remote command execution involving non-ASCII system names, BNU returns an error message.

BNU File and Directory Structure

BNU uses a structure of directories and files to keep track of their activities. This structure includes:

- Public directories
- Configuration files
- Administrative directories and files
- Lock files.

Most of the directories for BNU are created during the installation process. Some of the administrative directories and files are created by various BNU programs as they run.

BNU Public Directories

When specified, the BNU public directory (**/var/spool/uucppublic**) stores files that have been transferred to the local system from other systems. The files wait in the public directory until users claim them. The public directory is created when BNU is installed. Within the public directory, BNU creates a subdirectory for each remote system that sends files to the local system.

BNU Configuration Files

The BNU configuration files, also known as the BNU supporting database, reside in the **/etc/uucp** directory. The files must be configured specifically for your system. They are owned by the uucp login ID and can be edited only with root authority. The configuration files contain information about:

- Accessible remote systems
- Devices for contacting the remote systems
- Times to contact the remote systems
- What the remote systems are allowed to do on your system.

Some configuration files also specify limits on BNU activities that prevent your system from becoming overloaded.

The BNU configuration files include:

Devices Contains information about available devices, including both modems and direct connections.

Dialcodes	Contains dialing code abbreviations, which allow you to shorten phone numbers in the Systems file.
Dialers	Specifies calling command syntax for a specific modem type ("dialer").
Maxuuscheds	Limits simultaneous scheduled jobs.
Maxuuxqts	Limits simultaneous remote command executions.
Permissions	Contains access permission codes. This file is the primary file for determining the security for BNU.
Poll	Specifies when the BNU program should poll remote systems to initiate tasks.
Sysfiles	Lists files that serve as the Systems , Devices , and Dialers files for the BNU configuration. If this file is not used, the default files are /etc/uucp/Systems , /etc/uucp/Devices , and /etc/uucp/Dialers .
Systems	Lists accessible remote systems and information needed to contact them, including the device to use and the user name and password combinations you need to log in. Also specifies the times when the systems can be contacted.

The configuration files cross-reference each other when BNU is in use. For example:

- The **Devices** file contains a *Token* field that refers to entries in the **Dialers** file.
- The **Systems** file contains an entry for a *Class* of device. A device of each *Class* referred to in the **Systems** file must be defined in the **Devices** file.
- The **Poll** file contains entries for systems your system calls. Each of these systems must be defined in the **Systems** file.

Entries in the BNU configuration files depend on the types of connections between your system and each remote system. For example, special entries must be made if Transmission Control Protocol/Internet Protocol (TCP/IP) or direct connections are used to contact other systems. If modems are used to contact other systems, the modems must be defined in the **Dialers** file.

The **Systems**, **Devices**, and **Permissions** files must be configured on your system before you can contact remote systems using BNU. Other configuration files enable you to use BNU capabilities, such as automatic polling. Many of the configuration files must be modified periodically to reflect changes to your system or the systems you contact. The **Sysfiles** file can be used to specify files other than the default **Systems**, **Devices**, and **Dialers** files to fulfill the same role.

BNU Administrative Directories and Files

The BNU administrative directories and files are in subdirectories of the **/var/spool/uucp** directory. These directories and files contain two types of information:

- Data waiting to be transferred to other systems
- Log and error information about BNU activities.

Under the **/var/spool/uucp** directory, BNU creates the following directories:

.Admin	Contains four administrative files: <ul style="list-style-type: none"> • audit • Foreign • errors • xferstats
.Corrupt	These files contain error and log information about BNU activities.
.Log and .Old	Contains copies of files that cannot be processed by the BNU program.
.Status	Contain log files from BNU transactions.
.Workspace	Stores the last time the uucico daemon tried to contact remote systems.
.Xqtdir	Holds temporary files that the file transport programs use internally.
	Contains execute files with lists of commands that remote systems can run.

SystemName Contains files used by file transport programs. These files are:

- Command (**C.***)
- Data (**D.***)
- Execute (**X.***)
- Temporary (**TM.***)

BNU creates a *SystemName* directory for each remote system it contacts.

The directories whose names begin with a dot are *hidden*. They cannot be found with an **ls** or **li** command unless the **-a** flag is used. When the **uucico** daemon is started, it searches the **/var/spool/uucp** directory for work files and transfers the files from any directory that is not hidden. The **uucico** daemon sees only the *SystemName* directories, not the other administrative directories.

The files in the hidden directories are owned by the uucp login ID. These files can be accessed only with root authority or with a login ID with a UID of 5.

For further information about maintaining the BNU administrative directories, see Maintaining BNU.

BNU Lock Files

The BNU lock files are stored in the **/var/locks** directory. When BNU uses a device to connect to a remote computer, it places a lock file for that device in the **/var/locks** directory. When another BNU program or any other program needs the device, that program checks the **/var/locks** directory for a lock file. If a lock file exists, the program waits until the device is available or uses another device for the communication.

In addition, the **uucico** daemon places lock files for remote systems in the **/var/locks** directory. Before contacting a remote system, the **uucico** daemon checks the **/var/locks** directory for a lock file for that system. These files prevent other instances of the **uucico** daemon from establishing duplicate connections to the same remote system.

Note: Other software besides BNU, such as Asynchronous Terminal Emulation (ATE) and TCP/IP, uses the **/var/locks** directory.

BNU Security

Because other systems contact your system to log in, transfer files, and enter commands, BNU provides a means to establish security. BNU security enables you to restrict what users of remote systems can do on the local system (users of remote systems can also restrict what you can do on their systems). BNU runs several daemons to complete its activities and uses administrative directories to store the files it needs. BNU also keeps a log of its own activities.

BNU security works on several levels. When you configure BNU, you can determine:

- Who on your system has access to BNU files.
- Which remote systems your system can contact.
- How users on remote systems log in to your system.
- What users on remote systems can do on your system once they log in.

uucp Login ID

When BNU is installed, all of the configuration files, daemons, and many of the commands and shell procedures are owned by the uucp login ID. The uucp login ID has a user ID (UID) of 5 and a group ID (GID) of 5. The **cron** daemon reads the **/var/spool/cron/crontabs/uucp** file to schedule automatic jobs for BNU.

Usually, logging in as user uucp is not allowed. To change files that are owned by the uucp login ID, log in with root authority.

Attention: Allowing remote systems to log in to the local system with the uucp login ID seriously jeopardizes the security of the local system. Remote systems logged in with the uucp ID can display and possibly modify the local **Systems** and **Permissions** files depending on the other permissions specified in the LOGNAME entry. It is strongly recommended that you create other BNU login IDs for remote systems and reserve the uucp login ID for the person responsible for administering BNU on the local system. For the best security, each remote system that contacts the local system must have a unique login ID with a unique UID number.

The operating system provides a default nuucp login ID for transferring files.

BNU Login IDs

The startup shell for BNU login IDs is the **uucico** daemon (**/usr/sbin/uucp/uucico**). When remote systems call your system, they automatically start the **uucico** daemon on your system. Login IDs for BNU have a uucp group ID of 5.

Login IDs used by remote systems need passwords. In order to prevent security from prompting a new BNU login ID for a new password when the remote system logs in, you must set the password as soon as you create the account. To do this, use the **passwd** command followed by the **pwdadm** command. For example, to set a password for the login ID nuucp, log in as the root user and enter the following commands:

```
passwd nuucp
pwdadm -f NOCHECK
nuucp
```

The system prompts you for a password for the nuucp login ID. Completing these steps allows the remote system to log in without being immediately prompted for a new password (which the batch-oriented nuucp login ID can not provide).

After creating the login ID for a remote system, notify that system BNU administrator of the login ID and password to access your system.

Creating a BNU Administrative Login ID:

A user with root authority can set up a BNU administrative login ID. This is useful if you want to delegate BNU administration duties to a user without root authority. The BNU administrative login ID should have password security, a UID of 5, and be in a uucp group ID 5. The login shell for the administrative login should be the **/usr/bin/sh** program (instead of the **uucico** daemon). Giving the BNU administrative login a UID of 5 causes it to have the same privileges as the **uucp** login ID. Thus, for security, remote systems should not be allowed to log in as the BNU administrator.

Security and the **Systems** and **remote.unknown** Files

On most BNU systems, only remote systems listed in the **/etc/uucp/Systems** file or one of its substitutes (specified in the **Sysfiles** file) can log in to the local system. The **/usr/sbin/uucp/remote.unknown** script is executed whenever an unknown system attempts to call the local system. This script refuses to let the unknown system log in and makes an entry in the **/var/spool/uucp/.Admin/Foreign** file recording the time of the login attempt.

With root authority, or as a BNU administrator, you can modify the **remote.unknown** shell procedure to log more information about the remote system or to store the information in a different file. For example, you can modify the shell procedure to send mail to the BNU administrator whenever an unknown system tries to log in.

By taking away execute permissions on the **remote.unknown** shell procedure, you enable unknown machines to log in. In this case, you should add a **MACHINE=OTHER** entry to the **/etc/uucp/Permissions** file to establish permissions for the unknown machines.

Your system can contact only remote systems listed in the **Systems** file. This prevents users on your system from contacting unknown systems.

Security and the Permissions File

The **/etc/uucp/Permissions** file determines:

- Remote login user names for logging in to the local system
- Approved commands and privileges for remote systems logging in to the local system.

The **/etc/uucp/Permissions** file contains two types of entries:

LOGNAME	Defines login names and the privileges associated with them. LOGNAME entries take effect when a remote system calls the local system and attempts to log in.
MACHINE	Defines machine names and the privileges associated with them. MACHINE entries take effect when the remote system attempts to carry out commands on the local system.

Options in the **Permissions** file enable you to establish various levels of security for each remote system. For example, if many remote systems share one login ID on the local system, use the **VALIDATE** option to require each remote system to use a unique login ID. The **SENDFILES**, **REQUEST**, and **CALLBACK** options specify which system has control, keeping the local system in control of transactions if necessary.

The **READ**, **WRITE**, **NOREAD**, and **NOWRITE** options define access to specific directories on the local system. These options also control where on your system remote users can place data. The **COMMANDS** option limits the number of commands users on remote systems can execute on the local system. The **COMMANDS=ALL** option allows total privileges to systems closely associated with your system.

Attention: The **COMMANDS=ALL** option can seriously jeopardize the security of your system.

BNU Daemons

The BNU software includes four daemons stored in the **/usr/sbin/uucp** directory:

uucico	Facilitates file transfers.
uusched	Facilitates work request scheduling of files queued in the local spooling directory.
uuxqt	Facilitates remote command executions.
uucpd	Facilitates communications using TCP/IP.

The **uucico**, **uusched**, and **uuxqt** daemons are started by the **cron** daemon according to a schedule set by the BNU administrator. With root authority, you can also start these daemons manually. The **uucpd** daemon should be started by the TCP/IP **inetd** daemon.

Using the uucico Daemon

The **uucico** daemon transports the files required to send data from one system to another. The **uucp** and **uux** commands start the **uucico** daemon to transfer command, data, and execute files to the designated system. The **uucico** daemon is also started periodically by the BNU scheduler, the **uusched** daemon. When started by the **uusched** daemon, the **uucico** daemon attempts to contact other systems and execute the instructions in the command files.

How the Daemon Process Begins: To run the instructions in the command files, the **uucico** daemon first checks the **/etc/uucp/Systems** file (or one or more other files specified by **/etc/uucp/Sysfiles**) for the

system to be called. The daemon then checks the **Systems** file entry for a valid time to call. If the time is valid, the **uucico** daemon checks the *Type* and *Class* fields and accesses the **/etc/uucp/Devices** file (or one or more other files specified by **/etc/uucp/Sysfiles**) for a device that matches.

After finding a device, the **uucico** daemon checks the **/var/locks** directory for a lock file for the device. If one exists, the daemon checks for another device of the requested type and speed.

When no device is available, the daemon returns to the **Systems** files for another entry for the remote system. If one exists, the daemon repeats the process of searching for a device. If another entry is not found, the daemon makes an entry in the **/var/spool/uucp/.Status/SystemName** file for that remote system and goes on to the next request. The command file remains in the queue. The **uucico** daemon attempts the transfer again at a later time. The later attempt is called a *retry*.

When the Daemon Reaches the Remote System: When the **uucico** daemon reaches the remote system, it uses the instructions in the **Systems** files to log in. This causes an instance of the **uucico** daemon to be invoked on the remote system as well.

The two **uucico** daemons, one on each system, work together to make the transfer. The **uucico** daemon on the calling system controls the link, specifying the requests to be performed. The **uucico** daemon on the remote system checks the local permissions for whether they allow the request to be performed. If so, the file transfer starts.

After the **uucico** daemon on the calling system has finished transferring all requests it has for the remote system, it sends a hangup request. When the remote **uucico** daemon has transactions to send to the calling system, it denies the hangup request, and the two daemons reverse roles.

Note: Either the **/etc/uucp/Permissions** file on the local system or the **/etc/uucp/Permissions** file on the remote system can forbid the daemons to reverse roles. In this case, the remote system must wait to transfer files until it calls the local system.

When nothing is left to be transferred in either direction, the two **uucico** daemons hang up. At this point, the **uuxqt** daemon is called to execute remote command requests.

Throughout the transfer process, the **uucico** daemons on both systems log messages in the BNU log and error files.

Using the **uusched** Daemon

The **uusched** daemon schedules the transfer of files that are queued in the spooling directory on the local system. The spooling directory is **/var/spool/uucppublic**. When the **uusched** daemon is invoked, it scans the spooling directory for command files, then randomizes the files and starts the **uucico** daemon. The **uucico** daemon transfers the files.

Using the **uuxqt** Daemon

When a user issues the **uux** command to run a specified command on a designated system, the **uuxqt** daemon runs the command. After creating the necessary files, the **uux** command starts the **uucico** daemon, which transfers those files to the public spooling directory on the specified system.

The **uuxqt** daemon periodically searches the spooling directory for command-execution requests on every connected system. When it locates such a request, the **uuxqt** daemon checks for necessary files and permissions. Then, if permitted, the daemon runs the specified command.

Using the uucpd Daemon

The **uucpd** daemon must be able to run on the remote system before BNU can establish communications with a remote computer with Transmission Control Protocol/Internet Protocol (TCP/IP). The **uucpd** daemon is a subserver of the TCP/IP **inetd** daemon and is started by the **inetd** daemon.

By default, the **uucpd** daemon is commented in the **inetd.conf** file. To use it, you must remove the comment character and restart **inetd**. However, if this has been changed on your system, you might need to reconfigure the **inetd** daemon to start the **uucpd** daemon.

Configuring BNU

The following procedures detail how to configure Basic Network Utilities (BNU) for various types of connections, including hardwired, modem, and Transmission Control Protocol/Internet Protocol (TCP/IP) connections.

Prerequisites

- BNU must be installed on your system.
- You must have root user authority to edit the BNU configuration files.
- If you are using direct connections for BNU communications, the appropriate hardwired connections between your system and the remote systems must be set up.
- If you are using modems for BNU communications, you must have installed and configured each modem.
- If one or more of your connections uses TCP/IP, then TCP/IP must be running between your system and the appropriate remote systems.
- Collect the information you need to configure BNU. This information should include a list of remote systems and lists of devices and modems to use to connect to the systems.

Information to Collect before Configuring BNU

Before configuring BNU, gather the information listed:

- For each remote system your system will call, collect the following information:
 - System name
 - Login name your system should use on the remote system
 - Password for the login name
 - Login and password prompts on the remote system
 - Type of connection you will use to reach the remote system (TCP/IP, direct, or telephone).
- If the connection is direct, collect:
 - The bit rate of the connection
 - The port on the local system to which the connection is attached.
- If the connection is a telephone connection, collect:
 - The telephone number of the remote system
 - The speed of your modem that is compatible with that of the remote system.

Note: If any of the remote systems will call your system, ensure the BNU administrator on each of the remote systems has all the preceding information about your system.

- For each local modem that you will use for BNU connections, collect the following information:
 - The chat script for the modem (consult the modem documentation)

Note: For some modems, the chat script is already in the **/etc/uucp/Dialers** file.

- The local port for the modem.

Using the information you collect, make a list of each device you need to connect to a remote system. Following is a sample list for local system morgan:

```
direct:
hera 9600 tty5
zeus& 2400 tty2
ariadne 2400 tty1
hayes modem (tty3): apollo, athena
TCP/IP: merlin, arthur, percy
```

To connect to system hera, a direct connection at a speed of 9600 from port tty5 is used. To connect to system apollo, the hayes modem connected to port tty3 is used. TCP/IP is used to connect to systems merlin, arthur, and percy.

Procedure

For BNU to function correctly at your site, you must configure the remote communications facilities to:

- List the devices used to establish a hardwired, telephone, or modem communications link.
- List the modems used to contact remote systems over the telephone network.
- List the accessible remote systems.
- List the alphabetic abbreviations representing the prefixes of telephone numbers used to contact the specified remote systems (optional).
- Set access permissions specifying the ways in which local and remote systems may communicate.
- Schedule monitoring for the networked remote systems (optional).

To create these lists, permissions, schedules, and procedures:

- Modify the BNU configuration files.
- Edit the **/var/spool/cron/crontabs/uucp** file to remove the comment characters (#) from the beginnings of the lines that schedule the automatic maintenance routines.

You must configure the **Systems**, **Devices**, and **Permissions** files before BNU will run correctly at your site. However, it is not necessary to modify the BNU configuration files in any particular order.

To configure BNU on your system:

1. Make sure that BNU is installed on your system by running the command:

```
ls1pp -h bos.net.uucp
```

If BNU is installed, you will see bos.net.uucp in the output. If you do not see it, install it from the install tape.

2. Set up appropriate login IDs and passwords for remote systems that will call your system, and tell the person responsible for administering BNU or UNIX-to-UNIX Copy Program (UUCP) on each remote system the login and password you have provided. This is done by editing the **/etc/passwd**, **/etc/group**, **/etc/security/login.cfg**, and **/etc/security/passwd** files.

Attention: Allowing remote systems to log into the local system with the UUCP login ID seriously jeopardizes the security of your system. Remote systems logged in with the UUCP ID can display and possibly modify (depending on the permissions specified in the LOGNAME entry of the **Permissions** file) the local **Systems** and **Permissions** files. It is strongly recommended that you create other BNU login IDs for remote systems and reserve the uucp login ID for the person administering BNU on the local system. For the best security, each remote system that contacts the local system should have a unique login ID with a unique UID number. These login IDs should have GIDs of 5. By default, the operating system includes the nuucp login ID for transferring files.

- a. You have the option of maintaining separate logins or having one login for all BNU connections. If you need to maintain complete control over access by each individual machine, you must create separate login IDs, as well as combine the MACHINE and LOGNAME entries in the **Permissions** file. A few example **/etc/passwd** entries are shown here:

```
Umicrktk:!:105:5:micrktk uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
Ufloyd1:!:106:5:floyd1 uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
Uicus:!:107:5:icus uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
Urisctkr:!:108:5::/usr/spool/uucppublic:/usr/sbin/uucp/uucico
```

- b. If you want to have one set of permissions and do not want to maintain separate control for any of your UUCP connections, you can have a single login for all machines such as the following:

```
nuucp:!:6:5::/usr/spool/uucppublic:/usr/sbin/uucp/uucico
```

- c. The user ID (the third colon separated field) must be unique to avoid a security risk. The group ID (the fourth separated field) must be 5, the same group as uucp. You can change the home directory (the sixth field) to any valid directory, but the login shell (the seventh field) must be **/usr/sbin/uucp/uucico**.

- d. Make sure that the **/etc/group** file contains the new users. An example of such an entry is:

```
uucp:!:5:uucp,uucpadm,nuucp,Umicrktk,Uicus,Urisctkr
```

- e. You may want to add any users to group uucp who will be using modems to dial out with programs other than the **cu** command.

- f. After editing these files as root, set up a password for the new users with the command **passwd** **UserName**.

- g. Sometimes, the default herald with all of its Ctrl-J's, will cause a uucico process that is trying to login to give up. (You may see the message Enough al ready.) You can avoid that by commenting out (with asterisks) the default stanza, and defining a stanza for your tty something like this:

```
/dev/tty0:
    herald = "\nrisc001 login:"
```

- h. If you change a password from the root login, the flags entry in the stanza for the user in **/etc/security/passwd** will contain the following:

```
flags = ADMCHG
```

Change it to:

```
flags =
```

Otherwise, when the remote uucico logs in, it will be prompted to enter a new password, which it cannot do. Hence the login will fail.

- i. Using an ASCII text editor or the **uucpadm** command, edit the **Poll** file. Add an entry for each system your system will poll.

Note: The systems listed in the **Poll** file must also be listed in the **/etc/uucp/Systems** file.

- j. Using an ASCII text editor, edit the **/var/spool/cron/crontabs/uucp** file. Remove the comment characters (#) from the lines that run the **uudemon.hour** and **uudemon.poll** commands. You can change the times these commands are run. However, be sure to schedule the **uudemon.poll** command approximately five minutes *before* you schedule the **uudemon.hour** command.

- k. Check to make sure your changes took effect by running this command:

```
crontab -l
```

- l. Set up the 'BNU data files: Systems, Permissions, Devices, Dialers, and Sysfiles. You could use the **/usr/sbin/uucp/uucpadm** command to initially set up the files and then edit them to suit your exact needs. Note that the Sysfiles file allows you to specify files other than **/etc/uucp/Systems**, **/etc/uucp/Devices**, and **/etc/uucp/Dialers** for BNU configuration. See **Sysfiles** for more information.

3. Decide whether to use dial-code abbreviations for telephone numbers (see the **Dialcodes** file format). If you decide to use dial-code abbreviations in the **Systems** files, set up the **Dialcodes** entry for each abbreviation. Refer to Dialcodes File Format for BNU in *AIX 5L Version 5.1 Files Reference* for details. If you are using TCP/IP for your BNU connections, use the **netstat** command to see whether the **uucpd** daemon is runnable, by entering:

```
netstat -a
```

The **uucpd** daemon is started by the **inetd** daemon. If the **uucpd** daemon is not able to run, reconfigure the **inetd** daemon to start the **uucpd** daemon.

4. Using the list of devices you collected before beginning this procedure, modify the **Devices** file on your system. Make an entry for each modem and each direct connection. If you are using TCP/IP, make sure you uncomment the TCP/IP entry in the **Devices** file. You can configure the **/etc/uucp/Sysfiles** file to specify other files to use for Devices configuration. Refer to the Devices File Format for BNU in *AIX 5L Version 5.1 Files Reference* for details on the Devices file. Refer to Sysfiles File Format for BNU for details on the Sysfiles file in *AIX 5L Version 5.1 Files Reference*.

Also, if you are using TCP/IP, check to see whether the **/etc/services** file includes:

```
uucp      540/tcp      uucpd
```

If not, add the line.

5. Using your information about each remote system that you collected before beginning this procedure, modify the **Systems** file on your system. Use the commented examples in the **Systems** file as a guide when specifying your configuration. See the "BNU Systems File Format" in *AIX 5L Version 5.1 Files Reference* for details. If you are using TCP/IP, ensure the host-name table in the **/etc/hosts** file includes the name of the remote computer with which you want to connect. You can configure the **/etc/uucp/Sysfiles** file to specify other files to use for Systems configuration. Refer to Sysfiles File Format for BNU in *AIX 5L Version 5.1 Files Reference* for more information.
6. Using the information about devices and modems that you collected before beginning this procedure, make sure the **Dialers** file on your system contains an entry for each modem. If you are using TCP/IP and direct connections, make sure the TCP/IP entry and direct entries are present in the file. Refer to Dialers File Format for BNU in *AIX 5L Version 5.1 Files Reference* for details. You can configure the **/etc/uucp/Sysfiles** file to specify other files to use for Dialers configuration. Refer to Sysfiles File Format for BNU in *AIX 5L Version 5.1 Files Reference* for more information.
7. Decide how much access to your system you want to provide to each remote system you call and to each remote system that calls you. Set up appropriate entries for each system and each login name in the **Permissions** file. Refer to Permissions File Format for BNU in *AIX 5L Version 5.1 Files Reference* for details.
8. Issue the **uuccheck** command to verify that everything is in place:

```
/usr/sbin/uucp/uuccheck -v
```

The **uuccheck** command verifies that the directories, programs, and support files are set up properly and that the **Permissions** file entries are consistent. If the **uuccheck** command reports any errors, fix the errors.

9. If you wish, set up automatic monitoring of BNU operations and automatic polling of remote systems.

Setting Up Automatic Monitoring of BNU

Prerequisites

- Complete the steps in Configuring BNU.
- You must have root user authority to edit the **/var/spool/cron/crontabs/uucp** file.

Procedure

BNU uses the **cron** daemon to start BNU daemons and to monitor BNU activity. The **cron** daemon reads the **/var/spool/cron/crontabs/uucp** file for instructions about when to start BNU procedures.

1. Log in as a user with root user authority.
2. Using an ASCII text editor, edit the `/var/spool/cron/crontabs/uucp` file.
3. Uncomment the lines for the BNU maintenance procedures, `uudemon.admin` and `uudemon.cleanup`. You can change the times these procedures are run if your system needs maintenance at more or less frequent intervals. It is best, however, to run the `uudemon.admin` command at least once a day and the `uudemon.cleanup` command at least once a week.
4. You can use the `crontabs/uucp` file to schedule other BNU maintenance commands, such as the `uulog`, `uuclean`, or `uucleanup` commands. In addition, you can use the `crontabs/uucp` file to instruct the `cron` daemon to start the `uucico`, `uuxqt`, or `uusched` daemons at specific times.

Setting Up BNU Polling of Remote Systems

Prerequisites

1. Complete the steps in Configuring BNU.
2. You must have root authority to edit the `/var/spool/cron/crontabs/uucp` file and the `/etc/uucp/Poll` file.

Procedure

To enable BNU to poll remote systems for jobs, list the systems in the `/etc/uucp/Poll` file. In addition, run the `uudemon.hour` and `uudemon.poll` commands periodically.

1. Decide which remote systems to automatically poll. Decide how often you want to poll each one. Specify times for each system with the `Poll` file as seldom as once a day or as often as you wish.
2. Log in as a user with root authority.
3. Using an ASCII text editor or the `uucpadm` command, edit the `Poll` file. Add an entry for each system your system will poll.

Note: The systems listed in the `Poll` file must also be listed in the `/etc/uucp/Systems` file.

4. Using an ASCII text editor, edit the `/var/spool/cron/crontabs/uucp` file. Remove the comment characters (`#`) from the lines that run the `uudemon.hour` and `uudemon.poll` commands. You can change the times these commands are run. However, be sure to schedule the `uudemon.poll` command approximately five minutes *before* you schedule the `uudemon.hour` command.

BNU will now automatically poll the systems listed in the `Poll` file at the times you have specified.

Using the `/etc/uucp/Systems` File

The remote systems are listed in the `/etc/uucp/Systems` files. The `/etc/uucp/Systems` file is the default `Systems` file. The system administrator can specify additional files in the `/etc/uucp/Sysfiles` file.

Each entry in a `Systems` file contains:

- Name of the remote system
- Times when users can connect to the remote system
- Type of link (direct line or modem)
- Speed of transmission over the link
- Information needed to log in to the remote system.

Each entry in a `Systems` file represents one remote system. To establish communications, the remote system must be listed in the local `Systems` file. A `Systems` file must be present on every system that uses the BNU facility. Normally, only the root user can read the `Systems` files. Any user, however, can list the names of remote BNU systems using the `uname` command.

Editing Devices Files for Hardwired Connections

Prerequisites

You must have root authority to edit the `/etc/uucp/Devices` file or another file specified in `/etc/uucp/Sysfiles` as a **Devices** file.

Procedure to Set Up a System Name Entry

To set up a hardwired connection specifying a port and a remote system, make an entry as follows:

1. Enter the name of the remote system to which you want to connect the local computer over the hardwired line in the *Type* field in the second line of the entry.
2. Enter the device name appropriate for the hardwired connection used at your site in the *Line* field in both lines of the entry.
3. Enter a - (hyphen) for a placeholder in the *Line2* field in both lines of the entry.
4. Enter the transmission rate appropriate for the hardwired connection used at your site in the *Speed* field in both lines of the entry.
5. Enter `direct` (all lowercase) in the *Dialer-Token Pairs* field in both lines of the entry.

For example:

```
type device - speed direct
```

Continue adding entries to the **Devices** file until you have listed each hardwired device connecting the local system to a remote system.

Procedure to Set Up a Direct Entry

To set up a hardwired connection between two systems that use a permanent asynchronous serial connection, make a one-line entry as follows:

1. Enter the name of the remote system in the first (*Type*) field.
2. Enter the name of the tty device in the second (*Line*) field.
3. Enter a - (hyphen) for a placeholder in the third (*Line2*) field.
4. Enter the transmission rate appropriate for the hardwired connection used at your site in the fourth (*Class*) field.
5. Enter `direct` (all lowercase) in the fifth (*Dialer-Token Pairs*) field.

For example:

```
type device - speed direct
```

Continue adding entries to the **Devices** file until you have listed each hardwired device connecting the local system to a remote system.

Editing Devices File for Autodialer Connection

Prerequisites

You must have root authority to edit the `/etc/uucp/Devices` file or another file specified in `/etc/uucp/Sysfiles` as a **Devices** file.

Procedure

In telephone-connection entries, the *Type* field is specified as an automatic calling unit (ACU). Type ACU as the *Type* field entry in all remote connections established over a phone line. To set up **Device** file entries for autodialer connections, make a one-line entry for each modem:

1. Enter ACU in the first (*Type*) field.
2. The second (*Line*) field contains the name of the device that is attached to the modem. Enter the device name appropriate for your site.

3. Enter a - (hyphen) as a placeholder in the third (*Line2*) field, unless the autodialer is a standard 801 dialer. If the autodialer is a standard 801 dialer, enter 801.
4. In the fourth (*Speed*) field, enter the baud rate appropriate for your modem and line (this can be 300, 1200, 2400, or higher, depending on the modem) or the class of your modem (for example, D2400).

Note: If the modem can be used at more than one specific rate, make a separate entry in the **Devices** file for each rate. If the modem can be used at any rate, enter the word Any in the *Speed* field.

5. Enter the name of the modem as the *Dialer* field entry in the fifth (*Dialer-Token Pair*) field. If you are planning to include complete phone numbers in the **/etc/uucp/Systems** file or another **Systems** file specifies in **/etc/uucp/Sysfiles**, leave the *Token* field blank. (A blank instructs the BNU program to use the default \D token.) If you are planning to use dialing-code abbreviations specified in the **/etc/uucp/Dialcodes** file, enter the token \T.

For example:

```
type line - speed dialer - token pair
```

Continue adding entries to the **Devices** file until you have listed each connection between the local system and a remote system that uses a telephone line and a modem.

Editing Devices File for TCP/IP

Prerequisites

You must have root authority to edit the **/etc/uucp/Devices** file or another file specified in **/etc/uucp/Sysfiles** as a **Devices** file.

Procedure

If your site is using the TCP/IP system, include the relevant TCP/IP entry in the **Devices** file. To set up the file for use with the TCP/IP system, enter the following line in the **Devices** file:

```
TCP - - - TCP
```

Maintaining BNU

BNU must be maintained to work correctly on your system. To maintain BNU:

- Read and remove log files periodically.
- Use the **uucq** and **uustat** commands to check the BNU queues to ensure jobs are transferring to remote systems correctly.
- Schedule automatic commands that poll remote systems for jobs, return unsend files to users, and send you periodic messages about BNU status.
- Periodically update the configuration files to reflect changes in your system.

In addition, occasionally check with administrators of remote systems to keep up with changes on their systems that might affect your configuration. For example, if the supervisor of system venus changes your system password, you must put the new password in the **/etc/uucp/Systems** file (or the appropriate **Systems** file specified by **/etc/uucp/Sysfiles**) before your system can log in to system venus.

See BNU Files, Commands, and Directories Reference for a list of commands used to maintain BNU.

Working with BNU Log Files

BNU creates log files and error files to track its own activities. These files must be checked and removed periodically to keep them from filling the storage space on your system. BNU provides several commands for use in cleaning log files:

- **uulog**
- **uuclean**
- **uucleanup**
- **uudemon.cleanu**.

Run these commands manually or use entries in the `/var/spool/cron/crontabs/uucp` file to run the commands by the **cron** daemon.

Log Files in the `.Log` and `.Old` Directories

BNU creates individual log files in the `/var/spool/uucp/.Log` directory. BNU creates these log files for each accessible remote system, using the **uucp**, **uucico**, **uux**, and **uuxqt** commands. BNU places status information about each transaction in the appropriate log file each time someone on the system uses BNU. When more than one BNU process is running the system cannot access the log file. Instead, it places the status information in a separate file with a **.LOG** prefix.

The **uulog** command displays a summary of **uucp** or **uux** requests, by user or by system. The **uulog** command displays the files. However, you can also have BNU automatically combine the log files into a primary log file. This is called *compacting* the log files and can be done with the **uudemon.cleanu** command, usually run by the **cron** daemon.

The **cron** daemon runs the **uudemon.cleanu** command. The **uudemon.cleanu** command combines the **uucico** and **uuxqt** log files on the local system and stores them in the `/var/spool/uucp/.Old` directory. At the same time, the command removes old log files previously stored in the `.Old` directory. By default, the **uudemon.cleanu** command saves log files that are two days old.

If storage space is a problem, consider reducing the number of days that files are kept. To track BNU transactions over a longer period of time, consider increasing the number of days that files are kept. To change the default time for saving log files, modify the shell procedure for the **uudemon.cleanu** command. This script is stored in the `/usr/sbin/uucp` directory and can be modified with root authority.

Other BNU Log Files

BNU also collects information and stores it in the `/var/spool/uucp/.Admin` directory. This directory contains the **errors**, **xferstats**, **Foreign**, and **audit** files. These files must be checked and removed occasionally to save storage space. BNU creates each file when it is needed.

When another system contacts your system with the **uucico** daemon debugging mode on, it invokes the **uucico** daemon on your system with debugging turned on. The debugging messages generated by the daemon on the local system are stored in the **audit** file. This file can get quite large. Check and remove the **audit** file often.

The **errors** file records errors encountered by the **uucico** daemon. Checking this file can help you correct problems such as incorrect permissions on BNU work files.

The **xferstats** file contains information about the status of every file transfer. Check and remove this file occasionally.

The **Foreign** file is important to the security of your system. Whenever an unknown system attempts to log in to the local system, BNU calls the **remote.unknown** shell procedure. This shell procedure logs the attempt in the **Foreign** file. The **Foreign** file contains the names of the systems that have attempted to call the local system and been refused. If a system has been attempting frequent calls, use this information when considering whether to allow that system access.

Systemwide Log Files used by BNU

Because many BNU processes need root authority to complete their tasks, BNU creates frequent entries in the `/var/spool/sulog` log file. Similarly, using the **cron** daemon to schedule BNU tasks creates multiple entries in the `/var/spool/cron/log` file. When using BNU, check and clean these files.

BNU Maintenance Commands

The Basic Networking Utilities contain several commands for monitoring BNU activities and cleaning BNU directories and files.

Cleanup Commands

BNU contains three commands that clean directories and remove files that have not been sent:

uuclean	Deletes all files older than a specified number of hours, from the BNU administrative directories. Use the uuclean command to specify a directory to be cleaned or a type of file to be deleted. You can also instruct the command to notify the owners of the deleted files. The uuclean command is the Berkeley equivalent of the uucleanup command.
uucleanup	Performs functions similar to the uuclean command. However, the uucleanup command checks the age of files based on <i>days</i> rather than hours. Use the uucleanup command to send a warning message to users whose files have not been transferred, notifying them that the files are still in the queue. The uucleanup command also removes files relating to a specified remote system.
uudemon.cleanu	A shell procedure that issues the uulog and uucleanup commands to compress the BNU log files and remove log and work files over three days old. The uudemon.cleanu command is run by the cron daemon.

Status-checking Commands

BNU also provides commands for checking the status of transfers and log files:

uuq	Displays jobs currently in the BNU job queue. Use the uuq command to display the status of a specified job or of all jobs. With root authority, you can use the uuq command to delete a job from the queue.
uustat	Provides information similar to that provided by the uuq command, in a different format. Use the uustat command to check the status of jobs and delete jobs you own. With root authority, you can also delete jobs belonging to other users.
uulog	Displays a summary of uucp or uux requests, by user or by system. The uulog command displays the file names. See Working with BNU Log Files .
uupoll	Forces a poll of a remote system. This is helpful when work for that system is waiting in the queue and needs to be transferred, before the system is scheduled to be called automatically.
uusnap	Displays a very brief summary of BNU status. For each remote system, this command shows the number of files awaiting transfer. However, it does not show how long they have been waiting. The uusnap command is the Berkeley equivalent of the uustat command.

Shell Procedures

BNU is delivered with two shell procedures used for maintenance:

uudemon.cleanu	Discussed under "Cleanup Commands"
uudemon.admin	Issues the uustat command. The uustat command reports the status of BNU jobs. It sends the results to the uucp login ID as mail. You can modify the uudemon.admin shell procedure to send the mail elsewhere, or use a mail program to reroute all mail for the uucp login ID to the user responsible for BNU administration.

These shell procedures are stored in the **/usr/sbin/uucp** directory. Copy the procedures and modify the copy, if you want to change what they do. Run the procedures from the command line or schedule them to be run by the **cron** daemon.

To automatically run the **uudemon.cleanu** and **uudemon.admin** commands, remove the comment characters (#) from the beginning of the relevant lines in the **/var/spool/cron/crontabs/uucp** file.

Monitoring a BNU Remote Connection

Prerequisites

- The BNU program must be installed on your system.
- A link (hardwired, modem, or TCP/IP) must be set up between your system and the remote system.
- The BNU configuration files, including the **Systems** file, **Permissions** file, **Devices** file, and **Dialers** file (and **Sysfiles** file, if applicable), must be set up for communications between your system and the remote system.

Note: You must have root user authority to modify the BNU configuration files.

Procedure

The **Uutry** command can help you monitor the **uucico** daemon process if users at your site report file-transfer problems.

1. Issue the **uustat** command to determine the status of all the transfer jobs in the current queue as follows:

```
uustat -q
```

The system displays a status report like the following:

```
venus 3C (2) 05/09-11:02 CAN'T ACCESS DEVICE
hera 1C 05/09-11:12 SUCCESSFUL
merlin 2C 5/09-10:54 NO DEVICES AVAILABLE
```

This report indicates that three command (**C.***) files intended for remote system venus have been in the queue for two days. There could be several reasons for this delay. For example, perhaps system venus has been shut down for maintenance or the modem has been turned off.

2. Before you begin more extensive troubleshooting activities, issue the **Uutry** command as follows to determine whether your local system can contact system venus now:

```
/usr/sbin/uucp/Uutry -r venus
```

This command starts the **uucico** daemon with a moderate amount of debugging and the instruction to override the default retry time. The **Uutry** command directs the debugging output to a temporary file, `/tmp/venus`.

3. If your local system succeeds in establishing a connection to system venus, the debugging output contains a good deal of information. However, the final line in this script, which follows, is the most important:

```
Conversation Complete: Status SUCCEEDED
```

If the connection is successful, assume that the temporary file-transfer problems are now resolved. Issue the **uustat** command again to make certain that the files in the spooling directory have been transferred successfully to the remote system. If they have not, use the steps in Monitoring a BNU File Transfer to check for file-transfer problems between your system and the remote system.

4. If your local system cannot contact the remote system, the debugging output generated by the **Uutry** command contains the following type of information (the exact form of the output might vary):

```
mchFind called (venus)
conn (venus)
getto ret -1
Call Failed: CAN'T ACCESS DEVICE
exit code 101
Conversation Complete: Status FAILED
```

First, check the physical connections between the local and remote systems. Make sure that the remote computer is turned on and all cables are properly connected, that the ports are enabled or disabled (as appropriate) on both systems, and that the modems (if applicable) are working.

If the physical connections are correct and secure, then verify all the relevant configuration files on both the local and remote systems, including the following:

- Make certain that the entries in the **Devices**, **Systems**, and **Permissions** files (and **Sysfiles** file, if applicable) in the **/etc/uucp** directory are correct on both systems.
 - If you are using a modem, make sure that the **/etc/uucp/Dialers** file (or an alternate file specified in **/etc/uucp/Sysfiles**) contains the correct entry. If you are using dial-code abbreviations, be sure the abbreviations are defined in the **/etc/uucp/Dialcodes** file.
 - If you are using a TCP/IP connection, make sure that the **uucpd** daemon can be run on the remote system and that the configuration files contain the correct TCP entries.
5. Once you have checked the physical connections and configuration files, issue the **Uutry** command again. If the debugging output still reports that the connection failed, you might need to confer with a member of your systems support team. Save the debugging output produced by the **Uutry** command. This might prove helpful in diagnosing the problem.

Monitoring a BNU File Transfer

Prerequisites

1. The BNU program must be installed on and configured for your system
2. Establish a connection to a remote system using the steps given in Monitoring a BNU Remote Connection.

Monitoring a File Transfer

Use this procedure to monitor a file transfer to a remote system. Monitoring a file transfer is useful when file transfers to the remote system in question are failing for unknown reasons. The debugging information produced by the **uucico** daemon (called by the **Uutry** command) can help you find out what is working incorrectly.

The **Uutry** command enables you to monitor file transfers, as follows:

1. Prepare a file for transfer using the **uucp** command with the **-r** flag by entering:

```
uucp -r test1 venus!~/test2
```

The **-r** flag instructs the UUCP program to create and queue all necessary transfer files but *not* to start the **uucico** daemon.

2. Issue the **Uutry** command with the **-r** flag to start the **uucico** daemon with debugging turned on by entering:

```
/usr/sbin/uucp/Uutry -r venus
```

This instructs the **uucico** daemon to contact remote system *venus* overriding the default retry time. The daemon contacts system *venus*, logs in, and transfers the file, while the **Uutry** command produces debugging output that enables you to monitor the **uucico** process. Press the Interrupt key sequence to stop the debugging output and return to the command prompt.

The **Uutry** command also stores the debugging output in the **/tmp/SystemName** file. If you break out of the debugging output before the connection is complete, you can page through the output file to see the outcome of the connection.

Debugging BNU Problems

BNU error messages can be linked to a specific phase in the conversation flow. Use the "BNU Conversation Flow Diagram" and the following error descriptions to help diagnose your BNU problems. Some of the following messages might not be sent from BNU, but are included in case another UUCP version is in use.

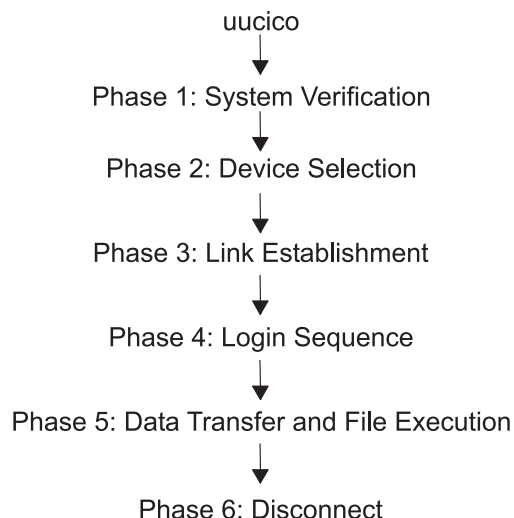


Figure 45. BNU Conversion Flow Diagram. This illustration shows the flow and different phases of BNU conversion. From `uucico` at the top, data is passed to Phase 1–System Verification, then Phase 2–Device Selection, and Phase 3–Link Establishment, then Phase 4–Login Sequence, next Phase 5–Data Transfer and File Execution and last, Phase 6–Disconnect.

PHASE 1 Status Messages

Assert Error	The local system unit is having problems. Check the error report for possible causes by issuing the command <code>errpt -a pg</code> .
System not in Systems	If you supply a remote system name that is not found in the Systems files, this status message is created, BNU will terminate. Use the <code>uname</code> command to check the system name again.
Wrong time to call	The Systems file has restrictions on times to allow outgoing calls. BNU will keep trying until the time is right. Check the Systems file.
Callback required	The network has restricted usage either for security or economic reasons, and access is denied at this time.
Cannot call	These errors mean BNU recently tried to call the remote system and failed.
No Call	It will not immediately try again. They can also be caused by an old system status file being retained thus keeping the <code>uucico</code> daemon from trying again.

PHASE 2 Status Messages

Dialer Script Failed	Your Dialers file script did not complete successfully.
No Device Available	The modem or the outgoing phone line from your system is busy. Check for an error in the device entry of the Systems file. Also, check the Devices and Dialers files to be sure logical devices have physical devices associated with them. The file <code>/etc/uucp/Sysfiles</code> might be specifying an alternate Systems , Devices , or Dialers file that is not correctly configured. Is the device in use by some other program? Check the <code>/var/locks</code> directory for lock on port. If a lock file exists (for example, <code>LCK..TTY0</code>), check to see if the process identified by the number in the lock file is still active. If not, you can remove it (for example, <code>rm /var/locks/LCK..TTY0</code>). Also check the permissions on the port.
Can't Access Device	

Dial Failed
Failed (call to system)

These errors appear when your system dials another successfully but the other system does not answer. It might also indicate a problem in the **Devices** files. Enter the command `uucico -r1 -x6 -s SystemName`. It could be that BNU is expecting some string that it is not receiving. Make the connection by hand to find out what needs to be incorporated into the **Systems** files entry to satisfy the request. Please keep "timing" in mind; perhaps some delays in the modem dial string are needed. This could also mean that the port is busy, you dialed an incorrect number, or BNU lost ownership of the port.

OK
Auto Dial

These are informative messages only and do not indicate an error.

PHASE 3 Status Messages

Handshake Failed (LCK)

The device is being used by someone else; the process could not create the **LCK** file. Sometimes **LCK** files must be manually removed by the administrator. After a number of retries, see your system administrator. See if another process has control of the port (for example, another instance of the **uucico** daemon).

Login Failed
Timeout

The login failed due to a bad connection or possibly a slow machine.

The remote system did not respond within a set period of time. This could also indicate a problem with the chat script.

Succeeded (Call to System)
BNU (continued)

The call was completed.

These are informative messages only and do not indicate an error.

PHASE 4 Status Messages

Startup Failed
Remote reject after login

After login, the **uucico** daemon is started on the remote system. If there is a problem initiating a conversation between the two systems, these messages are created. You might have also logged into the incorrect BNU account or the initial handshake failed.

Wrong machine name
Bad login/machine combination

A machine was called incorrectly or the machine name was changed.

The login to the remote system failed. The problem could be an incorrect phone number, an incorrect login or password, or an error in the chat script.

Remote has a LCK file for me

Both systems were simultaneously trying to call each other. The local request will fail temporarily.

OK
Talking

These are informative messages only and do not indicate an error.

LOGIN:
PASSWORD:

If the login or password prompt is in all capital letters, the modem might be in echo mode (E1 on Hayes compatibles). This causes the modem to echo back, or send, a RING to your system when an incoming call is received. The **getty** command receives the string and accordingly changes the `login:` or `password:` into all caps. Change the echo mode on the modem to off (use ATE0 for Hayes compatibles).

Note: Keep in mind that once this change is made, you should use ATE1 in the chat script of your **Dialers** files, or you will not get the expected OK back from the modem.

If the remote port is set for `delay` or `getty -r` and the chat script expects key input, then the ports set for `delay` are expecting one or more carriage returns before proceeding with the login. Try beginning the chat script on the dialing system with the following:

```
" " \r\d\r\d\r\d\r in:--in: ...
```

Interpreted, this chat script reads as follows: expect nothing, send return, delay, return, delay, return, delay, return.

PHASE 5 Status Messages

Alarm	The uucico daemon is having trouble with the connection. Either the connection is bad or "xon/xoff" is set to yes on the modem.
Remote access to path/file denied copy (failed)	These messages indicate a permission problem; check file and path permissions.
Bad read	The remote system ran out of space, most likely in the spool area, or the uucico daemon could not read or write to device.
Conversation failed	The modem carrier detect was lost. Possibly the modem was turned off, the cable is loose or disconnected, or the remote system crashed or is shut down. Telephone disconnection can also cause this error.
Requested Copy (succeeded)	These are informative messages only and do not indicate an error.

PHASE 6 Status Messages

OK (Conversation Complete)	The remote system can deny the hangup request and reverse the roles (meaning the remote system has work for the local system to do). Once the two uucico daemons agree that no more work exists, they hang up.
Conversation succeeded	This is an informative message only and does not indicate an error.

Debugging BNU Login Failures Using the uucico Daemon

Prerequisites

- BNU must be installed on your system.
- A link (hardwired, modem, or TCP/IP) must be set up between your system and the remote system.
- The BNU configuration files, including the **Sysfiles** file (if applicable), the **Systems** file, **Permissions** file, **Devices** file, and **Dialers** file, must be set up for communications between your system and the remote system.

Note: You must have root user authority to modify the BNU configuration files.

- You must have root user authority to invoke the **uucico** daemon in debugging mode.

Procedure

1. To produce debugging information about a local-to-remote system connection that is not working, start the **uucico** daemon with the **-x** flag as follows:

```
/usr/sbin/uucp/uucico -r 1 -s venus -x 9
```

where **-r 1** specifies the master, or caller mode; **-s venus**, the name of the remote system to which you are trying to connect; and **-x 9**, the debug level that produces the most detailed debugging information.

2. If the expect-send sequence entry in a **Systems** file in the format of **/etc/uucp/Systems** is:

```
venus Any venus 1200 - "" \n in:--in: uucp1 word:  
mirror
```

the **uucico** daemon connects the local system to the remote system **venus**. The debugging output is similar to:

```
expect: ""  
got it  
sendthem (^J^M)  
expect (in:)^  
M^Jlogin:got it  
sendthem (uucp1^M)
```



```

expect (word:)^
M^JPassword:got it
sendthem (mirror^M)
img > ^M^J^PShere^@Login Successful: System=venus

```

where:

expect: ""	Specifies that the local system will not wait for any information from the remote system.
got it	Acknowledges that the message has been received.
sendthem (^J^M)	Specifies that the local system will send the remote system a carriage return and a new line.
expect (in:)	Specifies that the local system expects to receive the remote system login prompt, which ends in the in: character string.
^M^Jlogin:got it	Confirms that the local system received the remote login prompt.
sendthem (uucp1^M)	Specifies that the local system will send the uucp1 login ID to the remote system.
expect (word:)	Specifies that the local system expects to receive the remote system password prompt, which ends in the word: character string.
^M^JPassword:got it	Confirms the local system received the remote password prompt.
sendthem (mirror^M)	Specifies that the local system will send the password for the uucp1 login ID to the remote system.
img > ^M^J^PShere^@Login Successful: System=venus	Confirms the local system is successfully logged in to remote system venus.

Notes:

1. The expect-send debugging output produced by the **uucico** command can come either from information in the **/etc/uucp/Dialers** file or from information in the **/etc/uucp/Systems** file. Information about communication with the modem comes from the **Dialers** file, while information about communication with the remote system comes from the **Systems** file. (Note that **/etc/uucp/Systems** and **/etc/uucp/Dialers** are default BNU configuration files. Other files can be specified in **/etc/uucp/Sysfiles** to serve the same role.)
2. To set up a connection with a remote system, you must be familiar with the login sequence of that system.

Contacting Connected UNIX Systems Using the tip Command

Use the **tip** command to contact any connected system running the UNIX operating system. The **tip** command is installed with the Basic Networking Utilities (BNU) and can use the same asynchronous connections used by BNU.

The **tip** command uses variables and escape signals, as well as flags, to control its operations. The flags can be entered at the command line. The escape signals can be used over a connection with a remote system to start and stop file transfers, change the direction of a file transfer, and exit to a subshell.

tip Command Variables

The **tip** command variables define settings such as the end-of-line character, the break signal, and the mode of file transfers. Variable settings can be initialized at run time using a **.tiprc** file. Variable settings can also be changed during execution using the **~s** escape signal. Some variables, such as the end-of-line character, can be set for an individual system in the system entry of the **remote** file.

The **tip** command reads three files, the **phones** file, **remote** file, and **.tiprc** file, to determine initial settings for its variables. The **.tiprc** file must always be in the user home directory. The names and locations of the **remote** and **phones** files can vary. The names of the **remote** file and the **phones** file can be determined by environment variables:

- PHONES** Specifies the name of the user phone file. The file can have any valid file name and must be set up in the format of the file **/usr/lib/phones-file**. The default file is **etc/phones**. If a file is specified with the **PHONES** variable, it is used in place of (not in addition to) the **/etc/phones** file.
- REMOTE** Specifies the name of the user remote system definition file. The file can have any valid file name and must be set up in the format of the **/usr/lib/remote-file** file. The default file is **etc/remote**. If a file is specified with the **REMOTE** variable, it is used in place of (not in addition to) the **/etc/remote** file.

To use an environment variable, set it before starting the **tip** command. As an alternative, the names of the **phones** and **remote** files can be determined using the **tip** command **phones** variable and **remote** variable, respectively, in the **.tiprc** file.

Note: The **tip** command reads only the *last* **remote** or **phones** file specified. Thus, if you specify a **remote** or **phones** file with a variable, the new file is used in place of (not in addition to) any previous files you specified.

The **tip** command uses variable settings in the following order:

1. The command checks the settings of the **PHONES** and **REMOTE** environment variables for the files to use as the **phones** and **remote** files.
2. The command reads the **.tiprc** file and sets all variables accordingly. If the **phones** or **remote** variable is set in the **.tiprc** file, this setting overrides the environment variable setting.
3. When a connection to a remote system is initiated, the command reads the **remote** file entry for that system. The settings in the **remote** file entry override settings made in the **.tiprc** file.
4. If the **- BaudRate** flag is used with the **tip** command, the specified rate overrides all previous baud rate settings.
5. A setting made with the **~s** escape signal overrides all previous settings of a variable.

Note: Any **tip** user can create a **.tiprc** file and use this file to specify initial settings for **tip** variables. The **.tiprc** file must be placed in the user **\$HOME** directory.

tip Command Configuration Files

Before the **tip** command can connect to a remote system, the **/etc/remote** and **/etc/phones** files must be established.

- /etc/remote** Defines attributes of remote systems such as the port and type of device to use to reach the system, as well as the signals to use to indicate the beginnings and endings of transmissions.
- /etc/phones** Lists telephone numbers used to contact remote systems over a modem line.

To establish one of these files, copy a sample file to the correct name and modify it to suit the needs of your site. Sample **remote** and **phones** files are delivered with the `bos.net.uucp` package. The sample **remote** file is named `/usr/lib/remote-file`. The sample **phones** file is named `/usr/lib/phones-file`. Copy `/usr/lib/remote-file` to `/etc/remote` and modify `/etc/remote`.

A **tip** user can also create customized **remote** and **phones** files. An individual **remote** file must be in the format of the `/usr/lib/remote-file` file and specified with the **remote** variable or the **REMOTE** environment variable. An individual **phones** file must be in the format of the `/usr/lib/phones-file` file and specified with the **phones** variable or the **PHONES** environment variable. If an individual **phones** or **remote** file is specified with one of the variables, that file is read in place of (not in addition to) the `/etc/phones` or `/etc/remote` file.

Users of **tip** can use combinations of individual **phones** and **remote** files. For example, a user could use the default **remote** file, `/etc/remote`, but use an individual **phones** file named with the **phones** variable.

BNU Configuration Files

Basic Network Utilities (BNU) uses the following configuration files:

<code>/etc/uucp</code>	Contains all the configuration files for BNU.
<code>/var/spool/uucppublic</code>	Contains files that have been transferred.
<code>/etc/uucp/Systems</code>	Contains a list of systems to which the uucico program can connect.
<code>/etc/uucp/Devices</code>	Defines the device type, location, speed, and other basic communication parameters for many system dial-out programs. Only dial-out connections use this file.
<code>/etc/uucp/Permissions</code>	Creates security control, with limitations, over machines attempting to communication with your machine.
<code>/etc/uucp/Dialers</code>	Specifies the dialer types. Each dialer uses a specific command set when attempting to dial the modem. The most common dialer types are <code>hayes</code> , <code>direct</code> , and <code>TCP</code> (Transmission Control Protocol).
<code>/etc/uucp/Dialcodes</code>	Makes standardized names for certain parts of a phone number. For example, if you frequently make calls to a certain area code in San Francisco, you could create the following entry: <code>SF09,1415</code> .
<code>/etc/uucp/Sysfiles</code>	Enables a BNU administrator to specify files to fill the role of BNU configuration files other than <code>/etc/uucp/Systems</code> , <code>/etc/uucp/Devices</code> , and <code>/etc/uucp/Dialers</code> . Distinctions can be made between what files are used for uucico traffic versus cu -related (cu , ct , slattach) activity.
<code>/usr/sbin/uucp/remote.unknown</code>	Defines a shell script. It is run by the BNU program when a remote computer that is not listed in the local permissions file attempts to communicate with that local system.
<code>/etc/uucp/Poll</code>	Schedules polling of passing systems. Its format is similar to the crontab file. Poll format is <code>SiteName</code> , a tab, and the hours to poll (0-23), separated by spaces.

Correlation of Files

Systems file:	<code>SystemName Any v32ibm 9600 555-1111</code>
Devices file:	<code>v32ibm tty0 - Any ibm \D</code>
Dialers file:	<code>ibm =, -, #" \d ATSF1\r\c#0K#AFE1SD3L2MIC0SCI\r\c#0K...</code>

BNU Configuration for a TCP/IP Connection Example

The following files are set up for a Transmission Control Protocol/Internet Protocol (TCP/IP) connection between systems zeus and hera, where zeus is considered the local system and hera the remote system.

Entries in the Local System Files

Files entries on local system zeus include the following:

Systems File: The **Systems** file on system zeus should contain the following entry to allow zeus to contact system hera:

```
hera Any TCP,t - - in:--in: uzeus word: birthday
```

This specifies that system zeus can call system hera at any time, using the **t** protocol for communications with system hera. System zeus logs in to system hera as uzeus with the password birthday.

Note: The **t** protocol supports the **tcp** protocol. Therefore, always use the **t** protocol for BNU communications over TCP/IP connections. However, the **t** protocol cannot be used when the *Type* field is ACU (automatic calling unit) or when a modem connection is being used.

BNU uses the *Type* and *Class* fields in the **Systems** file to find the appropriate device for the connection. Accordingly, it checks the **Devices** file for an entry of type TCP.

Devices File: A **Devices** file used by the **uucico** daemon on system zeus should contain the following entry for TCP/IP connections:

```
TCP - - - TCP
```

Because the device type is TCP, there are no *Class*, *Line*, or *Line2* entries. The *Dialer* is also specified as TCP. Accordingly, BNU looks in the **Dialers** files for a TCP entry.

Dialers File: The **Dialers** file used by the **uucico** daemon on system zeus should contain a TCP/IP entry as follows:

```
TCP
```

This entry specifies that no dialer configuration is required.

Note: Dialer configuration is never required over a TCP/IP connection.

Permissions File: The **Permissions** file on system zeus contains the following entry granting system hera access to system zeus:

```
LOGNAME=uhera SENDFILES=yes REQUEST=yes \  
MACHINE=zeus:hera VALIDATE=uhera /  
READ=/var/spool/uucppublic:/home/hera \  
WRITE=/var/spool/uucppublic:/home/hera COMMANDS=ALL
```

This combined LOGNAME and MACHINE entry provides the following permissions to system hera on system zeus:

- System hera can request and send files regardless of who initiated the call.
- System hera can read and write to the public directory and the **/home/hera** directory on system zeus.
- System hera can execute all commands on system zeus.
- System hera must log in to system zeus as user uhera and cannot use any other login ID for BNU transactions.

Note: Because the permissions are the same regardless of which system initiates the call, the preceding LOGNAME and MACHINE entries are combined. Separately, they are:

```
LOGNAME=uhera VALIDATE=hera SENDFILES=yes REQUEST=yes& \  
READ=/var/spool/uucppublic:/home/hera \  
WRITE=/var/spool/uucppublic:/home/hera
```

```
MACHINE=zeus:hera REQUEST=yes COMMANDS=ALL\  
READ=/var/spool/uucppublic:/home/hera \  
WRITE=/var/spool/uucppublic:/home/hera
```

Entries in the Remote System's Files

Files on remote system hera include the following:

Systems File: A **Systems** file on system hera should contain the following entry to allow hera to contact system zeus:

```
zeus Any TCP,t - - ogin:--ogin: uhera ord: lightning
```

This specifies that system hera can call system zeus at any time, using the **t** protocol for communications with system zeus. System hera logs in to system zeus as user uhera with the password lightning. Again, BNU next checks the **Devices** files for an entry of type TCP.

Note: The **t** protocol supports the **tcp** protocol. Therefore, always use the **t** protocol for BNU communications over TCP/IP connections. However, the **t** protocol cannot be used when the *Type* field is ACU or when a modem connection is being used.

Devices File: The **Devices** file used by the **uucico** daemon on system hera should contain the following entry for TCP/IP connections:

```
TCP - - - TCP
```

Because the device type is TCP, there are no *Class*, *Line*, or *Line2* entries. The *Dialer* is also specified as TCP. Accordingly, BNU looks in the **Dialers** files for a TCP entry.

Dialers File: The **Dialers** file used by the **uucico** daemon on system hera should contain a TCP/IP entry as follows:

```
TCP
```

This entry specifies that no dialer configuration is required.

Note: Dialer configuration is never required over a TCP/IP connection.

Permissions File: The **Permissions** file on system hera contains the following entry which grants system zeus access to system hera:

```
LOGNAME=uzeus SENDFILES=yes REQUEST=yes \  
MACHINE=hera:zeus VALIDATE=zeus COMMANDS=rmail:who:uucp
```

This combined LOGNAME and MACHINE entry provides the following permissions to system zeus on system hera:

- System zeus can request and send files regardless of who initiated the call.
- System zeus can read and write only to the public directory (the default).
- System zeus can run only the **rmail**, **who**, and **uucp** commands.
- System zeus must log in to system hera as user uzeus and cannot use any other login ID for BNU transactions.

Note: Separately, the LOGNAME and MACHINE entries are:

```
LOGNAME=uzeus VALIDATE=zeus SENDFILES=yes REQUEST=yes  
MACHINE=hera:zeus COMMANDS=rmail:who:uucp REQUEST=yes
```

BNU Configuration for a Telephone Connection Example

The following sample files are set up to connect systems `venus` and `merlin` over a telephone line using modems. System `venus` is considered the local system and system `merlin` the remote system.

On both systems, the device `tty1` is hooked to a Hayes modem at 1200 baud. The login ID used for system `venus` to log into system `merlin` is `uvenus`, and the associated password is `mirror`. The login ID for system `merlin` to log into system `venus` is `umerlin`, and the associated password is `oaktree`. The phone number for the modem attached to `venus` is 9=3251436; the number of the `merlin` modem is 9=4458784. Both computers include partial phone numbers in their **Systems** files and dial-codes in their **Dialcodes** files.

Entries on the Local System

Files containing telephone connection entries on local system `venus` include the following:

Systems File: The **Systems** file on `venus` should contain the following entry for `merlin`, including a phone number and a dialing prefix:

```
merlin Any ACU 1200 local8784 "" in:--in: uvenus word: mirror
```

System `venus` can call system `merlin` at any time, using an ACU device at 1200 baud and logging in as `uvenus` with the password `mirror`. The telephone number is expanded based on the code `local` in the **Dialcodes** file, and the device to be used is determined based on the *Type* and *Class* entries. Accordingly, BNU checks the **Devices** files for a device of type ACU and class 1200.

Dialcodes File: The **Dialcodes** file on system `venus` contains the following dial-code prefix for use with the number in the **Systems** file:

```
local 9=445
```

Given this code, the telephone number for system `merlin` in the **Systems** file is expanded to 9=4458784.

Devices File: The **Devices** file on system `venus` should contain the following entry for the connection to system `merlin`:

```
ACU tty1 - 1200 hayes \T
```

The port to be used is `tty1`, and the *Dialer* entry in the *Dialer-Token Pairs* field is `hayes`. The *Token* entry, `\T`, indicates that the telephone number is to be expanded using a code from the **Dialcodes** file. BNU checks the **Dialers** files for a `hayes` dialer type.

Dialers File: A **Dialers** file used by the `uucico` daemon on system `venus` should contain the following entry for the `hayes` modem:

```
hayes =,-, "" \DAT\r\c OK \PATDT\T\r\c CONNECT
```

Note: The expect-send characters are defined in the **Dialers** file format.

Permissions File: The **Permissions** file on system `venus` contains the following entries specifying the ways in which system `merlin` can conduct `uucico` and `uuxqt` transactions with system `venus`:

```
LOGNAME=umerlin REQUEST=yes SENDFILES=yes \  
READ=/var/spool/uucppublic:/home/merlin \  
WRITE=/var/spool/uucppublic:/home/merlin \  
MACHINE=venus:merlin VALIDATE=umerlin REQUEST=yes SENDFILES=yes \  
COMMANDS=ALL \  
READ=/var/spool/uucppublic:/home/merlin \  
WRITE=/var/spool/uucppublic:/home/merlin
```

System `merlin` logs in to system `venus` as `umerlin`, which is a unique login for system `merlin`. It can request and send files regardless of who initiated the call. Also, system `merlin` can read and write to the

`/var/spool/uucppublic` directory and the `/home/merlin` directory on system `venus`. It can issue all commands in the default command set on system `venus`.

Entries on the Remote System

Files containing telephone connection entries on remote system `merlin` include the following:

Systems File: A **Systems** file on `merlin` should contain the following entry for `venus`, including a phone number and a dialing prefix:

```
venus Any ACU 1200 intown4362 "" in:--in: umerlin word: oaktree
```

System `merlin` can call system `venus` at any time, using an ACU device at 1200 baud and logging in as user `umerlin` with the password `oaktree`. The telephone number is expanded based on the code `intown` in the **Dialcodes** file, and the device to be used is determined based on the *Type* and *Class* entries. Accordingly, BNU checks the **Devices** file(s) for a device of type ACU and class 1200.

Dialcodes File: The **Dialcodes** file on system `merlin` contains the following dial-code prefix for use with the number in the **Systems** file:

```
intown 9=325
```

Therefore, the expanded telephone number to reach system `venus` is 9=3254362.

Devices File: A **Devices** file on system `merlin` should contain the following entry for the connection to `venus`:

```
ACU tty1 - 1200 hayes \T
```

The ACU is attached to port `tty1`, and the dialer is `hayes`. The telephone number is expanded with information from the **Dialcodes** file. BNU checks the **Dialers** files for an entry for a `hayes` modem.

Dialers File: A **Dialers** file used by the `uucico` daemon on system `merlin` should contain the following entry for its modem:

```
hayes =,-, "" \dAT\r\c OK \pATDT\T\r\c CONNECT
```

Permissions File: The **Permissions** file on system `merlin` contains the following entries which grant system `venus` access to `merlin`:

```
LOGNAME=uvenus SENDFILES=call REQUEST=no \  
WRITE=/var/spool/uucppublic:/home/venus \  
READ=/var/spool/uucppublic:/home/venus \  
MACHINE=merlin:venus VALIDATE=uvenus \  
READ=/ WRITE=/ COMMANDS=ALL REQUEST=yes \  
NOREAD=/etc/uucp:/usr/etc/secure \  
NOWRITE=/etc/uucp:/usr/etc/secure
```

BNU Configuration for a Direct Connection Example

The following files are set up for a hardwired connection between systems `zeus` and `hera`, where `zeus` is considered the local system and `hera` the remote system. The hardwired device on system `zeus` is `tty5`; on system `hera` it is `tty1`. The speed of the connection is 1200 bps. The login ID for system `zeus` on system `hera` is `uzeus`, and the associated password is `thunder`. The login ID for system `hera` on system `zeus` is `uhera`, and the associated password is `portent`.

Entries in the Local System's Files

Files containing telephone connection entries on local system `zeus` include the following:

Systems File: A **Systems** file on `zeus` should contain the following entry for the remote system `hera`:

```
hera Any hera 1200 - "" \r\d\r\d\r in:--in: uzeus word: thunder
```


This entry specifies that system hera can log in to system zeus at any time, using a direct connection specified in the **Devices** files. To find the entry in the **Devices** files, BNU uses the third and fourth fields of the **Systems** entry. Thus, BNU looks for an entry in the **Devices** files with a *Type* of hera and a *Class* of 1200. System zeus logs in to system hera as user uzeus with the password thunder.

Devices File: A **Devices** file on zeus should contain the following entry in order to connect to the remote system hera:

```
hera    tty5 - 1200 direct
```

This entry specifies that system zeus uses the device tty5 at 1200 bps to communicate with system hera. Note that the *Dialer* in both *Dialer-Token Pairs* fields is direct. When connecting to system hera, BNU checks the **Dialers** file for a direct entry.

Dialers File: A **Dialers** file on system zeus must contain the following entry for direct connections:

```
direct
```

This specifies that no handshaking is required on the direct connection.

Permissions File: The **Permissions** file on the local system zeus contains the following entry specifying the ways in which the remote system hera can conduct **uucico** and **uuxqt** transactions with zeus:

```
LOGNAME=uhera MACHINE=hera VALIDATE=uhera REQUEST=yes \  
SENDFILES=yes MACHINE=zeus READ=/ WRITE=/ COMMANDS=ALL
```

This entry specifies that system hera logs in as uhera. Since the VALIDATE=uhera option is included, system hera cannot log in to system zeus with any other login ID, nor can any other remote system use the uhera ID. System hera can read and write to any directory on system zeus, and can send and request files regardless of who initiated the call. System hera can also initiate any commands on system zeus.

Note: Since the permissions that are granted are the same regardless of which system initiated the connection, the LOGNAME and MACHINE entries have been combined. Separately, they are:

```
LOGNAME=uhera REQUEST=yes SENDFILES=yes READ=/ WRITE=  
MACHINE=zeus:hera VALIDATE=uhera READ=/ WRITE=/ REQUEST=yes \  
COMMANDS=ALL
```

Attention: Providing the permissions in the preceding example is equivalent to giving any user on the remote system a login ID on the local system. Such liberal permissions can jeopardize your security and should usually be given only to well-trusted remote systems at the same site.

Entries in the Remote System Files

Files containing telephone connection entries on remote system hera include the following:

Systems File: A **Systems** file on system hera must contain the following entry for zeus:

```
zeus Any zeus 1200 - "" \r\d\r\d\r in:--in: uhera word: portent
```

This entry specifies that system hera can log in to system zeus at any time, using a direct connection specified in the **Devices** files. To find the entry in the **Devices** files, BNU uses the third and fourth fields of the **Systems** entry. Thus BNU looks for an entry in the **Devices** files with a *Type* of zeus and a *Class* of 1200. System hera logs in to system zeus as user uhera with the password portent.

Devices File: A **Devices** file on system hera must contain the following entry for communications with zeus:

```
zeus    tty1 - 1200 direct
```

This entry specifies that system hera uses the device tty1 at 1200 bps to communicate with system zeus. Since the *Dialer* is specified as direct, BNU checks the **Dialers** files for a direct entry.

uudemon.hour	Initiates file transport calls to remote systems.
uudemon.poll	Polls remote systems listed in the /etc/uucp/Poll file.
/var/spool/uucp/audit	Contains audit information from BNU activities.
/var/spool/uucp/Foreign	Contains error information about BNU activities.
/var/spool/uucp/errors	Contains error information about BNU activities.
/var/spool/uucp/xferstats	Contains statistical information about BNU activities.
/var/spool/uucp/Corrupt	Contains copies of files that cannot be processed by the BNU program.
/var/spool/uucp/.Log	Contains log files from current BNU transactions.
/var/spool/uucp/.Old	Contains log files from old BNU transactions.
/var/spool/uucp/.Status	Stores the last time the uucico daemon tried to contact remote systems.
/var/spool/uucp/SystemName/C.*	These files are the commands allowed when connected to <i>SystemName</i> .
/var/spool/uucp/SystemName/D.*	These files are data files associated with <i>SystemName</i> .
/var/spool/uucp/SystemName/X.*	Executable files on <i>SystemName</i> .
/var/spool/uucp/SystemName/TM.*	Temporary files used when connected to <i>SystemName</i> .

BNU Commands

ct	Connects to another system over a telephone line.
cu	Connects to another system.
tip	A variation of cu that requires special configuration.
uucp	Copies files from one system to another running BNU or a version of the UNIX-to-UNIX Copy Program (UUCP).
uudecode	Reconstructs a binary file encoded with uuencode .
uuencode	Encodes a binary file into ASCII form for transmission using BNU.
uuname	Provides information about accessible systems.
uupoll	Forces a call to a remote system.
uuq	Displays the BNU job queue.
uusend	Sends a file to a remote host running BNU or UUCP.
uusnap	Displays a brief summary of BNU status.
uustat	Reports the status of BNU operations.
uuto	Copies files to another system using BNU or UUCP.
uux	Runs a command on a remote system.
uucheck	Checks the /etc/uucp/Permissions file for correct configuration.
uuname	Shows the names of all systems that can be reached through BNU.
uuclean	Cleans BNU spooling directories.
uucleanup	Cleans BNU spooling directories.
uukick	Contacts a remote system with debugging enabled.
uulog	Displays BNU log files.
uutry	Contacts a remote system with debugging enabled; allows override of retry time.
uucpadm	Administers the BNU system.
uupick	Enables users to claim files in the /var/spool/uucppublic directory.
uucp	Login ID with full administrative authority over the BNU subsystem.
Uutry	Contacts a remote system with debugging turned on, and saves the debugging output in a file.

BNU Daemons

uucico	Contacts remote systems and transfers files.
uucpd	Allows BNU to run on top of Transmission Control Protocol/Internet Protocol (TCP/IP).
uusched	Schedules BNU jobs.
uuxqt	Runs command requests from remote systems.

Chapter 9. Network Management

The Network Management facility provides comprehensive management of system networks through the use of Simple Network Management Protocol (SNMP) enabling network hosts to exchange management information. SNMP is an internetworking protocol designed for use with TCP/IP-based internets. The following sections provide information to assist the network manager in understanding and working with SNMP:

- SNMP for Network Management
- SNMP Access Policies
- SNMP Daemon
- Configuring the SNMP Daemon
- SNMP Daemon Processing
- SNMP Daemon Support of EGP Family of MIB Variables
- SNMP Daemon RFC Conformance
- SNMP Daemon Implementation Restrictions
- SNMP Daemon Logging Facility
- Problem Determination for the SNMP Daemon.

You may also want to consult the information in *SNMP Overview for Programmers*, in *AIX 5L Version 5.1 Communications Programming Concepts*.

SNMP for Network Management

SNMP network management is based on the familiar client/server model that is widely used in TCP/IP-based network applications. Each host that is to be managed runs a process called an *agent*. The agent is a server process that maintains the Management Information Base (MIB) database for the host. Hosts that are involved in network management decision-making can run a process called a manager. A *manager* is a client application that generates requests for MIB information and processes responses. In addition, a manager may send requests to agent servers to modify MIB information.

For more information on SNMP, see *Simple Network Management Protocol (SNMP)* in *AIX 5L Version 5.1 Communications Programming Concepts*. The following RFCs might also be of use:

RFC 1155	Structure and Identification of Management Information (SMI) for TCP/IP-based internets.
RFC 1157	Simple Network Management Protocol (SNMP).
RFC 1213	Management Information Base (MIB) for Network Management of TCP/IP-based internets.
RFC 1227	Simple Network Management Protocol (SNMP) single multiplexer (SMUX) protocol and Management Information Base (MIB).
RFC 1228	Simple Network Management Protocol-Distributed Program Interface (SNMP-DPI).
RFC 1229	Extensions to the generic interface Management Information Base (MIB).
RFC 1231	IEEE 802.5 token-ring Management Information Base (MIB).
RFC 1398	Definitions of Managed objects for the Ethernet. For example, interface type.
RFC 1512	Fiber Distributed Data Interface (FDDI) Management Information Base (MIB).

SNMP Access Policies

The **snmpd** agent uses a simple authentication scheme to determine which Simple Network Management Protocol (SNMP) manager stations can access its Management Information Base (MIB) variables. This authentication scheme involves the specification of SNMP access policies. An SNMP access policy is an administrative relationship involving an association among an SNMP community, an access mode, and an MIB view.

An *SNMP community* is a group of one or more hosts and a community name. A community name is a string of octets that an SNMP manager must embed in an SNMP request packet for authentication purposes.

The *access mode* specifies the access the hosts in the community are allowed with respect to retrieving and modifying the MIB variables from a specific SNMP agent. The access mode must be one of: *none*, *read-only*, *read-write*, or *write-only*.

A *MIB view* defines one or more MIB subtrees that a specific SNMP community can access. The MIB view can be the entire MIB tree or a limited subset of the entire MIB tree.

When the SNMP agent receives a request, the agent verifies the community name with the requesting host Internet Protocol (IP) address to determine if the requesting host is a member of the SNMP community identified by the community name. If the requesting host is indeed a member of the SNMP community, the SNMP agent then determines if the requesting host is allowed the specified access for the specified MIB variables as defined in the access policy associated with that community. If all criteria are met, the SNMP agent attempts to honor the request. Otherwise, the SNMP agent generates an *authenticationFailure* trap or returns the appropriate error message to the requesting host.

The SNMP access policies for the **snmpd** agent are user-configurable and are specified in the **/etc/snmpd.conf** file. To configure the SNMP access policies for the **snmpd** agent, see the **/etc/snmpd.conf** file.

SNMP Daemon

The Simple Network Management Protocol (SNMP) daemon is a background server process that can be run on any Transmission Control Protocol/Internet Protocol (TCP/IP) workstation host. The daemon, acting as SNMP agent, receives, authenticates, and processes SNMP requests from manager applications. See Simple Network Management Protocol, How a Manager Functions, and How an Agent Functions in *AIX 5L Version 5.1 Communications Programming Concepts* for more detailed information on agent and manager functions.

Note: The terms SNMP daemon, SNMP agent, and agent are used interchangeably.

The **snmpd** daemon requires the loopback TCP/IP interface to be active for minimal configuration. Enter the following command before starting TCP/IP:

```
ifconfig lo0 loopback up
```

Configuring the SNMP Daemon

The Simple Network Management Protocol (SNMP) daemon will attempt to bind sockets to certain well-known User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) ports, which must be defined in the **/etc/services** file as follows:

```
snmp          161/udp
snmp-trap     162/udp
smux          199/tcp
```

The **snmp** service must be assigned port 161, as required by RFC 1157. The **/etc/services** file assigns ports 161, 162, and 199 to these services. If the **/etc/services** file is being serviced off another machine, these assigned ports must be made available in the served **/etc/services** file on the server before the **SNMP** daemon can run.

The **SNMP** daemon reads a configuration file, **/etc/snmpd.conf**, on startup and when a **refresh** command (if the **snmpd** daemon is invoked under System Resource Controller control) or **kill -1** signal is issued.

This configuration file specifies the community names and associated access privileges and views, hosts for trap notification, logging attributes, **snmpd**-specific parameter configurations, and single multiplexer (SMUX) configurations for the **SNMP** daemon. See the **/etc/snmpd.conf** file for more information.

SNMP Daemon Processing

The Simple Network Management Protocol (**SNMP**) daemon processes **SNMP** requests from manager applications. Read Simple Network Management Protocol (SNMP), How a Manager Functions, and How an Agent Functions in *AIX 5L Version 5.1 Communications Programming Concepts* for more detailed information on agent and manager functions.

Message Processing and Authentication

All requests, traps, and responses are transmitted in the form of ASN.1-encoded messages. A message, as defined by RFC 1157, has the following structure:

Version Community PDU

where *Version* is the SNMP version (currently version 1), *Community* is the community name, and *PDU* is the protocol data unit that contains the SNMP request, response, or trap data. A PDU is also encoded according to ASN.1 rules.

The **SNMP** daemon receives and transmits all **SNMP** protocol messages through the Transmission Control Protocol/Internet Protocol (TCP/IP) User Datagram Protocol (UDP). Requests are accepted on well-known port 161. Traps are transmitted to the hosts listed in the trap entries in the **/etc/snmpd.conf** file that are listening on well-known port 162.

When a request is received, the source IP address and the community name are checked against a list containing the IP addresses, community names, permissions, and views as specified in the community and view entries in the **/etc/snmpd.conf** file. The **snmpd** agent reads this file at startup and on a **refresh** command or a **kill -1** signal. If no matching entry is found, the request is ignored. If a matching entry is found, access is allowed according to the permissions specified in the community and view entries for that IP address, community, and view name association in the **/etc/snmpd.conf** file. Both the message and the PDU must be encoded according to the ASN.1 rules.

This authentication scheme is not intended to provide full security. If the **SNMP** daemon is used only for get and get-next requests, security might not be a problem. If set requests are allowed, the set privilege can be restricted.

See the **/etc/snmpd.conf** file for further information. See Management Information Base (MIB) for further information.

Request Processing

There are three types of request PDUs that can be received by the **SNMP** daemon. The request types are defined in RFC 1157, and the PDUs all have the following format:

Request PDU Format			
request-ID	error-status	error-index	variable-bindings
GET	0	0	<i>VarBindList</i>
GET-NEXT	0	0	<i>VarBindList</i>
SET	0	0	<i>VarBindList</i>

The request-ID field identifies the nature of the request; the error-status field and error-index field are unused and must be set to 0 (zero); and the variable-bindings field contains a variable-length list of numeric-format instance IDs whose values are being requested. If the value of the request-ID field is SET, the variable-bindings field is a list of pairs of instance IDs and values.

Read Using the Management Information Base (MIB) Database for a discussion of the three request types.

Response Processing

Response PDUs have nearly the same format as request PDUs:

Response PDU Format			
request-ID	error-status	error-index	variable-bindings
GET-RESPONSE	<i>ErrorStatus</i>	<i>ErrorIndex</i>	<i>VarBindList</i>

If the request was successfully processed, the value for both the error-status and error-index field is 0 (zero), and the variable-bindings field contains a complete list of pairs of instance IDs and values.

If any instance ID in the variable-bindings field of the request PDU was not successfully processed, the SNMP agent stops processing, writes the index of the failing instance ID into the error-index field, records an error code in the error-status field, and copies the partially completed result list into the variable-bindings field.

RFC 1157 defines the following values for the error-status field:

Values for the Error-Status Field		
Value	Value	Explanation
<i>noError</i>	0	Processing successfully completed (error-index is 0).
<i>tooBig</i>	1	The size of the response PDU would exceed an implementation-defined limit (error-index is 0).
<i>noSuchName</i>	2	An instance ID does not exist in the relevant MIB view for GET and SET request types or has no successor in the MIB tree in the relevant MIB view for GET-NEXT requests (nonzero error-index).
<i>badValue</i>	3	For SET requests only, a specified value is syntactically incompatible with the type attribute of the corresponding instance ID (nonzero error-index).
<i>readOnly</i>	4	Not defined.
<i>genErr</i>	5	An implementation-defined error occurred (nonzero error-index); for example, an attempt to assign a value that exceeds implementation limits.

Trap Processing

Trap PDUs are defined by RFC 1157 to have the following format:

Trap PDU Format					
enterprise	agent- address	generic- trap	specific- trap	time-stamp	variable- bindings
<i>Object ID</i>	<i>Integer</i>	<i>Integer</i>	<i>Integer</i>	<i>TimeTicks</i>	<i>VarBindList</i>

The fields are used as follows:

<i>enterprise</i>	The object identifier assigned to the vendor implementing the agent. This is the value of the sysObjectID variable, and it is unique for each implementer of an SNMP agent. The value assigned to this implementation of the agent is 1.3.6.1.4.1.2.3.1.2.1.1.3 , or risc6000snmpd.3 .
<i>agent-address</i>	IP address of the object generating the trap.
<i>generic-trap</i>	Integer, as follows: <ul style="list-style-type: none"> 0 <i>coldStart</i> 1 <i>warmStart</i> 2 <i>linkDown</i> 3 <i>linkUp</i> 4 <i>authenticationFailure</i> 5 <i>egpNeighborLoss</i> 6 <i>enterpriseSpecific</i>
<i>specific-trap</i>	Unused, reserved for future development.
<i>time-stamp</i>	Elapsed time, in hundredths of a second, from the last reinitialization of the agent to the event generating the trap.
<i>variable-bindings</i>	Extra information, dependent on <i>generic-trap</i> type.

The following generic-trap values indicate that certain system events have been detected:

<i>coldStart</i>	The agent is reinitializing. Configuration data or MIB variable values, or both, might have changed. Restart the measurement epochs.
<i>warmStart</i>	The agent is reinitializing but configuration data or MIB variable values have not changed. In this implementation of the SNMP agent, a <i>warmStart</i> trap is generated when the /etc/snmpd.conf file is reread. The configuration information in the /etc/snmpd.conf file is for agent configuration that has no side effects on SNMP manager databases. Measurement epochs should not be restarted.
<i>linkDown</i>	The agent has detected that a known communications interface has been disabled.
<i>linkUp</i>	The agent has detected that a known communications interface has been enabled.
<i>authenticationFailure</i>	A message was received that could not be authenticated.
<i>egpNeighborLoss</i>	An Exterior Gateway Protocol (EGP) neighbor was lost. This value is only generated when the agent is running on a host that runs the gated daemon using EGP.
<i>enterpriseSpecific</i>	Not implemented; reserved for future use.

The *linkDown* and *linkUp* traps contain a single instance ID/value pair in the variable-bindings list. The instance ID identifies the **ifIndex** of the adapter that was disabled or enabled, and the value is the **ifIndex** value. The trap for *egpNeighborLoss* also contains a binding consisting of the instance ID and value of *egpNeighAddr* for the lost neighbor.

Generation of linkUp and linkDown Traps

Note: In the following sections, interfaces applies to both a TCP/IP interface and a Common Data Link Interface (CDLI) token-ring, Ethernet, or Fiber Distributed Data Interface (FDDI) device. CDLI allows **snmpd** to monitor the Ethernet, token-ring, and FDDI devices even if they are not running TCP/IP. The SNMP daemon stills needs TCP/IP loopback to run, but it is no longer a requirement on interfaces.

The *linkUp* and *linkDown* traps are generated when the **snmpd** agent determines there is a change in the state for a known interface. If the current state of a known interface is down and the interface state changes to up, a *linkUp* trap will be generated. Likewise, if the current state of a known interface is up and the interface changes to down, a *linkDown* trap is generated.

There is no concept of up or down to a CDLI device if there is not a TCP/IP interface running on that device. CDLI devices with a TCP/IP interface layer attached is always considered to be up unless the device is removed from the system. The other values of the interfaces table are also affected by whether a CDLI device has a TCP/IP interface or not. If a CDLI device has a TCP/IP interface, then all the statistics in the interfaces table pertain to the TCP/IP interface and the device specific MIB are used to retrieve statistics about the device. If there is not a TCP/IP interface layer running on the CDLI device, the iftable entries for the device are retrieved from the CDLI device itself.

If an interface is known to the **snmpd** agent, there is an entry for that interface in the **snmpd** interfaces table. When an interface is attached or detached by the **ifconfig** command, the entries in the **snmpd** agent interfaces table change. A *coldStart* trap is generated to indicate the configuration change. The purpose of generating a *coldStart* trap is to guarantee the receiving host understands that crucial MIB variable values might have changed. In particular, measurement epochs should be restarted. Even though links might be enabled or disabled, no *linkUp* or *linkDown* traps are generated.

For instance, a **netstat -in** command might provide the following information for a host network configuration:

Name	Mtu	Network	Address	Ipkts	Ierrs	Opkts	Oerrs
Coll							
lo0	1536	127	127.0.0.1	6228	0	6228	0 0
en0	1500	192.100.154	192.100.154.7	585287	0	666636	0 0
tr0	1500	129.35.32	129.35.42.141	3976323	0	2414030	0 0

In this example, network configuration, the **snmpd** agent has three entries in its interface table from TCP/IP interfaces: one for *lo0*, one for *en0*, and one for *tr0*. In the interface index table, ifIndex.1 might refer to *lo0*, ifIndex.2 might refer to *en0*, and ifIndex.3 might refer to *tr0*. The term "might refer to" implies that since interfaces are dynamic, the actual entry number might vary. In the example, the TCP/IP kernel does not know about an additional token-ring adapter in the workstation; the interface for this adapter is named *tr1*. Because, the token-ring adapter is a CDLI device, there is an entry in the interfaces table, for example, ifIndex.4. The workstation has a serial optical device. The serial optical device has not been configured for TCP/IP and it is not a CDLI device. Neither TCP/IP nor **snmpd** recognize this device. Its interface name is *so0*.

If you issue the **ifconfig tr1** command, TCP/IP attaches *tr1*, but does not mark the interface as up. The **snmpd** agent changes the method for reporting interface table statistics to the TCP/IP interface layer from the CDLI device level statistics. The **snmpd** agent then generates a *coldStart* trap. This action has not added a new entry, because an entry for *tr1* already existed.

If you issue the **ifconfig so0** command, TCP/IP attaches *so1*, but does not mark the interface as up. The **snmpd** agent then adds a fifth entry for this new interface to its interfaces table and sets *ifIndex.5* to refer to *so0*. The SNMP agent then generates a *coldStart* trap. Since, *so0* is not a CDLI device, it was not in the interface table as a result of device configuration, so it had to be added when the TCP/IP interface layer was configured.

The crucial interface index (*ifIndex*) values that an SNMP manager might have stored for the four original entries do not change. But the *coldStart* trap is a signal to SNMP managers that their MIB databases need to be updated. Upon refreshing its database, an SNMP manager learns about this new entry in the **snmpd** agent interface table.

If the system administrator issued an **ifconfig tr1 up** command, the new interface is marked up and the method of obtaining statistics is changed. The **snmpd** agent sends a *coldStart* trap; a *linkUp* trap is not sent because the state of the device did not change. A CDLI device is always considered up, until it gets a TCP/IP interface.

The *coldStart* trap indicates to the SNMP managers that the **snmpd** agent configuration had changed, so the managers should update their MIB databases. A *linkUp* trap generated after the *coldStart* trap is meaningless to the SNMP manager because the manager database already has the information from its database refresh.

The system administrator elected to detach the *en0* interface from the previous network configuration example. Once detached, the **snmpd** agent updates its statistics gathering method to use the CDLI device statistics (Ethernet is a CDLI device). The agent generates a *coldStart* trap to inform the SNMP managers that something in the interfaces is different. In this case, the crucial *ifIndex* values have not changed, just the method of statistic retrieval.

The system administrator has elected to detach the *so0* interface from the above network configuration. Upon the detachment of an interface, the **snmpd** agent updates its interfaces table. In this example, all the indexes remain the same except the fifth entry has been removed. If the third entry is removed, the fourth and fifth entry are renumbered to be the third and fourth entries. In either case, the **snmpd** agent generates a *coldStart* trap.

In this case, any crucial *ifIndex* values that an SNMP manager might have stored for the original entries will change. The SNMP manager should refresh its MIB database to reflect the changes in the **snmpd** agent interfaces table. In this situation, a *linkDown* trap is not generated. An SNMP manager cannot take immediate action upon receipt of a *linkDown* trap because the *ifIndex* values in its database are no longer valid.

In order for the **snmpd** agent to catch all interface status changes, the **snmpd** agent periodically checks the TCP/IP kernel and CDLI device list to determine the status of the interfaces. The time interval at which these checks are run is user-configurable.

In the event that the **snmpd** agent has received a request for a MIB variable in the interfaces table and the **snmpd** agent determines that an interface has changed state to the degree that a *coldStart* should be generated, the **snmpd** agent returns a *genErr* and issues the *coldStart* trap.

See User Datagram Protocol (UDP), Exterior Gateway Protocol (EGP), and TCP/IP Addressing for more information on protocols and Internet addresses.

SNMP Daemon Support for the EGP Family of MIB Variables

If the agent host is running the **gated** daemon with the Exterior Gateway Protocol (EGP) enabled, there are several Management Information Base (MIB) variables in the EGP group supported by the **gated** daemon which the **snmpd** agent can access.

The following EGP MIB variables have a single, unique instance:

egpInMsgs	Number of EGP messages received without error.
egpInErrors	Number of EGP messages received in error.
egpOutMsgs	Total number of EGP messages transmitted by the gated daemon running on the agent's host.
egpOutErrors	Number of EGP messages that could not be sent by the agent host gated daemon because of resource limitations.
egpAs	Autonomous system number of the agent host gated daemon.

The following EGP MIB variables have an instance for each EGP peer or neighbor acquired by the agent host **gated** daemon:

egpNeighState	The state of this EGP peer: <table><tr><td>1</td><td>idle</td></tr><tr><td>2</td><td>acquisition</td></tr><tr><td>3</td><td>down</td></tr><tr><td>4</td><td>up</td></tr><tr><td>5</td><td>cease.</td></tr></table>	1	idle	2	acquisition	3	down	4	up	5	cease.
1	idle										
2	acquisition										
3	down										
4	up										
5	cease.										
egpNeighAddr	IP address of this EGP peer.										
egpNeighAs	Autonomous system number of this EGP peer. Zero (0) indicates the autonomous system number of this peer is not yet known.										
egpInNeighMsgs	Number of EGP messages received without error from this EGP peer.										
egpNeighInErrs	Number of EGP messages received in error from this EGP peer.										
egpNeighOutMsgs	Number of locally generated EGP messages to this EGP peer.										
egpNeighOutErrs	Number of locally generated EGP messages not sent to this EGP peer because of resource limitations.										
egpNeighInErrMsgs	Number of EGP-defined error messages received from this EGP peer										
egpNeighOutErrMsgs	Number of EGP-defined error messages sent to this EGP peer.										
egpNeighStateUp	Number of EGP state transitions to the UP state with this EGP peer.										
egpNeighStateDowns	Number of EGP state transitions from the UP state to any other state with this EGP peer.										
egpNeighIntervalHello	Interval between EGP Hello command retransmissions in hundredths of a second.										
egpNeighIntervalPoll	Interval between EGP poll command retransmissions in hundredths of a second.										
egpNeighMode	Polling mode of this EGP peer. The mode can be either active (1) or passive (2).										
egpNeighEventTrigger	Control variable triggers operator-initiated start and stop events to this EGP peer. This MIB variable can be set to start (1) or stop (2).										

If the **gated** daemon is not running, or if the **gated** daemon is running but is not configured to communicate with the **snmpd** agent, or if the **gated** daemon is not configured for EGP, get and set requests for the values of these variables will return the *noSuchName* error response code.

The **gated** daemon configuration file, **/etc/gated.conf**, should contain the following statement:

```
snmp      yes;
```

The **gated** daemon is internally configured to be an Simple Network Management Protocol (SNMP) single multiplexer (SMUX) protocol peer, or proxy agent, of the **snmpd** daemon. When the **gated** daemon starts up, it registers the *ipRouteTable* MIB variable tree with the **snmpd** agent. If the **gated** daemon is configured for EGP, then the **gated** daemon also registers the EGP MIB variable tree. After this

registration is complete, an SNMP manager can successfully make requests to the **snmpd** agent for the *ipRouteTable* and EGP MIB variables supported by this agent host **gated** daemon. When the **gated** daemon is running, all MIB routing information is obtained using the **gated** daemon. In this case, set requests to the *ipRouteTable* are not allowed.

The SMUX communication between the **gated** daemon and the **snmpd** daemon takes place over the well-known Transmission Control Protocol (TCP) port 199. If the **gated** daemon terminates, **snmpd** immediately unregisters the trees the **gated** daemon previously registered. If the **gated** daemon is started before the **snmpd** daemon, the **gated** daemon periodically checks for the **snmpd** daemon until the SMUX association can be established.

To configure the **snmpd** agent to recognize and allow the SMUX association with the **gated** daemon client, the user must add a SMUX entry to the */etc/snmpd.conf* file. The client object identifier and password specified in this SMUX entry for the **gated** daemon must match those specified in the */etc/snmpd.peers* file.

The **snmpd** agent supports set requests for the following read-write MIB I and MIB II variables:

sysContact

The textual identification of the contact person for this agent host. This information includes the name of this person and how to contact this person: for example, "Bob Smith, 555-5555, ext 5." The value is limited to 256 characters. If, for a set request, the string for this MIB variable is greater than 256 characters, the **snmpd** agent will return the error *badValue*, and the set operation is not performed. The initial value of *sysContact* is defined in */etc.snmp.conf*. If nothing is defined, the value is null string.

Instance	Value	Action
0	"string"	The MIB variable is set to "string".

sysName

The host name for this agent host. Typically this is the fully qualified domain name of the node. The value is limited to 256 characters. If, for a set request, the string for this MIB variable is greater than 256 characters, the **snmpd** agent returns the error *badValue*, and the set operation is not performed.

Instance	Value	Action
0	"string"	The MIB variable is set to "string".

sysLocation

A textual string stating the physical location of the machine on which this **snmpd** agent resides: for example, "Austin site, building 802, lab 3C-23." The value is limited to 256 characters. If, for a set request, the string for this MIB variable is greater than 256 characters, the **snmpd** agent returns the error *badValue*, and the set operation is not performed. The initial value of *sysLocation* is defined in */etc/snmp.conf*. If nothing is defined, the value is null string.

Instance	Value	Action
0	"string"	The MIB variable is set to "string".

ifAdminStatus

The desired state of an interface adapter on the agent's host. Supported states are up and down. The state can be set to testing but such an action has no effect on the operational state of the interface.

Instance	Value	Action
----------	-------	--------

f	1	The interface adapter with ifIndex f is enabled.
---	---	---

Note: It is possible that the *ifAdminStatus* value can be set to up or down, yet the actual operational change of the interface failed. In such a case, a get request of the *ifAdminStatus* might reflect *up* while an *ifOperStatus* for that interface might reflect *down*. If such a situation occurs, the network administrator would issue another set request to set *ifAdminStatus* to up to attempt the operational change again.

atPhysAddress

The hardware address portion of an address table binding on the agent host (an entry in the Address Resolution Protocol table). This is the same MIB variable as *ipNetToMediaPhysAddress*.

Instance	Value	Action
f.1.n.n.n.n	hh:hh:hh:hh:hh:hh	For the interface with ifIndex f , any existing ARP table binding for IP address n.n.n.n is replaced with the binding (n.n.n.n, hh:hh:hh:hh:hh:hh). If a binding did not exist, the new binding is added. hh:hh:hh:hh:hh:hh is a twelve-hexadecimal-digit hardware address.

atNetAddress

The IP address corresponding to the hardware or physical address specified in *atPhysAddress*. This is the same MIB variable as *ipNetToMediaNetAddress*.

Instance	Value	Action
f.1.n.n.n.n	m.m.m.m	For the interface with ifIndex f , an existing ARP table entry for IP address n.n.n.n is replaced with IP address m.m.m.m.

ipForwarding

Indicates whether this agent host is forwarding datagrams. See SNMP Daemon Implementation Restrictions for more information on this MIB variable.

Instance	Value	Action
0	1	If the agent host has more than one active interface, then the TCP/IP kernel is configured to forward packets. If the agent host has only one active interface, the set request fails.
	2	The TCP/IP kernel on the agent host is configured to not forward packets.

ipDefaultTTL

The default time-to-live (TTL) value inserted into IP headers of datagrams originated by the agent host.

Instance	Value	Action
0	n	The default time-to-live value used by IP protocol support is set to the integer n.

ipRouteDest

The destination IP address of a route in the route table.

Instance	Value	Action
n.n.n.n	m.m.m.m	The destination route for route n.n.n.n is set to the IP address m.m.m.m.

ipRouteNextHop

The gateway by which a destination IP address can be reached from the agent host (an entry in the route table).

Instance	Value	Action
n.n.n.n	m.m.m.m	A route table entry to reach network n.n.n.n using gateway m.m.m.m is added to the route table. The host portion of the IP address n.n.n.n must be 0 to indicate a network address.

ipRouteType

The state of a route table entry on the agent host (used to delete entries).

Instance	Value	Action
h.h.h.h	1	Any route to host IP address h.h.h.h is deleted.
n.n.n.n	2	Any route to host IP address n.n.n.n is deleted.

ipNetToMediaPhysAddress

The hardware address portion of an address table binding on the agent host (an entry in the ARP table). This is the same MIB variable as *atPhysAddress*.

Instance	Value	Action
f.1.n.n.n.n	hh:hh:hh:hh:hh:hh	For the interface with ifIndex f , any existing ARP table binding for IP address n.n.n.n is replaced with the binding (n.n.n.n, hh:hh:hh:hh:hh:hh). If a binding did not exist, the new binding is added. hh:hh:hh:hh:hh:hh is a 12-hexadecimal-digit hardware address.

ipNetToMediaNetAddress

The IP address corresponding to the hardware or physical address specified in *ipNetToMediaPhysAddress*. This is the same MIB variable as *atNetAddress*.

Instance	Value	Action
f.1.n.n.n.n	m.m.m.m	For the interface with ifIndex f , an existing ARP table entry for IP address n.n.n.n is replaced with IP address m.m.m.m.

ipNetToMediaType

The type of mapping from the IP address to the physical address.

Instance	Value	Action
----------	-------	--------

f.1.n.n.n.n	1	For the interface with ifIndex f , for an existing ARP binding from IP address to physical address, the mapping type is set to 1, or other.
	2	For the interface with ifIndex f , for an existing ARP binding from IP address to physical address, the mapping type is set to 2, or invalid. As a side effect, the corresponding entry in the ipNetMediaTable is invalidated; that is, the interface is disassociated from this ipNetToMediaTable entry.
	3	For the interface with ifIndex f , for an existing ARP binding from IP address to physical address, the mapping type is set to 3, or dynamic.
	4	For the interface with ifIndex f , for an existing ARP binding from IP address to physical address, the mapping type is set to 4, or static.

snmpEnableAuthenTraps

Indicates whether the **snmpd** agent is configured to generate *authenticationFailure* traps.

Instance	Value	Action
0	1	The snmpd agent will generate authentication failure traps.
	2	The snmpd agent will not generate authentication failure traps.

smuxPstatus

The status of an SMUX protocol peer (used to delete SMUX peers).

Instance	Value	Action
n	1	snmpd agent does nothing.
	2	snmpd agent stops communicating with SMUX peer n.

smuxTstatus

The status of a SMUX MIB tree (used to delete MIB tree mounts).

Instance	Value	Action
/m.m.m._ _ _ .p	1	snmpd agent does nothing.
	2	Unmounts SMUX mounting of MIB tree m.m.m... where <i>l</i> is the length of MIB tree instance and <i>p</i> is the smuxTpriority .

The following variables are the settable variables as defined in RFC 1229. The **snmpd** daemon allows the user to set these variables. The underlying device might not allow the setting of such variables. Check with each device to see what is and is not supported.

ifExtnsPromiscuous

The status of the promiscuous mode on a given device. This is used to enable and disable promiscuous mode on a given device. The **snmpd** action is final and complete. When **snmpd** is told to turn off, promiscuous mode is turned completely off regardless of the other applications on the machine.

Instance	Value	Action
n	1	Turns on the promiscuous mode for device n.
	2	Turns off the promiscuous mode for device n.

ifExtnsTestType

The test initiation variable. When this variable is set, the appropriate test is run for that device. An Object Identifier is the value of the variable. The specific value is dependent on the device type

and the test that is to be run. Currently, the only defined test that **snmpd** knows to run is the testFullDuplexLoopBack test.

Instance	Value	Action
n	oid	Start the test specified by oid.

ifExtnsRcvAddrStatus

The address status variable. When this variable is set, the specified address comes into existence with the appropriate level of duration. **snmpd** only allows the setting of temporary addresses because it is not able to set device Object Data Manager (ODM) records and it is only allowed to set multicast or broadcast addresses.

Instance	Value	Action
n.m.m.m.m.m.m	1	Add the address as something other than a temporary or permanent address.
	2	Remove the address from usage.
	3	Add the address as a temporary address.
	4	Add the address as a permanent address.

The variables listed below are the settable variables as defined in RFC 1231. The **snmpd** daemon allows the user to set these variables. The underlying device might not allow the setting of such variables. You should check with each device to see what is supported.

dot5Commands

The command the token-ring device is to perform.

Instance	Value	Action
n	1	Does nothing. Returned.
	2	Tells the token-ring device to open.
	3	Tells the token-ring to reset.
	4	Tells the token-ring device to close.

dot5RingSpeed

The current ring speed or bandwidth.

Instance	Value	Action
n	1	An unknown speed.
	2	1 megabit ring speed.
	3	4 megabit ring speed.
	4	16 megabit ring speed.

dot5ActMonParticipate

The object specifies whether the device participates in the active monitor selection process.

Instance	Value	Action
n	1	Participates.
	2	Not participate.

dot5Functional

The functional mask that allows the token-ring device to specify what addresses it receives frames from.

Instance	Value	Action
n	m.m.m.m.m.m	Functional mask to be set.

The following complex timer manipulations variables are defined in the RFC as read-only but you are encouraged to make them read-write. Review the RFC to gain a full understanding of their interactions. **snmpd** allows the requestor to set them, but the device might not. Check the device driver documentation for more information. The variables are:

- dot5TimerReturnRepeat
- dot5TimerHolding
- dot5TimerQueuePDU
- dot5TimerValidTransmit
- dot5TimerNoToken
- dot5TimerActiveMon
- dot5TimerStandbyMon
- dot5TimerErrorReport
- dot5TimerBeaconTransmit
- dot5TimerBeaconReceive.

The SNMP daemon allows the user to set the following variables. The daemon uses the FDDI Station Management (SMT) 7.2 protocol standard to get the information and is determined at the microcode level. Check the microcode on the FDDI documentation to ensure that the SMT 7.2 microcode is being used.

fddimibSMTUserData

A variable holding 32 bytes of user information.

Instance	Value	Action
n	string	Stores 32 bytes of user information.

fddimibSMTConfigPolicy

The status of the configuration policies, specifically the hold policy usage.

Instance	Value	Action
n	0	Do not use the hold policy.
	1	Use the hold policy.

fddimibSMTConnectionPolicy

The status of the connection policies in the FDDI node. See RFC 1512 for more information about the specific settable values.

Instance	Value	Action
n	k	Defines the connection policies.

fddimibSMTTNotify

The timer, expressed in seconds, used in the Neighbor Notification protocol. It has a range of 2 seconds to 30 seconds, and its default value is 30 seconds.

Instance	Value	Action
n	k	Defines the timer value.

fddimibSMTStatRptPolicy

The status of the status reporting frame generation.

Instance	Value	Action
n	1	Indicates that the node generates status reporting frames for implemented events.
	2	Indicates that the node does not create status reporting frames.

fddimibSMTTraceMaxExpiration

This variable defines the maximum timer expiration value for trace.

Instance	Value	Action
n	k	Defines the maximum timer expiration in milliseconds.

fddimibSMTStationAction

This variable causes the SMT entity to take a specific action. See the RFC to get specific information about this variable.

Instance	Value	Action
n	k	Defines an action on the SMT entity. Values range from 1 to 8.

fddimibMACRequestedPaths

Defines the paths the medium access control (MAC) should be inserted.

Instance	Value	Action
n.n	k	Defines the requested path for the MAC.

fddimibMACFrameErrorThreshold

Threshold for when a MAC status report is generated. Defines the number of error that must occur before a report is generated.

Instance	Value	Action
n.n	k	Defines the number of errors that must be observed before a MAC status report is generated.

fddimibMACMAUnitdataEnable

This variable determines the value of the **MA_UNITDATA_Enable** flag in RMT. The default and initial value of this flag is true (1).

Instance	Value	Action
n.n	1	Marks the MA_UNITDATA_Enable flag true.
	2	Marks the MA_UNITDATA_Enable flag false.

fddimibMACNotCopiedThreshold

A threshold for determining when a MAC condition report is generated.

Instance	Value	Action
----------	-------	--------

n.n	k	Defines the number of errors that must be observed before a MAC condition report is generated.
-----	---	--

The following three variables are timer variables that are interactive among themselves. Before changing any of these variables, you should have a good understanding of their meaning as defined in **RFC 1512**.

- fddimibPATHTVXLowerBound
- fddimibPATHHTMaxLowerBound
- fddimibPATHMaxTReq.

fddimibPORTConnectionPolicies

Specifies the connection policies for the specified port.

Instance	Value	Action
n.n	k	Defines the connection policies for the specified port.

fddimibPORTRequestedPaths

This variable is a list of permitted paths where each list element defines the port permitted paths. The first octet corresponds to 'none', the second octet to 'tree', and the third octet to 'peer'.

Instance	Value	Action
n.n	ccc	Defines the port paths.

fddimibPORTLerCutoff

The link error rate estimate at which a link connection is broken. It ranges from 10^{*-4} to 10^{*-15} and is reported as the absolute value of the base 10 logarithm (default of 7).

Instance	Value	Action
n.n	k	Defines the port LerCutoff.

fddimibPORTLerAlarm

The link error rate estimate at which a link connection generates an alarm. It ranges from 10^{*-4} to 10^{*-15} and is reported as the absolute value of the base 10 logarithm of the estimate (default is 8).

Instance	Value	Action
n.n	k	Defines the port LerAlarm.

fddimibPORTAction

This variable causes the port to take a specific action. See the RFC to get specific information about this variable.

Instance	Value	Action
n	k	Defines an action on the defined port. The values range from 1 to 6.

Note: RFC 1213 describes all variables in the *atEntry* and *ipNetToMediaEntry* tables as read-write. Set support is implemented only for the *atEntry* variables *atPhysAddress* and *atNetAddress*, and the *ipNetToMediaEntry* variables *ipNetToMediaPhysAddress*, *ipNetToMediaNetAddress*, and *ipNetToMediaType*. To accept set requests that might specify the remaining unsupported attributes in these two tables, set requests for the remaining variables are accepted in *atIfIndex* and

ipNetToMediaIfIndex. No error response is returned to the set request originator, but a subsequent get request will show that the original values are retained.

In the *ipRouteEntry* table, RFC 1213 describes all variables except *ipRouteProtoas* read-write. As mentioned above, set support is implemented only for the variables *ipRouteDest*, *ipRouteNextHop*, and *ipRouteType*. To accept set requests that might specify several unsupported route attributes, set requests for the remaining variables in the *ipRouteEntry* table are accepted: *ipRouteIfIndex*, *ipRouteMetric1*, *ipRouteMetric2*, *ipRouteMetric3*, *ipRouteMetric4*, *ipRouteMetric5*, *ipRouteAge*, and *ipRouteMask*. No error response is returned to the set request originator, but a subsequent get request will show that the original values are retained. The **snmpd** daemon does not coordinate routing with the **routed** daemon. If the **gated** daemon is running and has registered the *ipRouteTable* with the **snmpd** daemon, set requests to the *ipRouteTable* are not allowed.

RFC 1229 describes settable variables that **snmpd** allows. See the previous entries for actual deviations.

Examples

The following examples use the **snmpinfo** command. It is assumed that the **snmpinfo** default community name, **public**, has read-write access for the respective MIB subtree.

```
snmpinfo -m set sysContact.0="Primary contact: Bob Smith, office phone: 555-5555,  
beeper: 9-123-4567. Secondary contact: John Harris, phone: 555-1234."
```

This command sets the value of *sysContact.0* to the specified string. If an entry for *sysContact.0* already exists, it is replaced.

```
snmpinfo -m set sysName.0="bears.austin.ibm.com"
```

This command sets the value of *sysName.0* to the specified string. If an entry for *sysName.0* already exists, it is replaced.

```
snmpinfo -m set sysLocation.0="Austin site, building 802, lab 3C-23, southeast  
corner of the room."
```

This command sets the value of *sysLocation.0* to the specified string. If an entry for *sysLocation.0* already exists, it is replaced.

```
snmpinfo -m set ifAdminStatus.2=2
```

This command disables the network interface adapter which has the *ifIndex* of 2. If the assigned value is 1, the interface adapter is enabled.

```
snmpinfo -m set atPhysAddress.2.1.192.100.154.2=02:60:8c:2e:c2:00  
snmpinfo -m set ipNetToMediaPhysAddress.2.1.192.100.154.2=02:60:8c:2e:c2:00
```

These two commands change the hardware address in the ARP table entry for 192.100.154.2 to 02:60:8c:2e:c2:00. These two commands affect the same ARP table entry. The MIB variable *atPhysAddress* is a deprecated variable and is being replaced with the MIB variable *ipNetToMediaPhysAddress*. Thus, *atPhysAddress* and *ipNetToMediaPhysAddress* access the same structure in the TCP/IP kernel ARP table.

```
snmpinfo -m set atNetAddress.2.1.192.100.154.2=192.100.154.3  
snmpinfo -m set ipNetToMediaNetAddress.2.1.192.100.154.2=192.100.154.3
```

These commands change the IP address in the ARP table entry for 192.100.154.2 to 192.100.154.3. These two commands affect the same ARP table entry. The MIB variable *atNetAddress* is a deprecated variable and is being replaced with the MIB variable *ipNetToMediaNetAddress*. Thus, *atNetAddress* and *ipNetToMediaNetAddress* access the same structure in the TCP/IP kernel ARP table.

```
snmpinfo -m set ipForwarding.0=1
```

This command sets the TCP/IP kernel so that it can forward packets if the agent host has more than one interface that is up. If the host has only one active interface, then the set request fails and the **snmpd** agent returns the error, *badValue*.

```
snmpinfo -m set ipDefaultTTL=50
```

This command allows an IP datagram using default time-to-live (TTL) to pass through up to 50 gateways before being discarded. When each gateway processes a datagram, the gateway subtracts 1 from the time-to-live field. In addition, each gateway decrements the time-to-live field by the number of seconds the datagram waited for service at that gateway before passing the datagram on to the next destination.

```
snmpinfo -m set ipRouteDest.192.100.154.0=192.100.154.5
```

This command sets the destination IP address of the route associated with 192.100.154.0 to the IP address 192.100.154.5, assuming route 192.100.154 already existed.

```
snmpinfo -m set ipRouteNextHop.192.100.154.1=129.35.38.47
```

This command sets a route to host 192.100.154.1 using the gateway host 129.35.38.47, assuming route 192.100.154.1 already existed.

```
snmpinfo -m set ipRouteNextHop.192.100.154.0=192.100.154.7
```

This command sets a route to the class C network 192.100.154 using the gateway host 192.100.154.7, assuming route 192.100.154.0 already existed. Note that the host part of the address must be 0 to indicate a network address.

```
snmpinfo -m set ipRouteType.192.100.154.5=2
```

This command deletes any route to host 192.100.154.5.

```
snmpinfo -m set ipRouteDest.129.35.128.1=129.35.128.1
                ipRouteType.129.35.128.1=3
                ipRouteNextHop.129.35.128.1=129.35.128.90
```

This command creates a new route from host 129.35.128.90 to 129.35.128.1 as a gateway.

```
snmpinfo -m set ipNetToMediaType.2.1.192.100.154.11=4
```

This command sets the ARP table entry for 192.100.154.11 to static.

```
snmpinfo -m set snmpEnableAuthenTraps=2
```

This command causes the **snmpd** agent on the specified host to not generate *authenticationFailure* traps.

```
snmpinfo -m set smuxPstatus.1=2
```

This command invalidates the SMUX peer 1. The result is that the connection between the **snmpd** agent and this SMUX peer is terminated.

```
snmpinfo -m set smuxTstatus.8.1.3.6.1.2.1.4.21.0=2
```

This command invalidates or removes the mounting of the SMUX tree 1.3.6.1.2.1.4.21, the *ipRoute* Table. The first number in the instance indicates the number of levels in the SMUX tree identifier. The final number in the instance indicates the *smuxTpriority*. In this example, there are 8 levels in the SMUX tree identifier: 1.3.6.1.2.1.4.21. The priority, 0, is the highest priority.

```
snmpinfo -m set ifExtnsPromiscuous.1=1 ifExtnsPromiscuous.2=2
```

This command turns on promiscuous mode for the first device in the interfaces table and turns off promiscuous mode for the second device in the interfaces table.

```
snmpinfo -m set ifExtnsTestType.1=testFullDuplexLoopBack
```

This command starts the *testFullDuplexLoopBack* test on interface 1.

```
snmpinfo -m set ifExtnsRcvAddrStatus.1.129.35.128.1.3.2=2
```

This command tells interface 1 to remove physical address 129.35.128.1.3.2 from its list of acceptable addresses.

```
snmpinfo -m set dot5Commands.1=2
```

This command tells the first interface to do an open.

```
snmpinfo -m set dot5RingSpeed.1=2
```

This command tells the first interface to set its ring speed to 1 megabit.

```
snmpinfo -m set dot5ActMonParticipate.1=1
```

This command tells the first interface to participate in the active monitor selection process.

```
snmpinfo -m set dot5Functional.1=255.255.255.255.255
```

This command sets the functional address mask to allow everything.

```
snmpinfo -m set fddimibSMTUserData.1="Greg's Data"
```

This command sets the user data on the first SMT entity to "Greg's Data".

```
snmpinfo -m set fddimibMACFrameErrorThreshold.1.1=345
```

This command sets the threshold for frame errors to 345 on the first MAC of the first SMT entity.

Note: All of the variables described previously fall into one of the listed methods used to set the variable.

See Address Resolution Protocol and Internet Addresses for more information on protocols and Internet addresses.

SNMP Daemon RFC Conformance

RFC 1157 requires that each set request variable assignment "should be effected as if simultaneously set with respect to all other assignments specified in the same message" (on pages 25 and 26 of the RFC). This means that a set request with multiple instance ID/value pairs is processed in an *all-or-none* fashion. That is, either all the new values are assigned without error, or else none of the variables in the request have modified values. This requirement is also known as "atomic commit with rollback."

Note: RFC 1157 does not address the problems of order dependency or consistency. Ordering dependencies can exist, such as in:

```
snmpinfo -m set -h host1 ipNetToMediaPhysAddress.f.1.n.n.n=hh:hh:hh:hh:hh:hh\  
ifAdminStatus.f=1
```

```
snmpinfo -m set -h host1 iproutenextHop.n.n.n.n=m.m.m.m ifAdminStatus.f=1
```

The adapter with **ifIndex f** must be enabled before an address table entry can be associated with it or before a route can be established that reaches a gateway by using that adapter. In these cases, the order of the variables is significant and must be the reverse of that shown in the example. If the RFC 1157 atomic commit policy were followed exactly, ordered set requests would have undefined effects.

For the Management Information Base (MIB) variables supported, the **snmpd** agent prechecks the specified values for the MIB variables in the set request. If a specified value does not meet the precheck requirements, the set request is rejected. The actual implementation for the set request is technically a best effort, not a true atomic commit and rollback.

During the set process when the **snmpd** agent is actually modifying MIB variable values, if a failure occurs, the original values of the MIB variables already set are not restored.

RFC 1213 describes all variables in the `ipRouteEntry` table as read-write. As described previously, set support is only implemented for `ipRouteDest`, `ipRouteType`, and `ipRouteNextHop`. Set requests that may specify several unsupported route attributes (such as the `ipRouteMetric1` or the `ipRouteProto` attributes), are accepted. No error response is returned to the request originator, but a subsequent get request shows that the original values have been retained.

SNMP Daemon Implementation Restrictions

The current implementation of the Simple Network Management Protocol (SNMP) agent does *not* have:

- User interface for proxy agent support
- Nontrivial authentication support.

Not all values defined in RFC 1213 are supported, as illustrated in the following note:

Note: A set request for `ipRouteType 2` (invalid) causes deletion of a route. A get request never returns this value for an existing route.

Always observe the following implementation restrictions:

- An interface must be enabled by a locally issued `ifconfig` command before it can be enabled or disabled by the `ifAdminStatus` variable. An Internet Protocol (IP) address cannot be associated with an interface except with a locally issued `ifconfig` command.
- The `snmpd` agent does not provide a way to alter the IP address associated with an interface adapter. This must be done with a locally issued `ifconfig` command.
- The maximum message size that can be sent is 9KB (9216 bytes). The maximum message that can be received is 40KB (40960 bytes).

If you set the value of the MIB variable `ipForwarding` to 1, indicating that the TCP/IP kernel is to forward packets, the `snmpd` agent performs some interface checks before it allows the set operation to succeed. If more than one interface is active, the MIB variable `ipForwarding` is set to 1 and the Transmission Control Protocol/Internet Protocol (TCP/IP) kernel configurable option `ipforwarding` is set to 1. Otherwise, the `snmpd` agent does not perform the set operation and an error `badValue` is returned.

SNMP Daemon Logging Facility

Logging activities for the `snmpd` daemon can be handled in three ways:

- From the `snmpd` command line
- From the `snmpd` configuration file
- By the `syslogd` daemon.

The debug level of the `snmpd` logging messages is user-configurable. The debug level can be one of the following:

- 0 All NOTICES, EXCEPTIONS, and FATAL messages
- 1 Level 0 plus DEBUG messages
- 2 Level 1 plus a hexadecimal dump of the request and response packets
- 3 Level 2 plus an English version of the request and response packets.

The default debug level is 0. Level 3 messages are not written by the `syslogd` daemon.

If the `snmpd` daemon is configured for logging, logging can be toggled on and off by issuing a `SIGUSR1` signal, or a `kill -30` to the `snmpd` daemon. If the `snmpd` daemon is invoked under System Resource Controller (SRC) control, the SRC `traceson` and `tracesoff` commands can also be used to start and stop

the logging process. If logging is activated, the **tracesoff** command stops logging. Likewise, if logging is temporarily disabled, the **traceson** command can be used to restart the logging. If the **snmpd** daemon is not configured for logging, these commands have no effect. These commands have no effect on logging by the **syslogd** daemon.

Should the log file reach the file size limit, the **snmpd** daemon rotates the log file. The **snmpd** daemon keeps up to four levels of rotated log files. For example, if you name the log file LogFile, the files are rotated as follows:

- LogFile.3 is deleted.
- LogFile.2 is renamed LogFile.3.
- LogFile.1 is renamed LogFile.2.
- LogFile.0 is renamed LogFile.1.
- LogFile is renamed LogFile.0.
- Logging continues in LogFile.

If you are logging with **syslogd** at the same time, a log message is logged by the **syslogd** daemon stating that the log files are rotating. The **syslogd** log file is not rotated by the **snmpd** daemon.

If the **snmpd** daemon is invoked under SRC control, the **lssrc** command with the long status option lists the **snmpd** logging parameters. The **lssrc** command does not display information about the **syslogd** logging activities.

Logging Directed from the snmpd Command Line

To direct logging from the **snmpd** command line, the **-f** flag must be specified at the **snmpd** invocation. If the **-f** flag is not specified, logging cannot be directed from the command line. If the **-f** flag is specified, the full path name and file name of the logging file must be specified as the **snmpd** daemon forks and changes to the root directory at startup. If the **snmpd** daemon cannot open the file, the **snmpd** daemon directs logging from the configuration file. If **syslogd** is also handling **snmpd** log messages, an EXCEPTIONS message is logged to the **syslogd** log file stating the reason why the **snmpd** log file could not be opened. If the log file specified with the **-f** flag is successfully opened, the log file cannot be changed during execution of the **snmpd** daemon.

The debug level is specified on the **snmpd** command line with the **-d** flag. If the **-d** flag is not specified, the debug level defaults to 0. The debug level cannot change during execution of the **snmpd** daemon.

The size of the log file is **unlimited**; that is, it will be the system maximum file size.

If logging is directed from the command line, logging entries in the configuration file are ignored, both at startup and during any refresh of the **snmpd** daemon.

See the **snmpd** command in in *AIX 5L Version 5.1 Commands Reference*.

Logging Directed from the Configuration File

To direct logging from the configuration file, the **-f** flag must not be specified on the **snmpd** command line. If the **-d** flag is specified on the **snmpd** command line, the debug level specified with the **-d** flag becomes the default debug level. If the **-d** flag is not specified on the **snmpd** command line, the default debug level is 0.

The logging parameters are specified in logging entries in the **snmpd** configuration file. The configurable logging parameters include: *log file name*, *maximum log file size*, *debug level*, and *enablement*. If the log file name is not specified, logging is not enabled. Because the **snmpd** daemon forks and changes to the

root directory at startup, the full path name of the log file must be specified. The default enablement is disabled. Thus, if the enablement parameter is not specified as enabled, logging to the log file does not take place.

The default file size is **unlimited**; that is, it defaults to the system maximum file size.

The **snmpd** configuration file is read at **snmpd** startup and upon a refresh. Thus, logging parameters do not need to be specified before the **snmpd** daemon is invoked. Because the file is reread when the **snmpd** daemon receives a refresh (if **snmpd** is started under SRC control) or a SIGHUP signal (**kill -1**), the logging parameters can be specified at any time. In addition, the logging parameters can be changed at any time during the running of the **snmpd** daemon.

See the **snmpd.conf** file for more information on how to configure the **snmpd** daemon for logging directed from the configuration file.

Logging by the syslogd Daemon

Logging by the **syslogd** daemon can take place alone or in conjunction with logging directed from either the **snmpd** command line or configuration file.

To configure the **syslogd** daemon to log messages for the **snmpd** daemon, you must be the root user. Edit the **/etc/syslog.conf** file and add an entry such as the following:

```
daemon.debug      /var/tmp/snmpd.syslog
```

The **/var/tmp/snmpd.syslog** file must exist before the **syslogd** daemon rereads the **/etc/syslog.conf** configuration file in order for the **syslogd** daemon to log the **snmpd** daemon log messages to this file. To create this file, issue the following command:

```
touch /var/tmp/snmpd.syslog
```

Then issue the following command to force the **syslogd** daemon to reread its configuration file:

```
refresh -s syslogd
```

Note that the **syslogd** daemon logs *all* daemon messages to this log file, not just the **snmpd** log messages.

If the **syslogd** daemon is configured to log messages from the daemon facility at the **syslogd** LOG_DEBUG severity level and higher, all messages at **snmpd** debug level 2 or lower from the **snmpd** daemon can be logged into a **syslogd** configured file. If level 3 is specified as the **snmpd** debug level, the **syslogd** daemon logs only **snmpd** level 2 messages.

If logging is to be handled solely by the **syslogd** daemon, first the **syslogd** daemon must be configured for logging as described previously. The **snmpd** daemon debug level must be specified on the command line with the **-d** option, or else in the configuration file in a logging entry in the *level=Value* field. If no debug level is specified, the default level of 0 is used. If logging is directed from the **snmpd** configuration file, the debug level can change during running of the **snmpd** daemon.

The **SIGUSR1** signal (**kill -30**) and the SRC **traceson** and **tracesoff** commands have no effect on logging by the **syslogd** daemon.

No logging takes place if the **snmpd** daemon or the **syslogd** daemon are not configured to log messages as directed by the **snmpd** command line or configuration file.

Problem Determination for the SNMP Daemon

If the **snmpd** agent is not behaving as expected, the following are some hints to help determine and correct the problem. It is strongly recommended that you start up the **snmpd** agent with some type of logging. If invoking the **snmpd** daemon causes problems, it is strongly recommended that the **syslogd** daemon be set up for logging at the daemon facility and DEBUG severity level. See SNMP Daemon Logging Facility , the **snmpd** command, and the **snmpd.conf** file for more information on **snmpd** logging.

Daemon Termination Problem

If the **snmpd** daemon terminates as soon as it is invoked, the following are possible reasons for failure and probable solutions:

- The reason the **snmpd** daemon terminated will be logged in the **snmpd** log file or the configured **syslogd** log file. Check the log file to see the **FATAL** error message.
Solution: Correct the problem and restart the **snmpd** daemon.
- The **snmpd** command line usage was incorrect. If the **snmpd** command was invoked without the System Resource Controller (SRC), the required usage statement is echoed to the screen. If the **snmpd** daemon was invoked under SRC control, the usage message is not echoed to the screen. Check the log file to see the usage message.
Solution: Invoke the **snmpd** command with the correct usage statement.
- The **snmpd** daemon must be invoked by the root user.
Solution: Switch to the root user and restart the **snmpd** daemon.
- The **snmpd.conf** file must be owned by the root user. The **snmpd** agent verifies the ownership of the configuration file. If the file is not owned by the root user, the **snmpd** agent terminates with a fatal error.
Solution: Make sure you are the root user, change the ownership of the configuration file to the root user, and restart the **snmpd** daemon.
- The **snmpd.conf** file must exist. If the **-c** flag is not specified in the configuration file on the **snmpd** command line, the **/etc/snmpd.conf** file does not exist. If the **/etc/snmpd.conf** file is accidentally removed, reinstall the **bos.net.tcp.client** image or else reconstruct the file with the appropriate configuration entries as defined in the **snmpd.conf** file management page. If the configuration file is specified with the **-c** flag on the **snmpd** command line, make sure that the file exists and that the file is owned by the root user. The full path and file name of the configuration file must be specified or else the default **/etc/snmpd.conf** file will be used.
Solution: Make sure the specified configuration file exists and that this file is owned by the root user. Restart the **snmpd** daemon.
- The **udp port 161** is already bound. Make sure that the **snmpd** daemon is not already running. Issue the **ps -eaf | grep snmpd** command to determine if an **snmpd** daemon process is already executing. Only one **snmpd** agent can bind to **udp port 161**.
Solution: Either kill the existing **snmpd** agent or do not try to start up another **snmpd** daemon process.

Daemon Failure Problem

If the **snmpd** daemon fails when you issue a **refresh** or a **kill -1** signal, the following are possible reasons for failure and probable solutions:

- The reason the **snmpd** daemon terminated is logged in the **snmpd** log file or the configured **syslogd** log file. Check the log file to see the **FATAL** error message.
Solution: Correct the problem and restart the **snmpd** daemon.
- Make sure that the complete path and file name of the configuration file is specified when the **snmpd** daemon is invoked. The **snmpd** daemon forks and changes to the root directory at invocation. If the complete path name of the configuration file is not specified, the **snmpd** agent cannot find the file on a refresh. This is a fatal error and will cause the **snmpd** agent to terminate.

Solution: Specify the complete path and file name of the **snmpd** configuration file. Make sure the configuration file is owned by the root user. Restart the **snmpd** daemon.

- Make sure that the **snmpd** configuration file still exists. The file may have been accidentally removed after the **snmpd** agent was invoked. If the **snmpd** agent cannot open the configuration file, the **snmpd** agent terminates.

Solution: Recreate the **snmpd** configuration file, make sure the configuration file is owned by the root user, and restart the **snmpd** daemon.

MIB Variable Access Problem

If Management Information Base (MIB) variables cannot be accessed from the **snmpd** agent; if the **snmpd** agent is running, but the Simple Network Management Protocol (SNMP) manager application times out waiting for a response from the **snmpd** agent, try the following:

- Check the network configuration of the host on which the **snmpd** agent is running using the **netstat -in** command. Verify the lo0, loopback, device is up. If the device is down, an * (asterisk) displays to the left of the lo0. The lo0 must be up for the **snmpd** agent to service requests.

Solution: Issue the following command to start up the loopback interface:

```
ifconfig lo0 inet up
```

- Verify that the **snmpd** daemon has a route to the host where the requests are issued.

Solution: On the host where the **snmpd** daemon is running, add a route to the host where the **route add** command is issued. See the **route** command for more information.

- Check to see that the host name and the host IP address are the same value.

Solution: Reset the hostname to correspond to the host IP address.

- Check to see that *localhost* is defined to be the lo0 IP address.

Solution: Define *localhost* to be the same address used by the lo0 IP address (usually 127.0.0.1).

MIB Variable Access in Community Entry Problem

If a community entry is specified in the configuration file with a MIB view name, but MIB variables cannot be accessed, check the following:

- Make sure that you have correctly specified the community entry. If you have specified a view name in the community entry, all fields in the community are absolutely required.

Solution: Specify all fields in the community entry in the configuration file. Refresh the **snmpd** agent and try your request again.

- Make sure the access mode in the community entry corresponds with your request type. If you are issuing a **get** or **get-next** request, make sure that the community has read-only or read-write permission. If you are issuing a **set** request, make sure that the community has read-write permission.

Solution: Specify the correct access mode in the community entry. Refresh the **snmpd** agent and try your request again.

- Make sure that a view entry for the specified view name is specified in the community entry in the configuration file. If there is a specified view name in the community entry, but there is no corresponding view entry, the **snmpd** agent does not allow access for that community. A view entry is absolutely required for a view name specified in a community entry in the configuration file.

Solution: Specify a view entry for the view name specified in the community entry. Refresh the **snmpd** agent and try your request again.

- If *iso* is specified as the MIB subtree for the view entry, verify that *iso.3* is specified. The instance of 3 is required for the **snmpd** agent to access the *org* portion of the *iso* tree.

Solution: Specify the MIB subtree as *iso.3* in the view entry. Refresh the **snmpd** agent and try your request again.

- Check the *IP address* and *network mask* in the community entry. Verify that the host issuing the SNMP request is included in the community being specified with the community name.

Solution: Change the *IP address* and *network mask* fields in the community entry in the configuration file to include the host that is issuing the SNMP request.

No Response from Agent Problem

If the *IP address* in the community is specified as 0.0.0.0, but there is no response from the **snmpd** agent, try the following:

- Check the *network mask* field in the community entry. For general access to this community name, the *network mask* must be **0.0.0.0**. If the *network mask* is specified to be **255.255.255.255**, the **snmpd** agent is configured to not allow any requests with the specified community name.

Solution: Specify the *network mask* in the community entry to 0.0.0.0. Refresh the **snmpd** agent and try the request again.

- Make sure the access mode in the community entry corresponds with the request type. When issuing a **get** or **get-next** request, make sure that the community has read-only or read-write permission. If you are issuing a **set** request, make sure that the community has read-write permission.

Solution: specify the correct access mode in the community entry. Refresh the **snmpd** agent and try your request again.

noSuchName Problem

If, in attempting to set an MIB variable that the **snmpd** agent is supposed to support, a `noSuchName` error message is returned, the following might be the reason:

The set request issued did not include a community name for a valid community with write access. The SNMP protocol dictates that a set request with a community with inappropriate access privileges be answered with the `noSuchName` error message.

Solution: Issue the set request with a community name for a community that has write privileges and includes the host from which the set request is issued.

Chapter 10. Network File System

This chapter provides information on the Network File System (NFS), a mechanism for storing files on a network. The following topics are discussed:

- Network File System Overview
- NFS Installation and Configuration
- PC-NFS
- WebNFS
- Network Lock Manager
- Secure NFS
- NFS Problem Determination
- NFS Reference

Network File System Overview

The Network File System (NFS) is a distributed file system that allows users to access files and directories located on remote computers and treat those files and directories as if they were local. For example, users can use operating system commands to create, remove, read, write, and set file attributes for remote files and directories.

The NFS software package includes commands and daemons for NFS, Network Information Service (NIS), and other services. Although NFS and NIS are installed together as one package, each is independent and each is configured and administered individually. See *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide* for details on NIS and NIS+.

This operating system supports the latest NFS protocol update, NFS Version 3, and provides an NFS Version 2 client and server. The operating system is, therefore, backward compatible with an existing install base of NFS clients and servers.

The following topics are discussed in this section:

- NFS Services
- NFS Access Control Lists (ACL) Support
- Cache File System Support
- NFS Mapped File Support
- Three Types of Mounts
- NFS Mounting Process
- The `/etc/exports` File
- The `/etc/xtab` File
- Implementation of NFS
- Controlling NFS.

NFS Services

NFS provides its services through a client-server relationship. The computers that make their *file systems*, or *directories*, and other resources available for remote access are called *servers*. The act of making file systems available is called *exporting*. The computers and their processes that use server resources are considered *clients*. Once a client *mounts* a file system that a server exports, the client can access the individual server files (access to exported directories can be restricted to specific clients).

The major services provided by NFS are:

Mount service	From the <code>/usr/sbin/rpc.mountd</code> daemon on the server and the <code>/usr/sbin/mount</code> command on the client.
Remote File access	From the <code>/usr/sbin/nfsd</code> daemon on the server and the <code>/usr/sbin/biod</code> daemon on the client.
Remote execution service	From the <code>/usr/sbin/rpc.rexd</code> daemon on the server and the <code>/usr/bin/on</code> command on the client.
Remote System Statistics service	From the <code>/usr/sbin/rpc.rstatd</code> daemon on the server and the <code>/usr/bin/rup</code> command on the client.
Remote User Listing service	From the <code>/usr/lib/netsvc/rusers/rpc.rusersd</code> daemon on the server and the <code>/usr/bin/rusers</code> command on the client.
Boot Parameters service	Provides startup parameters to Sun Operating System diskless clients from the <code>/usr/sbin/rpc.bootparamd</code> daemon on the server.
Remote Wall service	From the <code>/usr/lib/netsvc/rwall/rpc.rwalld</code> daemon on the server and the <code>/usr/sbin/rwall</code> command on the client.
Spray service	Sends a one-way stream of Remote Procedure Call (RPC) packets from the <code>/usr/lib/netsvc/spray/rpc.sprayd</code> daemon on the server and the <code>/usr/sbin/spray</code> command on the client.
PC authentication service	Provides a user authentication service for PC-NFS from the <code>/usr/sbin/rpc.pcnfsd</code> daemon on the server.

Note: A computer can be both an NFS server and an NFS client simultaneously.

An NFS server is *stateless* meaning that an NFS server does not have to remember any transaction information about its clients. NFS transactions are atomic. Single NFS transaction corresponds to a single, complete file operation. NFS requires that the client remember any information needed for later NFS use.

NFS Access Control Lists (ACL) Support

Although NFS supports access control lists (ACLs), they are no longer used as the default. To use access control lists with NFS, use the `acl` option with the NFS `-o` flag, as shown in the following example:

```
mount -o acl
```

This support is handled by an RPC program that exchanges information about ACLs between clients and servers. The ACL support does not change the NFS protocol specification; it is a separate function.

The operating system adds ACLs to the regular file system. Since the normal NFS protocol does not support ACLs, they cannot be seen by normal NFS clients and unexpected behavior can result. A user on an NFS client might presume access to a file after looking at the permission bits, but the permissions could have been altered by the ACL associated with the file. Permissions on a server are enforced by the server according to the ACL on the server. Therefore, a user on the client machine could receive a permissions error.

When a client first attempts to access a remote mounted file system, it attempts to contact the ACL RPC program on the server.

If the server is a AIX 3.2 server, the client consults the ACL associated with a file before granting access to the program on the client. This provides the expected behavior on the client when the request is sent over to the server. In addition, the `aclget`, `aclput`, and `alcredit` commands can be used on the client to manipulate ACLs.

Cache File System (CacheFS) Support

The Cache File System (CacheFS) is a general-purpose file system caching mechanism that improves NFS server performance and scalability by reducing server and network load. Designed as a layered file system, CacheFS provides the ability to cache one file system on another. In an NFS environment, CacheFS increases the client-per-server ratio, reduces server and network loads and improves performance for clients on slow links, such as Point-to-Point Protocol (PPP).

A cache is created on the client machine so file systems specified to be mounted in the cache can be accessed locally instead of across the network. Files are placed in the cache when a user first requests access to them. The cache does not get filled until the user requests access to a file or files. Initial file requests may seem slow, but subsequent uses of the same files are faster.

Notes:

1. You cannot cache the / (root) or /usr file systems.
2. You can mount only file systems that are shared. (See the **exportfs** command.)
3. There is no performance gain in caching a local Journaled File System (JFS) disk file system.
4. You must have root or system authority to do the tasks in the following table.

CacheFS Tasks			
Task	SMIT Fast Path	Command or File	Web-based System Manager Management Environment.
Set up a cache	cachefs_admin_create	cfsadmin -c MountDirectoryName¹.	Software → File Systems → Cached File Systems → New Cached File System
Specifying Files for Mounting	cachefs_mount	mount -F cachefs -o backfstype=FileSysType, cachedir=CacheDirectory[,options] BackFileSystem MountDirectoryName² or edit /etc/filesystems.	Software → File Systems → Overview and Tasks → Mount a File System.
Modify the Cache	cachefs_admin_change	remove the cache, then recreate it using appropriate mount command options.	Software → File Systems → Cached File Systems → Selected → Delete. Proceed by setting up a cache as previously shown in the first row of this table.
Display Cache Information	cachefs_admin_change	cfsadmin -l MountDirectoryName.	Software → File Systems → Cached File Systems → Selected → Properties.
Remove a Cache	cachefs_admin_remove	<ol style="list-style-type: none"> 1. Unmount the file system: umount MountDirectoryName 2. Determine the cache ID: cfsadmin -l MountDirectoryName 3. Delete the file system: cfsadmin -d CacheID CacheDirectory 	Software → File Systems → Cached File Systems → Selected → Delete.

CacheFS Tasks			
Check File System Integrity	cachefs_admin_check	fsck_cachefs <i>CacheDirectory</i> ³ .	Software → File Systems → Cached File Systems → Selected → Check integrity of Cache.

Notes:

1. After you have created the cache, do not perform any operations within the cache directory (**cachedir**) itself. This causes conflicts within the CacheFS software.
2. If you use the **mount** command option to specify files for mounting, the command must be reissued each time the system is restarted.
3. Use the **-m** or **-o** options of the **fsck_cachefs** command to check the file systems without making any repairs.

NFS Mapped File Support

NFS mapped file support allows programs on a client to access a file as though it were in memory. By using the **shmat** subroutine, users can map areas of a file into their address space. As a program reads and writes into this region of memory, the file is read into memory from the server or updated as needed on the server.

Mapping files over NFS is limited in three ways:

- Files do not share information well between clients.
- Changes to a file on one client using a mapped file are not seen on another client.
- Locking and unlocking regions of a file is not an effective way to coordinate data between clients.

If an NFS file is to be used for data sharing between programs on different clients, use record locking and the regular **read** and **write** subroutines.

Multiple programs on the same client can share data effectively using a mapped file. Advisory record locking can coordinate updates to the file on the client, provided that the entire file is locked. Multiple clients can share data-using mapped files only if the data never changes, as in a static database.

Three Types of Mounts

There are three types of NFS mounts:

1. Predefined
2. Explicit
3. Automatic.

Predefined mounts are specified in the **/etc/filesystems** file. Each stanza (or entry) in this file defines the characteristics of a mount. Data such as the host name, remote path, local path, and any mount options are listed in this stanza. Predefined mounts are used when certain mounts are always required for proper operation of a client.

Explicit mounts serve the needs of the root user. Explicit mounts are usually done for short periods of time when there is a requirement for occasional unplanned mounts. Explicit mounts can also be used if a mount is required for special tasks and that mount is not generally available on the NFS client. These mounts are usually fully qualified on the command line by using the **mount** command with all needed

information. Explicit mounts do not require updating the `/etc/filesystems` file. File systems mounted explicitly remain mounted unless explicitly unmounted with the `umount` command or until the system is restarted.

Automatic mounts are controlled by the `automount` command, which causes the **AutoFS** kernel extension to monitor specified directories for activity. If a program or user attempts to access a directory that is not currently mounted, then **AutoFS** intercepts the request, arranges for the mount of the file system, then services the request.

NFS Mounting Process

Clients access files on the server by first mounting server exported directories. When a client mounts a directory, it does not make a copy of that directory. Rather, the mounting process uses a series of remote procedure calls to enable a client to transparently access the directories on the server. The following describes the mounting process:

1. When the server starts, the `/etc/rc.nfs` script runs the `exportfs` command, which reads the server `/etc/exports` file, and then tells the kernel which directories are to be exported and what access restrictions they require.
2. The `rpc.mountd` daemon and several `nfsd` daemons (8, by default) are then started by the `/etc/rc.nfs` script.
3. When the client starts, the `/etc/rc.nfs` script starts several `biode` daemons (8, by default), which forward client mount requests to the appropriate server.
4. Then the `/etc/rc.nfs` script executes the `mount` command, which reads the file systems listed in the `/etc/filesystems` file.
5. The `mount` command locates one or more servers that export the information the client wants and sets up communication between itself and that server. This process is called *binding*.
6. The `mount` command then requests that one or more servers allow the client to access the directories in the client `/etc/filesystems` file.
7. The server `rpc.mountd` daemon receives the client mount requests and either grants or denies them. If the requested directory is available to that client, the `rpc.mountd` daemon sends the client kernel an identifier called a *file handle*.
8. The client kernel then ties the file handle to the mount point (a directory) by recording certain information in a *mount record*.

Once the file system is mounted, the client can perform file operations. When the client does a file operation, the `biode` daemon sends the file handle to the server, where the file is read by one of the `nfsd` daemons to process the file request. Assuming the client has access to perform the requested file operation, the `nfsd` daemon returns the necessary information to the client `biode` daemon.

`/etc/exports` File

The `/etc/exports` file indicates all directories that a server exports to its clients. Each line in the file specifies a single directory. The server automatically exports the listed directories each time the NFS server is started. These exported directories can then be mounted by clients. The syntax of a line in the `/etc/exports` file is:

```
directory    -options[,option]
```

The `directory` is the full path name of the directory. Options can designate a simple flag such as `ro` or a list of host names. See the specific documentation of the `/etc/exports` file and the `exportfs` command for a complete list of options and their descriptions. The `/etc/rc.nfs` script does not start the `nfsd` daemons or the `rpc.mountd` daemon if the `/etc/exports` file does not exist.

The following example illustrates entries from an `/etc/exports` file:

```
/usr/games -ro,access=ballet:jazz:tap
/home     -root=ballet,access=ballet
/var/tmp
/usr/lib  -access=clients
```

The first entry in this example specifies that the `/usr/games` directory can be mounted by the systems named `ballet`, `jazz`, and `tap`. These systems can read data and run programs from the directory, but they cannot write in the directory.

The second entry in this example specifies that the `/home` directory can be mounted by the system `ballet` and that root access is allowed for the directory.

The third entry in this example specifies that any client can mount the `/var/tmp` directory. (Notice the absence of an access list.)

The fourth entry in this example specifies an access list designated by the netgroup `clients`. In other words, these machines designated as belonging to the netgroup `clients` can mount the `/usr/lib` directory from this server. (A *netgroup* is a network-wide group allowed access to certain network resources for security or organizational purposes. Netgroups are controlled by using NIS or NIS+. For more information, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.)

/etc/xtab File

The `/etc/xtab` file has a format identical to the `/etc/exports` file and lists the currently exported directories. Whenever the `exportfs` command is run, the `/etc/xtab` file changes. This allows you to export a directory temporarily without having to change the `/etc/exports` file. If the temporarily exported directory is unexported, the directory is removed from the `/etc/xtab` file.

Note: The `/etc/xtab` file is updated automatically, and is not to be edited.

Implementation of NFS

NFS is implemented on a wide variety of machine types, operating systems, and network architectures. NFS achieves this independence using the *Remote Procedure Call (RPC)* protocol.

Remote Procedure Call (RPC) Protocol

RPC is a library of procedures. The procedures allow one process (the client process) to direct another process (the server process) to run procedure calls as if the client process had run the calls in its own address space. Because the client and the server are two separate processes, they need not exist on the same physical system (although they can).

NFS is implemented as a set of RPC calls in which the server services certain types of calls made by the client. The client makes such calls based on the file system operations that are done by the client process. NFS, in this sense, is an RPC application.

Because the server and client processes can reside on two different physical systems which may have completely different architectures, RPC must address the possibility that the two systems might not represent data in the same way. For this reason, RPC uses data types defined by the eXternal Data Representation (XDR) protocol.

eXternal Data Representation (XDR) Protocol

XDR is the specification for a standard representation of various data types. By using a standard data type representation, a program can be confident that it is interpreting data correctly, even if the source of the data is a machine with a completely different architecture.

In practice, most programs do not use XDR internally. Rather, they use the data type representation specific to the architecture of the computer on which the program is running. When the program needs to communicate with another program, it converts its data into XDR format before sending the data. Conversely, when it receives data, it converts the data from XDR format into its own specific data type representation.

The portmap Daemon

Each RPC application has associated with it a program number and a version number. These numbers are used to communicate with a server application on a system. When making a request from a server, the client needs to know what port number the server is accepting requests on. This port number is associated with the User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) that is being used by the service. The client knows the program number, the version number, and the system name or host name where the service resides. The client needs a way to map the program number and version number pair to the port number of the server application. This is done with the help of the **portmap** daemon.

The **portmap** daemon runs on the same system as the NFS application. When the server starts running, it registers with the **portmap** daemon. As a function of this registration, the server supplies its program number, version number, and UDP or TCP port number. The **portmap** daemon keeps a table of server applications. When the client tries to make a request of the server, it first contacts the **portmap** daemon to find out which port the server is using. The **portmap** daemon responds to the client with the port of the server that the client is requesting. Upon receipt of the port number, the client is able to make all of its future requests directly to the server application.

Controlling NFS

The NFS, NIS, and NIS+ daemons are controlled by the System Resource Controller (SRC). This means you must use SRC commands such as **startsrc**, **stopsrc**, and **lssrc** to start, stop, and check the status of the NFS, NIS, and NIS+ daemons.

Some NFS daemons are not controlled by the SRC: specifically, **rpc.rexd**, **rpc.rusersd**, **rpc.rwalld**, and **rpc.rsprayed**. These daemons are started and stopped by the **inetd** daemon.

The following table lists the SRC-controlled daemons and their subsystem names.

Daemons and Their Subsystems		
<i>File Path</i>	<i>Subsystem Name</i>	<i>Group Name</i>
/usr/sbin/nfsd	nfsd	nfs
/usr/sbin/biod	biod	nfs
/usr/sbin/rpc.lockd	rpc.lockd	nfs
/usr/sbin/rpc.statd	rpc.statd	nfs
/usr/sbin/rpc.mountd	rpc.mountd	nfs
/usr/lib/netsvc/yp/ypserv	ypserv	yp
/usr/lib/netsvc/yp/ypbind	ypbind	yp
/usr/lib/netsvc/rpc.yppasswdd	yppasswdd	yp
/usr/lib/netsvc/rpc.yppupdated	ypupdated	yp
/usr/sbin/keyserv	keyserv	keyserv
/usr/sbin/portmap	portmap	portmap

NIS+ daemons are described in *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*. Each of these daemons can be specified to the SRC commands by using their subsystem name or the appropriate group name. These daemons support neither the long-listing facility of SRC nor the SRC trace commands.

For more information on using the SRC, see System Resource Controller Overview in *AIX 5L Version 5.1 System Management Concepts: Operating System and Devices*.

Change the Number of biod and nfsd Daemons

Use the **chnfs** command to change the number of **biod** or **nfsd** daemons running on the system. For example, to set the number of **nfsd** daemons to 10 and the number **biod** daemons to 4, run the command:

```
chnfs -n 10 -b 4
```

This command temporarily stops the daemons currently running on the system, modifies the SRC database code to reflect the new number, and restarts the daemons.

Note: In this implementation of NFS, the number of **biod** daemons are controllable only per mount point using the **biod -o** option. Specification using **chnfs** is retained for compatibility purposes only and has no real effect on the number of threads performing I/O.

Change Command Line Arguments for Daemons Controlled by SRC

Many NFS, NIS, and NIS+ daemons have command-line arguments that can be specified when the daemon is started. Since these daemons are not started directly from the command line, you must update the SRC database so that the daemons can be started correctly. To do this, use the **chssys** command. The **chssys** command has the format:

```
chssys -s Daemon -a 'NewParameter'
```

For example:

```
chssys -s nfsd -a '10'
```

changes the **nfsd** subsystem so that when the daemon is started, the command line looks like `nfsd 10`. The changes made by the **chssys** command do not take effect until the subsystem has been stopped and restarted.

Start the NFS Daemons at System Startup

The NFS daemons, by default, are not started during installation. When installed, all of the files are placed on the system, but the steps to activate NFS are not taken. You can start the NFS daemons at system startup through:

- The Web-based System Manager, **wsm**
- The SMIT fast path, **smit mknfs**
- The **mknfs** command.

All of these methods place an entry in the **inittab** file so that the **/etc/rc.nfs** script is run each time the system restarts. This script, in turn, starts all NFS daemons required for a particular system.

Start the NFS Daemons

The file size limit for files located on an NFS server is taken from the process environment when **nfsd** is started. To use a specific value, edit the **/etc/rc.nfs** file. Using the **ulimit** command with the desired limit before the **startsrc** command for **nfsd**.

The NFS daemons can be started individually or all at once. To start NFS daemons individually, run:

```
startsrc -s Daemon
```

where *Daemon* is anyone of the SRC-controlled daemons. For example, to start the **nfsd** daemons, run:

```
startsrc -s nfsd
```

To start all of the NFS daemons, run:

```
startsrc -g nfs
```

Note: If the **/etc/exports** file does not exist, the **nfsd** and the **rpc.mountd** daemons will not be started. You can create an empty **/etc/exports** file by running the command **touch /etc/exports**. This will allow the **nfsd** and the **rpc.mountd** daemons to start, although no file systems will be exported.

Stop the NFS Daemons

The NFS daemons can be stopped individually or all at once. To stop NFS daemons individually, run:

```
stopsrc -s Daemon
```

where *Daemon* is anyone of the SRC-controlled daemons. For example, to stop the **rpc.lockd** daemon, run:

```
stopsrc -s rpc.lockd
```

To stop all NFS daemons at once, run:

```
stopsrc -g nfs
```

Get the Current Status of the NFS Daemons

You can get the current status of the NFS daemons individually or all at once. To get the current status of the NFS daemons individually, run:

```
lssrc -s Daemon
```

where *Daemon* is any one of the SRC-controlled daemons. For example, to get the current status of the **rpc.lockd** daemon, run:

```
lssrc -s rpc.lockd
```

To get the current status of all NFS daemons at once, run:

```
lssrc -a
```

NFS Installation and Configuration

For information on installing the Network File System (NFS), see the *AIX 5L Version 5.1 Installation Guide*.

Checklist for Configuring NFS

Once the NFS software is installed on your systems, you are ready to configure NFS.

1. Determine which systems in the network are to be servers and which are to be clients (a system can be configured as both a server and a client).
2. For each system (whether client or server), follow the instructions in Start the NFS Daemons at System Startup.
3. For each NFS server, follow the instructions in Configuring an NFS Server.
4. For each NFS client, follow the instructions in Configuring an NFS Client.
5. If you want personal computers on your network to have access to your NFS servers (beyond being able to mount file systems), configure PC-NFS by following the instructions in PC-NFS.

Configuring an NFS Server

To configure an NFS server:

1. Start NFS using the instructions in Start the NFS Daemons Using SRC.
2. Create the **/etc/exports** file.

Configuring an NFS Client

1. Verify that NFS is the default remote file system. (If this is not done, specify the **-v nfs** flag when using the **mount** command.) Using a text editor, open the **/etc/vfs** file and search for the following entry:

```
#%defaultvfs jfs nfs
#nfs 2 /sbin/helpers/nfsmntheip none remote
```

Delete any lines that begin with a pound sign (#).

2. Start NFS using the instructions in Start the NFS Daemons.
3. Establish the local mount point using the **mkdir** command. For NFS to complete a mount successfully, a directory that acts as the mount point (or place holder) of an NFS mount must be present. This directory should be empty. This mount point can be created like any other directory, and no special attributes are needed.

Note:With one exception, the mount points for all NFS mounts must exist on your system before mounting a file system. If the **automount** daemon is used, it might not be necessary to create mount points. See the **automount** documentation for details.

4. Establish and mount the predefined mounts by following the instructions in Establishing Predefined NFS Mounts.

Exporting an NFS File System

You can export an NFS file system using the Web-based System Manager Network application, or you can use one of the following procedures.

- To export an NFS file system using the System Management Interface Tool (SMIT):
 1. Verify that NFS is already running by typing the command `lssrc -g nfs`. The output should indicate that the **nfsd** and the **rpc.mountd** daemons are active. If they are not, start NFS using the instructions in Start the NFS Daemons.
 2. At a command line, type the following and press **Enter**:

```
smit mknfsexp
```
 3. Specify appropriate values in the **PATHNAME** of directory to export, **MODE** to export directory, and **EXPORT** directory now, system restart or both fields.
 4. Specify any other optional characteristics you want, or accept the default values by leaving the remaining fields as they are.
 5. When you have finished making your changes, SMIT updates the **/etc/exports** file. If the **/etc/exports** file does not exist, it is created.
 6. Repeat steps 3 through 5 for each directory you want to export.
- To export an NFS file system using a text editor:
 1. Open the **/etc/exports** file with your favorite text editor.
 2. Create an entry for each directory to be exported using the full path name of the directory. List each directory to be exported starting in the left margin. No directory should include any other directory that is already exported. See the **/etc/exports** file documentation for a description of the full syntax for entries in the **/etc/exports** file.
 3. Save and close the **/etc/exports** file.
 4. If NFS is running, type the following command and press **Enter**:

```
/usr/sbin/exportfs -a
```

The **-a** option tells the **exportfs** command to send all information in the **/etc/exports** file to the kernel. If NFS is not running, start NFS using the instructions in Start the NFS Daemons.

- To temporarily export an NFS file system (without changing the **/etc/exports** file), type the following command and press **Enter**:

```
exportfs -i dirname
```

where *dirname* is the name of the file system you want to export. The **exportfs -i** command specifies that the **/etc/exports** file is not to be checked for the specified directory, and all options are taken directly from the command line.

Unexporting an NFS File System

You can unexport an NFS directory using the Web-based System Manager Network application, or you can use one of the following procedures.

- To unexport an NFS directory using SMIT:
 1. Type the following at a command prompt and press **Enter**:

```
smit rnmfsexp
```
 2. Enter the appropriate path name in the PATHNAME of exported directory to be removed field. The directory is now removed from the **/etc/exports** file and is unexported.
- To unexport an NFS directory by using a text editor:
 1. Open the **/etc/exports** file with your favorite text editor.
 2. Find the entry for the directory you wish to unexport, and delete that line.
 3. Save and close the **/etc/exports** file.
 4. If NFS is currently running, enter:

```
exportfs -u dirname
```

where *dirname* is the full path name of the directory you just deleted from the **/etc/exports** file.

Changing an Exported File System

Change an exported NFS file system using the Web-based System Manager Network application, or use one of the following procedures.

- To change an exported NFS file system using SMIT:
 1. To unexport the file system, **Enter**:

```
exportfs -u dirname
```
 - where *dirname* is the name of the file system you want to change.
 2. **Enter**:

```
smit chnfsexp
```
 3. Enter the appropriate path name in the PATHNAME of exported directory field.
 4. Make whatever changes you want.
 5. Exit SMIT.
 6. Re-export the file system by entering:

```
exportfs dirname
```

where *dirname* is the name of the file system you just changed.

- To change an exported NFS file system by using a text editor:
 1. To unexport the file system, **Enter**:


```
exportfs -u /dirname
```

where *dirname* is the name of the file system you want to change.

2. Open the **/etc/exports** file with your favorite text editor.
3. Make whatever changes you want.
4. Save and close the **/etc/exports** file.
5. Re-export the file system by entering:

```
exportfs /dirname
```

where *dirname* is the name of the file system you just changed.

Enabling Root User Access to an Exported File System

When a file system is exported, by default, the root user is not granted root access to that exported file systems. When a root user on one host requests access to a particular file from NFS, the user ID of the requester is mapped by NFS to the user ID of user *nobody* (*nobody* is one of the user names placed in the **/etc/passwd** file by default). The access rights of user *nobody* are the same as those given to the public (*others*) for a particular file. For example, if *others* only has run permission for a file, then user *nobody* can only run the file.

To enable root user access to an exported file system, follow the instructions in Changing an Exported File System. If you use the Web-based System Manager or SMIT method, specify in the HOSTS allowed root access field the name of the host to which you want to grant root access. If you edit the file with a text editor, add the qualifier `-root=hostname` to the file system entry. For example,

```
/usr/tps -root=hermes
```

specifies that the root user on host *hermes* may access the `/usr/tps` directory with root privileges.

Mounting an NFS File System Explicitly

To mount an NFS directory explicitly, use the Web-based System Manager, **wsm**, or use the following procedure:

1. Verify that the NFS server has exported the directory:

```
showmount -e ServerName
```

where *ServerName* is the name of the NFS server. This command displays the names of the directories currently exported from the NFS server. If the directory you want to mount is not listed, export the directory from the server.

2. Establish the local mount point using the **mkdir** command. A null (empty) directory that acts as the mount point (or place holder) of an NFS mount must be present for NFS to complete a mount successfully. This mount point can be created like any other directory, and no special attributes are needed.

3. Enter:

```
mount ServerName:/remote/directory /local/directory
```

where *ServerName* is the name of the NFS server, `/remote/directory` is the directory on the NFS server you want to mount, and `/local/directory` is the mount point on the NFS client.

4. On the client machine, enter the following SMIT fast path:

```
smit mknfsmnt
```

5. Make changes to the following fields that are appropriate for your network configuration. Your configuration might not require completing all of the entries on this screen.

Note: If the ASCII SMIT interface is being used, press the Tab key to change to the correct value for each field, but do *not* press Enter until completing step 7.

- PATHNAME of mount point.
 - PATHNAME of remote directory.
 - HOST where remote directory resides.
 - MOUNT now, add entry to /etc/filesystems or both?
 - /etc/filesystems entry will mount the directory on system RESTART.
 - MODE for this NFS file system.
6. Change or use the default values for the remaining entries, depending on your NFS configuration.
 7. When you finish making all the changes on this screen, SMIT mounts the NFS file system.
 8. When the **Command:** field shows the OK status, exit SMIT.

The NFS file system is now ready to use.

Using AutoFS to Automatically Mount a File System

AutoFS relies on the use of the **automount** command to propagate the automatic mount configuration information to the AutoFS kernel extension and start the **automountd** daemon. Through this configuration propagation, the extension automatically and transparently mounts file systems whenever a file or a directory within that file system is opened. The extension informs the **automountd** daemon of mount and unmount requests, and the **automountd** daemon actually performs the requested service.

Because the name-to-location binding is dynamic within the **automountd** daemon, updates to a Network Information Service (NIS) map used by the **automountd** daemon are transparent to the user. Also, there is no need to premount shared file systems for applications that have hard-coded references to files and directories, nor is there a need to maintain records of which hosts must be mounted for particular applications.

AutoFS allows file systems to be mounted as needed. With this method of mounting directories, all file systems do not need to be mounted all of the time; only those being used are mounted.

For example, to mount an NFS directory automatically:

1. Verify that the NFS server has exported the directory by entering:

```
showmount -e ServerName
```

where *ServerName* is the name of the NFS server. This command displays the names of the directories currently exported from the NFS server.

2. Create an **AutoFS** map file. **AutoFS** mounts and unmounts the directories specified in this map file. For example, suppose you want to use **AutoFS** to mount the /usr/local/dir1 and /usr/local/dir2 directories as needed from the serve1 server onto the /usr/remote/dir1 and /usr/remote/dir2 directories, respectively. In this example, the map file name is /tmp/mount.map.

```
dir1          -rw          serve1:/usr/local/dir1
dir2          -rw          serve1:/usr/local/dir2
```

3. Ensure that the **AutoFS** kernel extension is loaded and the **automountd** daemon is running. This can be accomplished in two ways:
 - a. Using **SRC**: Issue `lssrc -s automountd`. If the **automountd** subsystem is not running, issue `startsrc -s automountd`.
 - b. Using the **automount** command: Issue `/usr/bin/automount -v`.

Define the map file using the command line interface by entering:

```
/usr/sbin.automount /usr/remote /tmp/mount.map
```

where `/usr/remote` is the **AutoFS** mount point on the client. If a user runs the `cd /usr/remote/dir1` command, the **AutoFS** kernel extension intercepts access to the directory and issues a remote procedure call to the **automountd** daemon, which mounts the `/usr/remote/dir1` directory and then allows the `cd` command to complete.

```
/usr/sbin/automount /usr/remote /tmp/mount.map
```

where `/usr/remote` is the mount point on the NFS client. If a user runs the `cd /usr/remote/dir1` command, the **automount** daemon mounts the `/usr/remote/dir1` directory and then allows the `cd` command to complete.

4. To stop the **automount** daemon, issue the `stopsrc -s automountd` command.

If, for some reason, the **automountd** daemon was started without the use of **SRC**, issue:

```
kill automountd_PID
```

where `automountd_PID` is the process ID of the **automountd** daemon. (Running the `ps -e` command displays the process ID of the **automountd** daemon.) The `kill` command sends a SIGTERM signal to the **automountd** daemon.

Establishing Predefined NFS Mounts

You can establish predefined NFS mounts using the Web-based System Manager Network application, or you can use one of the following procedures.

Attention: Define the **bg** (background) and **intr** (interruptible) options in the `/etc/filesystems` file when establishing a predefined mount that is mounted during system startup. Mounts that are noninterruptible and running in the foreground can hang the client if the network or server is down when the client system starts up. If a client cannot access the network or server, the user must start the machine again in maintenance mode and edit the appropriate mount requests.

- To establish predefined mounts through SMIT:
 1. Enter:

```
smit mknfsmnt
```
 2. Specify values in this screen for each mount you want to predefine. Specify a value for each required field (those marked with an asterisk (*) in the left margin). Also specify values for the other fields or accept their default values. This method creates an entry in the `/etc/filesystems` file for the desired mount and attempts the mount.
- To establish the NFS default mounts by editing the `/etc/filesystems` file:
 1. Open the `/etc/filesystems` file with a text editor.
 2. Add entries for each of the remote file systems to be mounted when the system is started. For example:

```
/home/jdoe:  
dev = /home/jdoe  
mount = false  
vfs = nfs  
nodename = mach2  
options = ro,soft  
type = nfs_mount
```

This stanza directs the system to mount the `/home/jdoe` remote directory over the local mount point of the same name. The file system is mounted as read-only (`ro`). Because it is also mounted as `soft`, an error is returned in the event the server does not respond. By specifying the `type` parameter as `nfs_mount`, the system attempts to mount the `/home/jdoe` file (along with any other file systems that are specified in the `type = nfs_mount` group) when the `mount -t nfs_mount` command is issued.

The example stanza below directs the system to mount the **/usr/games** file system at system startup time. If the mount fails, the system continues to attempt to mount in the background.

```

/usr/games:
dev = /usr/games
mount = true
vfs = nfs
nodename = gameserver
options = ro,soft,bg
type = nfs_mount

```

The following parameters are required for stanzas pertaining to NFS mounts:

<code>dev=filesystem_name</code>	Specifies the path name of the remote file system being mounted.
<code>mount=[true false]</code>	If <i>true</i> the NFS file system is mounted when the system boots. If <i>false</i> , the NFS file system is not be mounted when the system boots.
<code>nodename=hostname</code>	Specifies the host machine on which the remote file system resides.
<code>vfs=nfs</code>	Specifies that the virtual file system being mounted is an NFS file system.

The following parameters are optional for stanzas pertaining to NFS mounts:

<code>type=type_name</code>	Defines the file system being mounted as part of the <i>type_name</i> mount group. This parameter is used with the mount -t command, which mounts groups of specified file systems at the same time.
<code>options=options</code>	<p>Specifies one or more of the following <i>options</i> parameters:</p> <p>biods=N Specifies the number of biod daemons to start. The default is 6. <i>N</i> is an integer.</p> <p>bg Specifies to try the mount again in the background if the first mount attempt fails.</p> <p>fg Specifies to try the mount again in the foreground if the first mount attempt fails.</p> <p>noacl Disables, for this mount only, the Access Control List (ACL) support provided by the NFS journaled file system.</p> <p>When used between two systems, NFS supports access control lists. If the <code>noacl</code> option is used when mounting a file system, NFS does not use ACLs. The effect of the <code>noacl</code> option equals what happens when an NFS client on a system mounts from an NFS server that does not support ACLs.</p> <p>For more information about ACLs, refer to NFS Access Control List (ACL) Support.</p> <p>retry=n Sets the number of times to try the mount.</p> <p>rsize=n Sets the read buffer size to the number of bytes specified by <i>n</i>.</p> <p>wsiz=n Sets the write buffer size to the number of bytes specified by <i>n</i>.</p>

	<p>timeo=<i>n</i> Sets the NFS time out to the tenths of a second specified by <i>n</i>. Use this variable to avoid situations that can occur in networks where the server load can cause inadequate response time.</p> <p>retrans=<i>n</i> Sets the number of NFS retransmissions to the number specified by <i>n</i>.</p> <p>port=<i>n</i> Sets the server port to the number specified by <i>n</i>.</p> <p>soft Returns an error if the server does not respond.</p> <p>hard Continues to try the request until the server responds.</p> <p style="padding-left: 40px;">Note: When you specify a hard mount, it is possible that the process can hang while waiting for a response. To be able to interrupt the process and end it from the keyboard, use the <code>intr</code> variable in the mount variables.</p> <p>intr Allows keyboard interrupts on hard mounts.</p> <p>ro Sets the read-only variable.</p> <p>rw Sets the read-write variable. Use the <code>hard</code> variable along with this variable to avoid error conditions that can conflict with applications if a <code>soft</code> mount is attempted as read-write. See NFS Problem Determination for information on hard- and soft-mounted problems.</p> <p>secure Specifies to use a more secure protocol for NFS transactions.</p> <p>actimeo=<i>n</i> Extends flush time by <i>n</i> seconds for both regular files and directories.</p> <p style="padding-left: 40px;">Note: The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be erased. If the file is modified before the flush time, then the flush time is extended by the time since the previous modification (under the assumption that recently changed files are likely to change again soon). There are minimum and maximum flush time extensions for regular files and for directories.</p> <p>acregmin=<i>n</i> Holds cached attributes for at least <i>n</i> seconds after file modification.</p> <p>acregmax=<i>n</i> Holds cached attributes for no more than <i>n</i> seconds after file modification.</p> <p>acdirmin=<i>n</i> Holds cached attributes for at least <i>n</i> seconds after directory update.</p> <p>acdirmax=<i>n</i> Holds cached attributes for no more than <i>n</i> seconds after directory update.</p>
--	---

	<p>Note: If you do not set the following options, the kernel automatically sets them to these default values:</p> <pre> biods=6 fg retry=10000 rsize=8192 wsize=8192 timeo=7 retrans=3 port=NFS_PORT hard secure=off acregmin=3 acregmax=60 acdirmin=30 acdirmax=60 </pre>
--	---

3. Remove any directory entries that you do not want to mount automatically at system startup.
4. Save and close the file.
5. Run the **mount -a** command to mount all the directories specified in the **/etc/filesystems** file.

Unmounting an Explicitly or Automatically Mounted File System

To unmount an explicitly or automatically mounted NFS directory, enter:

```
umount /directory/to/unmount
```

Removing Predefined NFS Mounts

You can remove a predefined NFS mount using the Web-based System Manager Network application, or you can use one of the following procedures.

- To remove a predefined NFS mount through SMIT:
 1. Enter:

```
smit rmnfsmnt
```
- To remove a predefined NFS mount by editing the **/etc/filesystems** file:
 1. Enter the command: `umount /directory/to/unmount`.
 2. Open the **/etc/filesystems** file with your favorite editor.
 3. Find the entry for the directory you just unmounted, and then delete it.
 4. Save and close the file.

PC-NFS

PC-NFS is a program for personal computers that enables the personal computer to mount file systems exported by a Network File System (NFS) server. The personal computer can also request network addresses and host names from the NFS server. Additionally, if the NFS server is running the **rpc.pcnfsd** daemon, the personal computer can access authentication and print-spooling services.

You might want to configure the **rpc.pcnfsd** daemon on the following:

- Systems that perform user authentication services
- Systems that offer print-spooling
- All Network Information Service (NIS) master and slave servers.

Note: Because NIS networks are typically configured so that PC-NFS can pick any NIS server as the default server, it is important that all servers have the **rpc.pcnfsd** daemon running. If running this daemon on all NIS servers is not practical, or if you want to limit requests to a specific server, add a **net pcnfsd** command to the **autoexec.bat** file on each personal computer to force it to use a specific NIS server. For more information, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

PC-NFS Authentication Service

By default, PC-NFS presents itself to NFS servers as the nobody user. With nobody privileges, all personal computer user files appear as owned by nobody, and consequently you cannot distinguish between different personal computer users. The authentication capability of the **rpc.pcnfsd** daemon allow you to monitor system resources and security by recognizing individual users and assigning them different privileges.

With the **rpc.pcnfsd** daemon running, a PC-NFS user can issue the **net name** command from a personal computer to log in to PC-NFS in the same manner as a user can log in to this operating system. The user name and password are verified by the **rpc.pcnfsd** daemon. This authentication procedure does not make a server more secure, but it does provide more control over access to files that are available through NFS.

PC-NFS Print-Spooling Service

The print-spooling service of the **rpc.pcnfsd** daemon enables personal computers running PC-NFS to print to printers not directly attached to the personal computer. Specifically, PC-NFS redirects files intended for personal computer printers to a file on an NFS server. This file is placed in a spool directory on the NFS server. The **rpc.pcnfsd** daemon then invokes the server printing facility. (The spooling directory must be in an exported file system so that PC-NFS clients can mount it.) When PC-NFS requests that the **rpc.pcnfsd** daemon print the file, it provides the following information:

- Name of the file to be printed
- Login ID of the user on the client
- Name of the printer to be used.

Configuring the rpc.pcnfsd Daemon

To configure the **rpc.pcnfsd** daemon:

1. Install PC-NFS program on your personal computer.
2. Select a location for the spool directory on the NFS server. The default spool directory is **/var/tmp**. The spool directory must have at least 100K bytes of free space.
3. Export the spool directory. Do not put access restrictions on the exported directory that could cause access problems in your network. For details of this procedure, see *Exporting an NFS File System*.
4. Start the **rpc.pcnfsd** daemon by following the instructions in *Starting the rpc.pcnfsd Daemon*.
5. Verify that the **rpc.pcnfsd** daemon is accessible by following the instructions in *Verifying the rpc.pcnfsd Daemon Is Accessible*.

Note: Because printer-redirection requests sometimes cause file listings of zero length to be left in the PC-NFS spool directories, periodically clear spooling directories of these entries.

Starting the rpc.pcnfsd Daemon

To start the **rpc.pcnfsd** daemon using the default spooling directory:

1. With a text editor, uncomment the following entry in the **/etc/inetd.conf** file:

```
pcnfsd sunrpc_udp udp wait root /usr/sbin/rpc.pcnfsd pcnfsd 150001 1
```

2. Save the file and exit the text editor.

To start the **rpc.pcnfsd** daemon using a directory that is different from the default:

1. Use a text editor to add the following entry to the **/etc/rc.nfs** file:

```
if [ -f /usr/sbin/rpc.pcnfsd ] ; then
/usr/sbin/rpc.pcnfsd -s spooldir ; echo ' rpc.pcnfsd\'
fi
```

where *spooldir* specifies the full path name of the spool directory.

2. Save the file and exit the text editor.
3. Using a text editor, comment the following entry in the **/etc/inetd.conf** file:

```
#pcnfsd sunrpc_udp udp wait root /usr/sbin/rpc.pcnfsd pcnfsd 150001 1
```

Placing a pound sign (#) at the beginning of the line prevents the **inetd** daemon from starting the **rpc.pcnfsd** daemon using the default spool directory.

4. Start the **rpc.pcnfsd** daemon print spooler by typing the following at the command line:

```
/usr/sbin/rpc.pcnfsd -s spooldir
```

where *spooldir* specifies the full path name of the spool directory.

For more information on updating the **inetd** configuration database, see [Configuring the inetd Daemon](#).

Note: The default directory that the **rpc.pcnfsd** daemon uses cannot be changed from the **inetd.conf** file.

Verifying the rpc.pcnfsd Daemon Is Accessible

To verify that the **rpc.pcnfsd** daemon is accessible, enter:

```
rpcinfo -u host 150001
```

where *host* specifies the host name of the system on which you are configuring **rpc.pcnfsd**, and 15001 is the RPC program number of the **rpc.pcnfsd** daemon. After you enter the command, you will receive a message that the program is ready and waiting.

WebNFS

The operating system provides NFS server capability for WebNFS. Defined by Sun Microsystems, WebNFS is a simple extension of the NFS protocol that allows easier access to servers and clients through Internet firewalls.

A WebNFS-enhanced web browser can use an NFS universal resource locator (URL) to access data directly from the server. An example NFS URL is:

```
nfs://www.YourCompany.com/
```

WebNFS works in tandem with existing web-based protocols to provide data to clients.

WebNFS also takes advantage of the scalability of NFS servers.

Network Lock Manager

The network lock manager is a facility that works in cooperation with the Network File System (NFS) to provide a System V style of advisory file and record locking over the network. The network lock manager (**rpc.lockd**) and the network status monitor (**rpc.statd**) are network-service daemons. The **rpc.statd** daemon is a user level process while the **rpc.lockd** daemon is implemented as a set of kernel threads (similar to the NFS server). Both daemons are essential to the ability of the kernel to provide fundamental network services.

Note: Mandatory or enforced locks are not supported over NFS.

Network Lock Manager Architecture

The network lock manager contains both server and client functions. The client functions are responsible for processing requests from the applications and sending requests to the network lock manager at the server. The server functions are responsible for accepting lock requests from clients and generating the appropriate locking calls at the server. The server will then respond to the locking request of the client.

In contrast to NFS, which is stateless, the network lock manager has an implicit state. In other words, the network lock manager must remember whether the client currently has a lock. The network status monitor, **rpc.statd**, implements a simple protocol that allows the network lock manager to monitor the status of other machines on the network. By having accurate status information, the network lock manager can maintain a consistent state within the stateless NFS environment.

Network File Locking Process

When an application wants to obtain a lock on a local file, it sends its request to the kernel using the **lockf**, **fcntl**, or **flock** subroutines. The kernel then processes the lock request. However, if an application on an NFS client makes a lock request for a remote file, the Network Lock Manager client generates a Remote Procedure Call (RPC) to the server to handle the request.

When the client receives an initial remote lock request, it registers interest in the server with the client's **rpc.statd** daemon. The same is true for the network lock manager at the server. On the initial request from a client, it registers interest in the client with the local network status monitor.

Crash Recovery Process

The **rpc.statd** daemon on each machine notifies the **rpc.statd** daemon on every other machine of its activities. When the **rpc.statd** daemon receives notice that another machine crashed or recovered, it notifies its **rpc.lockd** daemon.

If a server crashes, clients with locked files must be able to recover their locks. If a client crashes, its servers must hold the client locks while it recovers. Additionally, to preserve the overall transparency of NFS, the crash recovery must occur without requiring the intervention of the applications themselves.

The crash recovery procedure is simple. If the failure of a client is detected, the server releases the failed client locks on the assumption that the client application will request locks again as needed. If the crash and recovery of a server is detected, the client lock manager retransmits all lock requests previously granted by the server. This retransmitted information is used by the server to reconstruct its locking state during a grace period. (The grace period, 45 seconds by default, is a time period within which a server allows clients to reclaim their locks.)

The **rpc.statd** daemon uses the host names kept in **/etc/sm** and **/etc/sm.bak** to keep track of which hosts must be informed when the machine needs to recover operations.

Starting the Network Lock Manager

By default, the `/etc/rc.nfs` script starts the `rpc.lockd` and `rpc.statd` daemons along with the other NFS daemons. If NFS is already running, you can verify that the `rpc.lockd` and `rpc.statd` daemons are running by following the instructions in [Get the Current Status of the NFS Daemons](#). The status of these two daemons should be *active*. If the `rpc.lockd` and `rpc.statd` daemons are not active, and therefore not running, do the following:

1. Using your favorite text editor, open the `/etc/rc.nfs` file.
2. Search for the following lines:

```
if [ -x /usr/sbin/rpc.statd ]; then
    startsrc -s rpc.statd
fi
if [ -x /usr/sbin/rpc.lockd ]; then
    startsrc -s rpc.lockd
fi
```

3. If there is a pound sign (#) at the beginning of any of these lines, delete the character, then save and exit the file. Then start the `rpc.statd` and `rpc.lockd` daemons by following the instructions in [Start the NFS Daemons](#).

Note: Sequence is important. Always start the `statd` daemon first.

4. If NFS is running and the entries in the `/etc/rc.nfs` file are correct, stop and restart the `rpc.statd` and `rpc.lockd` daemons by following the instructions in [Stop the NFS Daemons and Start the NFS Daemons](#).

Note: Sequence is important. Always start the `statd` daemon first.

If the `rpc.statd` and `rpc.lockd` daemons are still not running, see [Troubleshooting the Network Lock Manager](#).

Troubleshooting the Network Lock Manager

If you receive a message on a client similar to:

```
clnttcp_create: RPC: Remote System error - Connection refused
rpc.statd:cannot talk to statd at {server}
```

then the machine thinks there is another machine which needs to be informed that it might have to take recovery measures. When a machine restarts, or when the `rpc.lockd` and the `rpc.statd` daemons are stopped and restarted, machine names are moved from `/etc/sm` to `/etc/sm.bak` and the `rpc.statd` daemon tries to inform each machine corresponding to each entry in `/etc/sm.bak` that recovery procedures are needed.

If the `rpc.statd` daemon can reach the machine, then its entry in `/etc/sm.bak` is removed. If the `rpc.statd` daemon cannot reach the machine, it will keep trying at regular intervals. Each time the machine fails to respond, the timeout generates the above message. In the interest of locking integrity, the daemon will continue to try; however, this can have an adverse effect on locking performance. The handling is different, depending on whether the target machine is just unresponsive or semi-permanently taken out of production. To eliminate the message:

1. Verify that the `statd` and `lockd` daemons on the server are running by following the instructions in [Get the Current Status of the NFS Daemons](#). (The status of these two daemons should be *active*.)
2. If these daemons are not running, start the `rpc.statd` and `rpc.lockd` daemons on the server by following the instructions in [Start the NFS Daemons](#).

Note: Sequence is important. Always start the `statd` daemon first.

After you have restarted the daemons, remember that there is a grace period. During this time, the **lockd** daemons allow reclaim requests to come from other clients that previously held locks with the server, so you might not get a new lock immediately after starting the daemons.

Alternatively, eliminate the message by:

1. Stop the **rpc.statd** and **rpc.lockd** daemons on the client by following the instructions in Stop the NFS Daemons.
2. On the client, remove the target machine entry from **/etc/sm.bak** file by entering:

```
rm /etc/sm.bak/TargetMachineName
```

This action keeps the target machine from being aware that it might need to participate in locking recovery. It should only be used when it can be determined that the machine does not have any applications running that are participating in network locking with the affected machine.

3. Start the **rpc.statd** and **rpc.lockd** daemons on the client by following the instructions in Start the NFS Daemons.

If you are unable to obtain a lock from a client, do the following:

1. Use the **ping** command to verify that the client and server can reach and recognize each other. If the machines are both running and the network is intact, check the host names listed in the **/etc/hosts** file for each machine. Host names must exactly match between server and client for machine recognition. If a name server is being used for host name resolution, make sure the host information is exactly the same as that in the **/etc/hosts** file.
2. Verify that the **rpc.lockd** and **rpc.statd** daemons are running on both the client and the server by following the instructions in Get the Current Status of the NFS Daemons. The status of these two daemons should be *active*.
3. If they are not active, start the **rpc.statd** and **rpc.lockd** daemons by following the instructions in Start the NFS Daemons.
4. If they are active, you might need to reset them on both clients and servers. To do this, stop all the applications that are requesting locks.
5. Next, stop the **rpc.statd** and **rpc.lockd** daemons on both the client and the server by following the instructions in Stop the NFS Daemons.
6. Now, restart the **rpc.statd** and **rpc.lockd** daemons, first on the server and then on the client, by following the instructions in Start the NFS Daemons.

Note: Sequence is important. Always start the **statd** daemon first.

If the procedure does not alleviate the locking problem, run the **lockd** daemon in debug mode, by doing the following:

1. Stop the **rpc.statd** and **rpc.lockd** daemons on both the client and the server by following the instructions in Stop the NFS Daemons.
2. Start the **rpc.statd** daemon on the client and server by following the instructions in Start the NFS Daemons.
3. Start the **rpc.lockd** daemon on the client and server by typing:

```
/usr/sbin/rpc.lockd -d1
```

When invoked with the **-d1** flag, the **lockd** daemon provides diagnostic messages to syslog. At first, there will be a number of messages dealing with the grace period; wait for them to time out. After the grace period has timed out on both the server and any clients, run the application that is having lock problems and verify that a lock request is transmitted from client to server and server to client.

Secure NFS

In addition to the standard UNIX authentication system, the Network Information Services NIS and NIS+ and the Network File System (NFS) provide a means to authenticate users and machines in networks on a message-by-message basis. This additional authentication system uses Data Encryption Standard (DES) encryption and public key cryptography.

This section discusses the following topics:

- Secrecy
- Secrecy in NFS
- Naming Network Entities for DES Authentication
- The `/etc/publickey` File
- Booting Considerations of Public Key Systems
- Performance Considerations
- Checklist for Administering Secure NFS
- How to Configure Secure NFS
- How to Export a File System Using Secure NFS
- How to Mount a File System Using Secure NFS.

Secrecy

Throughout history, various groups of people have sought to communicate in such a way that only the sender and receiver know the contents of a given message. To achieve this secrecy, senders and receivers use a *cipher*, a scheme for converting a *plaintext* message into *ciphertext* and back again. *Encryption* is the process of converting plain text into cipher text, and *decryption* is the process of converting cipher text into plain text.

One of the earliest ciphers, the *Caesar cipher*, is attributed to Julius Caesar. In this cipher, one letter is substituted for another. For example, 'A' becomes 'C', 'B' becomes 'D', ..., 'Y' becomes 'A', and 'Z' becomes 'B'. So, the Caesar cipher encrypts the phrase **ATTACK AT DAWN** as **CVVCEM CV FCYP**.

If the Carthaginians could *cryptanalyze* the Caesar cipher and break it, the Roman *cryptographers* would have to invent an entirely new cipher. Since cipher development is an expensive process, the Romans might use a cipher *key* in order to get more use out of their cipher. For example, instead of specifying a letter-for-letter substitute, the Romans might specify a key *K*, where *K* indicates the number of positions to shift a letter. That is, if $K = 2$, then 'A' becomes 'C'. If $K = 4$, then 'A' becomes 'E', and so on. With this scheme, if the Carthaginians break the cipher, all the Romans must do is change the key. Of course, the Carthaginians might realize what algorithm the Italians were using, and exhaustively try every value of *K* from 1 to 26. If the Carthaginians had a computer, their task would be a trivial programming exercise.

Data Encryption Standard

Modern ciphers are designed to address the fact that computers can be powerful tools for an intruder attempting to break a cipher. In 1977, the U.S. government adopted a cipher as its Data Encryption Standard. This cipher is widely used in industry. DES is a highly complex algorithm which converts 64-bit blocks of plain text into 64-bit blocks of cipher text using a 56-bit key. Because of the complexity of the algorithm and the size of the cipher key, DES is essentially unbreakable. For example, if an intruder had a computer which could compute the DES algorithm at a rate of one key per microsecond, the computer would need over two thousand years to try every possible key.

Public Key Cryptography

A significant weakness in any encryption algorithm is the key that is used. If the sender and receiver are to communicate securely using a cipher, both the sender and receiver must know the key. They must agree upon a key either using a separate communications link, which itself must be secure, or in person.

To address this problem, two researchers (Diffie and Hellman) developed a technique by which the sender and receiver can exchange keys publicly without compromising the security of their communication. Their technique has three requirements:

- $\text{Decipher}(\text{Encipher}(\text{plaintext}, E), D) = \text{plaintext}$
where E is the encryption key (known to the public), and D is the decryption key (known only by the receiver).
That is, Encipher and Decipher functions are inverses of each other. Therefore, if you take the encrypted text string returned by $\text{Encipher}(\text{plaintext}, E)$, and use it along with the key D for the Decipher function, Decipher returns the original plain text.
- An intruder cannot deduce Decipher() from Encipher().
- Encipher() is unbreakable.

The following outline describes how a sender sends a secret message to a receiver.

1. The sender obtains the receiver public encryption key.
2. The sender converts the plain text message into cipher text by computing the result:
 $\text{ciphertext} = \text{Encipher}(\text{plaintext}, E)$
3. The sender sends the cipher text message to the receiver.
4. The receiver receives the cipher text message and converts it into plaintext by computing the result:
 $\text{plaintext} = \text{Decipher}(\text{ciphertext}, D)$

Even if an intruder intercepts the message, the intruder will not be able to decipher it because the intruder does not have the decipher key. (For that matter, the sender cannot decipher the cipher text message either.)

Authentication

A primary application of secrecy is *authentication*. One common method of authentication (which is the standard UNIX method of authentication) uses a password. When a user wants to log in, the operating system requires the user to provide a password known only by the operating system and the user. If the user provides the correct password, the operating system concludes that the user is who the user claims to be. Note that this method requires the operating system to store the user passwords in a file on the system, although in encrypted form. This means that two different entities know a single password.

Public key cryptography provides an alternative to password authentication. Suppose a sender wants to send a message, and the receiver wants to be certain that the message is from the sender and not from an intruder pretending to be the sender. The authentication process will occur in the following manner:

1. First, the sender enciphers a "request to send" message using the public key of the receiver, and then sends the request.
2. The receiver receives the "request to send" message and deciphers it using the receiver private key.
3. The receiver enciphers a "token" message using the public key of the sender, and then sends the token.
4. The sender receives the token and deciphers it with the sender private key. When the sender sends messages to the receiver, the sender will begin each message with the token, signifying that the sender is, in fact, the sender. If an intruder attempts to send messages in the name of the sender, the receiver will reject them because the intruder does not know what the token is.

Note that, unlike password authentication, the receiver is able to authenticate the sender without knowing the sender private key. For more information on authentication systems, see Understanding RPC Authentication in *AIX 5L Version 5.1 Communications Programming Concepts*.

Secrecy in NFS

NFS, NIS, and NIS+ use the DES algorithm for different purposes. NFS uses DES to encrypt a time stamp in the Remote Procedure Call (RPC) messages sent between NFS servers and clients. This encrypted time stamp authenticates machines just as the "token" authenticates the sender, as described in Authentication. (For information about NIS and NIS+ see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.)

Because NFS can authenticate every single RPC message exchanged between NFS clients and servers, this provides an additional, optional, level of security for each file system. By default, file systems are exported with the standard UNIX authentication. To take advantage of this additional level of security, you can specify the **secure** option when you export a file system.

Public Key Cryptography for Secure NFS

Both the public key and the secret key of the user are stored and indexed by net name in the **publickey.byname** map. The secret key is DES-encrypted with the user login password. The **keylogin** command uses the encrypted secret key, decrypts it with the login password, then gives it to a secure local key server to save for use in future RPC transactions. Users are not aware of their public and secret keys because the **yppasswd** command, in addition to changing the login password, generates the public and secret keys automatically.

The **keyserv** daemon is an RPC service that runs on each NIS and NIS+ machine. For information on how NIS+ uses **keyserv**, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*. Within NIS, **keyserv** executes the following three public key subroutines:

- **key_setsecret** subroutine
- **key_encryptsession** subroutine
- **key_decryptsession** subroutine.

The **key_setsecret** subroutine tells the key server to store the secret key of the user (SK_A) for future use; it is normally called by the **keylogin** command. The client program calls the **key_encryptsession** subroutine to generate the encrypted conversation key, which is passed in the first RPC transaction to a server. The key server looks up the server public key and combines it with the secret key of the client (set up by a previous **key_setsecret** subroutine) to generate the common key. The server asks the key server to decrypt the conversation key by calling the **key_decryptsession** subroutine.

Implicit in these subroutine calls is the name of the caller, which must be authenticated in some manner. The key server cannot use DES authentication to do this, because it would create a deadlock. The key server solves this problem by storing the secret keys by the user ID (UID) and only granting requests to local root processes. The client process then executes a root user owned **setuid** subroutine which makes the request on the part of the client, telling the key server the real UID of the client.

Authentication Requirements

Secure NFS authentication is based on the ability of a sender to encrypt the current time, which the receiver can then decrypt and check against its own clock. This process has two requirements:

- The two agents must agree on the current time.
- The sender and receiver must be using the same DES encryption key.

Agreeing on the Current Time: If the network uses time synchronization, then the **timed** daemon keeps the client and server clocks synchronized. If not, the client computes the proper time stamps based on the server clock. To do this, the client determines the server time before starting the RPC session, and then

computes the time difference between its own clock and that of the server. The client then adjusts its time stamp accordingly. If, during the course of an RPC session, the client and server clocks become unsynchronized to the point where the server begins rejecting the client requests, the client will redetermine the server time.

Using the Same DES Key: The client and server compute the same DES encryption key by using public key cryptography. For any client A and server B, there is a key that only A and B can deduce. This key is called the *common key*. The client derives the common key by computing the following formula:

$$K_{AB} = PK_B^{SK_A}$$

where *K* stands for the *common Key*, *PK* stands for *Public Key*, and *SK* stands for *Secret Key*, and each of these keys is a 128-bit number. The server derives the same common key by computing the following formula:

$$K_{AB} = PK_A^{SK_B}$$

Only the server and client can calculate this common key since doing so requires knowing one secret key or the other. Because the common key has 128 bits, and DES uses a 56-bit key, the client and server extract 56 bits from the common key to form the DES key.

Authentication Process

When a client wants to talk to a server, it randomly generates a key used for encrypting the time stamps. This key is known as the *conversation key (CK)*. The client encrypts the conversation key using the DES common key (described in Authentication Requirements) and sends it to the server in the first RPC transaction. This process is illustrated in the following figure.

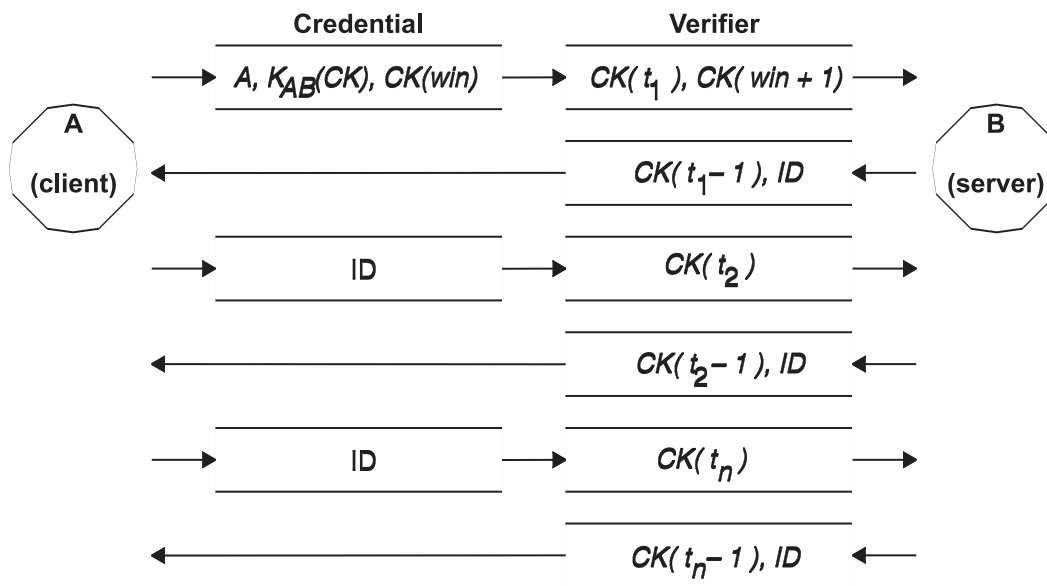


Figure 46. Authentication Process. This figure illustrates the authentication process which is described in the surrounding text.

This figure shows client A connecting to server B. The term $K(CK)$ means *CK* is encrypted with the DES common key *K*. In its first request, the client RPC credential contains the client name (*A*), the conversation key (*CK*), and the variable called *win* (window) encrypted with *CK*. (The default window size is 30 minutes.) The client verifier in the first request contains the encrypted time stamp and an encrypted verifier of the specified window, *win + 1*. The window verifier makes guessing the right credential much more difficult, and increases security.

After authenticating the client, the server stores the following items in a credential table:

- Client name, *A*
- Conversation key, *CK*
- Window
- Time stamp.

The server only accepts time stamps that are chronologically greater than the last one seen, so any replayed transactions are guaranteed to be rejected. The server returns to the client in the verifier an index ID into the credential table, plus the client time stamp minus one, encrypted by *CK*. The client knows that only the server could have sent such a verifier, because only the server knows what time stamp the client sent. The reason for subtracting one from the time stamp is to ensure that it is not valid and cannot be reused as a client verifier. After the first RPC transaction, the client sends just its ID and an encrypted time stamp to the server, and the server sends back the client time stamp minus one, encrypted by *CK*.

Naming Network Entities for DES Authentication

DES authentication does its naming by using net names. The following paragraphs describe how NIS handles DES authentication. For information on how NIS+ handles DES authentication, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

A *net name* is a string of printable characters to authenticate. The public and secret keys are stored on a per-net-name rather than a per-user-name basis. The **netid.byname** NIS map maps the net name into a local UID and group-access list.

User names are unique within each domain. Net names are assigned by concatenating the operating system and user ID with the NIS and Internet domain names. A good convention for naming domains is to append the Internet domain name (com, edu, gov, mil) to the local domain name.

Network names are assigned to machines as well as to users. A net name of a machine is formed much like that of a user. For example, a machine named `hal` in the `eng.ibm.com` domain has the net name `unix.hal@eng.ibm.com`. Correct authentication of machines is important for diskless machines that need full access to their home directories over the network.

To authenticate users from any remote domain, make entries for them in two NIS databases. One is an entry for their public and secret keys; the other is for their local UID and group-access list mapping. Users in the remote domain can then access all of the local network services, such as the NFS and remote logins.

/etc/publickey File

The **/etc/publickey** file contains names and public keys, which NIS and NIS+ use to create the **publickey** map. The **publickey** map is used for secure networking. Each entry in the file consists of a network user name (which refers to either a user or a host name), followed by the user public key (in hexadecimal notation), a colon, and the user encrypted secret key (also in hexadecimal notation). By default, the only user in the **/etc/publickey** file is the user `nobody`.

Do not use a text editor to alter the **/etc/publickey** file because the file contains encryption keys. To alter the **/etc/publickey** file, use either the **chkey** or **newkey** commands.

Booting Considerations of Public Key Systems

When restarting a machine after a power failure, all of the stored secret keys are lost, and no process can access secure network services, such as mounting an NFS. Root processes could continue if there were someone to enter the password that decrypts the secret key of the root user. The solution is to store the root user decrypted secret key in a file that the key server can read.

Not all **setuid** subroutine calls operate correctly. For example, if a **setuid** subroutine is called by owner A, and owner A has not logged into the machine since it started, the subroutine cannot access any secure network services as A. However, most **setuid** subroutine calls are owned by the root user, and the root user secret key is always stored at startup time.

Performance Considerations

Secure NFS affects system performance in two ways.

•

First, both the client and server must compute the common key. The time it takes to compute the common key is about one second. As a result, it takes about two seconds to establish the initial RPC connection, because both client and server have to perform this operation. After the initial RPC connection, the key server caches the results of previous computations, and so it does not have to recompute the common key every time.

•

Second, each RPC transaction requires four DES encryption operations:

1. The client encrypts the request time stamp
2. The server decrypts it
3. The server encrypts the reply time stamp
4. The client decrypts it.

Because system performance is reduced by secure NFS, weigh the benefits of increased security against system performance requirements.

Administering Secure NFS Checklist

The following is a checklist of items to help ensure that secure NFS operates properly:

- When mounting a file system with the **-secure** option on a client, the server name must match the server host name in the **/etc/hosts** file. If a name server is being used for host name resolution, make sure the host information returned by the name server matches the entry in the **/etc/hosts** file. Authentication errors result if these names do not match because the net names for machines are based on the primary entries in the **/etc/hosts** file and keys in the **publickey** map are accessed by net name.
- Do not mix secure and nonsecure exports and mounts. Otherwise, file access might be determined incorrectly. For example, if a client machine mounts a secure file system without the **secure** option or mounts a nonsecure system with the **secure** option, users have access as nobody, rather than as themselves. This condition also occurs if a user unknown to NIS or NIS+ attempts to create or modify files on a secure file system.
- Because NIS must propagate a new map after each use of the **chkey** and **newkey** commands, only use these commands when the network is lightly loaded.
- Do not delete the **/etc/keystore** file or the **/etc/.rootkey** file. If you reinstall, move, or upgrade a machine, save the **/etc/keystore** and **/etc/.rootkey** files.
- Instruct users to use the **yppasswd** command rather than the **passwd** command to change passwords. Doing so keeps passwords and private keys synchronized.
- Because the **login** command does not retrieve keys out of the **publickey** map for the **keyserv** daemon, the user must execute the **keylogin** command. You may want to place the **keylogin** command in each user **profile** file to execute the command automatically during login. Note that the **keylogin** command requires the user to enter their password again.
- When you generate keys for the root user at each host with either the **newkey -h** or **chkey** command, you must run the **keylogin** command to pass the new keys to the **keyserv** daemon. The keys are stored in the **/etc/.rootkey** file, which is read by the **keyserv** daemon each time the daemon is started.

- Periodically verify that the **yppasswdd** and **ypupdated** daemons are running on the NIS master server. These daemons are necessary for maintaining the **publickey** map.
- Periodically verify that that the **keyserv** daemon is running on all machines using secure NFS.

Configuring Secure NFS

To configure secure NFS on NIS master and slave servers, use the Web-based System Manager Network application or use the following procedure. For information on using NFS with NIS+ see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

1. On the NIS master server, create an entry for each user in the NIS **/etc/publickey** file using the **newkey** command. This command has two options.

- For a regular user, enter:

```
smit newkey
```

or

```
newkey -u username
```

commands.

For a root user on a host machine, enter the:

```
newkey -h hostname
```

command.

- Alternatively, users can establish their own public keys using the **chkey** or **newkey**

2. Create the NIS **publickey** map by following the instructions in *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*. The corresponding NIS **publickey.byname** map resides only on the NIS servers.
3. Uncomment the following stanzas in the **/etc/rc.nfs** file:

```
#if [ -x /usr/sbin/keyserv ]; then
#  startsrc -s keyserv
#fi
#if [ -x /usr/lib/netsvc/yp/rpc.yppupdated -a -d /etc/yp/'domainname' ]; then
#  startsrc -s yppupdated
#fi
#DIR=/etc/passwd
#if [ -x /usr/lib/netsvc/yp/rpc.yppasswdd -a -f $DIR/passwd ]; then
#  startsrc -s yppasswdd
#fi
```

4. Start the **keyserv**, **ypupdated**, and **yppasswdd** daemons using the **startsrc** command.

To configure secure NFS on NIS clients, start the **keyserv** daemon using the **startsrc** command.

Exporting a File System Using Secure NFS

You can export a secure NFS using the Web-based System Manager Network application or by using one of the following procedures.

- To export a secure NFS file system using SMIT:

1. Verify that NFS is already be running by issuing the **lssrc -g nfs** command. The output should indicate that the **nfsd** and the **rpc.mountd** daemons are active. If they are not, start NFS using the instructions in Starting the NFS Daemons.
2. Verify that the **publickey** map exists and that the **keyserv** daemon is running. See Configuring Secure NFS for more information.
3. Run the **smit mknfsexp** fast path.

4. Specify the appropriate values for the `PATHNAME` of directory to export, `MODE` to export directory, and `EXPORT` directory now, system restart or both fields. Specify **yes** for the Use SECURE option field.
 5. Specify any other optional characteristics, or simply accept the default values by leaving the remaining fields as they are.
 6. Exit SMIT. If the `/etc/exports` file does not exist, then it will be created.
 7. Repeat steps 3 through 6 for each directory you want to export.
- To export a secure NFS file system using a text editor:
 1. Open the `/etc/exports` file with your favorite text editor.
 2. Create an entry for each directory to be exported, using the full path name of the directory. List each directory to be exported starting in the left margin. No directory should include any other directory that is already exported. See the `/etc/exports` file documentation for a description of the full syntax for entries in the `/etc/exports` file, including how to specify the **secure** option.
 3. Save and close the `/etc/exports` file.
 4. If NFS is currently running, enter:


```
/usr/sbin/exportfs -a
```

The **-a** option tells the **exportfs** command to send all information in the `/etc/exports` file to the kernel. If NFS is not running, start NFS using the instructions in Start the NFS Daemons.

- To export an NFS file system temporarily (that is, without changing the `/etc/exports` file):
Enter:


```
exportfs -i -o secure /dirname
```

where *dirname* is the name of the file system you want to export. The **exportfs -i** command specifies that the `/etc/exports` file is not to be checked for the specified directory, and all options are taken directly from the command line.

Mounting a File System Using Secure NFS

To mount a secure NFS directory explicitly:

1. Verify that the NFS server has exported the directory. Do this by issuing the command:


```
showmount -e ServerName
```

where *ServerName* is the name of the NFS server. This command displays the names of the directories currently exported from the NFS server. If the directory you want to mount is not listed, export the directory from the server.

2. Establish the local mount point using the **mkdir** command. For NFS to complete a mount successfully, a directory that acts as the mount point (or place holder) of an NFS mount must be present. This directory should be empty. This mount point can be created like any other directory, and no special attributes are needed for this directory.
3. Verify that the **publickey** map exists and that the **keyserv** daemon is running. See Configuring Secure NFS for more information.
4. Enter:

```
mount -o secure ServerName:/remote/directory /local/directory
```

where *ServerName* is the name of the NFS server, */remote/directory* is the directory on the NFS server you want to mount, and */local/directory* is the mount point on the NFS client.

Note: Only the root user can mount a secure NFS.

See Establishing Predefined NFS Mounts for more details on mounting with NFS.

NFS Problem Determination

As with other network services, problems can occur on machines that use the Network File System (NFS). Troubleshooting for these problems involves understanding the strategies for tracking NFS problems, recognizing NFS-related error messages, and selecting the appropriate solutions. When tracking down an NFS problem, isolate each of the three main points of failure to determine which is not working: the server, the client, or the network itself.

Note: See Troubleshooting the Network Lock Manager for file lock problems.

Identifying Hard-Mounted and Soft-Mounted File Problems

When the network or server has problems, programs that access hard-mounted remote files fail differently from those that access soft-mounted remote files.

If a server fails to respond to a hard-mount request, NFS prints the message:

```
NFS server hostname not responding, still trying
```

Hard-mounted remote file systems cause programs to hang until the server responds because the client retries the mount request until it succeeds. Use the **-bg** flag with the **mount** command when performing a hard mount so if the server does not respond, the client will retry the mount in the background.

If a server fails to respond to a soft-mount request, NFS prints the message:

```
Connection timed out
```

Soft-mounted remote file systems return an error after trying unsuccessfully for a while. Unfortunately, many programs do not check return conditions on file system operations, so you do not see this error message when accessing soft-mounted files. However, this NFS error message prints on the console.

Identifying NFS Problems Checklist

If a client is having NFS trouble, do the following:

1. Verify that the network connections are good.
2. Verify that the **inetd**, **portmap**, and **bioid** daemons are running on the client, by following the instructions in Getting the Current Status of the NFS Daemons.
3. Verify that a valid mount point exists for the file system being mounted. For more information, see Configuring an NFS Client.
4. Verify that the server is up and running by running the following command at the shell prompt of the client:

```
/usr/bin/rpcinfo -p server_name
```

If the server is up, a list of programs, versions, protocols, and port numbers is printed, similar to the following:

program	vers	proto	port	
100000	2	tcp	111	portmapper
100000	2	udp	111	portmapper
100005	1	udp	1025	mountd
100001	1	udp	1030	rstatd
100001	2	udp	1030	rstatd
100001	3	udp	1030	rstatd
100002	1	udp	1036	rusersd
100002	2	udp	1036	rusersd
100008	1	udp	1040	walld
100012	1	udp	1043	sprayd
100005	1	tcp	694	mountd
100003	2	udp	2049	nfs

100024	1	udp	713	status
100024	1	tcp	715	status
100021	1	tcp	716	nlockmgr
100021	1	udp	718	nlockmgr
100021	3	tcp	721	nlockmgr
100021	3	udp	723	nlockmgr
100020	1	udp	726	llockmgr
100020	1	tcp	728	llockmgr
100021	2	tcp	731	nlockmgr

If a similar response is not returned, log in to the server at the server console and check the status of the **inetd** daemon by following the instructions in *Get the Current Status of the NFS Daemons*.

5. Verify that the **mountd**, **portmap** and **nfsd** daemons are running on the NFS server by entering the following commands at the client shell prompt:

```
/usr/bin/rpcinfo -u server_name mount
/usr/bin/rpcinfo -u server_name portmap
/usr/bin/rpcinfo -u server_name nfs
```

If the daemons are running at the server, the following responses are returned:

```
program 100005 version 1 ready and waiting
program 100000 version 2 ready and waiting
program 100003 version 2 ready and waiting
```

The program numbers correspond to the commands, respectively, as shown in the previous example. If a similar response is not returned, log in to the server at the server console and check the status of the daemons by following the instructions in *Get the Current Status of the NFS Daemons*.

6. Verify that the **/etc/exports** file on the server lists the name of the file system that the client wants to mount and that the file system is exported. Do this by entering the command:

```
showmount -e server_name
```

This command lists all the file systems currently exported by the *server_name*.

Asynchronous Write Errors

When an application program writes data to a file in an NFS-mounted file system, the write operation is scheduled for asynchronous processing by the **biod** daemon. If an error occurs at the NFS server at the same time that the data is actually written to disk, the error is returned to the NFS client and the **biod** daemon saves the error internally in NFS data structures. The stored error is subsequently returned to the application program the next time it calls either the **fsync** or **close** functions. As a consequence of such errors, the application is not notified of the write error until the program closes the file. A typical example of this event is when a file system on the server is full, causing writes attempted by a client to fail.

NFS Error Messages

The following sections explain error codes that can be generated while using NFS.

nfs_server Error Message

Insufficient transmit buffers on your network can cause the following error message:

```
nfs_server: bad sendreply
```

To increase transmit buffers, use the Web-based System Manager, **wsm**, or the System Management Interface Tool (SMIT) fast path, **smit commodev**. Then select your adapter type, and increase the number of transmit buffers.

mount Error Messages

A remote mounting process can fail in several ways. The error messages associated with mounting failures are as follows:

mount: ... already mounted

The file system that you are trying to mount is already mounted.

mount: ... not found in /etc/filesystems

The specified file system or directory name cannot be matched.

If you issue the **mount** command with either a directory or file system name but not both, the command looks in the **/etc/filesystems** file for an entry whose file system or directory field matches the argument. If the **mount** command finds an entry such as the following:

```
/dancer.src:
    dev=/usr/src
    nodename = d61server
    type = nfs
    mount = false
```

then it performs the mount as if you had entered the following at the command line:

```
/usr/sbin/mount -n dancer -o rw,hard /usr/src /dancer.src
```

... not in hosts database

On a network without Network Information Service, this message indicates that the host specified in the **mount** command is not in the **/etc/hosts** file. On a network running NIS, the message indicates that NIS could not find the host name in the **/etc/hosts** database or that the NIS **ypbind** daemon on your machine has died. If the **/etc/resolv.conf** file exists so that the name server is being used for host name resolution, there might be a problem in the **named** database. See Name Resolution on an NFS Server.

Check the spelling and the syntax in your **mount** command. If the command is correct, your network does not run NIS, and you only get this message for this host name, check the entry in the **/etc/hosts** file.

If your network is running NIS, make sure that the **ypbind** daemon is running by entering the following at the command line:

```
ps -ef
```

You should see the **ypbind** daemon in the list. Try using the **rlogin** command to log in remotely to another machine, or use the **rcp** command to remote-copy something to another machine. If this also fails, your **ypbind** daemon is probably stopped or hung.

If you only get this message for this host name, check the **/etc/hosts** entry on the NIS server.

mount: ... server not responding: port mapper failure - RPC timed out

Either the server you are trying to mount from is down or its port mapper is stopped or hung. Try restarting the server to activate the **inetd**, **portmap**, and **ypbind** daemons.

If you cannot log in to the server remotely with the **rlogin** command but the server is up, check the network connection by trying to log in remotely to some other machine. Also check the server network connection.

mount: ... server not responding: program not registered

This means that the **mount** command got through to the port mapper, but the **rpc.mountd** NFS mount daemon was not registered.

mount: access denied ...

Your machine name is not in the export list for the file system you are trying to mount from the server.

You can get a list of the server exported file systems by running the following command at the command line:

```
showmount -e hostname
```

If the file system you want is not in the list, or your machine name or netgroup name is not in the user list for the file system, log in to the server and check the **/etc/exports** file for the correct file system entry. A file system name that appears in the **/etc/exports** file, but not in the output from the **showmount** command, indicates a failure in the **mountd** daemon. Either the daemon could not parse that line in the file, it could not find the directory, or the directory name was not a locally mounted directory. If the **/etc/exports** file looks correct and your network runs NIS, check the **ypbind** daemon on the server. It may be stopped or hung. For more information, see *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide*.

mount: ...: Permission denied

This message is a generic indication that some part of authentication failed on the server. It could be that, in the previous example, you are not in the export list, the server could not recognize your machine **ypbind** daemon, or that the server does not accept the identity you provided.

Check the **/etc/exports** file on the server and, if applicable, the **ypbind** daemon. In this case, you can just change your host name with the **hostname** command and retry the **mount** command.

mount: ...: Not a directory

Either the remote path or the local path is not a directory. Check the spelling in your command and try to run on both directories.

mount: ...: You are not allowed

You must have root authority or be a member of the system group to run the **mount** command on your machine because it affects the file system for all users on that machine. NFS mounts and unmounts are only allowed for root users and members of the system group.

Identifying the Cause of Slow Access Times for NFS

If access to remote files seems unusually slow, ensure that access time is not being inhibited by a runaway daemon, a bad **tty** line, or a similar problem.

Checking Processes

On the server, enter the following at the command line:

```
ps -ef
```

If the server appears to be operating correctly and other users are getting timely responses, make sure your **biobd** daemons are running. Try the following steps:

1. Run the **ps -ef** command and look for the **biobd** daemons in the display.
If they are not running, continue with steps 2 and 3.
2. Stop the **biobd** daemons that are in use by issuing the following command:

```
stopsrc -x biobd -c
```
3. Start the **biobd** daemons by issuing the following command:

```
startsrc -s biobd
```

To determine if one or more **biobd** daemons are not responding, the user can run **nfsstat -c** several times during the period when they suspect that one or more **biobd** daemons are hung. If there is no noticeable change in the number of Remote Procedure Call (RPC) client reads or writes, one or more of the **biobd** daemons are not performing their task. You can determine only that one or more **biobd** daemons are inactive; you cannot determine which one is inactive.

Checking Network Connections

If the **biod** daemons are working, check the network connections. The **nfsstat** command determines whether you are dropping packets. Use the **nfsstat -c** and **nfsstat -s** commands to determine if the client or server is retransmitting large blocks. Retransmissions are always a possibility due to lost packets or busy servers. A retransmission rate of five percent or more is considered high.

The probability of retransmissions can be reduced by changing communication adapter transmit queue parameters. The System Management Interface Tool (SMIT) can be used to change these parameters.

The following values are recommended for NFS servers.

Communication Adapter Maximum Transmission Unit (MTU) and Transmit Queue Sizes		
Adapter	MTU	Transmit Queue
Token Ring	1500	50
	4Mb	40 (Increase if the nfsstat command times out.)
	16Mb	40 (Increase if the nfsstat command times out.)
	8500	40 (Increase if the nfsstat command times out.)
Ethernet	1500	40 (Increase if the nfsstat command times out.)

The larger MTU sizes for each token-ring speed reduce processor use and significantly improve read/write operations.

Notes:

1. Apply these values to NFS clients if retransmissions persist.
2. All nodes on a network must use the same MTU size.

Setting MTU Sizes

To set MTU size, use the Web-based System Manager, **wsm**, or the SMIT fast path, **smit chif**. Select the appropriate adapter and enter an MTU value in the Maximum IP Packet Size field.

The **ifconfig** command can be used to set MTU size (and *must* be used to set MTU size at 8500). The format for the **ifconfig** command is:

```
ifconfig trn NodeName up mtu MTUSize
```

where *trn* is your adapter name, for example, *tr0*.

Another method of setting MTU sizes combines the **ifconfig** command with SMIT.

1. Add the **ifconfig** command for token rings, as illustrated in the previous example, to the **/etc/rc.bsdnet** file.
2. Enter the **smit setbootup_option** fast path. Toggle the Use BSD Style field to **yes**.

Setting Transmit Queue Sizes

Communication adapter transmit queue sizes are set with SMIT. Enter the **smit chgtok** fast path, select the appropriate adapter, and enter a queue size in the Transmit field.

Fixing Hung Programs

If programs hang during file-related work, the NFS server could have stopped. In this case, the following error message may be displayed:

```
NFS server hostname not responding, still trying
```

The NFS server (*hostname*) is down. This indicates a problem with the NFS server, the network connection, or the NIS server.

Check the servers from which you have mounted file systems if your machine hangs completely. If one or more of them is down, do not be concerned. When the server comes back up, your programs continue automatically. No files are destroyed.

If a soft-mounted server dies, other work is not affected. Programs that time out while trying to access soft-mounted remote files fail with the `errno` message, but you are still able to access your other file systems.

If all servers are running, determine whether others who are using the same servers are having trouble. More than one machine having service problems indicates a problem with the `nfsd` daemons on the server. In this case, log in to the server and run the `ps` command to see if the `nfsd` daemon is running and accumulating CPU time. If not, you might be able to stop and then restart the `nfsd` daemon. If this does not work, you have to restart the server.

Check your network connection and the connection of the server if other systems seem to be up and running.

Permissions and Authentication Schemes

Sometimes, after mounts have been successfully established, there are problems in reading, writing, or creating remote files or directories. Such difficulties are usually due to permissions or authentication problems. Permission and authentication problems can vary in cause depending on whether NIS is being used and secure mounts are specified.

The simplest case occurs when nonsecure mounts are specified and NIS is not used. In this case, user IDs (UIDs) and group IDs (GIDs) are mapped solely through the server `/etc/passwd` file and client `/etc/group` file. In this scheme, for a user named B to be identified both on the client and on the server as B, the user B must have the same UID number in the `/etc/passwd` file. The following is an example of how this might cause problems:

```
User B is uid 200 on client foo.  
User B is uid 250 on server bar.  
User G is uid 200 on server bar.
```

The `/home/bar` directory is mounted from server bar onto client foo. If user B is editing files on the `/home/bar` remote file system on client foo, confusion results when he saves files.

The server bar thinks the files belong to user G, because G is UID 200 on bar. If B logs on directly to bar by using the `rlogin` command, he may not be able to access the files he just created while working on the remotely mounted file system. G, however, is able to do so because the machines arbitrate permissions by UID, not by name.

The only permanent solution to this is to reassign consistent UIDs on the two machines. For example, give B UID 200 on server bar or 250 on client foo. The files owned by B would then need to have the `chown` command run against them to make them match the new ID on the appropriate machine.

Because of the problems with maintaining consistent UID and GID mappings on all machines in a network, NIS or NIS+ is often used to perform the appropriate mappings so that this type of problem is avoided. See *AIX 5L Version 5.1 Network Information Services (NIS and NIS+) Guide* for more information.

Name Resolution on an NFS Server

When an NFS server services a mount request, it looks up the name of the client making the request. The server takes the client Internet Protocol (IP) address and looks up the corresponding host name that matches that address. Once the host name has been found, the server looks at the exports list for the requested directory and checks the existence of the client name in the access list for the directory. If an entry exists for the client and the entry matches exactly what was returned for the name resolution, then that part of the mount authentication passes.

If the server is not able to perform the IP address-to-host-name resolution, the server denies the mount request. The server must be able to find some match for the client IP address making the mount request. If the directory is exported with the access being to all clients, the server still must be able to do the reverse name lookup to allow the mount request.

The server also must be able to look up the correct name for the client. For example, if there exists an entry in the `/etc/exports` file like the following:

```
/tmp      -access=silly:funny
```

the following corresponding entries exist in the `/etc/hosts` file:

```
150.102.23.21      silly.domain.name.com
150.102.23.52      funny.domain.name.com
```

Notice that the names do not correspond exactly. When the server looks up the IP address-to-host-name matches for the hosts `silly` and `funny`, the string names do not match exactly with the entries in the access list of the export. This type of name resolution problem usually occurs when using the **named** daemon for name resolution. Most **named** daemon databases have aliases for the full domain names of hosts so that users do not have to enter full names when referring to hosts. Even though these host-name-to-IP address entries exist for the aliases, the reverse lookup might not exist. The database for reverse name lookup (IP address to host name) usually has entries containing the IP address and the full domain name (not the alias) of that host. Sometimes the export entries are created with the shorter alias name, causing problems when clients try to mount.

Limitations on the Number of Groups in the NFS Structure

On systems that use NFS Version 3.2, users cannot be a member of more than 16 groups without complications. (Groups are defined by the **groups** command.) If a user is a member of 17 or more groups, and the user tries to access files owned by the 17th (or greater) group, the system does not allow the file to be read or copied. To permit the user access to the files, rearrange the group order.

Mounting from NFS Servers That Have Earlier Version of NFS

When mounting a file system from a pre-Version 3 NFS server onto a Version 3 NFS client, a problem occurs when the user on the client executing the mount is a member of more than eight groups. Some servers are not able to deal correctly with this situation and deny the request for the mount. The solution is to change the group membership of the user to a number less than eight and then retry the mount. The following error message is characteristic of this group problem:

```
RPC:  Authentication error; why=Invalid client credential
```

Problems That Occur If the NFS Kernel Extension Is Not Loaded

Some NFS commands do not execute correctly if the NFS kernel extension is not loaded. Some commands with this dependency are: **nfsstat**, **exportfs**, **mountd**, **nfsd**, and **biod**. When NFS is installed on the system, the kernel extension is placed in the `/usr/lib/drivers/nfs.ext` file. This file is then loaded as the NFS kernel extension when the system is configured. The script that does this kernel extension loads

the `/etc/rc.net` file. There are many other things done in this script, one of which is to load the NFS kernel extension. It is important to note that Transmission Control Protocol/Internet Protocol (TCP/IP) kernel extension should be loaded before the NFS kernel extension is loaded.

Note: The `gfsinstall` command is used to load the NFS kernel extension into the kernel when the system initially starts. This command can be run more than once per system startup and it will not cause a problem. The system is currently shipped with the `gfsinstall` command used in both the `/etc/rc.net` and `/etc/rc.nfs` files. There is no need to remove either of these calls.

NFS Reference

List of Network File System (NFS) Files

<code>bootparams</code>	Lists clients that diskless clients can use for booting.
<code>exports</code>	Lists the directories that can be exported to NFS clients.
<code>networks</code>	Contains information about networks on the Internet network.
<code>pcnfsd.conf</code>	Provides configuration options for the <code>rpc.pcnfsd</code> daemon.
<code>rpc</code>	Contains database information for Remote Procedure Call (RPC) programs.
<code>xtab</code>	Lists directories that are currently exported.
<code>/etc/filesystems</code>	Lists all the file systems that are mounted at system restart.

List of NFS Commands

<code>chnfs</code>	Starts a specified number of <code>bioid</code> and <code>nfsd</code> daemons.
<code>mknfs</code>	Configures the system to run NFS and starts NFS daemons.
<code>nfsd</code>	Configures NFS network options.
<code>automount</code>	Mounts an NFS file system automatically.
<code>chnfsexp</code>	Changes the attributes of an NFS-exported directory.
<code>chnfsmnt</code>	Changes the attributes of an NFS-mounted directory.
<code>exportfs</code>	Exports and unexports directories to NFS clients.
<code>lsnfsexp</code>	Displays the characteristics of directories that are exported with NFS.
<code>lsnfsmnt</code>	Displays the characteristics of mounted NFS systems.
<code>mknfsexp</code>	Exports a directory using NFS.
<code>mknfsmnt</code>	Mounts a directory using NFS.
<code>rmnfs</code>	Stops the NFS daemons.
<code>rmnfsexp</code>	Removes NFS-exported directories from a server's list of exports.
<code>rmnfsmnt</code>	Removes NFS-mounted file systems from a client's list of mounts.

List of NFS Daemons

Locking Daemons

<code>lockd</code>	Processes lock requests through the RPC package.
<code>statd</code>	Provides crash-and-recovery functions for the locking services on NFS.

Network Service Daemons and Utilities

<code>bioid</code>	Sends the client read and write requests to the server.
<code>mountd</code>	Answers requests from clients for file system mounts.
<code>nfsd</code>	Starts the daemons that handle a client requests for file system operations.
<code>pcnfsd</code>	Handles service requests from PC-NFS clients.
<code>nfsstat</code>	Displays information about the ability to receive calls for a particular machine.
<code>on</code>	Executes commands on remote machines.

portmap	Maps RPC program numbers to Internet port numbers.
rex	Accepts request to run programs from remote machines.
rpcgen	Generates C code to implement an RPC protocol.
rpcinfo	Reports the status of RPC servers.
rstatd	Returns performance statistics obtained from the kernel.
rup	Shows the status of a remote host on the local network.
rusers	Reports a list of users logged on to the remote machines.
rusersd	Responds to queries from the rusers command.
rwall	Sends messages to all users on the network.
rwalld	Handles requests from the rwall command.
showmount	Displays a list of all clients that have mounted remote file systems.
spray	Sends a specified number of packets to a host.
sprayd	Receives packets sent by the spray command.

Secure Networking Daemons and Utilities

chkey	Changes the user encryption key.
keyenvoy	Provides an intermediary between user processes and the key server.
keylogin	Decrypts and stores the user secret key.
keyserv	Stores public and private keys.
mkkeyserv	Starts the keyserv daemon and uncomments the appropriate entries in the /etc/rc.nfs file.
newkey	Creates a new key in the publickey file.
rmkeyserv	Stops the keyserv daemon and comments the entry for the keyserv daemon in the /etc/rc.nfs file.
ypupdated	Updates information in Network Information Service (NIS) maps.

Sun Diskless Client Support

bootparamd	Provides information necessary for booting to diskless clients.
-------------------	---

NFS Subroutines

cbc_crypt , des_setparity , or ecb_crypt	Implements Data Encryption Standard (DES) routines.
---	---

Chapter 11. AIX Fast Connect

Note: AIX Fast Connect is a licensed program product (LPP). You must have purchased this software separately to use the features described in this section.

AIX Fast Connect is server software that lets AIX servers and workstations share files and printers with personal computer clients running Windows 2000, Windows NT, Windows 98, Windows 95, Windows for Workgroups, or OS/2 operating systems. This chapter discusses the following topics:

- AIX Fast Connect Overview
 - Features
 - Requirements
 - Packaging and Installation
 - Limitations
- Windows Networking Concepts (NetBIOS, SMB, WINS)
- AIX Fast Connect Configuration and Administration
 - Overview
 - Configurable Parameters
 - Configuration of File and Print Shares (Exports)
 - User Administration
 - Basic Server Administration
 - NetBIOS Name Service (NBNS)
- Configuring Client PCs for use with AIX Fast Connect
 - TCP/IP Configuration
 - NetBIOS Name Resolution
 - Workgroups, Domains, and User Accounts
 - Enabling Windows Clients for Plain Text Passwords
 - Browsing the Network
 - Mapping Drives
 - Using AIX Fast Connect Printers
 - Support for Windows 2000
 - Support for Windows Terminal Server
- Advanced AIX Fast Connect Features
 - AIX-based User Authentication (Plain Text Passwords)
 - CIFS Password Encryption Protocol
 - NT Passthrough Authentication
 - Network Logon to AIX Fast Connect
 - DCE/DFS Support
 - Guest Logon
 - Share-Level Security
 - User Name Mappings
 - AIX Fast Connect User Management and File Access
 - Mapping Long AIX File Names to 8.3 DOS File Names
 - Support for DOS File Attributes
 - Specifying NetBIOS Aliases for HACMP-support
 - Performance Considerations

- AIX Fast Connect Problem Determination
 - Traces
 - Logs
 - Troubleshooting Connection Problems
- Configuring Network Logon for AIX Fast Connect
- AIX Fast Connect Configurable Parameters for the net Command
- Migrating to AIX Fast Connect from AIX Connections

AIX Fast Connect Overview

Because AIX Fast Connect uses industry-standard Microsoft networking protocols, PC clients can access AIX files and printers using their native networking client software. PC users can use remote AIX file systems directly from their machines like local file systems, and access AIX print queues like local printers. AIX Fast Connect provides these services by implementing the SMB networking protocol on top of NetBIOS over TCP/IP (RFC-1001/1002). See Windows Networking Concepts for more information about these concepts.

Features

Important features of AIX Fast Connect include:

AIX-application standard and advanced features, including:

- Tight integration with AIX, using AIX features such as threads, kernel I/O, filesystem, and security
- Maintenance and administration using SMIT, command-line, or Web-based System Manager
- Streamlined configuration
- Trace and log capabilities
- SendFile API support
- DCE/DFS integration
- Support for JFS-ACLs
- HACMP support, using server name aliases.

Advanced SMB/NetBIOS features, including:

- SMB-based file and print services
- Passthrough authentication to NT
- Resource Browsing Protocol (Network Neighborhood)
- Network Logon support, including Roaming user-profiles
- WINS client and proxy, and NBNS-server
- Opportunistic locking (oplock)
- B-node support
- Guest Logon support
- Share-Level Security support
- Messaging from server to client
- Mapping of AIX long filenames to DOS 8.3 filenames
- Unicode representation of share, user, file, and directory names.
- Mapping of PC-client usernames to AIX usernames
- Multiplexed SMB-sessions (for Windows Terminal Server support).

Requirements

This section includes hardware and software requirements, both for the AIX server and for its PC clients.

Server Hardware Requirements

AIX Fast Connect runs on any machine that supports the AIX 4.3.3 (or later) of the operating system, except for diskless or dataless machines. This server machine must have:

- 32 MB of RAM Minimum (64 MB is preferred)
- 50 MB of available disk space
- TCP/IP-supported LAN adapters physically connected to a network.

Server Software Requirements

The following are the server software requirements for AIX Fast Connect:

- AIX 4.3.3 or higher
- The size of `/var` should be large enough to temporarily store the largest file that can be printed by the print service
- Fileset **bos.net.tcp.client** version 4.3.2.0 or higher must be installed and configured.
- Fileset **bos.rte.loc** version 4.3.2.2 or higher must be installed and configured.

Client Hardware Requirements

Each client PC must have an installed LAN adapter and should be physically connected to a network.

Client Software Requirements

To use Fast Connect, all client PCs must have one of the following operating systems:

- Windows 2000 (with Service Pack 1 or higher)
- Windows NT 4.0 (with Service Pack 3 or higher)
- Windows 98
- Windows 95 (with Service Pack 1 or higher)
- Windows for Workgroups 3.11 or higher
- OS/2 Warp 4.0 or higher

To use the Web-based System Manager, a web browser with forms support (for example, Netscape) is required.

Known Conflicts with other Server Software

Like other NetBIOS servers, AIX Fast Connect cannot share ownership of the TCP/IP ports used for NetBIOS (on a single machine). The following NetBIOS-based server-software is known to conflict with AIX Fast Connect. These products should be uninstalled before installing AIX Fast Connect:

Fileset	Description
SAMBA.*	Samba server
netbios.*	NetBIOS/ix for AIX
connect.*	AIX Connections
TAS.*	TotalNet Advanced Server for AIX
ASU.*	Advanced Server for UNIX

Migrating from AIX Connections

Before uninstalling AIX Connections (and NetBIOS/ix) and before installing AIX Fast Connect, read Migrating to AIX Fast Connect from AIX Connections to preserve configuration-data from AIX Connections and NetBIOS/ix.

Also, note that AIX Fast Connect supports only NB-realm networking using NetBIOS over TCP/IP (RFC 1001/1002). AIX Fast Connect does not support IPX/SPX, NetBEUI, or Netware protocols.

Packaging and Installation

This section describes the AIX Fast Connect packaging images and installation requirements.

Packaging

AIX Fast Connect packaging includes the following images:

Image	Description
cifs.base	Server Utilities
cifs.client	Client Utilities
cifs.msg.*	Server Messages (by language)
cifs.websm	Web-based System Manager Utilities (AIX 5.1 or later)
-and-	
cifs.advanced-demo	Demo Version (for Windows and OS/2 clients)
-or- cifs.advanced	Advanced Server (for Windows and OS/2 clients)
-or- cifs.basic	Server (for Windows clients only)

Note: The install files, **cifs.basic** and **cifs.advanced**, are mutually exclusive. Standard distributions of AIX Fast Connect contain only one of these images.

These images contain the following filesets:

Image	Fileset	Fileset Description
cifs.base	cifs.base.websm	Web-based System Manager support (AIX 4.3.3only)
	cifs.base.smit	SMIT support
	cifs.base.cmd	Commands
	cifs.base.ldap	Active Directory Support
cifs.client	cifs.client.rte	Client support
cifs.websm	cifs.websm.apps	Web-based System Manager support (AIX 5.1 and later)
cifs.msg.*	cifs.msg.*	Server messages (by language)
cifs.advanced-demo	cifs.advanced-demo.rte	Demo Version files (for Windows and OS/2 clients)
cifs.advanced	cifs.advanced.rte	Advanced server files (for Windows and OS/2 clients)
cifs.basic	cifs.basic.rte	Server files (for Windows clients only)

Installation

Installation of AIX Fast Connect creates the following files on the server:

File	Type	Path	Description
net	binary	/usr/sbin	Command-line administration command
cifsClient	binary	/usr/sbin	Command-line utility for sending messages to PC clients
cifsLdap	binary	/usr/sbin	Command-line utility for Active Directory support
rc.cifs	script	/etc	Start/stop shell script
cifsServer	binary/link	/usr/sbin	Main server daemon (one main server process, owned by root)
cifsServerAdv	binary	/usr/sbin	Main server daemon (from cifs.advanced)
cifsServerAdvDemo	binary	/usr/sbin	Main server daemon (from cifs.advanced-demo)
cifsUserProc	link	/usr/sbin	Client-session daemon (one process per PC-client session)

File	Type	Path	Description
cifsPrintServer	binary	/usr/sbin	Print server daemon
cifsPrintServerDCE	binary	/usr/sbin	Print server daemon (for DCE/DFS support)
cifsConfig	text	/etc/cifs	Server configuration file
cifsPasswd	text	/etc/cifs	User-database file
README	HTML	/etc/cifs	Additional documentation
cifsLog	text	/var/cifs	Log file
cifsTrace *	text	/var/cifs	Trace file
sm_smb.cat	message catalog	/usr/lib/nls/msg	Runtime message catalogs (by language)

Installation notes:

- For DCE/DFS support, you must install **dce.client.*** before installing AIX Fast Connect. When AIX Fast Connect is installed, it creates the **cifsUserProc** file as a soft link to **cifsPrintServer** or **cifsPrintServerDCE**, as appropriate.
- If **cifs.advanced** or **cifs.advanced-demo** is installed, then file **cifsServer** is created as a soft link to **cifsServerAdv** or **cifsServerAdvDemo**, as appropriate.
- The **cifsTrace** file does not appear on the system until tracing is enabled using the **net trace** command.

Configuration of Network Interfaces

Whenever the AIX Fast Connect server is started, it automatically supports RFC1001/1002 (NetBIOS over TCP/IP) on all AIX TCP/IP interfaces that are currently defined and operational ("up"). No special or additional configuration is required to support these interfaces.

Initial Configuration

During installation, AIX Fast Connect preconfigures itself as an SMB/NetBIOS file server with the following default parameters:

Parameter	Initial Value
servername	<i>hostname</i> (TCP/IP hostname)
comment	"Fast Connect server on <i>hostname</i> "
domainname	WORKGROUP
encrypt_passwords	0 (PlainText passwords)
guestlogonsupport	0 (disabled)
networklogon	0 (disabled)
share_level_security	0 (disabled)

In addition, the **HOME** file share is predefined, and it maps to **\$HOME**, the AIX Fast Connect user's home directory on AIX.

Other server parameters are initially at the default values.

Limitations

The following limitations apply to AIX Fast Connect:

- The maximum file size is 4GB. (Individual files must be less than 4GB.)
- All AIX user names that access AIX Fast Connect must have an AIX home directory specified. Otherwise, access is not granted to that user name.
- Users of OS/2 or other clients that do not support Unicode must ensure client and server locales match.
- Disk quota and **ulimit** are not checked for each user. A single user can fill the shared file system.

- AIX Fast Connect does not allow multiple printer share names for a single AIX print queue name. If you try to create a printer share for an AIX print queue that is already being mapped to another printer share, the system displays the message Operation could not be performed.
- Some AIX back-end printer drivers add controls to the file that is being printed; others do not. Windows clients always send print jobs in a format that needs no controls. So, if your AIX printer driver adds controls, set the **-o -dp** printer share options when you create the printer share.
- Guest Logon support is mutually exclusive to DCE/DFS authentication ("dce_auth=1"). Also, Guest Logon support is mutually-exclusive to NT Domains Passthrough Authentication.
- Network Logon support is mutually-exclusive to NT Domains Passthrough Authentication. Network Logon is supported for Windows NT clients only through *IBM Networks Primary Logon Client for Windows NT* (http://service.boulder.ibm.com/asd-bin/doc/en_us/winntcl2/f-feat.htm). Network Logon is not supported for Windows 2000.
- Share names and comments can only be in ASCII.
- **LC_MESSAGES=C@lft** environment variable does not support multibyte characters. If AIX Fast Connect is running in a multibyte environment and **LC_MESSAGES** is set to **C@lft**, either unset it or set this variable to the correct locale at the beginning of the AIX Fast Connect program.
- There are four known Japanese characters that are not supported due to differences in Unicode mapping between IBM cp943 and Microsoft ms932. These are:

SJIS Code	Character Name
815C	EM DASH
8160	WAVE DASH
8161	DOUBLE VERTICAL LINE
817C	MINUS SIGN

Windows Networking Concepts (NetBIOS, SMB, WINS)

The following definitions explain some common Windows networking terms:

B-Nodes

(Broadcast nodes)

A Broadcast Node (B-node) is a type of NetBIOS end node that supports NetBIOS service and contains applications. B-nodes communicate using a mix of UDP datagrams and TCP connections. B-nodes can freely interoperate with one another within a broadcast area; normally a single LAN segment. Other standard end nodes are point-to-point nodes (P-nodes) and Mixed-mode nodes (M-nodes).

Browsing

Browsing refers to viewing the resources available on a network. The browse list on a Windows network is the list of other hosts and domains available on a network. Windows maintains the browse list to present other hosts offering network services through a point-and-click user interface rather than asking users to remember the names of remote hosts and services. Windows clients use the browse list to construct the view of the network shown in the Network Neighborhood (renamed *My Network Places* in Windows 2000) and Windows Explorer. The browse list is also accessible from the command line using the NET VIEW command.

Windows for Workgroups and Windows NT domains maintain the browse list on a computer called the Master Browser. Whenever a computer offers a network service for the first time, it broadcasts a server announcement packet. The Master Browser receives this packet and adds the computer's name to its browse list. In response, the Master Browser transmits a list of backup browsers to the new computer.

Each domain or NT group contains at least one backup browser. A copy of the browse list is maintained on the backup browser to eliminate the need to rebuild the browse list if the Master

Browser goes down. For more information about NT domains and network browsing, see the related Microsoft **technet** site on the World Wide Web.

CIFS CIFS stands for Common Internet File System protocol. CIFS provides an open cross-platform mechanism for client systems to request file services from server systems over a network. It is based on the Server Message Block protocol widely in use by PCs and workstations running a wide variety of operating systems. It is a draft submitted by Microsoft to the Internet Engineering Task Force for transparent file access across the Internet.

NetBIOS

NetBIOS, or Network Basic Input/Output System, is a vendor-independent network interface originally designed for IBM PC computer systems running PC-DOS or MS-DOS. NetBIOS is a software interface, not an actual networking protocol. It specifies the services that should be available without putting any restrictions on the protocol used to implement those services.

There is no officially defined NetBIOS standard. The original version, as described by IBM in 1984 in the *IBM PC Network Technical Reference Manual*, is treated as the de facto standard. Since its introduction, three main versions of NetBIOS have emerged, each using its own transport protocol: NetBEUI, NetBIOS over IPX, and NetBIOS over TCP/IP.

AIX Fast Connect uses NetBIOS over TCP/IP.

NetBIOS Interface to Application Programs

On PCs, NetBIOS includes both a set of services and an exact program interface to those services. There are three types of NetBIOS services:

Name Service

NetBIOS resources are referenced by name. Lower level addresses are not available to NetBIOS applications. An application representing a resource registers one or more names that it wants to use.

The name space is flat and not hierarchically organized. It uses 15 alphanumeric characters, plus a 16th "subcode" byte. Names cannot start with an asterisk (*).

Registration implies bidding for use of a name. The bid may be for exclusive (unique) or shared (group) ownership. Each application contends with other applications in real time. No two applications on the NetBIOS network can use the same unique name until the originating application requests that its name be deleted or the host is powered off or reset.

The three primitive operations provided by Name Service are **Add Name**, **Add Group Name**, and **Delete Name**.

Session Service

A *session* is a full-duplex, sequenced, and reliable message exchange conducted between a pair of NetBIOS applications. Data is organized into messages.

Multiple sessions can exist between any two applications. Both applications participating in the session have access to the name of the remote application. No specification is given for resolving session requests to a group name into a data connection. A service is provided for the detection of a session failure by an application.

The Session Service primitives are **Call**, **Listen**, **Hang Up**, **Send**, **Receive**, and **Session Status**.

Datagram Service

The Datagram Service is an unreliable, nonsequenced, and connectionless communication between two NetBIOS applications. It is analogous to UDP service under TCP/IP.

Datagrams are sent under cover of a name properly registered to the sender. Datagrams can be sent to a specific name or be explicitly broadcast.

Datagrams sent to an exclusive name are received, if at all, by the holder of that name. Datagrams sent to a group name are multicast to all holders of that name. The sending application cannot distinguish between group and unique names and thus must act as if all nonbroadcast datagrams are multicast.

As with the Session Service, the receiver of the datagram is told the sending and receiving names.

The Datagram Service primitives are **Send Datagram**, **Send Broadcast Datagram**, **Receive Datagram**, and **Receive Broadcast Datagram**.

NetBIOS Name Resolution

Name Resolution refers to mapping a NetBIOS name to its corresponding IP address. The techniques commonly used for name resolution are the Windows Internet Name Service (WINS), the LMHOSTS file, and the domain name system (DNS). DNS is explained in TCP/IP Name Resolution. The other techniques are defined below:

WINS/NBNS

When a new service is made available on the network, such as when a Windows machine boots or when AIX Fast Connect is started, the service must be registered with a WINS server before it can be available to clients located on other subnets. The WINS server records the name of the host, the NT domain the host is part of, and the IP address of the host. Whenever a machine attempts to resolve a host name, it first checks with the WINS server. If the host is not registered there, it attempts to find the host using a broadcast. If the host is still not found, the system returns a computer or share name could not be found. AIX Fast Connect properly registers itself with any WINS server.

WINS also includes a method for replicating its database of host names with other WINS servers to create a backup WINS server that can host queries if the primary WINS server is unavailable. It also allows large networks that are encumbered by slow links to distribute WINS servers closer to clients and provide faster name resolution. (WINS is a proprietary Microsoft protocol.)

AIX Fast Connect can be configured to act as a NBNS (NetBIOS Name Service) server, providing most WINS functionality. AIX Fast Connect can also be configured to act as a WINS proxy to other WINS or NBNS servers. (See NetBIOS Name Service for details.)

LMHOSTS

LMHOSTS stands for LanManager Hosts and is analogous to the UNIX */etc/hosts* file. The LMHOSTS file allows specific NetBIOS server names to be mapped to IP addresses. It also provides a syntax for defining the domain a NetBIOS server resides in, as well as loading a LMHOSTS file from a shared directory on a server.

For more details on the LMHOSTS file, refer to the *Windows NT Networking Guide* or the *Windows 95 Resource Kit*.

NetBIOS over TCP/IP

NetBIOS over TCP/IP was first proposed in RFCs 1001 and 1002, which were submitted to the Internet Engineering Task Force in 1987. These RFCs describe an implementation of NetBIOS using Transmission Control Protocol (TCP) for connection-oriented session services and User Datagram Protocol (UDP) for datagram services.

This design has some significant advantages over NetBEUI and NetBIOS over IPX. First, it uses the existing TCP/IP protocols, so it can be routed across the global Internet and any other wide area networks. Secondly, software implementing the NetBIOS interface can be built using existing TCP/IP implementation without requiring any new network drivers. Since most operating systems already support TCP/IP, most are capable of supporting NetBIOS with minimal additional effort.

NetBIOS Scope

A NetBIOS Scope is the population of computers across which a registered NetBIOS name is known. NetBIOS broadcast and multicast datagram operations must reach the entire extent of the NetBIOS scope.

net Command

The **net** command and its subcommands can be used to configure and administer the AIX Fast Connect Server from the command line. Alternatively, the Web-based System Manager and SMIT offer menu-driven interfaces for the same tasks. For detailed information about the **net** command, refer to the *AIX 5L Version 5.1 Commands Reference, Volume 4*.

Passthrough Authentication

Passthrough authentication is a mechanism employed by the AIX Fast Connect server to validate user credentials with a domain controller and, if validated, to grant the user access to a resource on the AIX Fast Connect server. For more details on passthrough authentication, refer to Internet Draft, *CIFS Login and Passthrough Authentication, Prelim Draft, Jan 3, 1997*.

SMB SMB stands for Server Message Block. It is the protocol used on top of NetBIOS to implement Windows file sharing and print services.

With this protocol, clients exchange messages (called Server Message Blocks) with a server to access resources on that server. Every SMB message has a common format: it consists of a fixed-sized header followed by a variable-sized parameter and data component.

SMB messages can be broken into the following types:

- Session control messages start, authenticate, and terminate sessions.
- File and printer messages control file and printer access, respectively.
- Message commands allow an application to send or receive messages to or from another host.

When an SMB client negotiates a connection with an SMB server, the two parties decide on a common protocol to use for communication. This capability allows protocol extensions but can make SMB quite complex.

Shares

Shares are resources exported to the network by the AIX Fast Connect server. The two types of shares supported by AIX Fast Connect are AIX files and printers.

Workgroups

A workgroup is a logical collection of workstations and servers that do not belong to a domain. In a workgroup, each computer stores its own copy of user and group account information. Therefore, in workgroups, users can only log directly onto machines on which they have accounts. Workgroup members are able to view and use resources on other systems. To do this, resources are shared in the workgroup and network users are validated by the machine owning the resource.

AIX Fast Connect Configuration and Administration

This chapter discusses basic configuration and operation of AIX Fast Connect. Some examples are given, using the AIX Fast Connect command-line interface, the **net** command. (AIX Fast Connect also supports the system-management tools SMIT and Web-based System Manager.)

Note: Unless otherwise noted, all references to the **net** command in this section refer to the AIX Fast Connect command (**/usr/sbin/net**) not the NET command used on DOS, OS/2, and Windows. (Examples of the NET command use on PC clients are shown in the next section, Configuring Client PCs for use with AIX Fast Connect.)

Overview

You can use the Web-based System Manager, SMIT, the **net** command, or a combination of these methods to configure and administer the AIX Fast Connect server for your site.

As indicated in AIX Fast Connect Packaging and Installation, AIX Fast Connect preconfigures itself to provide basic access to AIX user home directories (as defined in */etc/passwd*) using plain-text network passwords. When started, the AIX Fast Connect server responds to SMB/NetBIOS requests on all operational TCP/IP interfaces.

Configurable Parameters

AIX Fast Connect is designed for ease of administration, but provides a sufficient set of customizable parameters to support various configurations. Several of these parameters are dynamically configurable and do not require the server to be stopped and restarted for the changes to become effective.

These parameters are found in the */etc/cifs/cifsConfig* file, and can be configured by using the **net config** command with the following syntax:

```
net config /parameter_name:parameter_value
```

The entire list of these configurable parameters is shown in the Table of AIX Fast Connect Configurable Parameters or by typing: **net config help** on the command line.

Note: Use the Web-based System Manager or SMIT for most changes to the AIX Fast Connect configuration parameters, both to avoid spelling mistakes and because some of these parameters must be changed simultaneously. However, examples of the `net config` command are shown below, for AIX Fast Connect system administrators who prefer this method.

- **To show the current configuration** (an abbreviated list), type:

```
net config
```

This command shows some of the most important parameters, including *servername*, *domainname*, and *primary_wins_ipaddr*.

- **To show a single parameter** (for example, parameter *servername*), type:

```
net config /parm:servername
```

- **To change a parameter** (for example, changing the *domainname*, the *autodisconnect* timeout, and the server *comment*), type:

```
net config /domainname:testdomain
```

```
net config /autodisconnect:60
```

```
net config /comment:"String parameter containing Spaces"
```

Configuration of File and Print Shares (Exports)

There are two types of shares that can be configured and exported by AIX Fast Connect: File Shares and Print Shares. Whenever the AIX Fast Connect server is started, a file share with the network name HOME is created by default. This special file share maps to **\$HOME**, the AIX home directory (from */etc/passwd*) of any PC-client user that connects to AIX Fast Connect. (Additionally, the file shares IBMLAN\$ and ADMIN\$ may be created by default, to support the Network Logon feature of AIX Fast Connect.) More file or print shares can be added by the system administrator using Web-based System Manager, SMIT, or the **net** command.

Note: The default shares HOME, IBMLAN\$, and ADMIN\$ cannot be changed or deleted.

Each file or print share represents an object that AIX Fast Connect is exporting to the Windows network, accessed by its *netname*. File shares are exported AIX directories. Print shares are exported AIX print queues.

- **To list all shares** currently exported by AIX Fast Connect, type:

```
net share
```

- **To add a new file share** (for example, to export AIX directory */tmp* as network-name NETTEMP), type:

```
net share /add /type:f /netname:NETTEMP /path:/tmp /desc:"File share test"
```


- **To add a new printer share** (for example, to export AIX print queue **psColor1** as network name PSCOLOR1), type:

```
net share /add /type:p /netname:PSCOLOR1 /printq:psColor1 /desc:"Print share test"
```

Note: AIX names for files, directories, and print-queues are case-sensitive, but network-names used by Windows networking are *not* case-sensitive.

- **To delete a share** (for example, share NETTEMP listed above), type:

```
net share /delete /netname:NETTEMP
```

Note: If files seem to be missing in the directory when viewed from a PC client, AIX Fast Connect uses the AIX file permission bits to encode DOS file attributes (ReadOnly, Archive, System, Hidden). For more information, see Support for DOS File Attributes. Also, you might want to review Mapping Long AIX File Names to DOS File Names.

User Administration

Access to AIX Fast Connect shares is managed internally by AIX user security mechanisms. For example, if an AIX user has write access to a particular AIX subdirectory that is being exported by AIX Fast Connect, then any PC client connecting to AIX Fast Connect (as that AIX user) would then have write access to that same subdirectory. (There are cases when an external PC client accesses AIX Fast Connect with a client username that is different than the server username being used for access checking, for example guestmode, share-level security, and username mapping.)

User accounts can be configured on the server using Web-based System Manager, SMIT, or the **net** command. Each defined AIX Fast Connect user must also be a defined AIX user. AIX Fast Connect supports user-level authentication using several mechanisms described in the following section. Resource access is permitted based on the authenticated AIX user credentials.

Note: Every AIX username used for AIX Fast Connect authentication *must* have an AIX home directory specified. Otherwise, that user cannot access the AIX Fast Connect server.

Overview of User-Authentication Mechanisms

AIX Fast Connect supports several different types of user-authentication for access to the AIX Fast Connect server. Whichever authentication method you choose depends on your existing network environment and your network policies. These authentication methods are discussed briefly in this section. For more information, see Advanced Server Administration.

AIX-based User Authentication (using plain text network passwords)

When the AIX Fast Connect server is configured for plain text passwords (and *not* NT-Passthrough authentication), then incoming SMB username/password logins are sent to standard AIX system services for user authentication, which includes integrated DCE-login, if specified for that AIX-user.)

To enable Plain Text passwords for AIX Fast Connect, type:

```
net config /encrypt_passwords:0
```

Note: SMB networking does not support mixed case for plain text passwords. In plain text mode, every AIX user accessing AIX Fast Connect must have all uppercase or all lowercase AIX passwords.

CIFS Password Encryption Protocols

When the AIX Fast Connect server is configured for encrypted passwords (and *not* NT-Passthrough authentication), then incoming SMB username/encrypted_password logins are validated by AIX Fast Connect against the **/etc/cifs/cifsPasswd** file, which is a database of AIX Fast Connect users (and their encrypted passwords). The **/etc/cifs/cifsPasswd** file is initialized and maintained by the **net user** command (see Configuring Encrypted Passwords).

To enforce encrypted passwords for AIX Fast Connect, type:

```
net config /encrypt_passwords:2
```

NT Passthrough Authentication

When the AIX Fast Connect server is configured for NT-Passthrough Authentication, then the `encrypt_passwords` parameter is ignored, and incoming PC client login requests are routed through the network to an external Windows NT server for user authentication. (Normally, the PC-client uses encrypted passwords to authenticate with the external Windows NT server.) This method is often used when an NT server is already being used as a Network Logon server for the Windows network.

To enable AIX Fast Connect to authenticate to an external NT server (located at TCP/IP address *IPAddress*), type:

```
net config /passthrough_authentication_server:IPaddress
```

You can also designate a backup server for NT authentication with the following command:

```
net config /backup_passthrough_authentication_server:IPaddress2
```

Network Logon to AIX Fast Connect

AIX Fast Connect itself can be configured to act as a Network Logon server. (Windows NT clients require the IBM Primary Logon Client for NT to use this feature. Windows 2000 clients cannot use this feature of AIX Fast Connect.) For more information about Network Logon, see *Advanced Server Administration*, and *Configuring Network Logon for AIX Fast Connect*.

DCE/DFS Support

AIX Fast Connect can be configured for DCE/DFS support using plain text or incrypted passwords. In this mode, Fast Connect uses DCE-authentication mechanisms to validate PC-clients for DFS access.

See *Advanced Server Administration* for more details.

Guest Logon

AIX Fast Connect can support guest-mode logon when configured for either plain-text or encrypted passwords. If AIX Fast Connect is enabled for guest-mode logins, then an incoming PC client username (which AIX Fast Connect must recognize as *not* a standard AIX Fast Connect user) is granted guest-mode access rights based on the AIX Fast Connect username specified as the *guest-user* (parameter *guestname*).

See *Advanced Server Administration* for more details.

Share-Level Security

When the AIX Fast Connect server is configured for share-level security, then passwords are associated with individual file and print shares, not with PC client usernames. In this mode, AIX Fast Connect provides access rights to PC clients based on a share-mode username specified as parameter *share_level_security_username*, similar to the *guest-logon* access mode.

See *Advanced Server Administration* for more details.

Client-to-Server Username Mappings

As an extension of the `net user` command, AIX Fast Connect can map PC client usernames (or *sets* of PC client usernames) to AIX usernames, for user-mode authentication and file access.

See *Advanced Server Administration* for more details.

Configuring Encrypted Passwords

When the AIX Fast Connect server is configured for encrypted passwords, AIX Fast Connect attempts to authenticate all incoming SMB username/encrypted_password logins against the AIX Fast Connect `/etc/cifs/cifsPasswd` file, which is database of AIX Fast Connect users (and their encrypted passwords). This file is initialized and maintained by the command `net user`.

Note: When AIX Fast Connect is configured to use encrypted passwords, only AIX Fast Connect usernames configured to use encrypted passwords by `net user` are able to login to AIX Fast

Connect. These passwords are distinct from (and may differ from) the standard AIX passwords in **/etc/security**. When an AIX user changes their password (using **/usr/bin/passwd**), the AIX Fast Connect password for that user does not automatically change. Nevertheless, you may want to use encrypted passwords on your network to enhance network security or to simplify configuration of recent Windows clients (who assume encrypted passwords, by default).

- **To enforce Encrypted Passwords** for AIX Fast Connect, type:

```
net config /encrypt_passwords:2
```

- **To list all users** configured in **/etc/cifs/cifsPasswd**, type:

```
net user
```

- **To configure a new user for encrypted passwords**, type:

```
net user username password /add
```

-Or-

```
net user username -p /add
```

The **-p** flag prompts for a no-echo password.

- **To change a user's encrypted password**, and also update his AIX password, type:

```
net user username password /changeaixpwd:yes
```

-Or-

```
net user username -p /changeaixpwd:yes
```

- **To delete a user** from the encrypted-passwords database, type:

```
net user username /delete
```

- For security reasons, the default **/etc/cifs/cifsPasswd** file maps the client user name *root* to the server user name *nobody*. If it is desired to allow the user name *root* to map to itself (as a server user name), then this default mapping must be deleted using:

```
net user /delete root
```

Then, the user name *root* can be added as a Fast Connect user with its own encrypted password.

Basic Server Administration

You can use Web-based System Manager, SMIT, or the **net** command to manage AIX Fast Connect server operations. The following sections show basic server operations, using the AIX Fast Connect **net** command, and highlights the fast paths for SMIT at the end of the section.

Starting and Stopping the AIX Fast Connect Server

- **To load the server-daemon**, and enable PC-clients to connect, type:

```
/etc/rc.cifs start
```

- **To stop the server**, (and unload the server-daemon), type:

```
/etc/rc.cifs stop
```

Note: When the server-daemon (**cifsServer**) is not loaded, the AIX Fast Connect **net** command does not function. To configure AIX Fast Connect parameters offline, you might need to load the server daemon manually by typing **/usr/sbin/cifsServer** on the command line. This enables the **net** command to function, but does not start the server. PC clients are not able to connect until the **/etc/rc.cifs start** command is issued.

- **To temporarily reject new SMB-sessions** (without disturbing existing connections), type:

```
net pause
```

- **To re-enable the server to accept new connections**, type:

```
net resume
```

Showing Server Status Information

AIX Fast Connect provides several mechanisms for displaying current server status, including general status, configuration information, statistical information, and user-session information.

- **To query the server's operational status**, type:
`net status`
- **To show general configuration information**, type:
`net config`
- **To show statistical information** (packets delivered, etc.), type:
`net statistics`

Note: You can reset the statistics counts by typing `net statistics /reset` on the command line.

- **To query the status of logged-in user-sessions**, type:
`net session`

Web-based System Manager, SMIT fast paths, and net commands

You can use the Web-based System Manager PC Services container to administer AIX Fast Connect, or you can use the SMIT fast paths and `net` commands shown in the following table.

Administering AIX Fast Connect		
Web-based System Manager: PC Services container		
-OR-		
<i>Task</i>	<i>SMIT Fast Path</i>	<i>Command or File</i>
Starting the Server	<code>smit smbadminstart</code>	<code>net start</code>
Stopping the Server	<code>smit smbadminstop</code>	<code>net stop</code>
Pausing the Server		<code>net pause</code>
Resuming the Server		<code>net resume</code>
Changing Parameters	<code>smit smbcfghatt</code>	<code>net config</code>
Changing Resources	<code>smit smbcfgresi</code>	<code>net config</code>
Adding Users	<code>smit smbcfgusradd</code>	<code>net user</code>
Changing Users	<code>smit smbchgusrlis</code>	<code>net user</code>
Changing a User Password	<code>smit smbusrpwd</code>	<code>net user</code>
Removing a User	<code>smit smbbrmusrlis</code>	<code>net user</code>
Configuring nbns	<code>smit smbwcfgn</code>	
Listing All Shares	<code>smit smbsrvlisall</code>	<code>net share</code>
Listing All File Shares	<code>smit smbsrvfilist</code>	<code>net share</code>
Adding a File Share	<code>smit smbsrvfiladd</code>	<code>net share</code>
Changing a File Share	<code>smit smbsrvfilchg</code>	<code>net share</code>
Deleting a File Share	<code>smit smbsrvfilrm</code>	<code>net share</code>
Adding Printer Share	<code>smit smbsrvprtadd</code>	<code>net share</code>
Changing Printer Share	<code>smit smbsrvprchg</code>	<code>net share</code>
Deleting Printer Share	<code>smit smbsrvprtrm</code>	<code>net share</code>
Showing Server Status	<code>smit smbadminstatu</code>	<code>net status</code>
Showing the Configuration	<code>smit smbcfg</code>	<code>net config</code>
Showing Statistics	<code>smit smbadminstats</code>	<code>net statistics</code>

Administering AIX Fast Connect		
Showing Share	smit smbsrvlisall	net share
Getting Help	(smit help-panels)	net help

NetBIOS Name Service (NBNS)

NetBIOS Name Service (NBNS) for AIX Fast Connect provides name resolution services. It also supports some functions of Windows Internet Name Service (WINS), such as registration of multihomed name and Internet group name.

- **To activate NBNS**, type:

```
net config /nbns:1
```

- **To turn off NBNS**, type:

```
net config /nbns:0
```

Note: The *nbns* parameter is static, not dynamic. The AIX Fast Connect server must be shutdown and restarted to enable NBNS service.

Administering NBNS Tasks		
Task	SMIT Fast Path	Command or File
List all names in the NetBIOS Name Table		net nblastnames
Add a static NetBIOS Name	smit smbwcfgadd	net nbaddname /name:NBname /ipaddress:IPaddress [/sub:XX] or net nbaddgroup or net nbaddmulti
Delete a NetBIOS name in Name Table	smit smbwcfgdel	net nbdelname /name:NBname [/sub:XX]
Delete by Name and Address	smit smbwcfdadd	net nbdeladdr /name:NBname /ipaddress:IPaddress
Backup the NBNS Name Table to a File	smit smbwcfgbak	net nbbackup [/file:filename]
Restore the NBNS Name Table from Backup	smit smbwcfgres	net nbrestore [/file:filename]

Notes:

1. The value of *IPaddress* can be any number in IP address range.
2. The subcode value *XX* is any two-digit hexadecimal number in the range *00-FF*.

Configuring Client PCs for use with AIX Fast Connect

The steps shown in the following sections are required to connect a PC client to the AIX Fast Connect server.

TCP/IP Configuration

To access the AIX Fast Connect server, each client PC must be configured for NetBIOS over TCP/IP (RFC1001/1002). This can be accomplished for the various clients as shown in the following sections.

Windows 95, Windows 98 Clients

1. From the Start button, select **Settings -> Control Panel -> Network**.
2. On the *Configuration* tabbed panel (initially shown), verify that the following entries exist:

- An entry for your networking-card (hardware driver)
- TCP/IP (protocol)
- Client for Microsoft Networks (client).

If any are missing, add it from your Windows installation media.

3. Click on the TCP/IP entry and select Properties.

The TCP/IP Properties dialog box has several tabbed panels. Verify the following:

IP Address panel

Configure as needed. (For initial testing, you might want to disable DHCP and manually specify unique IP addresses for each PC.)

Bindings panel

Select Client for Microsoft Networks.

Additionally, you might want to enable WINS support, DNS support, and/or gateway support for each client. If so, configure each as needed.

4. Test the client TCP/IP configuration by **ping**-ing (by IP address) from the PC client DOS prompt to the AIX Fast Connect server, and vice versa.

Windows NT Clients

Note: You must be logged in as an Administrator.

1. From the Start button, select **S**ettings -> **C**ontrol Panel -> **N**etwork.
2. On the **Adapters** tabbed panel, verify that you have a correctly-configured entry for your networking card (hardware driver).
3. On the **Services** tabbed panel, verify that there are entries for the following services:
 - Computer Browser
 - NetBIOS Interface
 - Workstation.

If any are missing, add it from your Windows NT CD.

4. On the **Protocols** panel, add TCP/IP (if missing), then select Properties.

The TCP/IP Properties dialog box has several tabbed panels. Verify the following:

IP Address panel

Configure as needed. (For initial testing, you might want to disable DHCP and manually specify unique IP addresses for each PC.)

You might also want to configure DNS, WINS Address, and Routing.

5. Test the client TCP/IP configuration by **ping**-ing (by IP address) from the PC client DOS prompt to the AIX Fast Connect server and vice versa.

Windows 2000 Clients

Note: You must be logged in as an Administrator.

1. From the Start button, select **S**ettings -> **C**ontrol Panel -> **N**etwork and **D**ialup **C**onnections.
2. Right-click on the Local Area Connection icon of the network adapter to be configured. Select **Properties**.
3. On the **General** tabbed panel, verify that there are checked entries for the following components:
 - Your networking card (hardware driver) entry
 - Client for Microsoft Networks
 - Internet Protocol (TCP/IP).

If any is missing, add it from your Windows 2000 CD.

4. Select the TCP/IP entry, then select the Properties button. Configure as needed.
(For initial testing, you may want to disable DHCP and manually specify unique IP addresses for each PC.)
5. Select **Advanced...** -> **WINS**, to verify that NetBIOS over TCP/IP is enabled.
6. Test the client TCP/IP configuration by **ping**-ing (by IP address) from the PC client DOS prompt to the AIX Fast Connect server and vice versa.

Windows For Workgroups (Windows 3.11) Clients

1. From Network group (within Program Manager), run Network Setup.
2. Verify that the following entries exist:
 - Microsoft Windows Network (version 3.11) (network)
 - An entry for your LAN adapter card (device-driver)
 - Microsoft TCP/IP-32 3.11b (protocol)

You might need to install the TCP/IP protocol. TCP/IP is not included on the Windows 3.11 installation media. You can download a copy of Microsoft TCP/IP-32 3.11b from the Microsoft web site at www.microsoft.com.)

To set up the TCP/IP configuration, double-click on Microsoft TCP/IP-32 3.11b. Configure the IP Address, Subnet Mask, Default Gateway, WINS Server(s), DNS, and other options as needed. (**LMHOSTS** and DNS enablement are available as Advanced options.)

3. Test the client TCP/IP configuration by **ping**-ing (by IP address) from the PC client DOS prompt to the AIX Fast Connect server and vice versa.

OS/2 Clients

1. Install TCP/IP and NetBIOS support during OS/2 installation.
2. Use the TCP/IP configuration program to verify and configure TCP/IP.
3. Use the Multi-Protocol Transport Services program (MPTS) to verify and configure the following protocols for your network adapter:
 - IBM OS/2 TCP/IP
 - IBM OS/2 NetBIOS OVER TCP/IP

These protocols should have the same LAN adapter number and should match your TCP/IP interface.

Note: The default installation is IBM OS/2 NetBIOS. Be sure to add IBM OS/2 NetBIOS OVER TCP/IP if not already listed.

4. Test the client TCP/IP configuration by **ping**-ing (by IP address) from the PC client DOS prompt to the AIX Fast Connect server and vice versa.

NetBIOS Name Resolution

In addition to being able to **ping** the AIX Fast Connect server over TCP/IP, each client PC also must be able to resolve the NetBIOS name of the AIX Fast Connect server (the AIX Fast Connect *servername*) to an IP address. This can be done using UDP-Broadcast, **LMHOSTS** files, DNS, or WINS.

UDP-Broadcast (B-node)

The simplest NetBIOS name resolution (both in terms of setup and functionality) is UDP-Broadcast (B-node name resolution). No additional setup is required on the PC client as long as the client is on the same physical network segment (Ethernet, Token Ring, etc.) as the AIX Fast Connect server. The PC client broadcasts a UDP NetBIOS query to the local network, to which the AIX Fast Connect server responds.

Note: This mechanism does not work across TCP/IP routers, gateways, etc. Larger networks typically use DNS or WINS.

LMHOSTS files

Windows PCs can provide local **LMHOSTS** files for resolving NetBIOS names. Similar to **/etc/hosts** on AIX, each PC can have an **LMHOSTS** file to statically resolve NetBIOS names to IP addresses. (This mechanism might be unsuitable for DHCP environments or networks with many client PCs, because every **LMHOSTS** file must change whenever the AIX Fast Connect servers' IP addresses change.)

The following is an example of editing an **LMHOSTS** file on Windows 95, Windows 98, or Windows 3.11. From the DOS prompt:

```
DOS> cd \windows
DOS> edit lmhosts      (LMHOSTS.SAM is included with Windows as an example.)
```

On a Windows NT or Windows 2000 machine, do the following:

```
NT> cd \winnt\system32\drivers\etc
NT> edit lmhosts
```

After editing the **LMHOSTS** file, run the Windows PC command **nbtstat -R**.

DNS If your network is running the domain name service (DNS) for TCP/IP and your AIX Fast Connect *servername* is registered in the DNS, then each client PC can be configured to use DNS for NetBIOS name resolution. (This is the default on Windows 95, but must be enabled under TCP/IP Properties vfor Windows NT.)

During installation, the AIX Fast Connect *servername* defaults to match the AIX *hostname*.

WINS Your Windows network might use Windows Internet Naming Service (WINS) for NetBIOS name resolution. Similar to DNS for TCP/IP, WINS allows NetBIOS service names to be resolved to IP addresses across multiple LAN segments. When this is the case, each Client PC is configured to use the WINS server(s) under TCP/IP Properties.

Additionally, use the SMIT fast path **smit smbcfghatt** to set the WINS Address entry and Backup WINS Server for the AIX Fast Connect server. The AIX Fast Connect server uses these IP addresses to automatically register its NetBIOS *servername* with the WINS servers.

You can configure one or more AIX Fast Connect servers to act as NBNS/WINS servers. For more information, see NetBIOS Name Service.

At this point, if you have LMHOSTS, DNS, or WINS correctly configured, you should be able to **ping** from the client PC by using the NetBIOS server name.

Workgroups, Domains, and User Accounts

AIX Fast Connect supports several different types of user authentication/access mechanisms. (See Basic User Administration and Advanced Server Administration.) Each client PC should be configured to match the AIX Fast Connect user-access scheme you have chosen for your network.

For ease of use, client PCs should be in the same Windows workgroup or NT domain as the AIX Fast Connect server (or vice versa). Windows 3.11, Windows 95, and Windows NT all use WORKGROUP as a default workgroup name, and AIX Fast Connect server initializes itself to use WORKGROUP, also. If your network uses NT domain login authentication, you can configure the AIX Fast Connect server to verify AIX Fast Connect access using the NT domain authentication servers.

Whether you use Workgroups or NT domains, access to AIX Fast Connect is managed by user security. You must set up AIX user accounts for each Windows user who is accessing AIX Fast Connect. It is easiest to use if the user accounts (and passwords) on AIX match the Windows or NT domain user accounts (and passwords).

- **On the AIX Fast Connect server**, use the SMIT fast path:

```
smit smbcfghatt
```

Within the SMIT panel, enter the following:

- To use Workgroups, enter the workgroup name in the Domain Name field.
- To use NT domain validation, enter the IP addresses for the NT domain authentication server(s) in the Passthrough Authentication Server and Backup Passthrough Authentication Server fields.
- **On PC clients running Windows 95 or Windows 98**, do the following:
 1. Select **Start button -> Settings -> Control Panel -> Network**.
 2. On the **Identification** panel, enter the computer name for that PC.
 3. Configure the domain:
 - To use workgroups, enter the workgroup name in the Workgroup field.
 - To use NT domain validation,
 - a. Go to the **Configuration** tabbed panel.
 - b. Select Client for Microsoft Networks, and click on Properties.
 - c. Check the NT domain checkbox, and enter the NT domain name. (To join an NT domain, you must have Domain Administrator privileges.)
- **On PC clients running Windows NT**, make sure you are logged on as Administrator. Then:
 1. Select **Start button -> Settings -> Control Panel -> Network**.
 2. On the **Identification** panel, select the **Change...** button.
 3. Enter the Computer Name for that PC.
 4. Enter the appropriate workgroup or domain name. (To join an NT domain, you must have Domain Administrator privileges.)
- **On PC clients running Windows 2000**, make sure you are logged on as Administrator. Then:
 1. Select **Start button -> Settings -> Control Panel -> System**.
 2. On the **Network Identification** panel, select the **Properties** button.
 3. Enter the Computer Name for that PC.
 4. Enter the appropriate workgroup or domain name. (To join an NT domain, you must have Domain Administrator privileges.)
- **On PC clients running Windows for Workgroups**, configuration for the workgroup occurs during Windows 3.11 install/setup, but can be changed in the Network applet of the Control Panel.
- **On PC clients running OS/2**, configuration for the workgroup occurs during OS/2 installation, but can be changed in the DOMAIN parameter of the **IBMLAN.INI** file.

Note: Use the OS/2 command **LOGON** to use NetBIOS services such as network browsing, NET VIEW, and NET USE.

Enabling Windows Clients for Plain Text Passwords

For security reasons, Microsoft has disabled support for nonencrypted (plain text) network passwords in all recent versions of Windows (Windows 95C, Windows 98, Windows NT 4.0 w/ Service Pack 3, Windows 2000). If you want to use plain text passwords on your network, these clients must be upgraded with the following Registry patches.

Note: Microsoft has recommended the current System Registry be saved as a backup before any manual changes are made to it. For details, see Microsoft's **technet** web site.

- **To enable plain text passwords on Windows 95 or Windows 98**,
 1. Use **EDIT** or the **NOTEPAD** accessory to create the following text file, named **W98plain.reg**, as a local file on the Windows 98 machine:

REGEDIT4

; Registry file to allow plaintext passwords on Windows 98

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"EnablePlainTextPassword"=dword:00000001
```

2. Using Windows Explorer, double click on the **W98plain.reg** file name in the directory where you saved it. This action will update the Windows Registry for that client to allow plain text passwords.
 3. Shutdown/Restart the Windows 98 machine. (Shutdown/Restart is required for this patch to take effect.)
- **To enable plain text passwords on Windows NT 4.0**, logon as Administrator. Then:
 1. Use **EDIT** or the **NOTEPAD** accessory to create the following text file, named **NT4plain.reg**, as a local file on the Windows NT machine:

REGEDIT4

; Registry file to allow plaintext passwords on Windows NT 4.0

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\Parameters]
EnablePlainTextPassword=dword:00000001
```

2. Using Windows NT Explorer, double click on the **NT4plain.reg** file name in the directory where you saved it. This action will update the Windows Registry for that client to allow plain text passwords.
3. Shutdown/Restart the Windows NT machine. (Shutdown/Restart is required for this patch to take effect.)

Note: Even with the previous patch installed, all Windows NT 4.0 clients still require the user to type his/her password every time the user first connects to the AIX Fast Connect server (by browsing, mapping drives, etc.). Once the user is successfully connected, additional browsing or drive mapping operations proceed without hindrance. The initial "Password Invalid" message occurs because Windows NT 4.0 attempts to use encrypted passwords while connecting to AIX Fast Connect server, rather than plain text passwords.

- **To enable plain text passwords on Windows 2000**, logon as Administrator. Then:
 1. From the **Start** button, select **Programs -> Administrative_Tools -> Local_Security_Policy**.
 2. On the Tree view, select **Local_Policies -> Security_Options**.
 3. On the Policy list (right-hand panel), find the entry "Send unencrypted password to connect to third-party SMB servers", and enable it.
 4. Shutdown/Restart the Windows 2000 machine.

Browsing the Network

AIX Fast Connect supports browser operations such as NET VIEW and Network Neighborhood (renamed My Network Places on Windows 2000). These operations show the user a list of file and printer shares exported by each server.

Network Neighborhood can also be used as a convenient way to map drives. (Right-click on a file share name, then select Map Network Drive from the pop-up menu.)

However, note the following limitations on network browsing:

- To see the AIX Fast Connect server in Network Neighborhood, a client PC needs to be able to see the Master Browser for the workgroup or domain for which that AIX Fast Connect server is configured. (AIX Fast Connect does not act as a browse master itself, but any Windows 95, Windows 98 or Windows NT clients in that workgroup can perform this role.)

Network browsing generally works best if the client PC and the AIX Fast Connect server are in the same workgroup/domain.

- The browse list database that is maintained by the Master Browser is not always up-to-date. The list can show AIX Fast Connect server names for servers that are currently down, off, physically

disconnected, or otherwise unreachable. The Master Browser does not delete a server from the browse list until that servername's refresh timeout has expired, which can take several days. However, if a user tries to access that servername (by browsing share names, mapping drives, etc.), a disconnected AIX Fast Connect server is detected as unavailable.

Mapping Drives

Normally, PC clients must define drive mappings to use the AIX Fast Connect-exported file shares. These drive mappings can be done from Windows or from the DOS command prompt.

You can use the following mechanisms to define/undefine mappings between PC drive letters and AIX Fast Connect file shares. For the sake of the following examples, assume that the NetBIOS servername is `cifs01`, and that file share `apps` is defined.

From DOS:

```
DOS> net help                (help info for DOS)
DOS> net use H: \\cifs01\home (pre-defined AIX Fast Connect share)
DOS> net use F: \\cifs01\apps
DOS> copy F:\oldfile H:\newfile (uses the mapped drives)
DOS> net use F: /delete       (delete the drive-mapping)
```

From Windows:

1. Find the Map Network Drive dialog box.
 - Select **Windows Explorer -> Tools -> Map Network Drive**.
 - or-
 - Right-click on Network Neighborhood and select **Map Network Drive**.
2. Select the desired drive from the Drive: drop-down list, then
 - Enter the path: (for example, `\\cifs01\apps`).
 - or-
 - Use the Shared Directories (browse tree) panel to select the network share desired.

Using AIX Fast Connect Printers

For printing, DOS and Windows mappings are somewhat different. For the following examples, assume that AIX Fast Connect server `cifs01` has print shares `netprint1` and `pscolor` defined.

For DOS applications, the following simple device-mappings can be used:

```
DOS> net use LPT1: \\cifs01\netprint1
DOS> net use LPT2: \\cifs01\pscolor
```

To test these DOS printer-mappings, use the following:

```
DOS> COPY text_file LPT1:
DOS> COPY Postscript_file LPT2:
```

Note: During print-spooling, neither DOS nor AIX Fast Connect auto-convert Postscript to text, or vice versa; however, this auto-detection/auto-convert feature can be enabled using AIX print-spooling options, if desired.

For Windows applications, a Windows printer driver needs to be installed, and mapped to the network printer, as follows:

1. Select **Start -> Settings -> Printers -> Add Printer**.
2. Select **Network Printer**.

3. Enter the AIX Fast Connect print share name (for example, \\cifs01\netprint1) or use the browse list to select the desired print share.
4. Select the correct Windows printer driver for that network printer (for example, IBM 4039 Laser Printer PS), which is installed from your Windows installation disks.

Test Windows printer-driver functionality by printing a test-file from any Windows application (for example, Notepad), or by using the "Print Test Page" feature as follows:

1. Select **Start -> Settings -> Printers**.
2. Select the printer driver (for example, pscolor).
3. From the Menu Bar, select **File -> Properties**.
4. From the Tabbed-panel labelled "General", select **Print Test Page**.

For Windows 3.11, install the desired printer driver through Control Panel, and use the Connect... button to map it to the AIX Fast Connect print share.

Support for Windows 2000 Clients

Windows 2000 clients are now supported using CIFS/SMB protocol on NetBIOS over TCP/IP. Network Logon is not yet supported for Windows 2000 clients.

Support for Windows Terminal Server

AIX Fast Connect is compatible with the Windows Terminal Server program. This program allows multiple PC clients running Windows Terminal Client software to login to the Windows Terminal Server and establish a remote console session. Any network drive (or network printer) mapping made within that console session gets forwarded by Windows Terminal Server to other NetBIOS servers, as required.

Windows Terminal Server (and other similar terminal-server programs) must accommodate multiple net-mappings by multiple usernames, coming from multiple client PCs. Windows Terminal Server (and other terminal servers) can multiplex these requests to AIX Fast Connect using the following mechanisms:

- Multiple TCP/IP sessions (from a single Windows Terminal Server PC) to AIX Fast Connect.
- Multiple SMB sessions multiplexed into a single TCP/IP session.

Both of these mechanisms are supported by AIX Fast Connect version 3.0 and later.

To enable Windows Terminal Server support, set **multiuserlogin=1**.

Refer to your Windows Terminal Server documentation for specific information about setup and use of Windows Terminal Server and Windows Terminal Client.

Advanced AIX Fast Connect Features

This section discusses advanced AIX Fast Connect features used for customized configurations. See AIX Fast Connect Configuration and Administration for basic administrative procedures.

Note: Several of the features described in this section cannot be used simultaneously.

AIX Fast Connect supports the following advanced features:

- User name mappings from client PC user names to AIX-server user names.
- User-session management using **net session**
- Resource management
- Per-share options
- Support for AIX JFS ACLs
- Support for DOS (8.3) filenames (for Windows 3.11 clients, etc.)

- Servername aliases for HACMP-support

Several performance considerations for AIX Fast Connect are also discussed in this section.

Many choices for the above features depend on the authentication method selected. Each type has its advantages and disadvantages. Which authentication method or methods you choose depends on your environment, your administration policy, and the ease of administration and use. The following methods for user authentication are described in detail in this section:

- AIX-based user authentication using plain text CIFS passwords.
- CIFS password encryption protocols
- NT Passthrough authentication
- Network Logon to AIX Fast Connect (using IBM Network Client for NT-clients)
- DCE/DFS authentication (using plain text passwords)
- Guest Logon
- Share-level security

AIX-based User Authentication (Plain Text Passwords)

AIX-based authentication uses AIX user definitions and passwords. All AIX authentication grammars are supported, including DCE and LDAP. Following session setup, a AIX Fast Connect session gets the authenticated AIX user's credentials (UID, GID, and secondary groups).

The following requirements apply:

- Clients must be able to negotiate plain text passwords. This might require updating registry entries on all Windows NT, 95, and 98 clients. (See Enabling Windows Clients for Plain Text Passwords.)
- AIX Fast Connect must be enabled for plain text passwords. To do this type:


```
net config /encrypt_passwords:0
```

Plain text passwords have the following advantages:

- Low administrative overhead. (It uses existing AIX user information.)
- AIX tools for managing users can be used.

Plain text passwords have the following disadvantages:

- Windows registry update might be required, on a per-client basis.
- Windows might require user ID and passwords to be retyped, on a per-SMB-login basis.
- Clear-text passwords are sent over the network.

Note: SMB networking does not support mixed case for plain text passwords. Every AIX user accessing AIX Fast Connect must have all uppercase or all lowercase AIX passwords.

CIFS Password Encryption Protocols

The CIFS password encryption protocol method uses AIX Fast Connect user definitions and encrypted passwords for user authentication. Each user must be defined under the same user name as an AIX user as well. AIX Fast Connect encrypts passwords and saves them in its user database (*/etc/cifs/cifsPasswd*) for use during session setup. (See Configuring Encrypted Passwords.) Following session setup, a AIX Fast Connect session gets the authenticated user's credentials (UID, GID and secondary groups).

CIFS password encryption protocol method has the following requirements:

- Users must be defined to AIX Fast Connect using Web-based System Manager, SMIT, or the **net user** command.

(User passwords do not have to be the same on both systems.)
- AIX Fast Connect must be enabled for encrypted passwords. To do this, type:

```
net config /encrypt_passwords:2
```

- Changing AIX Fast Connect passwords requires root authority.

This method has the following advantages:

- No additional logon, beyond logging into the Windows or OS/2 workstation, is required.
- Clear text passwords are not sent over the network, which provides additional security.

This method has the following disadvantages:

- Additional administrative tasks are needed for AIX Fast Connect users.
- Root authority is needed to update passwords in AIX.

NT Passthrough Authentication

This authentication method uses AIX user definitions and NT server user authentication. In this mode, each AIX Fast Connect user must also be defined as an AIX user. Passthrough authentication is enabled using Web-based System Manager, SMIT, or the **net** command by specifying an IP address for the NT Passthrough Authentication Server. To configure this mode using the **net** command, type:

```
net config /passthrough_authentication_server:IPaddress
```

You can also designate a backup server for NT authentication by typing:

```
net config /backup_passthrough_authentication_server:IPaddress2
```

During session setup, AIX Fast Connect forwards the session setup request to the NT server. If the NT server authenticates the user, AIX Fast Connect grants access. Following session setup, a AIX Fast Connect session gets the authenticated user's credentials (UID, GID and secondary groups).

Passthrough authentication has the following requirements:

- User must be defined on the passthrough authentication server.
- AIX Fast Connect must be enabled for passthrough authentication.
- NT user name must match AIX user name, although passwords can be different.

This method has the following advantages:

- No additional logon, other than logging into the Windows or OS/2 workstation is required.
- Clear text passwords are not sent over the network, which provides additional security.
- Uses NT user definition, therefore less administrative overhead is needed.

This method has the following disadvantage:

- Requires an NT authentication server, which must be a secure system.

Notes:

- If passthrough authentication fails to authenticate a AIX Fast Connect user, user authentication continues with normal authentication on the AIX Fast Connect server. Depending on the value of the **encrypt_passwords** option, the server attempts to authenticate the PC client using either plain text or encrypted passwords.
- When passthrough authentication is enabled, guest logon support cannot work. These options are mutually exclusive. Disable guest logon by typing:

```
net config /guestlogon:0
```

- When passthrough authentication is enabled, AIX Fast Connect's network logon feature cannot work. These options are mutually exclusive. (Frequently, the external NT authentication server is also acting as a Network Logon server, or even a Primary Domain Controller for NT-domains.) Disable AIX Fast Connect's network logon feature by typing:

```
net config /networklogon:0
```

Network Logon to AIX Fast Connect

AIX Fast Connect can be configured to act as a Network Logon server. In this mode, Windows-based PCs are configured for Network-Logon, rather than Local-Logon, which provides the following benefits:

Network Password

Each PC user can log in to any network workstation using his network password, without having separate Local-Logon passwords per workstation.

Startup Scripts

During network login, startup scripts can be executed from the Network Logon server, based on user name and workstation name.

Roaming Profile

After network login, each PC user's desktop environment is automatically initialized to the correct network settings, regardless of which workstation that user is using.

Home Directories

After network login, each PC user's "home directory" is available, regardless of which workstation that user is using.

The following restrictions apply to AIX Fast Connect's network logon feature:

- Windows 95 and Windows 98 clients can use either:
 - Microsoft Client for Microsoft Networks, or
 - IBM Client for IBM Networks
- Windows NT clients must use the IBM Primary Logon Client for Windows NT.
- Windows 2000 clients are not supported by this method.
- Passthrough-authentication to NT must be disabled.

AIX Fast Connect's Network Logon feature is enabled (or disabled) using the *networklogon* parameter. This feature has multiple configuration settings, many of which are rarely used. For more information, see *Configuring Network Logon for AIX Fast Connect*.

DCE/DFS Support

AIX Fast Connect can be configured to provide access to DFS for Windows clients. Each AIX Fast Connect user name is used as a DCE principal name. Mixed case user names or passwords are only supported in encrypted passwords are used..

DCE support is automatically installed if the DCE filesets are installed *before* installing AIX Fast Connect. (**cifsUserProc** is then linked to **cifsPrintServerDCE** rather than **cifsPrintServer**.)

DCE support is controlled through the **dce_auth** configuration option, which can be set to 0 or 1. A value of 1 indicates that DCE authentication option is enabled. When **dce_auth=1** (and **cifsPrintServerDCE** is being used), all incoming PC client logins are sent to DCE for authentication. This requires plain text passwords, and all PC-client user names and passwords must also be valid DCE user names and passwords. (UID, GID, and groupset are defined by the DCE authentication.)

When **dce_auth=0**, AIX Fast Connect can still provide some access to DFS files:

- If AIX-based authentication is being used (plain text passwords), then all AIX accounts configured for Integrated Login to DCE are allowed DCE-authenticated access to DFS when connecting to AIX Fast Connect.
- In all other cases, AIX Fast Connect users are allowed non-authenticated access to DFS, using the *any_other* ACL.

Notes:

- When DCE integration is enabled and the user's AIX UID is different from DCE UID, the user might not have the same access rights as an AIX login shell.
- Older versions of AIX Fast Connect (prior to 2.1.1.20) required that the root user be logged in as the cell administrator when loading the **cifsServer** daemon, for access to certain files. This is not a restriction in the current version of AIX Fast Connect.
- DCE/DFS authentication (**dce_auth=1**) is mutually exclusive with NT Passthrough authentication.
- DCE/DFS authentication (**dce_auth=1**) is mutually exclusive with the guest logon feature.

Guest Logon

AIX Fast Connect can support guest-mode logins when configured for either plain text or encrypted passwords. To enable guest-mode logins, two parameters must be configured:

```
net config /guestlogonsupport:1 (enables guest logons)
net config /guestname:GuestID (AIX guestid with null password)
```

When guest logon support is enabled (**guestlogonsupport=1**), and the **guestname** field is set, non-AIX users can connect to the AIX Fast Connect Server. The credentials for guest clients is set to those of the **guestname** attribute.

The AIX account specified by **guestname** must have a null AIX password — it is being used for guest-mode access to the AIX file-system. This guest account can access all of the file system directories exported by AIX Fast Connect (as File Shares); therefore, this guest account should probably be in its own unique AIX group, to simplify access control.

Guest access is only given to user names that are *not* defined AIX Fast Connect users with passwords that are *not* null.

Incoming login requests are authenticated as follows:

1. If the incoming user name is recognized as a valid user, then the password is checked. If the password is correct, then standard user-mode access is granted; otherwise, the login attempt fails.
2. If the incoming user name is *not* recognized as a valid user, then the password is checked. If the password is not null, then guest-mode access is granted; otherwise, the login attempt fails.

To disable guest logon support, type:

```
net config /guestlogonsupport:0
```

Note:

- When guest logon support and encrypted passwords are both enabled, the **guestname** user does not have to be added to the AIX Fast Connect user database (**/etc/cifs/cifsPasswd**), but still must have a null AIX password.
- Guest logon support *does* cooperate with Network Logon support (**networklogon=1**). Whenever guest-mode access is granted, then the profile, startup scripts, and home directory of the **guestname** user are used for the network logon.
- If **dce_auth=1**, guest logon support does not work.
- If NT-passthrough authentication is configured, guest logon support does not work.
- If **share_level_security=1**, guest logon support does not work.

Share-Level Security

When the AIX Fast Connect server is configured for share-level security, then passwords are associated with individual file and print shares, not with PC client user names. In this mode, AIX Fast Connect provides access rights to PC clients based on a share-mode user name specified as the **share_level_security_username** parameter, similar to the guest logon access mode.

Note: When share-level security is enabled, all user-level authentication mechanisms are disabled.

To enable share-level security, type:

```
net config /share_level_security:1 (enable share-level security)
net config /share_level_security_username:AIXuser (configure share user)
```

In share-level security mode, AIX Fast Connect supports both ReadWrite passwords and ReadOnly passwords. When a PC client tries to connect to a share, the following can occur:

1. If that client provides the ReadWrite password for a share (or if that share's ReadWrite password is null or undefined), then that client is granted ReadWrite access to the share.
2. If that client fails to get ReadWrite access, but provides the ReadOnly password for a share (or if that share's ReadOnly password is null or undefined), then that client is granted ReadOnly access to the share.

Note: These access modes are also affected by the access credentials of the *share_level_security_username* for that share, and by the **mode** share option, both of which can effectively change ReadWrite access to ReadOnly access.

- To create a NETTEMP share with a ReadWrite password of write-is-okay, type:

```
net share /add /netname:NETTEMP /path:/tmp /rw_password:"write-is-okay"
```

- To create a USERS share with both ReadWrite and ReadOnly passwords, type:

```
net share /add /netname:USERS /path:/home /rw_password:writeme /ro_password:readme
```

Note: Specifying a ReadOnly password without specifying a ReadWrite password normally allows all clients to get ReadWrite access (if the ReadWrite password is null).

- To disable share-level security (to use other user-authentication mechanisms), type:

```
net config /share_level_security:0
```

- If Windows Terminal services is used with Share Level Security, (**multiuserlogin=1 AND share_level_security=1**), then only the *first* user that connects to a share will prompt for the share's password — all successive users that connect to that share will not be prompted for a password (and no password will be sent to the server, even if specified). This is a problem with Windows Terminal Services. See Microsoft KnowledgeBase article Q260853 for more information.

User Name Mappings

This feature allows AIX Fast Connect to map PC client user names (or *sets* of PC client user names) to server (AIX) user names, for purposes of user-mode authentication and file access. When enabled, AIX Fast Connect tries to map every incoming client user name to a server user name, and then uses that server user name for further user authentication and AIX credentials. (All user-authentication mechanisms are supported: AIX-based, encrypted passwords, NT-passthrough, DCE, ...)

This feature is controlled by the *usernamemapping* parameter, and mappings are configured by the **net user /map** command.

- To enable the user name mappings feature, type:

```
net config /usernamemapping:1
```

- To define a mapping from *longclientname* to *aixname*, type:

```
net user /map longclientname aixname
```

- To define a *second* mapping to that same AIX user, type:

```
net user /map secondclientname aixname
```

- To delete a mapping, use the *client* user name, like:

```
net user /delete longclientname
```

- To disable this feature, type:

```
net config /usernamemapping:0
```

Notes:

- PC client usernames are restricted to 20 characters.
- When username mapping is enabled, the username *root* is mapped to the username *nobody* by default. This mapping can be changed.
- After mapping a client username *XXXX* to an AIX server username, then that client username cannot be defined as a *server* username (with its own unique encrypted password) until that username mapping is deleted by **net user/delete**.
- When username mapping is enabled, the username *root* is mapped to the username *nobody* by default. This mapping can be changed. If it is desired to allow the username *root* to map to itself (as a server user name), then this default mapping must be deleted with **net user/delete root**.

AIX Fast Connect User Management and File Access

AIX Fast Connect provides several additional features for file access and user management, which are described in the following sections.

User-Session Management Using **net session**

AIX Fast Connect supports the **net session** command, for displaying and managing logged-in user sessions.

- **To display all connected user sessions**, type:
`net session`
- **To display all share resources** currently mapped by a specific session, type:
`net session /user:username /workstation:IPaddress /shareinfo`
- **To display all open files** for a specific session, type:
`net session /user:username /workstation:IPaddress /fileinfo`
- **To abort a user's session**, type:
`net session /user:username /workstation:IPaddress /close`
- **To close a user's share-mapping**, type:
`net session /user:username /workstation:IPaddress /close /netname:sharename`
- **To close a user's file**, type:
`net session /user:username /workstation:IPaddress /close /file:filename`

Note: The *workstation* parameter works with NetBIOS names, also.

Establishing Resource Limits

AIX Fast Connect provides several parameters to specify limits on resource use:

maxusers	Maximum number of user-sessions (logins), at any given time
maxconnections	Maximum number of connections to a single share-resource
maxopens	Maximum number of open files allowed
maxsearches	Maximum number of open file-searches
autodisconnect	Autodisconnect time for idle sessions (in minutes)

See the **net config** command, or the Table of Configurable Parameters for the **net** Command, for more details.

Changing the *umask*

AIX Fast Connect provides a global parameter *umask* to control permission bits on all files created by all AIX Fast Connect users. The *umask* parameter is specified as an octal number (with a leading zero), and defaults to 022.

To change the `umask` to 002, type:

```
net config /umask:002
```

Specifying Per-Share Options

Several advanced features of AIX Fast Connect are available as per-share options. These options are encoded as bit fields within the `sh_options` parameter of each share definition. These options must be defined when the share is created with the `net share /add` command.

Per-share options currently allowed by `net share /add` are:

<i>parameter</i>	<i>values</i>	<i>default</i>	<i>description</i>
<code>sh_oplockfiles</code>	(0,1)	1	Enables oplocks on this share, if oplockfiles=1
<code>sh_searchcache</code>	(0,1)	0	Enables search caching on this share, if cache_searches=1
<code>sh_sendfile</code>	(0,1)	0	Enables SendFile API on this share, if send_file_api=1
<code>mode</code>	(0,1)	1	Allows ReadWrite access to this share. (0 implies ReadOnly mode.)

Example: To create a ReadOnly share that has SendFile enabled, type:

```
net share /add /netname:ROSHARE /path:/usr/etc /mode:0 /sh_sendfile:1
```

Support for AIX JFS ACLs

AIX Access Control Lists allows extended control of files and directories of AIX Journaled File System. AIX Fast Connect exploits this features by honoring AIX ACLs. AIX 4.3.3 adds graphical manipulation of ACLs using CDE `dtfile` application.

AIX Fast Connect extends this support by implementing ACL inheritance for AIX Fast Connect file shares. This feature can be used to implement default ACLs for created file objects. When **acl_inheritance** is enabled, the `umask` parameter is not effective.

ACL inheritance is enabled by setting the **acl_inheritance** option to 1. This option can be viewed and changed using the **net config** command. Once enabled, it applies to *all* the AIX Fast Connect file shares.

ACLs are inherited from the ACL defined on the base directory of the share. For example, if you have a share named TEMP mapped to the AIX directory `/tmp` (assuming a valid ACL is defined for this directory and `acl_inheritance=1`), all files created in this share now inherit the ACLs defined for `/tmp`.

- **To enable ACL inheritance** for all AIX Fast Connect file shares, type:

```
net config /acl_inheritance:1
```

- **To disable ACL inheritance** for all AIX Fast Connect file shares, type:

```
net config /acl_inheritance:0
```

- **To view the current setting** of the `acl_inheritance` flag, type:

```
net config /parm:acl_inheritance
```

Sending Messages to Clients

When necessary, the AIX Fast Connect administrator can use the **cifsClient** command to send messages to individual workstations, or to all user-sessions connected to AIX Fast Connect.

- **To send a message to all users** connected to AIX Fast Connect, type:

```
cifsClient send -a -m "message"
```

- **To send a message to a specific computer**, type:
`cifsClient send -c computer -m "message"`
- **To send a message to a specific connected user**, type:
`cifsClient send -u username -m "message"`
- **To send a message to a NetBIOS domain**, type:
`cifsClient send -d domainname -m "message"`

Notes:

- A file may be sent as the message using the `-f filename` option, or the message can be read from standard input.
- The `domainname` is optional. The default domain is the AIX Fast Connect server's domain.
- The target computer must be enabled to receive messages, using messaging software. On Windows NT clients, the messaging service is started by default. To start the messaging service on Windows 95, 98, or 3.11, run:
`WIN95> winpopup`
- When share-level security is enabled (`share_level_security=1`), then the user-specified messaging command "`cifsClient send -u username`" is not supported.

Mapping Long AIX File Names to 8.3 DOS File Names

Older PC client operating systems, such as Windows for Workgroups 3.11, do not support long filenames. Also, this restriction is true for many older (16-bit) applications running under Windows 95, Windows 98, and Windows NT. This restriction requires mapping long names of AIX files to DOS file name format. (The DOS format is also called *8.3* format because file names are limited to a maximum of eight characters followed by a period and a three-character extension.)

Simply truncating a long name to a shorter name is not the solution, because multiple files could get mapped to the same name whenever the first eight characters are same. AIX Fast Connect maps AIX file names (AFN) to DOS File Names (DFN) ensuring file name uniqueness. It maps AFNs to DFNs using Microsoft Windows NT method for mapping names (that is, name conflicts are handled by using a delimiting character in the short name followed by a unique numeric to make the name unique).

For example, consider two files in the root directory of an exported SMB share: `LongFileName1.txt` and `LongFileName2.txt`. Assume a Windows 3.11 client mounts this share and searches the directory. The resulting filenames are:

`LONGFI~1.TXT` for `LongFileName1.txt`

`LONGFI~2.TXT` for `LongFileName2.txt`

AIX Fast Connect generates a mapped name whenever the AFN needs to be passed back to a DOS client. DFNs generated by AIX Fast Connect are not remembered across server restarts. Filename mappings remain consistent until the AIX Fast Connect server is restarted.

AIX Fast Connect has a configuration option to turn off the mapping. When the mapping is turned off, no mapping is attempted. When disabled, any mapping of long names must be done by the PC client software.

- **To enable filename mapping** (default), type:
`net config /dosfilenamemapping:1`
- **To disable filename mapping**, type:
`net config /dosfilenamemapping:0`

Notes:

- AFN-to-DFN mapping might not map correctly if the server restarts. Given the previous example, assume a user on a Windows 3.11 client opens LONGFI~1.TXT, edits it, and saves the changes. Then the server shuts down. Someone then removes LongFileName1.txt from the server file system. Once the server is up and running, the user on the client again edits LONGFI~1.TXT. This time, however, the same file maps to LongFileName2.txt, not the previously deleted file name, and the client ends up editing the wrong file. To prevent this situation, after the network drive is reconnected following server restart, new file lists must be obtained before accessing any mapped names.
- If your site does not need this feature, turn **dosfilenamemapping** off (0) to reduce memory and CPU usage and thereby improve performance.

Support for DOS File Attributes

AIX Fast Connect provides optional support for the ReadOnly, Archive, System, and Hidden file attribute bits of DOS files. These bits are encoded by AIX Fast Connect into the AIX file permission bits of the AIX file system.

- The ReadOnly attribute is encoded by turning *off* the AIX User/Group/Other Write bits. (**chmod a-w filename**)
- The Archive attribute is encoded by turning on the AIX *User* Execute bit. (**chmod u+x filename**)
- The System attribute is encoded by turning on the AIX *Group* Execute bit. (**chmod g+x filename**)
- The Hidden attribute is encoded by turning on the AIX *Other* Execute bit. (**chmod o+x filename**)
- For directories, AIX Fast Connect does not support the Archive, System, or Hidden attributes — only the ReadOnly attribute is supported. (AIX directories use the Execute bits to allow "change directory" permission, so AIX Fast Connect does not use these bits on exported directories.)

AIX Fast Connect automatically handles these bits in the AIX file system; the examples listed above simply show how AIX Fast Connect interprets these AIX-permission bits, when reporting DOS file attributes to a PC client. If you have AIX Fast Connect configured to support DOS file attributes (the default), then you might need to manually turn *off* the Execute bits in your AIX directories that are being exported as AIX Fast Connect file shares.

- **To clear the Execute bits on files (in an entire *dirname* tree)**, so that these files are not listed as "System" or "Hidden", type:

```
find dirname -type f -exec chmod a-x "{}" ";" -print
```

- **To disable support for Archive, System, and Hidden bits**, type:

```
net config /dosattrmapping:0
```

Specifying NetBIOS Aliases for HACMP support

AIX Fast Connect supports server name aliases, which allows a AIX Fast Connect server to respond to multiple NetBIOS server names. This feature is helpful in HACMP mutual takeover. Server aliases can be configured using the **net name** command, as described below.

- **To show the primary AIX Fast Connect servername**, type:

```
net config /parm:servername
```

- **To list *alias* servernames**, type:

```
net name /list
```

- **To add an alias servername** (for example, *sname2*), type:

```
net name /add sname2
```

- **To delete an alias servername** (for example, *sname2*), type:

```
net name /delete sname2
```

Server aliases normally use NetBIOS subcodes 0x00 and 0x20, but other subcodes can be specified, for example:

```
net name /add test3 /sub:03
net name /delete sname2 /sub:2f
```

Notes:

- Whenever adding or deleting an alias name without specifying a subcode, or if subcode 0x00 or 0x20 is specified, the alias name is added or deleted with subcodes 0x00 and 0x20.
- **net name /list** uses angle-brackets ("*<*", "*>*") to show subcodes other than 0x00 and 0x20.
- To register alias name(s) to WINS or NBNS (including the local NBNS), the IP address of the WINS or NBNS server needs to be specified in parameters *primary_wins_ipaddr* or *secondary_wins_ipaddr*.
- When adding an alias name:
 - If someone on the same subnet is currently holding the name, adding fails.
 - If no one on the same subnet is holding the name, but it exists in name table of the NBNS, then the name cannot be registered to the NBNS, but is still added to the local name table.

Performance Considerations

This section discusses several issues affecting AIX Fast Connect performance.

Large Directories

Directory enumerations are frequent network operations on Windows clients. Whenever Network Neighborhood (or Windows Explorer) opens a network directory, that entire directory is enumerated over the network, for display in a Explorer-window. Usually, Windows Explorer waits to display the contents of the window until the entire network directory has been listed. For large directories containing many files, this delay is noticeable to the PC user, and can be frustrating. Remote file accesses from AIX (such as DCE/DFS or NFS) tend to aggravate this situation.

Try shielding your AIX Fast Connect users from having to access large directories to get to the network files they need. One possible solution is to define smaller-sized AIX directories to be exported by AIX Fast Connect. These directories can contain links to files in the large directories.

If large directories are needed but rarely change (for example, CD-ROM), then you might find the search caching features useful.

Search Caching

Directory searches are very frequent network operations on Windows clients. Every time a network file is opened, or renamed, or deleted, or listed, a directory search for that filename is performed. (For example, simply opening a document in Microsoft Word can cause multiple directory searches for that filename.)

AIX Fast Connect has a search-caching feature that allows directory searches to be temporarily cached to improve the performance of multiple-search scenarios like opening documents, as mentioned above. Also, for directories that change infrequently, but are accessed often, this feature enhances performance.

Search-caching is implemented in AIX Fast Connect by taking snapshots of directories and their modification times.

1. When AIX Fast Connect needs to perform a directory search, AIX Fast Connect first checks its search cache (if enabled).
2. If a search-cache entry is found, it is first validated. If that directory's current modification time is different than the cached time, the the feature determines the cache entry is invalid.
3. Whenever the search-cache table gets full, older entries are deleted, to make space for new entries.

Search caching is configured on AIX Fast Connect by several parameters:

parameter	default	description
cache_searches	0 (disabled)	Globally disable the search-caching feature. (Set to 1 to enable.)
sh_searchcache	0 (disabled)	Disable search caching on a per-share basis. (Set to 1 to enable.)

Note: To enable search caching on any file shares, the *cache_searches* parameter must be enabled (set to 1), and *sh_searchcache* must be enabled for every file share for which search caching is desired.

SendFile API support

For file transfers to clients, AIX Fast Connect can use the SendFile API for performance enhancement. The SendFile API is an AIX kernel extension that provides efficient file transfers and can do data caching.

SendFile API is configured on AIX Fast Connect by several parameters:

parameter	default	description
send_file_api	1 (enabled)	Flag to enable/disable the SendFile API to be used by AIX Fast Connect. Default is enable. To disable SendFile, set to 0.
send_file_cache_size	0 (disabled)	Maximum Read-Request size that is cached by the SendFile API.
send_file_size	4096	Minimum Read-Request size, before SendFile API is used.
sh_sendfile	0 (disabled)	Flag to enable/disable per-share option. Default is disable. To enable SendFile for that file share, set to 1.

Notes:

- To enable SendFile API on any file shares, the *send_file_api* must be enabled, and *sh_sendfile* must be enabled for every file share for which the SendFile API is desired.
- See the **no** command for system-wide SendFile configuration parameters.

AIX Fast Connect Problem Determination

Traces

The AIX Fast Connect server comes with the ability to create AIX trace files to isolate problems. When a trace facility is active, information about selected events is recorded in the trace file. To obtain trace files, you must have the trace command installed on your machine. The trace command is in the **bos.sysmgt.trace** package.

The following trace hooks are used by the AIX Fast Connect server:

2EE	CIFS Enter
2EF	CIFS Exit
2F0	CIFS-FSS
2F1	CIFS-LOGON
2F2	CIFS-NET
2F3	CIFS-SMB PARSER
2F4	CIFS-PSS
2F5	CIFS-SMS

Trace files can be created by using either through SMIT or the command line.

Using SMIT:

1. Type the **smit trcstart** fast path on the command line.
2. Select the CIFS hooks for ADDITIONAL event IDs to trace field, then exit SMIT. This creates a trace file named **trcfile** in the **/var/adm/ras** directory (default).
3. Recreate the problem.
4. Then type **smit trcstop** on the command line.
5. Exit SMIT.
6. Type **smit trcrpt** on the command line and select the output format. It displays the trace file into readable format.

Using commands:

1. Type the following on the command line:

```
trace -a -j 2EE,2EF,2F0,2F1,2F2,2F3,2F4,2F5 -o /tmp/cifs.trace
```

This creates a trace file named **cifs.trace** in the **/tmp** directory.

2. Recreate the problem.

3. Type:

```
trcstop
```

4. Type:

```
trcrpt -t /etc/trcfmt /tmp/cifs.trace
```

The **trcrpt** command formats the trace file into readable text and writes a report to standard output.

Logs

The AIX Fast Connect server writes information and error messages to a file in **/var/cifs** named **cifsLog**.

Troubleshooting Connection Problems

Cannot connect to server.

```
access is denied
password is invalid
password is not correct
not authorized to login
```

Check that the server has `passthrough_authentication_server` enabled. When you get one of these error messages on the client PC, try the following:

- Enter the correct password.
- Check logon user ID and its password on clients that should have an account on the AIX server. Log clients off and on with correct user ID and password.
- For clients with Window NT with Service Pack 3 installed, the NET VIEW command returns `access is denied`. See Enabling Windows Clients for Plain Text Passwords for more information.

Note: AIX Fast Connect does not support mixed-case passwords when **encrypt_passwords=0**.

When you get this error message on the client PC, try the following:

- Check the NetBIOS name of the AIX Fast Connect server.
- Check server status.
- See the Connection Checking Procedure.

System error 53 has occurred.
The network path was
not found.

When you get this error message on the client PC, check server status. It might be paused.

System error 51 has
occurred. The remote
computer is not
available.

Connection Error when using a Passthrough Server

- Be sure the *passthrough_authentication_server* parameter is set to the IP Address (rather than the hostname) by typing **net config /parm:passthrough_authentication_server**.
- Test the network connection by pinging this machine. See Connection Checking Procedure.
- Check that the *networklogon* or *guestlogonsupport* option is not being used. These options are mutually exclusive with passthrough authentication.

Cannot view Network Neighborhood from Entire Network on Windows clients

Each Windows Workgroup must have a master browser present for network browsing to work properly. By default, any Windows NT, Windows 95, or Windows 98 client is set up to act as a master browser. The *domainname* parameter on the AIX Fast Connect Server determines which workgroup this AIX Fast Connect server is a member of. AIX Fast Connect does not function as a master browser.

Windows Primary Domain Controller reports that it cannot be started when AIX Fast Connect is running:

If the *networklogon* parameter is set to 1, the AIX Fast Connect server acts as the Logon Server for Windows 95/98/NT clients. Set this parameter to 0 if you do not want this behavior.

Client reports

Account is not authorized to logon from this station

This error message occurs when AIX Fast Connect is configured for plaintext passwords but the client has not been configured to support plaintext passwords. See Enabling Windows Clients for Plain Text Passwords.

Server reports

Net:connect: A remote host refused an attempted connection, Can't start server: Operation could not be performed, or similar message.

- This usually means the server cannot be started. Ensure the server has not started by using **ps -ef | grep /usr/sbin/cifs**
- Check for other services using the NetBIOS port (**netstat -an | grep 139**)
- If services are found, then this problem was caused by the current installation of an application using the NetBIOS ports (AIX Connections, SAMBA, etc.). The intruding application must be removed so the port can be made available to AIX Fast Connect.
- Additionally, check to be sure you have sufficient disk space available in the */var* file system by using the **df /var** command, and that sufficient paging space is available and active using the **lsps -a** command.

Guest user cannot logon, Client reports Unknown user or password or similar error.

- Ensure *guestlogonsupport* is set to 1.
- Ensure *guestname* is set to a valid AIX user.
- Ensure that *guestname*'s AIX password is null.
- Additionally, check that **dce_auth** and **passthrough_authentication_server** are not being used. These are mutually exclusive with the guest logon option.

Client reports The credentials supplied conflict with existing credentials or similar error.

Client must logout and relogin with the user ID granting the desired access on the server. This usually happens when one client attempts to access the same server as two different users.

Client reports that it cannot create a file on the server.

- In most cases this is an AIX permissions problem. Check the AIX share path to ensure desired permissions are set. (Also, if *acl_inheritance* is set to 1, examine the AIX ACLs using the **acledit** or similar command.
- If permissions are not the problem, check that the file system where the share exists has enough space using the **df [share path]** command.
- The administrator might want to log on to the AIX machine as the user who is having problems and attempt to create a file in the path that is causing problems. AIX might provide a more descriptive error message.

Printing from client results in garbled printout

Some AIX back-end printer drivers add controls to the file that is being printed; others do not. Windows clients always send print jobs in a format that needs no controls. So, if your AIX printer driver adds controls, set the **-o -dp** printer share options when you create the printer share.

Technical Service Information

If you need to contact technical support, the following information can help them diagnose your problem.

1. Your machine type
2. Output from **oslevel** command - Operating System Level
3. Output from **netstat -an** command - Network Information
4. Output from **lsps -a** command - Paging Space Information
5. Amount of memory on the machine.
6. **/etc/cifs/cifsConfig** - Server Configuration File
7. **/var/cifs/cifsLog** - Server Error Log
8. Output from **lspp -l** command - Software Installed
9. Full output from **errpt** and **errpt -a** commands - System Errors
10. Output from **ps aux** and **ps -efl** commands - Process Listing
11. Output from trace.

Additionally, it might be helpful to have full core enabled, especially in the rare event that the AIX Fast Connect server crashes. To enable a full core, use the command **chdev -l sys0 -a fullcore='true'** and ensure you have plenty of space in your root (*/*) filesystem.

Connection Checking Procedure

1. **ping** the AIX Fast Connect server by IP address. If timeout occurs, check:
 - cable for physical connection
 - status of the AIX machine
 - TCP/IP configuration on clients and on the AIX server.
2. **ping** the AIX Fast Connect server with its NetBIOS name. If it fails, refer to NetBIOS Name Resolution for more information.
3. Check server status on the AIX machine using **net config**, **net status**, and **net statistics** commands.

Configuring Network Logon for AIX Fast Connect

AIX Fast Connect can be configured to support Network Logon. Network Logon support allows centralizing the user accounts, startup scripts, home directories, and configuration policy of Windows systems participating in a workgroup to a single AIX system running the AIX Fast Connect server. This support does not allow an AIX Fast Connect server to act as a Windows NT Domain Controller. However, with the IBM Networks Client software, both NT and Windows 95/98 clients can be configured to perform network logon to an AIX server using the Network Logon feature of AIX Fast Connect.

AIX Fast Connect Network Logon feature supports Windows 95/98 and NT clients. Windows 95/98 clients are supported using the standard Microsoft Client for Microsoft Networks or the IBM Client for IBM Networks. Windows NT clients require the IBM Networks Primary Logon Client for NT.

IBM Network Client can be downloaded from the following IBM Internet sites:

- http://service.boulder.ibm.com/asd-bin/doc/en_us/winntcl2/f-feat.htm for the **Windows NT** logon client. (Use the Primary Logon Client rather than the Coordinated Logon Client.)
- http://service.boulder.ibm.com/asd-bin/doc/en_us/win95cl/f-feat.htm for **Windows 95/98** clients.

Configuration Options

The following AIX Fast Connect configuration options are available for Network Logon feature customization.

Option	Default Value	Description
networklogon	0	This option is used to enable or disable the Network Logon feature of AIX Fast Connect — 1 indicates enabled, and 0 indicates disabled.
startup_script	startup.bat	This option specifies the filename of the startup script (in the NETLOGON share) used by the Microsoft Client for Windows 95/98 during network logon. Two meta tags in this string allow customization of the startup script filename during client logon — %U is expanded to the client's user name, and %N is expanded to the client's computer name. (IBM Networks clients always search for filename profile.bat , in directory \dcdb\users\username in the IBMLAN\$ file-share.)
profiles_path	/home	This string option specifies the AIX pathname for the PROFILES share, which the Network Logon feature uses to store user profiles and home directories.
netlogon_path	/var/cifs/netlogon	This string option specifies the UNIX path to the top of the NETLOGON and IBMLAN\$ shares. These shares are used to store the startup scripts. This is also where the Windows client searches for the configuration policy files at domain network logon time (for example: \\Server\netlogon\config.pol).

Enabling the Network Logon Feature

Enabling domain network logon support is simply a matter of setting the **networklogon** option to 1. This option can be enabled (or disabled) using Web-based System Manager, SMIT, or the **net** command. To enable the Network Logon feature, type:

```
net config /networklogon:1
```

Then restart the server. The AIX Fast Connect server then acts as a domain logon server for your workgroup.

Setting Up Startup Scripts

Startup scripts are DOS batch files that are executed automatically when client users logon to the domain through a domain logon server. Typically, these scripts are defined as user specific. By default, AIX Fast Connect installs a sample startup script (**/var/cifs/netlogon/startup.bat**), which can be customized as needed as a global startup script.

For Windows 95/98 clients using the Microsoft Networks client, the default installation of AIX Fast Connect configures **/var/cifs/netlogon/startup.bat** as a global startup script for all these clients. The parameter *startup_script* can be modified for these clients to support per-user or per-workstation scripts:

- Setting *startup_script* to **%N.bat** specifies that each login from *workstation* look for a startup script *workstation.BAT*, (in **/var/cifs/netlogon**, the NETLOGON share), regardless of the login user.
- Setting *startup_script* to **%U.bat** specifies that every login from *username* looks for a startup script *username.BAT*, (in **/var/cifs/netlogon**, the NETLOGON share), regardless of the PC workstation used.

- Setting *startup_script* to **dcdb\users\%U\profile.bat** provides compatibility with workstations configured for the IBM Networks client software, so every login goes to that user's profile directory, and executes the **profile.bat** startup script, regardless of which client software is configured.

For Windows 95/98/NT clients using the IBM Networks client, the IBM Networks client *always* uses **dcdb\users\username\profile.bat** (in share IBMLAN\$) as its startup script. By default, AIX Fast Connect sets **/var/cifs/netlogon/dcdb/users** as a link to **/home** (which is also the default for *profiles_path*). This allows the user-specific **profile.bat** files to reside in those users' profile directories (which are also AIX-user home directories, by default).

To setup a global startup script for all users using the IBM Networks client (and provide compatibility with Microsoft clients):

1. Edit the global startup script **/var/cifs/netlogon/startup.bat**
2. Create file links from **profile.bat** in every users' profile directory to the **/var/cifs/netlogon/startup.bat** file.

Setting Up Home Directories (Profile Directories)

Home directories, or profile directories, are used to store a Windows user's profile (**USER.DAT** and **USER.MAN**). Additionally, any application-specific settings and data are also stored in the Windows user's home directory. When the AIX Fast Connect server is configured as a domain network logon server, these home directories can reside on the AIX server.

AIX Fast Connect uses the *profiles_path* option to indicate where these profile directories are located. AIX Fast Connect expects the directory specified by *profiles_path* to contain a subdirectory for each AIX Fast Connect user. By default, AIX Fast Connect configures *profiles_path* to be **/home** (where most AIX user directories are kept).

If you want to change *profiles_path*, you must create subdirectories for each AIX Fast Connect user, with ownership and read/write permissions per user.

Windows Configuration Policy Files

When the AIX Fast Connect server is configured to support domain network logons, then Windows 95/98 and NT configuration policy files can be placed in the directory specified by the *netlogon_path* option. If **CONFIG.POL** or **NTCONFIG.POL** exist in the NETLOGON share at logon time, then the Windows client uses this policy file. By default, the location for these files is **/var/cifs/netlogon**.

Configuring Win 95/98 Clients for Network Logon

If IBM Network Client is being used, Follow the steps described in the IBM Network Client software README file.

If Microsoft Network Client is being used, select Client for Microsoft Networks as the default logon, and then change the Properties of this client software to logon to NT domains, using the AIX Fast Connect *domainname* as the NT-logon domain.

Configuring Network Logon for NT clients from Remote Subnets

The following are required to configure network logon from remote subnets:

- You must use encrypted passwords.
- The AIX Fast Connect logon server must have a domain name that is different from the NT domain controller's domain (if present), because the AIX Fast Connect logon server provides the logon services.
- If the client is not in same subnet as the AIX Fast Connect logon server, then you need either an LMHOSTS or an NBNS entry that maps AIX Fast Connect's *domainname*<00> to the AIX Fast Connect

logon server. You also need the entry *domainname<1C>*, which is automatically registered to the NBNS by the AIX Fast Connect NetLogon server. (The AIX Fast Connect NBNS server allows you to add *domainname<00>* as an Internet group name.)

- For browsing to work correctly, you need at least one master browser (NT workstation, for example) to be in the same workgroup as the AIX Fast Connect server domain name, on each network segment.

The location of the LMHOSTS file varies depending on the system configuration. It can be found on the client by typing **dir /s lmhosts** from the Windows base directory. If this file does not exist on the system, the default file LMHOSTS.SAM can be copied to LMHOSTS and then modified.

LMHOSTS example:

```
192.1.2.3 fcserver #PRE #DOM:fcdomain #AIX Fast Connect domain
192.1.2.3 "fcdomain \0x00" #PRE # 15 Bytes for the name, and
192.1.2.3 "fcdomain \0x1C" #PRE # the last byte is a hex subcode
```

These entries map the AIX Fast Connect name and domain to the server's IP-address. The #PRE operative indicates that this is to be preloaded, and the #DOM operative indicates the domain this server maps to. The other text above, after the '#' character is simply a comment statement. More details on this file can be found in the comment section of the LMHOSTS file.

After changing LMHOSTS, the PC client needs to be restarted, or run the command **nbtstat -R** to refresh the local name table.

Configuring LanServer (OS/2) Clients for Network Logon

The following restrictions apply to LanServer (OS/2) clients when accessing AIX Fast Connect as a network logon server:

- When using the OS/2 LOGON command to connect to AIX Fast Connect, specify the AIX Fast Connect domain as the OS/2 logon domain.
- LanServer clients always search for a startup script called **profile.cmd** rather than **profile.bat**.
- LanServer clients do not support roaming profiles and configuration files.

AIX Fast Connect NetLogon Limitations

The following restrictions apply to the AIX Fast Connect implementation of Network Logon:

- AIX Fast Connect logon server must be configured in the same IP subnet as the NT clients running the IBM Network client, or else follow the steps described above.
- AIX Fast Connect must be configured to use encrypted passwords to provide logon services to NT clients.

AIX Fast Connect Configurable Parameters for the net Command

AIX Fast Connect is designed for ease of administration, but provides a sufficient set of customizable parameters to support various configurations. Several of these parameters are dynamically configurable and do not require the server to be stopped and restarted for the changes to become effective.

These parameters are found in the **/etc/cifs/cifsConfig** file, and can be configured by using the **net** command with the following syntax:

```
net config /parameter_name:parameter_value
```

Details on usage can be found by typing: **net config help**.

A brief description of these parameters follows:

Parameter	Description	Type	(default,min,max)	S/D ¹
acl_inheritance	This value enables or disables the inheritance of AIX ACLs from the base path of a file share. Details of this feature can be found in Support for JFS ACLs.	int	(0, 0, 1)	S
aix_sharing_omodes	This option is used to enable or disable SMB ShareMode locking, using AIX file-locking modes. (<i>Not</i> related to <code>share_level_security</code> .) See also <code>oplockfiles</code> , <code>oplock_unix_lock</code>	int	0, 0, 1	S
alias_names	List of servername aliases. Use net name to list or update this parameter. Maximum length of each alias is 15 characters. See Specifying NetBIOS Aliases	String	NULL, n/a, n/a	D
autodisconnect	Timeout (in minutes) to disconnect inactive sessions. Value 0 indicates sessions will not timeout.	int	120, 0, 65535	D
backup_passthrough_authentication_server	IP address of the backup authentication server	String	NULL, n/a, n/a	S
cache_searches	Global enable/disable of the Search-caching feature. See Search Caching	int	0,0,1	S
casepreserve	When set to 1, AIX Fast Connect preserves mixed-case filenames when creating new files or directories for PC clients. When set to 0, AIX Fast Connect converts all filenames to lowercase when creating files and directories.	int	1,0,1	S
casesensitive	When set to 1, AIX Fast Connect filename searches are case sensitive. When set to 0 (the default), AIX Fast Connect filename searches are not case sensitive. Normally, this parameter should be set to the default because DOS and Windows use case-insensitive filename searches on their local file systems by default.	int	0,0,1	S
comment	Server description (for network browsing), up to a maximum of 49 characters.	String	n/a	S
dce_auth	Setting to enable AIX Fast Connect's support features for DCE and DFS. When enabled (set to 1), AIX Fast Connect uses DCE-authentication for all PC client logins and file-accesses. Requires AIX Fast Connect is installed <i>after</i> dce.client.* . See DCE/DFS Support for details.	int	(0,0,1)	S
domainname	Server domain (maximum of 15 characters).	String	WORKGROUP, n/a, n/a	S

Parameter	Description	Type	(default,min,max)	S/D ¹
dosattrmapping	DOS attribute mapping. If set to 1, the Archive, System, and Hidden attributes are mapped to User, Group, and Other execute bits. Otherwise, these attributes are not supported. This is only valid for files.	int	(1, 0, 1)	D
dosfilenamemapchar	The character used to map long file names to 8.3 DOS filename format. Valid values are tilde (~) and caret (^). Tilde (~) is the default.	char	~	S
dosfilenamemapping	DOS filename mapping. If set to 1, long file names are mapped to 8.3 format. Otherwise, no file name mapping is attempted. See Mapping Long AIX File Names to DOS File Names.	int	(1, 0, 1)	S
encrypt_passwords	Encrypted passwords. If set to 0, plain text passwords are used. A value of 1 will negotiate with the client. A value of 2 forces encrypted passwords.	int	(1, 0, 2)	S
guestlogonsupport	Guest Logon. A value of 1 will enable a guest user to access the server without an AIX Fast Connect password. This user will be connected with credentials defined by the user specified in the guestname parameter. A value of 0 disables this feature.	int	(0, 0, 1)	S
guestname	Guest Name (maximum 8 characters). This parameter specifies the user name that guest users will be connected as. The AIX Fast Connect password for this user should be null.	String	null, n/a, n/a	D
lm_encryption_level	Parameter to allow use of NT password encryption, when appropriate, instead of LM password encryption. The default is 0, meaning LM encryption only. If set to 1, allows NT encryption if the client supports it.	int	0,0,1	S
maxconnections ²	Maximum number of open connections allowed to a single resource (fileshare) on the server. (0 implies no limit.)	int	0, 0, 1000	D
maxopens ²	Maximum number of open files on the server.	int	0, 0, 1000	S
maxsearches ²	Maximum number of open searches on the server.	int	0, 0, 1000	S
maxsessessearches	Maximum number of open searches per session. For performance reasons, this number should be kept as small as practicable for your installation.	int	5,2,1000	S

Parameter	Description	Type	(default,min,max)	S/D ¹
maxusers ²	Maximum number of user sessions (logins) permitted.	int	0, 0, 1000	D
nbns	If set to 1, server acts as a NetBIOS name server.	int	1, 0, 1	S
netlogon_path	The AIX pathname for the NETLOGON and IBMLAN\$ shares (maximum 1023 characters), to store user startup scripts and policy files.	String	/var/cifs/netlogon, n/a, n/a	S
networklogon	Network Logon. This option is used to enable or disable the Network Logon feature of AIX Fast Connect.	int	0, 0, 1	S
oplock_unix_lock	Oplocks File Locking. Enable or disable AIX file-locking to be used for opportunistic locks. Enable this option if oplocks are enabled, and AIX applications need to share files with PC-clients. See also oplock_unix_lock_timeout , aix_sharing_omodes , oplockfiles .	int	0, 0, 1	S
oplock_unix_lock_timeout	Timeout in seconds, for <code>oplock_unix_lock</code> . (Time allowed to obtain AIX file lock.)	int	0, 0, 1	S
oplockfiles	Global parameter to define whether opportunistic locking is enabled (yes) or disabled (no). Opportunistic locking is a performance feature, allowing clients to lock entire files in non-exclusive mode. Controlled by oplocktimeout . See also sh_options , oplock_unix_lock .	Y/N	yes, no, yes	S
oplocktimeout	Timeout in seconds for opportunistic locking.	int	35, 35, 640	S
os2compatible	OS/2 Compatibility. If set to 0, READONLY means all readable but not writable files. If set to 1, READONLY means all readable files.	int	1, 0, 1	D
passthrough_authentication_server	IP address of the passthrough authentication server	String	NULL, n/a, n/a	S
primary_wins_ipaddr	IP address of the NBNS (WINS) server. When started, the AIX Fast Connect server will register its NetBIOS name(s) with this NBNS server. See also <code>wins_proxy</code> .	String	null, n/a, n/a	S
profiles_path	The AIX pathname for the PROFILES share (maximum 1023 characters), which the Network Logon feature uses to store user profiles and home directories.	String	/home, n/a, n/a	S
secondary_wins_ipaddr	IP address of secondary WINS address.	String	n/a	S

Parameter	Description	Type	(default,min,max)	S/D ¹
send_file_api	Boolean value to enable an enhanced system call to improve the performance in sending files over the network.	int	(1, 0, 1)	S
send_file_cache_size	Cache SendFile Option. If the send_file_api is 1 and the requested SMB read size is less than the value of this parameter, the send_file API caches the file. The default value is zero, which means send_file API will not cache the file.	int	(0, 0, 4194304)	S
send_file_size	Cache SendFile maximum size. If the send_file_api is 1 and the requested SMB read size is greater than the value of this parameter, then send_file API is used in the SMB operation.	int	(4096, 1, 4194304)	S
servername	NetBIOS name of the AIX Fast Connect server (maximum 15 characters).	String	TCP/IP hostname, n/a, n/a	S
sh_options	Data field (per share) to allow per-share options to be defined. This field should only be accessed with the net share command. See Per-Share Options.	int	n/a	S
share_level_security	Option to enable or disable share-level security (instead of user-level security). When enabled, share_level_security_username must also be specified. See Share-Level Security.	int	0,0,1	S
share_level_security_username	AIX username used for file-access credentials when share_level_security is enabled (maximum 8 characters). Similar to guestname, but used for share-level security mode.	String	NULL, n/a, n/a	S
startup_script	The filename of the startup script used when networklogon=1 (maximum 256 characters). Two meta tags in this string allow customization of the startup script filename during client logon — %U is expanded to the client's username, and %N is expanded to the client's computer name.	String	startup.bat, n/a, n/a	S
umask	Default permissions mask for files created from client machines. It is an octal number, and should always be prefixed with a zero.	octal	(022, 0, 0777)	D
usernamemapping	Option to enable/disable the Username Mapping feature, configured by net user /map.	int	0,0,1	S

Parameter	Description	Type	(default,min,max)	S/D ¹
wins_proxy	Proxy Option. A value of 1 enables the forwarding of NetBIOS name resolution requests to a WINS server specified by the <i>primary_wins_ipaddr</i> parameter.	int	0,0,1	S

Notes:

1. S stands for *static* and D for *dynamic*. Any changes to static parameters require a stop and restart of the AIX Fast Connect daemon before they take effect.
2. For maxusers, maxconnections, maxopens, and maxsearches, a default or minimum value of zero means unlimited (no restrictions).

Migrating to AIX Fast Connect from AIX Connections

For AIX Connections users to migrate to AIX Fast Connect, **netbios.*** filesets must be uninstalled, which also requires the **connect.*** prerequisite filesets to be uninstalled.

Note: AIX Fast Connect does *not* support the NetBEUI, IPX/SPX, Appletalk, or Netware protocols. AIX Fast Connect only supports SMB networking using NetBIOS over TCP/IP (RFC 1001/1002). If your network is configured for one of these other protocols, you might need to install TCP/IP and SMB-client software on your client PCs.

Before uninstalling AIX Connections, you might want to save the old configuration files. These are plain text configuration files that can be used as a reference when configuring AIX Fast Connect.

Saving ACONN Configuration Data Before ACONN Uninstall

AIX Connections (**connect.***) configuration files include:

/usr/tn/config.tn	Network/socket definitions
/usr/tn/profile.file	Export/share definitions
/usr/tn/services.NB	Service definitions for NB-realm
/usr/tn/services.NW	Service definitions for NW-realm
/usr/tn/services.AT	Service definitions for AT-realm
/usr/tn/lic.tot	Number of licensed users
/usr/tn/passwd.file.narrow	Encrypted passwords

To save these configuration files before uninstalling ACONN, simply copy/move these files to new names, **config.tn.save**, etc.

NOTE: Uninstalling **connect.*** deletes *only* those files that were originally installed by the AIX Connections installation.

NetBIOS/ix (**netbios.***) configuration files include:

/etc/mcstab	LANA definitions
/etc/mcs0	startup script (possibly customized)
/etc/mcsnet/wins.names	WINS data
/etc/inethosts	NIP cache (similar to LMHOSTS)

To save NetBIOS configuration data, the saved filenames *must not* begin with **mcs**, because **netbios.*** deletes all **mcs*** filenames during its uninstall process. For example:


```
mkdir /etc/nbix.save; cp -rph /etc/mcs* /etc/inethosts /etc/nbix.save
```

Appendix. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and

cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

Index

Special Characters

- /etc/exports file 369
- /etc/filesystems file 378
- /etc/gated.conf 47
- /etc/gateways 177
- /etc/hosts 23
- /etc/mail/aliases file 8
- /etc/named.ca 154
- /etc/named.data 154
- /etc/named.local 154
- /etc/named.rev 154
- /etc/protocols 50
- /etc/publickey file 391
- /etc/rc.bsdnet 217
- /etc/rc.net 24
- /etc/rc.tcpip 141
- /etc/resolv.conf 46
- /etc/sendmail.cf
 - TCP/IP 150
- /etc/services 50
- /etc/xtab file 370
- .netrc 201
- /usr/lib/security/audit/config 201
- /usr/lib/sendmail.cf 161
- /usr/lib/uucp/Devices 181

Numerics

- 802.3 64

A

- access control lists 366
- access times
 - NFS 398
- ACL (access control lists)
 - NFS support 366
- adapters
 - 2-port multiprotocol 297
 - isa/pci
 - wide area network 292
 - micro-channel 291
 - multiport/2 291
 - multiport/2
 - configuring 291
 - multiport model 2
 - configuring 294
 - object information and attributes 295
 - PCI adapters
 - ARTIC960HX 297
 - portmaster
 - configuring 291
- adding a CA root digital certificate 239
- Address Resolution Protocol 38
- addresses
 - network
 - general 4
 - TCP/IP 70

- administrative logon
 - BNU 311
- aliases
 - mail 8
- aliases file 8
- ARTIC960HX 297
- asinfo file 269
- assigned numbers 50
- asynchronous point-to-point protocol
 - user-level processes 185
- Asynchronous Point-to-Point Protocol
 - configuration 186
- Asynchronous Transfer Mode
 - connections 54
 - technology 54
- ATE (asynchronous terminal emulation)
 - customizing
 - changing defaults 283
 - characteristics 284
 - overview
 - management 283
 - setting up 283, 285
- ATM 54, 66
 - TCP/IP 55
- authentication services
 - PC-NFS 382
- autodialer connections
 - device files 319
- automount daemon
 - NFS (Network File System)
 - file systems 377

B

- Basic Networking Utilities 307
- binding
 - NFS (Network File System) 369
- BINLD 127
- biod daemons
 - NFS (Network File System) 372
- BNU
 - overview 307
- BNU (Basic Networking Utilities)
 - administrative login ID 311
 - daemons
 - overview 312
 - file transfer
 - monitoring 324
 - scheduling 313
 - log files 320
 - logon 311
 - logon failures
 - debugging 327
 - maintenance 320
 - monitoring
 - automatic 317
 - file transfer 324
 - remote connection 323

- BNU (Basic Networking Utilities) *(continued)*
 - setting up 317
 - polling
 - remote systems 318
 - remote systems
 - transporting files to 312
 - security 310
 - shell procedures 322
 - TCP/IP 314
 - tip command
 - variables 329
- BNU commands
 - cleanup 322
 - executing remote 313
 - maintenance 321
 - status-checking 322
- BNU configuration
 - files 308
 - general 314
- BNU directories
 - administrative 309
 - hidden 309
 - public directory 308
 - spooling 309
 - structure 308
- BNU examples
 - direct connection 334
 - modem connection 333
 - TCP/IP connection 331
- BNU files
 - administrative 309
 - configuration 308
 - devices files
 - autodialer connections 319
 - hardwired connections 319
 - TCP/IP 320
 - lock files 310
 - monitoring transfer 324
 - permissions 312
 - remote.unknown file 311
 - structure 308
 - systems files 311
- Boot Image Negotiation Layer daemon(BINLD) 127
- bridges
 - network 4

C

- cache file system support
 - NFS (Network File System) 367
- CacheFS
 - cache file system 367
- Certification Authority (CA)
 - adding root certificate to database 239
 - deleting root certificate from database 240
 - list of CAs 238
 - receiving certificate from 241
 - requesting certificate from 240
 - trust settings 239
- changing key database password 242

- clients
 - description 5
- commands
 - telnet 211
 - tic 211
 - touch 210
- communications
 - functions 1
 - network support 3
- configuration
 - TCP/IP 24
- creating a key database 238
- creating IKE tunnels with digital certificates 242

D

- daemons
 - network services 402
 - secure NFS 403
 - SRC 372
 - TCP/IP 141
- data link control (DLC)
 - device manager environment
 - components 300
 - structure 299
 - generic 299
- DDN 180
- debugging
 - BNU
 - logon failures 327
- default route 171
- deleting a CA root digital certificate 240
- deleting a personal digital certificate 241
- device driver
 - communications
 - 4-port multiprotocol 291
- digital certificates
 - adding root 239
 - creating IKE tunnels with 242
 - creating key database 238
 - deleting personal 241
 - deleting root 240
 - managing 237
 - receiving 241
 - requesting 240
 - trust settings 239
- direct connections
 - BNU configuration
 - example 334
- directories
 - BNU structure 308
- diskless support
 - NFS
 - SUN 403
- Distributed Computer Network Local-Network Protocol 49
- DLC (data link control) 299
- DNS (Domain Name Service) 146
- DOD 206
- domains
 - network
 - general 4

- Dynamic Host Configuration Protocol (DHCP)
 - addresses
 - TCP/IP 77
 - parameter assignments
 - TCP/IP 77
 - proxy daemon 111
- dynamic screen assignment 269

E

- error messages
 - NFS 396
- ESCDELAY 212
- establishing trust settings for key database 239
- Ethernet Version 2 64
- exporting
 - NFS (Network File System) 365
- exports file 369
- Exterior Gateway Protocol 47

F

- file formats
 - TCP/IP 217
- file handle
 - NFS (Network File System) 369
- file systems 365
- File Transfer Protocol 48
- file transfers
 - BNU
 - monitoring 324
- filesystems file 378
- filters
 - relationship to tunnels 225
 - rules 223
- filters, setting up 246
- FINGER 49
- flat network 23
- frames 26

G

- gateways
 - network 4
 - TCP/IP 172
- GDLC (generic data link control)
 - controls
 - installing 302
 - criteria 301
 - interface
 - implementing 301
 - ioctl operations 302
 - kernel services 304
 - overview 299
- generic data link control 299

H

- hardwired connections
 - devices files for 319
- hidden directories
 - BNU 309

- hierarchical network 23
- hop count 172
- host addresses 70
- host route 171

I

- IBM Key Manager 237
- IKE
 - features 221
- IKE tunnels
 - creating
 - using digital certificates 242
- IMAP server
 - configuring 18
- inetd daemon
 - debugging 210
- installation
 - TCP/IP 24
- interfaces
 - TCP/IP 63
- Internet Control Message Protocol 39
- Internet Engineering Task Force (IETF) 219
- Internet Key Exchange
 - see IKE 221
- Internet protocol 40
- Internet Protocol
 - security 219
 - features 220
 - IKE features 221
 - operating system 220
- Internet Protocol (IP) security 219
 - configuration 246
 - planning 224
 - installation 224
 - logging 251
 - predefined filter rules 250
 - problem determination 255
 - reference 263
- Internet Protocol Version 6 29
- IP
 - see Internet Protocol 219
- IP security
 - filters 223
 - and tunnels 225
 - SAs 226
 - security associations 221
 - tunnels
 - and filters 225
 - and SAs 226
 - choosing which type 226
 - tunnels and key management 222
- IP Security
 - Digital Certificate Support 224
- IPv4
 - also see Internet Protocol (IP) security 219
- IPv6 219
 - also see Internet Protocol Version 6 29

K

- kernel extension
 - NFS 401
- key management
 - and tunnels 222
- Key Manager tool
 - see IBM Key Manager 237
- keylogin command
 - secure NFS 389
- keys
 - changing database password 242
 - creating a database 238

L

- LAN (local area network)
 - description 3
- line discipline 266
- link station 303
- links
 - testing 304
 - tracing 304
- LLC (logical link control) 4
- local area network 3
- local-busy mode 303
- local node 5
- log files
 - BNU 320
- logging IP Security 251
- logical link control 4
- logon
 - BNU 311
 - UUCP 310
- LS (link station)
 - definition 303
 - statistics
 - querying 304

M

- MAC (medium access control) 4
- mail
 - /etc/mail/aliases file 8
 - aliases 8
 - how to compile database 9
 - local system 9
 - debugging 16
 - installation 7
 - internet message access protocol 17
 - list of
 - commands 19
 - files and directories 20
 - list of commands
 - IMAP and POP 21
 - log file
 - how to manage 15
 - logging 14
 - mailers 7
 - bellmail 7
 - BNU 7

- mail (*continued*)
 - mailers 7 (*continued*)
 - prog 16
 - statistics 15, 16
 - management tasks 7
 - message access programs 17
 - message routing program 7
 - post office protocol 17
 - protocol
 - IMAP 17
 - POP 17
 - queue 10
 - files 10
 - how to determine processing interval 13
 - how to force 12
 - how to move 13
 - how to specify processing interval 12
 - q control file 11
 - system management overview 7
 - traffic
 - how to log 15
 - user interface 7
- mailers 7
- managing TTY devices 266
- mapped file support
 - NFS (Network File System) 368
- medium access control 4
- methods
 - TCP/IP 217
- metric 172
- MIB (Management Information Base)
 - variables 345
- modems 273
 - AT command summary 280
 - dial modifiers 283
 - result codes summary 282
 - S-registersSummary 281
 - attaching a modem 275
 - commands
 - sending AT commands 276, 277
 - connections
 - BNU configuration example 333
 - data compression 274
 - speed
 - baud 273
 - bits per second (bps) 273
 - system compatibility 274
 - standards 273
 - ITU-TSS 274
 - Microcom Networking Protocol (MNP) 274
 - troubleshooting modem problems 279
- monitoring
 - BNU
 - automatic 317
 - file transfer 324
 - remote connection 323
- mount command
 - NFS (Network File System)
 - file systems 376
 - secure NFS
 - file systems 394

- mount points
 - NFS (Network File System) 374
- mounting process
 - NFS (Network File System) 369
- MTU
 - Path MTU Discovery 180
- Multiple Screen utility 268

N

- name resolution
 - TCP/IP 146
- national languages
 - BNU support of 308
- network, configuration of
 - hosts list
 - updating 141
- network adapter cards
 - TCP/IP 50, 54
- network addresses 70
- Network File System (NFS) 365
- network interfaces
 - TCP/IP 63
- network lock manager 384
- network planning
 - TCP/IP 23
- network route 171
- network services
 - daemons
 - list of 402
 - utilities
 - list of 402
- network status monitor 384
- network trusted computing base 204
- networks
 - local area networks 3
 - overview 1
 - physical 3
 - wide area networks 3
- NFS (Network File System)
 - /etc/exports file 369
 - /etc/filesystems file 378
 - /etc/publickey file 391
 - /etc/xtab file 370
 - access times 398
 - ACL (Access Control Lists) 366
 - automount daemon 377
 - binding 369
 - biod daemons
 - how to change number of 372
 - cache file system 367
 - checklist for configuring 373
 - clients
 - how to configure 374
 - controlling 371
 - directory 365
 - error messages 396
 - mount 397
 - nfs_server 396
 - exporting 365
 - file handle 369
 - file system 365

- NFS (Network File System) (continued)
 - file systems
 - how to change exported 375
 - how to enable root access 376
 - how to export 374
 - how to mount automatically 377
 - how to mount explicitly 376
 - how to unexport 375
 - how to unmount 381
 - groups 401
 - implementation 370
 - installing 373
 - kernel extension 401
 - mapped files 368
 - mount points 374
 - mounting process 369
 - mounts
 - predefined 378, 381
 - types of 368
 - network lock manager 384
 - architecture 384
 - crash recovery process 384
 - grace period 384
 - how to start 385
 - network file locking process 384
 - troubleshooting 385
 - network services
 - list of 365
 - network status monitor 384
 - nfsd daemons
 - how to change number of 372
 - overview 365
 - PC-NFS 381
 - authentication services 382
 - print-spooling services 382
 - portmap daemon 371
 - problem determination
 - authentication schemes 400
 - hard-mounted files 395
 - hung programs 400
 - list of commands 395
 - permissions 400
 - soft-mounted files 395
 - RPC 370
 - rpc.
 - how to configure 382
 - rpc.pcnfsd
 - how to start 382
 - how to verify accessibility 383
 - secure NFS 387
 - administering 392
 - authentication 388
 - authentication requirements 389
 - Caesar cipher 387
 - cipher 387
 - ciphertext 387
 - configuring 393
 - cryptanalyze 387
 - cryptographer 387
 - decryption 387
 - DES (Data Encryption Standard) 387

- NFS (Network File System) *(continued)*
 - encryption 387
 - file systems 394
 - how to export a file system 393
 - key 387
 - net name 391
 - network entities 391
 - networking daemons 403
 - networking utilities 403
 - performance 392
 - plaintext 387
 - public key cryptography 389
 - servers 365
 - how to configure 374
 - stateless servers 365
 - system startup
 - how to start 372
 - XDR 370
- NFS commands
 - list of 402
- NFS daemons
 - command line arguments
 - how to change 372
 - controlling 371
 - how to get current status 373
 - how to start 372
 - how to stop 373
 - locking
 - list of 402
 - secure NFS 403
- NFS diskless support
 - SUN
 - clients 403
- NFS files
 - list of 402
- NFS servers
 - hung programs 400
 - problem determination
 - name resolution 401
- nfsd daemons
 - NFS (Network File System) 372
- NIC (Network Information Center) 180
- nodes
 - local 5
 - network 5
 - remote 5

O

- other operating systems 5

P

- packets 26
- Path MTU Discovery 180
- PC-NFS 381, 382
- PCI adapters
 - ARTIC960HX 297
- permissions files 312
- point-to-point protocol
 - user-level processes 185

- polling
 - BNU
 - remote systems 318
- POP server
 - configuring 18
- portmap daemon
 - NFS (Network File System) 371
- protocols
 - gateway 173
 - network
 - general 4
- public directory
 - BNU 308
- public key cryptography
 - secure NFS 389

Q

- queue
 - mail 10

R

- receiving a digital certificate 241
- Remote Command Execution Protocol 50
- remote connections
 - BNU
 - monitoring 323
- Remote Login Protocol 50
- remote node 5
- Remote Shell Protocol 50
- remote systems
 - BNU
 - polling 318
- remote.unknown file 311
- requesting a digital certificate 240
- RFC 1010 37
- RFC 1100 37
- RFC 791 40
- route
 - definition of 171
- routers
 - TCP/IP 172
- routing
 - network 4
 - TCP/IP 171
- Routing Information Protocol 50
- routing table 171
- RPC
 - NFS 370
- rpcinfo command
 - NFS configuration 383

S

- SAP (service access point)
 - definition 303
 - statistics
 - querying 304
- secure NFS 387

- security
 - BNU 310
 - Internet Protocol (IP) 219
 - TCP/IP 199
- security associations (SA) 221
- security associations (SAs)
 - relationship to tunnels 226
- Security Parameters Index (SPI)
 - and security associations 221
- Serial Optical 66
- servers
 - configuring IMAP 18
 - configuring POP 18
 - description 5
 - NFS (Network File System) 365
 - stateless 365
- service access point 303
- shell procedures
 - BNU 322
- short-hold mode 304
- SLIP 66
- SNMP (Simple Network Management Protocol)
 - access policies 339
- SNMP daemon
 - configuring 340
 - logging facility 358
 - MIB variables support 345
 - overview 340
 - problem determination 361
 - processing 341
 - restrictions 358
 - RFC conformance 357
- spooling directory
 - BNU 309
- SRC (System Resource Controller)
 - control of TCP/IP 142
 - NFS (Network File System)
 - daemons 373
- statistics
 - querying
 - SAP 304
- subservers
 - TCP/IP 141
- subsystems
 - TCP/IP 141

T

- TCP/IP
 - /etc/ftpusers 203
 - /etc/gated.conf 47, 178
 - /etc/gateways 177, 208
 - /etc/hosts 23, 24, 46, 146, 148, 150, 152, 207
 - /etc/hosts.equiv 202
 - /etc/named.boot 154
 - /etc/named.ca 154
 - /etc/named.data 154
 - /etc/named.local 154
 - /etc/named.rev 154
 - /etc/networks 177, 178, 208
 - /etc/protocols 50

- TCP/IP (*continued*)
 - /etc/rc.bsdnet 217
 - /etc/rc.net 24
 - /etc/rc.tcpip 141, 177
 - /etc/resolv.conf 46, 150, 154, 207
 - /etc/sendmail.cf 150, 161
 - /etc/services 50
 - /etc/syslog.conf 207
 - .netrc 201
 - /usr/lib/security/audit/config 201
 - /usr/lib/sendmail.cf 161
 - addresses 70
 - broadcast 76
 - class A 71
 - class B 71
 - class C 72
 - comparison 75
 - DHCP 77
 - DHCP proxy daemon 111
 - host 70
 - local 70
 - local loopback 76
 - network 70
 - subnet 73
 - subnet masks 74
 - zeros 73
 - ATM 55
 - BINLD 127
 - BNU
 - devices files 320
 - BNU connections 314
 - client network services 144
 - commands
 - list of 25
 - configuration 24
 - checklist 25
 - daemons 141
 - how to configure gated 178
 - how to configure routed 177
 - inetd 143
 - SRC (System Resource Controller) 142, 209
 - subservers 141
 - subsystems 141
 - DNS name server
 - configuring dynamic zones 168
 - examples
 - BNU configuration 331
 - file formats 217
 - frames
 - definition 26
 - hosts 24
 - installation 24
 - interfaces 63
 - Internet Protocol Version 6 29
 - IP security
 - IKE features 221
 - installation 224
 - planning configuration 224
 - predefined filter rules 250
 - problem determination 255
 - reference 263

TCP/IP (continued)

- IP Security 219
- list of commands 216
- list of daemons 216
- list of files 217
- mail server 161
- methods 217
- name resolution 146
 - how to perform local 152
 - planning for domain 153
 - problem determination 207
 - process 150
- name server 148
 - caching-only 148
 - configuration files 154
 - forwarder/client 148
 - how to configure hint 160
 - how to configure host to use 166
 - how to configure mail server 161
 - how to configure master 155
 - how to configure slave 158
 - master 148
 - remote 148
 - slave 148
 - zone of authority 148
- naming 146
 - authority 146
 - conventions 147
 - DNS (Domain Name Service) 146
 - domain 146
 - flat network 23, 146
 - hierarchical network 23, 146
 - how to choose names 148
- network adapter cards 50, 54
 - ATM adapter 59
 - configuring 59
 - how to configure 51
 - how to install 51
- network interfaces 63
 - 802.3 64
 - ATM 66
 - automatic configuration 64
 - automatic creation 64
 - Ethernet Version 2 64
 - managing 67
 - manual creation 64
 - multiple 67
 - problem determination 212
 - Serial Optical 66
 - SLIP configuration 66
 - Token-Ring 65
- network planning 23
- packets
 - definition 26
 - headers 35, 36, 37
 - problem determination 215
 - tracing 35
- parameter assignments
 - DHCP 77
- point-to-point protocol 185, 186
 - used as an alternative to SLIP 185

TCP/IP (continued)

- point-to-point protocol 185, 186 (continued)
 - user-level processes 185
- problem determination 206
 - communication 207
 - ESCDELAY 212
 - name resolution 207
 - network interface 212, 213, 214
 - packet delivery 215
 - routing 208
 - SRC 209
 - telnet or rlogin 210
 - TERM 210
- protocols 26
 - application-level 46, 47, 48, 49, 50
 - assigned numbers 50
 - network-level 37, 38, 39, 40
 - transport-level 42, 43, 44
- RFCs
 - RFC 1010 37
 - RFC 1100 37
 - RFC 791 40
 - supported 217
- route
 - default 171
 - definition of 171
 - host 171
 - network 171
- routing 171
 - dynamic 171, 174
 - gated 171
 - gateways 25, 172, 174, 175
 - hop count 172
 - how to configure gated 178
 - how to configure routed 177
 - how to get an autonomous system number 180
 - metric 172
 - problem determination 208
 - protocols 50, 173
 - routed 171
 - routers 172
 - static 171, 174
- routing table 171
- security 199
 - data 206
 - DOD 206
 - NTCB 204, 205
 - operating system-specific 200, 201
 - remote command execution access 202
 - restricted FTP users 203
 - SAK 201
 - TCP/IP-specific 201, 204
 - trusted shell 201
- see Internet Protocol 220
- server network services 145
- servers 25
- SLIP
 - /usr/lib/uucp/Devices 181, 183
 - how to configure over modem 181
 - how to configure over null modem 183
 - how to deactivate a SLIP connection 185

- TCP/IP *(continued)*
 - tty
 - how to remove 185
 - used for SLIP over a modem 181
 - used for SLIP over a null modem 183
- TELNET 48
- telnet command 211
- telnet connection
 - debugging 211
- telnetd daemon
 - debugging 211
- TERM
 - TCP/IP
 - TERM 210
 - TERM environment variable 265
 - termcap conversion 265
 - terminal 265
 - terminfo database 265
 - tic command 211
 - Time Server Protocol 50
 - tip command
 - configuring 329
 - overview 328
 - variables
 - order of use 329
 - Token-Ring 65
 - touch command 210
 - Transmission Control Protocol 44
 - Transmission Control Protocol/Internet Protocol 23
 - Trivial File Transfer Protocol 49
 - tty 185
 - TTY
 - managing 266
 - tty (teletypewriter)
 - definition 265
 - examples 265
 - tty characteristics
 - setting 266
 - tty tasks
 - setting tty characteristics 266
 - using the Multiple Screen utility 268
- tunnels
 - and key management 222
 - choosing which type 226
 - relationship to filters 225
 - relationship to SAs 226

U

- umount command
 - NFS (Network File System)
 - file systems 381
- UNIX-to-UNIX copy program 307
- User Datagram Protocol 43
- utilities
 - network services 402
 - NFS
 - secure 403
- uucico daemon 312
- uuclean command 322
- uucleanup command 322

- UUCP (UNIX-to-UNIX Copy Program) 307, 310
 - uucpd daemon 314
 - uudemon.admin command 322
 - uudemon.cleau command 322
 - uupoll command 322
 - uuq command 322
 - uusched daemon 313
 - uusnap command 322
 - uustat command 322
 - Uutry command 323, 324
 - uuxqt daemon 313

V

- variables
 - tip command
 - order of use 329
- Virtual Private Network (VPN) 219
- VPN
 - benefits 219

W

- WAN (wide area network)
 - description 3
- wide area network 3

X

- XDR
 - NFS (Network File System) 370
- xtab file 370

Readers' Comments — We'd Like to Hear from You

AIX 5L Version 5.1
System Management Guide: Communications and Networks

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Corporation
Publications Department
Internal Zip 9561
11400 Burnet Road
Austin, TX
78758-3493

Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Printed in U.S.A