

HP 9000 Computers

**Administering ARPA
Services**

**HP 9000 Computers
Administering
ARPA Services**



**HEWLETT
PACKARD**

**Manual Part Number: B1014-90008
Printed in U.S.A., October 1992**

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material. Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

© Copyright 1991, 1992 Hewlett-Packard Company.

This document contains proprietary information, which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104.9(a). © Copyright 1980, 1984, AT&T, Inc. © Copyright 1979, 1980, 1983, 1985-1990, The Regents of the University of California. © Copyright, 1986, 1987, 1988 Sun Microsystems, Inc. This documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California. MS-DOS[®] is a registered trademark of Microsoft Corp. UNIX[®] is a U.S. registered trademark of AT&T in the U.S.A. and other countries. NFS is a trademark of Sun Microsystems, Inc.

**Hewlett-Packard Company
19420 Homestead Road
Cupertino, CA 95014 U.S.A.**

Contents

Chapter 1 Product Overview

Software Components	1-1
DTC Device File Access (DDFA) Software Component	1-4
Military Standards and Request for Comment Documents	1-7

Chapter 2 Installing ARPA Services

Installation Steps	2-1
1. Updating Your Network Map	2-2
2. Installing the ARPA Services Software	2-2
Files Created During Software Installation	2-3
3. Configure Host Name to Address Mapping	2-5
Choose Mapping Method	2-5
Configure Mapping	2-6
Verify Mapping	2-6
4. Configure the ARPA Services Software	2-7
Configuration Tasks	2-8
Adding Remote Connectivity (Editing /etc/hosts)	2-10
Using SAM to Add Remote Connectivity (Edit /etc/hosts)	2-11
Manually Adding Remote Connectivity (Editing /etc/hosts)	2-14
Mapping Verification	2-15
Example	2-15
Format for /etc/hosts	2-15
Permissions	2-15
Editing /etc/inetd.conf	2-16
Example	2-18
Format for /etc/inetd.conf	2-18
Permissions	2-19
Adding Security (Editing /usr/adm/inetd.sec)	2-19
Setting the Maximum Number of Remote Connections	2-20
Allowing and Denying Nodes Access to Services	2-20
Using SAM to Add Security (Edit /usr/adm/inetd.sec)	2-20
Manually Editing /usr/adm/inetd.sec	2-21
Format for /usr/adm/inetd.sec	2-22

Example	2-23
Permissions	2-24
Modifying FTP Access (Creating /etc/ftpusers)	2-24
Using SAM to Modify FTP Access (Create /etc/ftpusers)	2-24
Manually Modifying FTP Access (Editing /etc/ftpusers)	2-25
Permitting Easy Access from Remotes (Creating /etc/hosts.equiv)	2-26
Using SAM to Permit Easy Access from Remotes(Create /etc/hosts.equiv)	2-27
Manually Permitting Access from Remotes (Creating /etc/hosts.equiv)	2-28
Creating \$HOME/.rhosts	2-30
Using SAM to Create a \$HOME/.rhosts File for the Local Super-User Account	2-31
Manually Creating \$HOME/.rhosts	2-32
Creating \$HOME/.netrc	2-34
Format for \$HOME/.netrc	2-34
Example	2-35
Permissions	2-35
5. Rebooting Your System	2-36
6. Checking the ARPA Services Installation	2-36
Chapter 3 Administering ARPA Services	
Daemons and Servers	3-1
Daemons	3-1
rwhod Daemon	3-2
inetd Daemon	3-4
Servers	3-5
Remote Shell Server	3-6
Remote Login Server	3-7
Remote Execution Server	3-8
Finger Protocol Server	3-9
Telnet Protocol Server	3-9
Internet File Transfer Protocol Server	3-10
Altering the ftpd Timeout	3-10
Enabling ftpd Logging	3-11
Anonymous ftp Account	3-11
Using SAM to Create an Anonymous ftp Account	3-12
Manually Creating an Anonymous ftp Account	3-14
System Log Files	3-16
Configuring syslogd	3-16
Maintaining System Log Files	3-17

Enabling the Log File for inetd Connection Logging	3-18
Reconfiguring the Log File for inetd	3-18

Chapter 4 Configuring and Maintaining the BIND Name Server

Key Terms	4-3
Overview of the BIND Name Service	4-6
Domain Name System (DNS)	4-7
Name Servers	4-8
Resolvers	4-8
BIND Name Server versus Network Information Service	4-8
BIND Name Server versus /etc/hosts	4-9
Benefits of Using the BIND Name Server	4-10
How the BIND Name Server Works	4-10
Resolver Search Algorithm	4-12
Setting Up the BIND Name Service	4-14
Creating and Registering a New Domain	4-15
Name Server Information	4-16
Considerations for Selecting Name Servers	4-17
Name Server Data Files	4-18
Resource Record Format	4-19
Standard Resource Records	4-20
Configuring Name Servers	4-25
Configuring a Primary Name Server	4-25
Using hosts_to_named	4-26
Configuring a Secondary Name Server	4-27
Creating the Boot File with hosts_to_named	4-28
Creating the Boot File Manually	4-28
Configuring a Caching Only Name Server	4-30
Setting Up a Remote Name Server	4-30
Setting the Local Domain	4-32
Starting the Primary Server	4-32
Verifying the Name Server	4-33
Maintaining Network and Domain Data Files	4-34
Updating Network-Related Files	4-34
Updating Domain Data Files	4-35
Adding a Host	4-35
Deleting a Host	4-35
Delegating a Subdomain	4-36
Configuring a Root Server	4-37
Using Other Options	4-38
Wildcarding	4-38
Boot File Options	4-38

Troubleshooting the BIND Name Server	4-40
Troubleshooting Tools and Techniques	4-40
Standard networking commands	4-40
nslookup	4-41
syslogd	4-41
Name Server Debugging	4-41
Dump the name server database	4-42
Problem Symptoms	4-42
Name Server Problems	4-43
Understanding Name Server Debugging Output	4-50
Sending Signals to the Name Server	4-54
Name Server Debug Levels	4-55
Name Server Statistics	4-56

Chapter 5 Configuring gated

Overview	5-1
Advantages	5-2
When to Use gated	5-2
Protocols and Metrics	5-3
Configuration Options	5-4
Configuring Routing Protocols	5-5
RIP	5-5
EGP and HELLO	5-6
Customizing Routes	5-6
Specifying a Default Gateway	5-6
Installing Static Routes	5-7
Setting Interface Metrics or States	5-7
Limiting Routing Information that gated Sends Out	5-8
Specifying Interfaces that Cannot Send Information Out	5-8
Specifying Gateways that Receive Outgoing Information	5-8
Limiting Information Sent Out About Networks and Hosts	5-9
Limiting Routing Information That gated Takes In	5-10
Specifying Interfaces That Cannot Receive Information	5-10
Specifying Gateways that Send Incoming Information	5-10
Limiting Information Received About Networks and Hosts	5-11
Specifying Tracing Output	5-11
Configuring gated	5-12
Sample Configurations	5-14
Starting gated	5-17
To Find Out if gated is Running	5-19
Troubleshooting gated	5-20
Troubleshooting Tools and Techniques	5-20

syslog Output	5-20
gated Tracing	5-20
Dump the gated routing table	5-21
ripquery	5-21
Problems	5-21
Chapter 6 Internetwork Mail Routing	
Chapter Overview	6-1
Key Terms	6-2
sendmail Overview	6-4
Function	6-4
Features	6-5
How sendmail Works	6-6
Collecting Messages	6-8
From the Argument Vector and Standard Input	6-8
Using the Simple Mail Transfer Protocol (SMTP)	6-9
From the Queue	6-10
Routing Messages	6-11
Routing to Remote Systems	6-11
Routing to Local Destinations	6-12
Error Handling	6-12
Deciding Whether to Install sendmail	6-13
Installing sendmail	6-13
Using SAM to Install sendmail	6-14
Manually Installing sendmail	6-14
1. Installing a Configuration File	6-15
Default Routing	6-16
2. Making sendmail Executable	6-18
3. Creating System-wide Mail Aliases	6-19
4. Invoking the sendmail Daemon	6-20
5. Verifying sendmail Examples	6-21
Operating sendmail	6-24
The sendmail Daemon	6-24
Freezing the Configuration	6-26
sendmail on a Cluster	6-27
The Mail Queue	6-28
Queue Priorities	6-30
Load Limiting	6-31
Mail Queue File Format	6-31
Printing the Mail Queue	6-34
The System Log	6-35
Setting Log Levels	6-36

Aliasing and the Alias Database	6-38
Filename Aliases	6-40
Command Line Aliases	6-40
:include: Specifications	6-41
User-Controlled Aliasing: .forward	6-42
Aliasing Loops	6-43
The Alias Database	6-43
MX Records	6-46
How sendmail Uses MX Records	6-46
How to Use MX Records for Mail Relaying	6-47
MX Failures	6-48
Mail Error Handling	6-49
The sendmail Configuration File	6-51
C or F Lines – Class Definitions	6-52
Literal Class Definition	6-53
File Class Definition	6-53
File Class Definition from Program Output	6-54
D Lines – Macro Definitions	6-55
Macro Interpolation	6-55
Required Macro Definitions	6-56
Macros Defined at Run Time	6-57
Example of Macro Definitions	6-59
H Lines – Header Definitions	6-60
Default Processing of Header Lines	6-61
Special Processing of Standard Header Lines	6-62
Special Header Lines	6-64
M Lines – Mailer Definitions	6-65
Mailer Name	6-66
The Path Field	6-66
The Flags Field	6-66
The Sender and Recipient Fields	6-70
The Eol Field	6-71
The Argv Field	6-71
Example Delivery Agents Definitions	6-71
O Lines – Option Specifications	6-72
P Lines – Precedence Definitions	6-79
T Lines – Trusted User Definitions	6-80
S and R Lines – Address Rewriting	6-80
Recipient Address Rewriting	6-82
Header Address Rewriting	6-84
Rewriting Rules	6-85

Testing Address Rewriting Rulesets	6-93
Troubleshooting sendmail	6-97
Superuser	6-97
Consistency	6-97
Address Resolution and Aliasing	6-98
Message Delivery	6-99
Contacting the sendmail Daemon	6-100
sendmail Logging	6-101

Chapter 7 Configuring TFTP and BOOTP Servers

Chapter Overview	7-2
Key Terms	7-3
How BOOTP Works	7-4
Configuring TFTP and BOOTP Servers Using SAM	7-5
Procedure for Configuring tftpd and bootpd with SAM	7-5
Configuring the TFTP Server Manually	7-7
Procedure for Configuring tftpd	7-7
Verify Your tftpd Installation	7-9
Configuring the BOOTP Server Manually	7-10
Procedure for Configuring bootpd	7-10
Verify Your bootpd Installation	7-11
Adding a BOOTP Client Manually	7-12
Collect Information About the Client	7-12
Understanding Boot File Configurations	7-12
Parameter Tags and Descriptions	7-14
Examples of Adding BOOTP Clients	7-15
Example 1: Adding an HP 700/X Terminal as a Client	7-15
Example 2: Adding a Gateway as a Client	7-18
Command Options for Using TFTP	7-22
Troubleshooting BOOTP and TFTP Servers	7-24
Helpful Configuration Changes	7-24
Common bootpd Problems	7-25
Common tftpd Problems	7-31
Error Logging	7-34
Information Log Level	7-34
Notice Log Level	7-36
Error Log Level	7-37

Chapter 8 Troubleshooting ARPA Services

Chapter Overview	8-1
Characterizing the Problem	8-2
Diagnostic Tools Summary	8-3
Diagnosing Repeater and Gateway Problems	8-5

Flowchart Format	8-7
Troubleshooting the ARPA/Berkeley Services	8-8
Error Messages	8-8
Services Checklist	8-9
Flowchart 1. Checking for a Server	8-11
Flowchart 2. Security for ARPA Services	8-15
Flowchart 3. Security for Berkeley Services	8-18
Reporting Problems to Your Hewlett-Packard Support Contact	8-20
Appendix A Public Network Contacts	
Choosing a BIND Domain	A-2
Appendix B Standard Data Files and Experimental Resource Records	
Domain Data Files	B-1
Boot File	B-1
Standard Data File Format	B-1
Option Lines	B-6
Experimental Resource Records	B-6

Figures

Figure 2-1. Multi-Homed Host Network Scheme	2-11
Figure 3-1. Directory Structure for Public ftp Account	3-12
Figure 4-1. Resolution Algorithm	4-2
Figure 4-2. The Domain Name System	4-7
Figure 4-3. Local Query	4-11
Figure 4-4. Remote Query	4-12
Figure 4-5. Standard Data Files	4-19
Figure 5-1. Configuring gated	5-13
Figure 5-2. Internet Using Gated	5-14
Figure 6-1. Flow of Mail Through sendmail	6-7
Figure 6-2. Rulesets	6-82
Figure 8-1. Troubleshooting Networks that Use Repeaters	8-5
Figure 8-2. Flowchart 1. Checking for a Server	8-10
Figure 8-3. Flowchart 2. Security for ARPA Services	8-17
Figure 8-4. Flowchart 3. Security: Berkeley Services	8-19

Product Overview

The HP 9000 ARPA Services product enables your HP 9000 computer to transfer files, log into remote hosts, execute commands remotely, and send mail to and receive mail from remote hosts on the network.

A link product, such as LAN/9000 or X.25/9000, must be installed for ARPA to function. The link product provides the hardware and software needed for communication by an HP 9000 computer over an IEEE 802.3, Ethernet Local Area Network, or X.25 packet switch network. NS and NFS Services also require link software, and can run concurrently on the same node with ARPA Services.

Note The information in this manual applies to all HP 9000 computer systems unless specifically noted otherwise.

Software Components

The HP 9000 ARPA Services product combines services developed by the University of California at Berkeley (UCB), Cornell University, Carnegie-Mellon University (CMU), and Hewlett-Packard. The product includes subsets known as ARPA Services, Berkeley Services, and DTC Device File Access (DDFA).

ARPA Services include the set of services developed by UCB for the Advanced Research Projects Agency (ARPA), or those services which implement standard

Software Components

internet protocols. ARPA services are used to communicate with HP-UX, UNIX, and non-UNIX systems.

Berkeley Services include the set of services developed by UCB to implement UCB protocols. These services, sometimes referred to as the "Berkeley 'r' commands," are used to communicate with HP-UX or UNIX systems. Tables 1-1 and 1-2 list the services provided in the HP 9000 ARPA Services product.

Service	Program	Author
File Transfer Protocol	ftp	UCB
Telnet	telnet	UCB
Simple Mail Transfer Protocol (Internet Mail Routing)	sendmail	UCB
Domain Name System (Berkeley Internet Name Domain server)	named	UCB
Finger/Name Protocol	fingerd	UCB
Trivial File Transfer	tftp	UCB
Bootstrap Protocol	bootpd	CMU
Dynamic routing	gated	Cornell
DTC Device File Access	DDFA	HP

Service	Command
Remote copy	rcp
Remote login	rlogin
Remote shell	remsh
Remote uptime	ruptime
Remote who	rwho

Software Components

The following is an overview of the ARPA and Berkeley Services. For more information about these services, refer to the *Using ARPA Services* manual.

- ftp enables you to perform file management operations on remote nodes and copy files among nodes on the network that support ARPA Services.
- telnet allows you to log onto a remote host that supports ARPA Services.
- When installed, sendmail works with your network's mailers to perform internetwork mail routing among UNIX and non-UNIX hosts on the network.
- named implements the Domain Name System. This service (called the Berkeley Internet Name Domain (BIND) Name Service) provides a distributed host name lookup and facilitates internetwork mail.
- fingerd allows remote users to look up user information about your system over the network.
- bootpd allows some diskless systems, such as the HP 700/X terminal, to load network and configuration parameters from a server on the network.
- tftp is used to allow some diskless systems, such as the HP 700/X terminal, to transfer files containing bootstrap code, fonts, or other configuration information.
- gated dynamically determines routing over internets from one node to another.
- rexec is a library routine used to execute commands on a remote UNIX host on the network.
- rcp allows you to transfer files between UNIX hosts on the network.
- rlogin allows you to log onto a remote UNIX host.
- remsh allows you to execute commands on a remote UNIX host. remsh is the same command as rsh in 4.3 BSD.

DTC Device File Access (DDFA) Software Component

- **ruptime** lists information about specified UNIX nodes that are running the **rwwho** daemon. **ruptime** is not supported over X.25 networks or networks using the PPL (SLIP) product.
- **rwwho** lists information about specified UNIX nodes that are running the **rwwho** daemon. **rwwho** is not supported over X.25 networks or networks using the PPL (SLIP) product.

The ARPA Services product does not include hardware components.

DTC Device File Access (DDFA) Software Component

Devices on a Datacommunications and Terminal Controller (DTC) and the DTCs are configured by one of two DTC manager products listed below. Refer to the documentation with these products for information on configuring DTCs and devices connected to the DTC. The DTC is a modular and flexible LAN-based controller which provides asynchronous connectivity and PAD support for local and remote terminals and printers.

- **HP OpenView DTC Manager/UX**, which runs on an HP 9000 Series 800 host and manages DTCs for the local system. This DTC Manager is also known as the **HP-UX Host-Based DTC Manager**.
- **HP OpenView DTC Manager**, which runs on a dedicated PC and manages DTCs for access to multiple HP (HP 3000 and HP 9000) and non-HP systems. This DTC Manager is also known as the **PC-Based DTC Manager**.

The DTC Device File Access (DDFA) software allows access from HP-UX systems and user-written applications to HP DTCs using standard HP-UX structures. DDFA provides an interface to remote (LAN-connected) DTC ports which is similar to the interface for local MUX ports. DDFA is documented in the *DTC Device File Access Utilities Manual*.

DTC Device File Access (DDFA) Software Component

The basic principle is that a daemon is created for each configured port based on information in a configuration file (dp file). When the daemon is spawned, it takes a pty from the pty pool and creates a device file with the same major and minor number as the pty slave. The device file is known as the "pseudonym", and user applications use the pseudonym to access the server port using standard HP-UX intrinsics (open, close, write etc). The daemon listens on the pty until an application does an open(pseudonym) call. Then the daemon manages the connection to the server port. The end result is that the server port is addressed via a device file. The mechanism that makes it happen is transparent to the user. For example, the lpadmin command can be used to configure a printer on a remote server port by using the pseudonym. A second configuration file (pcf) contains information to profile the port's behavior.

DDFA consists of the following items:

- dp - Dedicated Port configuration file. This is a text file which contains the information DDFA needs to set up and manage a connection to a specified DTC port. It contains a one line entry for each configured DTC port, which specifies the following:
 - DTC IP address
 - DTC board and port numbers
 - pseudonym (the device file)
 - path to the Port Configuration File (pcf)

The dp file is parsed by the Dedicated Port Parser (dpp) which spawns an outgoing connection daemon (ocd) for each connection specified in the file.

dp is also used by the HP-UX telnet daemon (telnetd) to identify incoming connections. In this usage, the daemon negotiates the telnet environment option, and the DTC returns the board and port numbers of the connecting device. telnetd then looks up the port and board numbers in the dp file and maps them to the pseudonym. See dp(4) for more information.

There are two ways to specify a port: either by explicitly giving its IP address, or by giving the IP address of the server and then specifying the

DTC Device File Access (DDFA) Software Component

board and port. Alternately, the server's IP address and the TCP service port address of the port can be given.

- **pcf - Port Configuration File.** This file is used by DDFA to configure specific port parameters. "pcf" is the generic name of the template file. In practice it is renamed for each port, and the values are altered appropriately for the device attached to the port. The pcf is referenced by an entry in the Dedicated Ports file (dp). The Dedicated Port Parser (dpp) parses the dp file and calls the Outbound Connection Daemon (ocd) program to spawn a daemon for each valid line in the dp file. A valid line is one in which the fourth field is the name of a pcf.
- **dpp - Dedicated Port Parser.** dpp parses the Dedicated Ports file (dp) and calls the Outbound Connection Daemon (ocd) to spawn a daemon for each valid entry in dp. It can be run from the shell or it can be included in netlinkrc to automatically run the DDFA software each time the system is booted.

dpp has one mandatory argument, the path to the dp file. The other arguments specify an optional log file, the path to ocd if it is not the default, and an option to kill existing processes when the dp file is parsed. dpp can also be run in check mode. In this case, dpp parses the dp file and reports any errors.

- **ocd - Outbound Connection Daemon.** ocd manages the connection and data transfer to the remote DTC port. Normally, it is spawned by the Dedicated Port Parser (dpp), but it can be run directly from the shell. If it is run from the shell, the name or IP address of the server or port and the pseudonym (device file name) must be entered on the command line. The board/port number or TCP service port address must be entered if the IP address does not explicitly name a port.

ocd creates a device file called `ym` for the connection. If ocd has an error condition which causes it to exit, it will normally remove the device file it created. If a device file is accidentally deleted, the corresponding ocd will continue running for at least 30 seconds before it detects the absence of the device file. The ocd will then write an error message to `/usr/adm/syslog` and exit.

Military Standards and Request for Comment Documents

- **ocdebug** - Outbound Connection Daemon debug mode. **ocdebug** is a special build of **ocd** which contains debug code. **ocdebug** has to be run from the shell with all the arguments that would normally come from the **dp** file entered on the command line: apart from **-d**, the arguments are the same as the **ocd** arguments. The **-d** option specifies the level of debugging messages.

Military Standards and Request for Comment Documents

To obtain information about available MIL-STD specifications, contact:

Department of the Navy
Naval Publications and Forms Center
5801 Tabor Avenue
Philadelphia, PA 19120-5099

To obtain information about available RFCs, contact:

Network Information Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Installing ARPA Services

This chapter describes how to install the HP 9000 ARPA Services product on your system.

Note If you have already installed ARPA Services: When you update your system to a new revision of the ARPA Services software, refer to the installation instructions in the “Read Me First” document.

Installation Steps

To install the ARPA Services product, perform these steps in order:

1. Update your network map.
2. Install the ARPA Services software.
3. Configure the mapping of host names to addresses.
4. Configure the ARPA Services software.
5. Reboot your system.
6. Check the ARPA Services installation.

2. Installing the ARPA Services Software

Each step is described in detail in this chapter.

1. Updating Your Network Map

Before you install the ARPA Services product, it is important to take the time to update your network map to indicate that ARPA Services is installed on your node. A network map provides you with information about the configuration of the computers on the network. As node manager, it is your responsibility to keep the network map up to date when you add or delete computers or make cable changes.

Refer to your link installation manual for information about creating and maintaining a network map.

Note If you are using HP's OpenView Network Node Manager to maintain a map of your network: After you configure ARPA services on your node and generate network traffic, your node will be discovered automatically and added to the map.

2. Installing the ARPA Services Software

Before you begin to install the software, make sure you have the correct operating system on your computer. *The HP-UX operating system, the required link software, and the ARPA Services software must all be the same version.* You can check your HP-UX operating system version with the `uname -r` command.

2. Installing the ARPA Services Software

To install the ARPA Services software, use the HP-UX update program. This program installs the ARPA Services file set. Because installing ARPA Services requires no kernel changes, the update program does not generate a new kernel when you install.

The update program is fully documented in the *HP-UX System Administration Tasks Manual*. You should read this manual before using the update program to install the ARPA Services software.

Files Created During Software Installation

The following daemons and servers are created when the ARPA Services software is installed:

rwhod	The daemon process that maintains the database used by the rwho and ruptime services
sendmail	The Simple Mail Transfer Protocol (SMTP) daemon
ftpd	The File Transfer Protocol (FTP) server
remshd	The remote shell server
rexecd	The remote execution server
rlogind	The remote login server
telnetd	The Telnet Protocol server
tftpd	The Trivial File Transfer Protocol server
fingerd	The Finger Protocol server
named	The BIND name server daemon

2. Installing the ARPA Services Software

gated	The dynamic routing daemon process
bootpd	The Bootstrap Protocol (BOOTP) daemon
ocd	Outbound connection daemon (for DDFA). ocdebug is the outbound connection daemon in debug mode for DDFA.

An additional daemon, `inetd`, is used by the ARPA Services. This daemon is provided by the link product and is started by `/etc/netlinkrc`.

The ARPA Services initialization script (`/etc/netbsdsrc`) starts `rwhod`, `gated`, `named`, and the `sendmail` daemons (if they are executable). Other ARPA Service servers are started on demand by `inetd`. `/etc/netbsdsrc` is invoked from the link initialization script `/etc/netlinkrc`. You do not need to change `/etc/netbsdsrc` unless you want to enable `rwhod` or `gated`.

Note All of the files and directories needed to use the BIND name service, `gated`, and `sendmail` are provided when the ARPA Services software is installed. Because the BIND name service and `gated` are both optional, configuration information is not documented in this chapter.

If you want to configure the BIND name server, read Chapter 4, “Configuring and Maintaining the BIND Name Server,” after you complete the steps described in this chapter. To configure `gated`, read Chapter 5, “Configuring `gated`,” after you complete the steps described in this chapter.

Because it may not be appropriate to use `sendmail` on all systems, `sendmail` installation and configuration information is not documented in this chapter. If you want to install and configure

3. Configure Host Name to Address Mapping

sendmail, read Chapter 6, “Internetwork Mail Routing,” after you complete the steps described in this chapter.

All of the daemons and servers provided with the ARPA Services product are described in detail in “Daemons and Servers” in Chapter 3, “Administering ARPA Services.”

3. Configure Host Name to Address Mapping

Host name to address mapping refers to the way in which your system determines a network address when it is given the name of a host.

Avoid assigning duplicate IP addresses whenever a PC or workstation is moved or assigned a new owner. Keep a written record of the owner of each device’s IP address. If a duplicate IP address were assigned, messages would show up at a timed interval whenever either one of the nodes that has one of the duplicate addresses attempts to connect to another node or when another node attempts to connect to one of the nodes that has a duplicate address. If there is no written record of who has the duplicate address, it can be difficult to notify the owners of the duplicate addresses to correct the problem (especially if the IP address is on a PC).

Choose Mapping Method

HP-UX provides three ways of mapping host names to addresses:

- BIND

3. Configure Host Name to Address Mapping

- NIS (Network Information Service)
- /etc/hosts

If you configure the BIND name server, the name server is used to resolve all host names and internet addresses. If the name server is not configured and you have configured the NIS (Network Information Service), NIS is used for this purpose. If NIS is not configured, /etc/hosts is used.

Configure Mapping

To configure your host name to address mapping, refer to the appropriate documentation:

- For BIND, see Chapter 4 in this manual.
- For NIS, refer to *Installing and Administering NFS*.
- For /etc/hosts, refer to the next section in this chapter, “4. Configure the ARPA Services Software”.

Verify Mapping

To verify how your system is mapping host names to addresses, use the `nslookup` utility, which is included in the ARPA services.

In the following examples, `nslookup` is used to determine the address of a host named `hpcndr`.

When the system is using BIND, an `nslookup` looks like the following:

```
$ nslookup lpcodr
Name Server: lpcodav.cod.lp.com
Address: 15.19.8.60

Name: lpcodr.cod.lp.com
Address: 15.19.8.119
```

4. Configure the ARPA Services Software

When the system is using NIS, an `nslookup` looks like the following:

```
$ nslookup lpcodr
Default YP Server:      lpcodbm
Address: 15.19.8.65
Aliases: lpcodbm-cod

Name:      lpcodr
Address: 15.19.8.119
Aliases: lpcodr-cod, lpcodr.cod.lp.com
```

When the system is using `/etc/hosts` rather than BIND or NIS, an `nslookup` looks like the following:

```
$ nslookup lpcodr
Using /etc/hosts on: lpcodav.cod.lp.com

Name:      lpcodr
Address: 15.19.8.119
Aliases: lpcodr-cod, lpcodr.cod.lp.com
```

4. Configure the ARPA Services Software

Most of the configuration files used by ARPA Services are provided when link software is installed, or are created during link configuration. With the exception of `/etc/hosts`, no further editing of the files is necessary.

The following are configuration files used by ARPA Services:

- | | |
|-----------------------------|--|
| <code>/etc/services</code> | A configuration file that associates each service name and aliases with the port number and protocol that each service uses. |
| <code>/etc/protocols</code> | A configuration file that contains the protocol names of all the protocols known by the local host. |

4. Configure the ARPA Services Software

<code>/etc/netbsdsrc</code>	A configuration file that starts the <code>gated</code> , <code>named</code> , <code>rwhod</code> , and <code>sendmail</code> daemons.
<code>/etc/inetd.conf</code>	The configuration file used by the ARPA Services/9000 daemon <code>inetd</code> .
<code>/usr/adm/inetd.sec</code>	A security file that can be used by the ARPA Services/9000 daemon <code>inetd</code> .
<code>/etc/hosts</code>	A configuration file that contains the internet addresses, host names and aliases of remote hosts on the network.
<code>/etc/networks</code>	A configuration file that contains the network addresses and names of networks known by the local host.
<code>/etc/ddfa/dp</code>	A text file that contains information for DDFA to set up and manage a connection to a specified DTC port.
<code>/etc/ddfa/pcf</code>	A configuration file used by DDFA to configure DTC port parameters.

Note Configuration files for the BIND name service are explained in Chapter 4, “Configuring and Maintaining the BIND Name Server.” Configuration files for `gated` are explained in Chapter 5, “Configuring `gated`.” Configuration files for `sendmail` are explained in Chapter 6, “Internetwork Mail Routing.”

Configuration Tasks

The following task *must be performed* to configure the ARPA Services software:

- Edit the `/etc/hosts` configuration file. The `/etc/hosts` file associates

4. Configure the ARPA Services Software

internet addresses with mnemonic names and alias names. It contains the names of hosts with which the local host can communicate. The services use this file to translate host names into internet addresses.

The following tasks *may be performed* to configure the ARPA Services software. All of these tasks are optional.

- Edit the `/etc/inetd.conf` configuration file. This is the configuration file used by the ARPA Services daemon `inetd`. You can modify this file if you have special requirements, but it is properly configured when you receive the link product.
- Edit the `/usr/adm/inetd.sec` security file. This is a security file that can be used by the ARPA Services daemon `inetd`. This file provides an extra security layer beyond the security check done by the services. You should configure this file if you want this additional security.
- Create an `/etc/ftpusers` security file. This file is used by the ARPA Services daemon `ftpd`. It provides an extra security layer when remote users login to the local host. You should create this file if you want additional security.
- Create an `/etc/hosts.equiv` configuration file. This file associates remote hosts with the local host. The purpose of this association is to identify “equivalent” hosts frequently accessed by the same users. You should create this file for your host if you want to use this feature.
- Create `$HOME/.rhosts` files. These files can be created and configured by any user to specify remote login names which are equivalent to that particular user’s login name.
- Create `$HOME/.netrc` files. These files can be created and configured by any user to specify login names and passwords to remote hosts so that `rexec` and `ftp` can execute without prompting the user for a password.
- Edit `/etc/ddfa/dp`. If the file does not exist, copy `/etc/newconfig/ddfa/dp` to `/etc/ddfa/dp`. Refer to the `dp` man page.
- Edit `/etc/ddfa/pcf`. If the file does not exist, copy

4. Configure the ARPA Services Software

`/etc/newconfig/ddfa/pcf` to `/etc/ddfa/pcf`. Refer to the `pcf` man page.

Adding Remote Connectivity (Editing `/etc/hosts`)

As node manager, you must configure this file for your host. You can add entries to this file either automatically with the System Administration Manager (SAM) or manually by editing the file. If you want to communicate with a large number of remote systems, and another node on your network has a configured `/etc/hosts` file, you may do the following:

- Use SAM to add connectivity information about a remote system that has an `/etc/hosts` file you can copy to your system. You may want to add connectivity information about several other key remote systems as well. The procedure for adding this information is described in the section called “Using SAM to Add Remote Connectivity (Edit `/etc/hosts`)”.
- From the other remote system, use the ARPA Services `ftp` command to append the more extensive `/etc/hosts` file to the `/etc/hosts` file on your system. (Note that if this results in duplicate entries for the same remote system, only the information in the first entry will be used in reaching that remote system.)

If you overwrite your local `/etc/hosts` file with a copy from another host, you may need to bring it up to date by adding unofficial aliases or unknown hosts, including your own host. If your host has more than one IP address (for multiple network interfaces), you must add entries for every IP address. These entries must have the same official host name but different aliases. This is so that different IP addresses (network interfaces) are distinguished (and can be referenced) by different aliases or hostnames.

You can copy the official host data base maintained at the Network Information Control Center (NIC) for ARPA Internet networks. Refer to chapter one for information on how to contact the NIC. Be sure to check the format of files received from the NIC.

4. Configure the ARPA Services Software

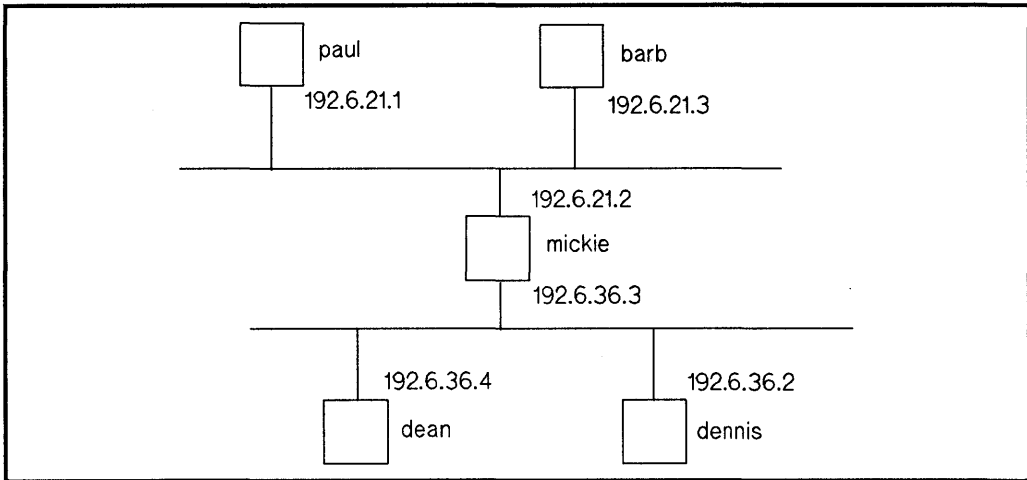


Figure 2-1. Multi-Homed Host Network Scheme

If your host accesses a multi-homed host (one with more than one link interface), make sure the internet address for that host is correct in the `/etc/hosts` file with respect to your host. For example, in the networking scheme shown in Figure 2-1, host paul and host barb access multi-homed host mickie via internet address 192.6.21.2. Hosts dean and dennis, on the other hand, access host mickie via internet address 192.6.36.3.

Using SAM to Add Remote Connectivity (Edit `/etc/hosts`)

SAM stands for System Administration Manager, a menu-driven utility for performing system administration tasks, including configuration of networking software.

Procedure

Note the following information before you begin:

- If your system is configured to use NFS NIS or the BIND Name Service for hostname-to-address mapping, you cannot use SAM to add or remove ARPA Services connectivity to a remote system. The `Add...` and `...Remove Connectivity Info About a Remote System` forms edit only the `/etc/hosts` file; they do not edit an NFS NIS or BIND Name Service

4. Configure the ARPA Services Software

database.

- If you must go through a gateway to reach the remote system that you are adding connectivity information about, SAM will prompt you for the gateway's hostname and IP address. With this information, SAM will automatically configure the necessary routing (by executing an `/etc/route add host` command and adding it to `/etc/netlinkrc`).
- If you use only one gateway to reach all systems on other parts of the network, use the Modify Default Gateway action (under the Internet Connectivity screen of the Remote Connectivity area) to avoid having to enter the same gateway information every time SAM prompts you for it.

The following steps tell how to use SAM to automatically add entries to `/etc/hosts`:

1. At the HP-UX prompt, type: `sam`
and wait for SAM's main menu to appear.
2. Select the Networking/Communications menu item.
3. Select the Remote System Connectivity menu item.
4. Select the Internet Connectivity menu item.
5. Select the Add action.

Note

The Remove action lets you delete `/etc/hosts` file entries. If you have to reach the remote system through a gateway, this action also removes the associated `/etc/route add host` command from the `/etc/netlinkrc` file.

6. Fill in the form according to its instructions. View the help screens for information about filling in the form.

4. Configure the ARPA Services Software

7. Select `apply` to enter additional names of systems to be configured (use `apply` as a shortcut to remain in the add screen). Then, press `OK` when you are done with the screen.
8. Repeat steps 4 through 7 to add connectivity to more remote systems.
9. Exit the Internet Connectivity screen by selecting `Exit` from the List menu. From the Remote System Connectivity screen, select `Exit SAM` to exit from SAM.

Verification

To view the list of remote systems you may communicate with, type the following command at the HP-UX prompt:

```
more /etc/hosts
```

To verify that `/etc/hosts` is being used to do host name to address mapping, use `nslookup` as described in the previous section, “3. Configure Host Name to Address Mapping.”

To view the destinations reached through gateways and the gateways used to reach those destinations, type the following command at the HP-UX prompt:

```
netstat -r
```

The listing from this command may appear slowly, as it attempts to find the names associated with the network addresses used to perform routing.

To verify that you can communicate with a remote system via ARPA Services, see the “Checking the ARPA Services Installation” section at the end of this chapter.

4. Configure the ARPA Services Software

Specifying a New Default Gateway

To replace the current default gateway (if there is one), select the Modify Default Gateway Action from the Internet Connectivity menu (under the Remote System Connectivity area).

Deleting the Default Gateway

If you want to delete the default gateway that you added with SAM's Specify the Default Gateway form, you must do it manually with the following:

1. Enter the following command at the HP-UX prompt:

```
/etc/route delete default gateway_hostname
```

2. Edit the `/etc/netlinkrc` file to remove the corresponding `/etc/route add default` entry for the gateway.

Manually Adding Remote Connectivity (Editing `/etc/hosts`)

Each host (including the local host) has a one line entry in the `/etc/hosts` file. Each entry in the `/etc/hosts` file takes the following form:

```
internet_address official_host_name [alias(es)]
```

internet_address The network address that uniquely identifies the node. *Internet_address* must be in dot notation. Refer to the "Configure Host Name to Address Mapping" and "Edit etc/hosts" sections earlier in this chapter for more information about internet addresses.

official_host_name The name of the node. Host names can contain any printable character except white spaces, newline, or the comment character (`#`). By convention, *official_host_name* should be the same as the system host name assigned with the HP-UX `hostname` command.

4. Configure the ARPA Services Software

alias(es) The common name or names for the node. An *alias* is a substitute for *official_host_name*. *Alias* names are optional and are not supported by all the commands that use `/etc/hosts`.

Mapping Verification

To verify that `/etc/hosts` is being used to map host names to addresses, use `nslookup` as described in the previous section, “3. Configure Host Name to Address Mapping.”

Example

The `/etc/hosts` entry for a node with the address 192.45.36.5, the official host name *hpdxsg*, and the alias name *bullfrog* looks like the following:

```
192.45.36.5 hpdxsg bullfrog
```

Format for `/etc/hosts`

- Lines cannot start with a white space (tabs or blanks).
- The fields can have any number of blanks or tab characters separating them.
- Comments are allowed and are designated by a “#” in front of the comment text.
- Trailing blanks and tab characters are allowed.
- Blank lines are allowed.

Permissions

The `/etc/hosts` file should be owned by user *root*, group *other* and have 0444 (r—r—r—) permission.

4. Configure the ARPA Services Software

Refer to the `/etc/hosts` file for examples of the actual format and contents. For more information on `/etc/hosts`, refer to the `hosts` entry in the *HP-UX Reference*.

Editing `/etc/inetd.conf`

The `/etc/inetd.conf` file is the configuration file for the internet daemon, `inetd`. This daemon listens on the well-known port associated with each service listed in this configuration file, and starts an appropriate server for each service when it receives a connection request on its port.

You can modify this file if you have special requirements, but it is properly configured when you receive it with the LAN product.

Anytime `inetd` is started up, it reads the `/etc/inetd.conf` file. If you modify the `/etc/inetd.conf` entry for a service, use the `inetd -c` command to reconfigure `inetd` while it is still running. The `inetd -c` command is explained in Chapter 3, "Administering ARPA Services."

The `/etc/inetd.conf` file contains an entry for each ARPA server started by `inetd` on your host, with the exception of `rcp`, whose server is `remshd`. `sendmail`, `named`, and `gated` provide their own daemons, and their servers are not started by `inetd`.

Each entry, which must be complete on a single line, contains the following fields:

service_name
socket_type
protocol
wait/nowait
user
server_program
program_number (for NFS and RPC services only)
version_number (for NFS and RPC services only)
server_prog_args

4. Configure the ARPA Services Software

<i>service_name</i>	Name of a valid service listed in the <code>/etc/services</code> file (or the name <code>rpc</code> for an NFS RPC service).
<i>socket_type</i>	<code>stream</code> or <code>dgram</code> , depending on the type of the service's server socket.
<i>protocol</i>	Name of a valid protocol with an entry in the <code>/etc/protocols</code> file.
<i>wait/nowait</i>	<code>wait</code> instructs <code>inetd</code> not to start another server of the type in this entry until the first one completes. (This is for datagram sockets only). <code>nowait</code> instructs <code>inetd</code> to start a server for the service whenever a request arrives. <code>nowait</code> must be used for all services using <code>stream</code> sockets.
<i>user</i>	Name of the user with whose permission the server should run.
<i>server_program</i>	Absolute pathname of the program that <code>inetd</code> should run when a request on that server's socket is made.
<i>program_number</i>	A unique number which defines a particular service grouping. This is used by NFS Services only. Refer to the NFS documentation for more information.
<i>version_number</i>	The version supported by the RPC service. This number can be a single value or a range if the program handles multiple versions, for example, 1 or 1-3. Version numbers enable old and new protocols to share the same server process. This is used by NFS Services only. Refer to the NFS documentation for more information.
<i>server_prog_args</i>	Normal program arguments, starting with <code>argv[0]</code> , which is the program name. For more information on program arguments, refer to the <code>exec</code> entry in the <i>HP-UX Reference</i> .

4. Configure the ARPA Services Software

Example

The `/etc/inetd.conf` entry for a service with the following:

- The service name `ftp`.
- The socket type `stream`.
- The protocol `tcp`.
- The wait/nowait value `nowait`.
- The user permissions of `root`.
- The server path name `/etc/ftpd`.
- The server program arguments `ftpd -l`.

looks like:

```
ftp stream tcp nowait root /etc/ftpd ftpd -l
```

Format for `/etc/inetd.conf`

Follow these guidelines when formatting `/etc/inetd.conf`:

- Each field must be separated by one or more spaces or tab characters.
- Entries can be continued to the next line if the first line terminates with a backslash (`\`) followed by a newline.
- Comments are allowed and are designated by a pound sign (`#`) in front of the comment text.
 - Trailing blanks and tab characters are allowed.
 - Blanks are allowed.

4. Configure the ARPA Services Software

Note If you make a change to `inetd.conf` and `inetd` is already running, you must execute `inetd -c` for `inetd` to recognize the changes.

Permissions

The `/etc/inetd.conf` file should be owned by user *root*, group *other* and have 0444 (`-r—r—r—`) permission. Refer to the `/etc/inetd.conf` files for examples of the actual format and contents. For more information on `/etc/inetd.conf`, refer to the `inetd.conf` entry in section 4 of the *HP-UX Reference*.

Adding Security (Editing `/usr/adm/inetd.sec`)

The `/usr/adm/inetd.sec` file is a security file used by the daemon `inetd`. As node manager, you must configure this file for your host if you want to use it.

The `/usr/adm/inetd.sec` file provides an extra security layer beyond any security check done by the services. It allows the node manager to determine how many remote services can run simultaneously on the local host and which hosts or networks are allowed to remotely use the local host. This check is done before the service's security check. The `inetd` daemon reads the `/usr/adm/inetd.sec` file and checks the address of any host requesting a service. The `inetd` daemon only allows the requesting host to access a particular service if it is not forbidden by `/usr/adm/inetd.sec`.

Note If `inetd` is running, it rereads `/usr/adm/inetd.sec` when you make changes to it. Your changes are applied to any services started up after the file is reread, but not to any services currently running.

4. Configure the ARPA Services Software

Setting the Maximum Number of Remote Connections

If you choose to put a limit on the number of services that can be used remotely at any one time, use the following entry in the first line of `/usr/adm/inetd.sec`:

```
MAXNUM number
```

where *number* is the maximum number of simultaneous remote services allowed.

If MAXNUM is declared, it must be the first line of data in the file. MAXNUM default is 1000.

Allowing and Denying Nodes Access to Services

You can allow and deny remote systems access to local ARPA Services by using SAM or by manually editing the `/usr/admin/inetd.sec` file.

Using SAM to Add Security (Edit `/usr/adm/inetd.sec`)

SAM does not allow you to set the maximum number of connections by which remote users access local services. If you want to set this number, edit `/usr/adm/inetd.sec` manually.

Procedure

The following steps tell how to use SAM to allow or deny remote systems access to local ARPA Services:

1. At the HP-UX prompt, type:

```
sam
```

and wait for SAM's main menu to appear.

2. Select the Networking/Communications menu item.
3. Select the Security menu item.

4. Configure the ARPA Services Software

4. Select the Internet Service menu item.
5. To modify a single service, highlight it, and then select the Modify action item. Alternatively, you can choose the Modify All Services action to change security for all services.
6. Fill in the form according to its instructions. View the help screens for information about filling in the form.
7. Select apply to enter additional names of systems to be configured (use apply as a shortcut to remain in the add screen). Then, press OK when you are done with the screen.
8. Repeat steps 5 through 7 to allow or deny remote systems access to other local ARPA Services.
9. Exit the Internet Service screen by selecting exit from the List menu. From the Security screen, select Exit SAM to exit from SAM.

Verification

Besides using SAM, you can also view current security restrictions on local ARPA Services by typing the following command at the HP-UX prompt:

```
more /usr/adm/inetd.sec
```

Manually Editing /usr/adm/inetd.sec

Each entry in /usr/adm/inetd.sec has the following form:

Syntax

```
service_name [allow] host_specifier(s)  
[deny ]
```

service_name name of a valid service with an entry in /etc/services. NFS Services only: The service name for RPC-based services is the name (not alias) of a valid service in the file /etc/rpc.

4. Configure the ARPA Services Software

<code>allow/deny</code>	<code>allow</code> instructs <code>inetd</code> to approve the host for access to the specified service. <code>deny</code> instructs <code>inetd</code> to deny host access to the specified service.
<i>host_specifier(s)</i>	names or IP addresses of hosts or networks. Alias names are not allowed. The wild card character "*" and the range character "-" can be used in any field of a host or network address. The host name (as returned by <code>gethostbyname</code>), the network name (as returned by <code>getnetbyname</code>), and the IP address of the host requesting a connection will be compared to each <i>host_specifier</i> .

If a remote service is not mentioned in the `/usr/adm/inetd.sec` file, then it is not restricted by `inetd`.

If an entry in the `/usr/adm/inetd.sec` file specifies the service name and nothing else, then `inetd` allows all hosts to attempt to access this service.

Format for `/usr/adm/inetd.sec`

Follow these guidelines when formatting `/usr/adm/inetd.sec`:

- If `MAXNUM` is declared it must be the first line of data in the file.
- Commented lines must have the comment character (`#`) in the first column. Comment characters are not allowed at the end of a line.
- Entries can continue to the next line if the first line is terminated with a backslash (`\`) followed by a newline.
- One or more blanks or tab characters should separate each *host_specifier* listed in one entry.
- If there is more than one entry for a service, only the first one is used.

4. Configure the ARPA Services Software

Example

The following example describes the type of security restrictions you may specify within `/usr/adm/inetd.sec`.

```
MAXNUM 15
login   allow 10.* 192.54.24.5
shell   deny 11.3-5.* lark 192.54.24.8
ftp
```

This example does the following:

- Allows access rights to `rlogin` from any network address which starts with 10.
- Allows access rights to `rlogin` from the host with the address 192.54.24.5.
- Denies access to `remsh` and `rcp` from the following hosts:
 - Hosts with network address 11, hosts 3.0.0 through 5.255.255.
 - All hosts in the network `lark` (as mapped by `getnetbyname`).
 - The host with the address 192.54.24.8.
- Allows access rights to `ftp` from all hosts.

If you have the following entry:	The following access is allowed:
<code>ftp allow *</code>	Access to <code>ftp</code> is allowed to everyone.
<code>ftp allow</code>	Entry is ignored and a warning message sent to the log file <code>/usr/adm/inetd.log</code> .
<code>ftp deny *</code>	Access to <code>ftp</code> is denied to everyone.

4. Configure the ARPA Services Software

Permissions

The `/usr/adm/inetd.sec` file should be owned by user *root*, group *other* and have 0444 (-r—r—r—) permission.

For more information on `/usr/adm/inetd.sec`, refer to the `inetd.sec` entry in section 4 of the *HP-UX Reference*.

Modifying FTP Access (Creating `/etc/ftpusers`)

The `/etc/ftpusers` file is a security file for `ftpd`. You must create this file if you want to use it. `ftpd` checks locally for this file before allowing a remote login to the local host. If the remote user specifies an account on the local host that is listed in `/etc/ftpusers`, the remote connection is denied.

Using SAM to Modify FTP Access (Create `/etc/ftpusers`)

You can use SAM to create `/etc/ftpusers` or you can manually create and edit this file.

Procedure

The following steps tell how to use SAM to prevent remote users from using `ftp` to perform file management operations and file transfers on your local host:

1. At the HP-UX prompt, type:

```
sam
```

and wait for SAM's main menu to appear.

2. Select the Networking/Communications menu item.
3. Select the Security menu item.
4. Select the Internet Service menu item.

4. Configure the ARPA Services Software

5. Select the ftp service and choose the Modify action. Highlight the "Select Denied Users" item and fill in the list of users you wish to deny access to the ftp service. Select Add to add each user to the list.
6. Select OK when you are done with the screen.
7. Exit the Internet Security screen by selecting exit from the List menu. From the Security screen, select Exit SAM to exit from SAM.

Verification

To view the remote users prevented from using ftp to perform file management operations and file transfers on your local host, type the following command at the HP-UX prompt:

```
more /etc/ftpusers
```

You may also perform steps 1-6 in the previous procedure.

Manually Modifying FTP Access (Editing /etc/ftpusers)

Each entry in the /etc/ftpusers file must take the following form:

user_name

Format for /etc/ftpusers

Follow these guidelines for editing /etc/ftpusers:

- One user name per line.
- No white space allowed.
- Lines with comments (or other information that does not exactly match user account names) are ignored.

Example

You should include user names that specify a restricted login shell. If the contents of an /etc/ftpusers file are the following:

4. Configure the ARPA Services Software

```
uucp
guest
```

The local `ftpd` denies an `ftp` login to the local accounts named `uucp` and `guest`.

Permissions

The `/etc/ftpusers` file should be owned by user `root`, group `other`, and have 0444 (`-r—r—r—`) permission.

For more information on `/etc/ftpusers`, refer to the `ftpusers` entry in section 4 of the *HP-UX Reference*.

Permitting Easy Access from Remotes (Creating `/etc/hosts.equiv`)

The `/etc/hosts.equiv` file associates remote hosts with the local host. `rcp`, `remsh` and `rlogin` may use `/etc/hosts.equiv`. The purpose of this association is to identify “equivalent” hosts frequently accessed by the same users. You must create `/etc/hosts.equiv` for your host if you wish to use it.

To illustrate how `/etc/hosts.equiv` works, assume that a user from a remote host attempts to log in to the local host using `rlogin`. If the remote host is listed in the local `/etc/hosts.equiv` file and the remote user’s login name matches a login name on the local host, then the user is not prompted for a password.

Note that if you are logged in as a super-user, `/etc/hosts.equiv` is bypassed. Only `.rhosts` in the super-user’s home directory is checked. With any other user name, `/etc/hosts.equiv` is checked before `$HOME/.rhosts`.

You can use SAM to create `/etc/hosts.equiv` or you may manually create and edit this file.

4. Configure the ARPA Services Software

Using SAM to Permit Easy Access from Remotes(Create /etc/hosts.equiv)

If you have already manually configured an `/etc/hosts.equiv` file with entries other than those of the form:

hostname
or
hostname username

do not use SAM to configure `/etc/hosts.equiv`. SAM does not recognize, display or add entries of other forms (such as +, -, %, or +@example_nfsnetgroup).

Procedure

The following steps tell how to use SAM to allow a remote system's users to bypass password security on your local host if their login name matches a local login name:

1. At the HP-UX prompt, type: `sam`
and wait for SAM's main menu to appear.
2. Select the Networking/Communications menu item.
3. Select the Security menu item.
4. Select the Remote Login menu item.
5. Select the Add action.
6. Fill in the form according to its instructions. View the help screens for information about filling in the form.
7. Select apply to enter additional names of systems to be configured (use apply as a shortcut to remain in the add screen). Then, press OK when you are done with the screen.

4. Configure the ARPA Services Software

8. Repeat steps 5 through 7 to allow or deny remote systems' easy access (via rcp, remsh, or rlogin) to the local system.
9. Exit the Remote Login screen by selecting exit from the List menu. From the Security screen, select Exit SAM to exit from SAM.

Note The Required choice deletes all `/etc/hosts.equiv` file entries for the remote system you specify. Use this choice to remove unwanted entries. You may also modify the list of users that are currently allowed access without a password. Choose Select - Not Required and change the list of remote login names as desired.

Verification

To view the list of remote systems whose users are allowed to bypass password security on your local host, type the following command at the HP-UX prompt:

```
more /etc/hosts.equiv
```

Note If an entry consists of a remote system's hostname only, then all remote users are allowed to bypass password security on your local host if their login name matches a local login name.

You may also perform steps 1-7 of the previous procedure, and enter the hostname of a remote system whose list of users you want to see.

Manually Permitting Access from Remotes (Creating `/etc/hosts.equiv`)

Follow these guidelines when creating `/etc/hosts.equiv`

- Each entry contains a valid official host name or a comment. If you have

4. Configure the ARPA Services Software

also installed the NFS Services product, an entry may contain a valid network group name.

- Host names can contain any characters except blanks, tabs, comment characters (#), or newline.
- If you have also installed the NFS Services product, network group names must be preceded by the characters +@ or -@. See the explanation below.
- If your system is a member of a diskless cluster, you may use % to represent all hosts in the cluster (the root server and the cluster nodes).

An entry in `/etc/hosts.equiv` takes the following form:

host_name

host_name corresponds to a valid official host name as returned by `gethostbyaddr`. For an explanation of how official host names are determined, refer to `gethostent` in section 3N of the *HP-UX Reference*.

If you have also installed the NFS Services product, an entry in `/etc/hosts.equiv` can also take the following form:

...+@group
...-@group
...+

4. Configure the ARPA Services Software

Where:	Means:
<i>group</i>	the valid network group as defined in /etc/netgroup. Refer to <i>Using and Administering NFS Services</i> and the netgroup entry in section 4 of the <i>HP-UX Reference</i> for details on /etc/netgroup.
+@ <i>group</i>	all hosts in that network group are allowed.
-@ <i>group</i>	all hosts in that network group are not allowed.
+	everyone is allowed.

Permissions

The /etc/hosts.equiv file should be owned by user *root*, group *other* and have 0444 (-r—r—r—) permission.

Note This file creates a significant security risk.

For more information on /etc/hosts.equiv, refer to the hosts.equiv entry in section 4 of the *HP-UX Reference*.

Creating \$HOME/.rhosts

\$HOME/.rhosts files can be created and configured by any user to specify remote login names that are equivalent to the user's login name. rcp, remsh and rlogin may use \$HOME/.rhosts. You must create this file for the home directory of the super-user account if you wish to use "equivalent" login names.

The local host allows a remote user with a login listed in a local \$HOME/.rhosts file to log in to that local user's account without specifying a password.

4. Configure the ARPA Services Software

`$HOME/.rhosts` must be owned by the local user and must not be a symbolic link.

The `/etc/hosts.equiv` file is checked before `$HOME/.rhosts`. If an entry is found in `/etc/hosts.equiv`, `$HOME/.rhosts` is not checked. For NFS Services only: `-@` entries are not absolute. If a user is excluded by a minus entry from `hosts.equiv` but included in `.rhosts`, then that user is allowed.

If you are logged in as super-user, the `/etc/hosts.equiv` file is bypassed and the `.rhosts` file in the super-user's home directory is checked if it exists.

Using SAM to Create a `$HOME/.rhosts` File for the Local Super-User Account

You can use SAM to create a `$HOME/.rhosts` file for the local super-user account only. (You cannot create `$HOME/.rhosts` files for local non-super-user accounts with SAM.)

If you have already manually configured a `$HOME/.rhosts` file and did not use entries of the following forms:

hostname
or
hostname username

do not use SAM to configure `$HOME/.rhosts`. SAM does not recognize, display or add entries of other forms (such as `+`, `-`, `%`, or `+@example_nfsnetgroup`).

Procedure

The following steps tell how to use SAM to let a remote system's user(s) become super-user on your local host without having to enter a password:

1. At the HP-UX prompt, type:

`sam`

and wait for SAM's main menu to appear.

4. Configure the ARPA Services Software

2. Select the Networking/Communications menu item.
3. Select the Security menu item.
4. Select the Remote Login menu item.
5. Select the Add action.
6. Fill in the form according to its instructions. View the help screens for information about filling in the form.

Note

If you are the super-user, the Required choice deletes all \$HOME/.rhosts file entries for the remote system you specify. Use this choice to remove unwanted entries.

7. Select apply to enter additional names of systems to be configured (use apply as a shortcut to remain in the add screen). Then, press OK when you are done with the screen.
8. Exit the Remote Login screen by selecting exit from the List menu. From the Security screen, select Exit SAM to exit from SAM.

Verification

To view the list of remote systems whose users are allowed to bypass password security on your local host, type the following command (as super-user) at the HP-UX prompt:

```
more $HOME/.rhosts
```

You may also perform steps 1-7 of the previous procedure, and enter the hostname of a remote system whose list of users you want to see.

Manually Creating \$HOME/.rhosts

Follow these guidelines when creating \$HOME/.rhosts:

4. Configure the ARPA Services Software

- Each entry contains a valid official host name and a remote login name.
- If you have also installed the NFS Services product, an entry can also be a valid network group name followed by a white space and a remote user name or group name.
- The host name and login name are separated by any number of tabs or blanks.
- If you have also installed the NFS Services product, network group names must be preceded by the characters +@ or -@. See the explanation below.
- If your system is a member of a diskless cluster, you may use % to represent all hosts in the cluster (the root server and the cluster nodes).
- Comments can follow the login name on any entry.

An entry looks like:

host_name remote_user_name

If you have also installed the NFS Services product, an entry can also look like:

+@group remote_user_name | remote_group_name
-@group remote_user_name | remote_group_name

Where:	Means:
<i>group</i>	the valid network group name defined in /etc/netgroup. Refer to <i>Using and Administering NFS Services</i> and the netgroup entry in the <i>HP-UX Reference</i> for details on /etc/netgroup.
<i>+@group</i>	the <i>remote_user_name</i> or all users in <i>remote_group_name</i> in that network group are allowed.
<i>+@group</i>	the <i>remote_user_name</i> or all users in <i>remote_group_name</i> in that network group are not allowed.

Permissions

4. Configure the ARPA Services Software

Each `$HOME/.rhosts` file should be owned by the user, with 0600 (-rw-----) permission. The user's home directory should be protected so that no one else can read it. Even if users do not want to use an `.rhosts` file, it is important to write-protect their home directories so no one can create an `.rhosts` file in someone else's directory and then have unauthorized permissions to use that account.

Caution This file creates a significant security risk. To disable its use on your host, refer to "Disabling Use of `$HOME/.rhosts`" later in this chapter.

For more information on `$HOME/.rhosts`, refer to the `hosts.equiv` entry in section 4 in the *HP-UX Reference*.

Creating `$HOME/.netrc`

`$HOME/.netrc` files can be created and configured by any user. This file is used to specify login names and passwords to remote hosts so that `rexec` and `ftp` can execute without prompting the user for a password.

Format for `$HOME/.netrc`

- Each entry contains a valid host name, a remote login name and an optional password. (These are each preceded by a keyword). If you do not specify a password, the service prompts for one. If there is no password for the user name, press **Return** when the prompt appears.
- The fields are separated by tabs, commas, or blanks.
- Each host name is followed by only one logical name and password.

An entry looks like:

```
machine host_name login remote_login_name password password
```

4. Configure the ARPA Services Software

Example

The `$HOME/.netrc` entry for the remote account:

- On the host `hpdxsg`.
- With the login name `carolyn`.
- With the password `driveway`.

looks like:

```
machine hpdxsg login carolyn password driveway
```

For information about other items that may be included in the `$HOME/.netrc` file, refer to the *HP-UX Reference*.

Permissions

Each `$HOME/.netrc` file should be owned by the user, with `0600 (-rw-----)` permission. The user's home directory should be protected so that no one else can read it or write to it.

Note Using `$HOME/.netrc` creates a serious security risk. Passwords in this file are unencrypted.

For more information, refer to the `netrc` entry in the fourth section of the *HP-UX Reference*.

6. Checking the ARPA Services Installation

5. Rebooting Your System

After you have completed Steps 1 through 4 (as shown in the “Installation Steps” section), you are ready to reboot your system with the ARPA Services product configured and initialized. Once rebooted, the ARPA Services product will be running.

To reboot your system, use the following shell script:

```
/etc/shutdown -r
```

6. Checking the ARPA Services Installation

Once your ARPA Services software is installed, configured, and initialized, ensure that the software is operating correctly by performing the following tasks:

- Ensure that the ARPA Services are actually installed. To do this, invoke the `ftp` program by typing `ftp` on the command line. For more information, refer to the `ftp` entry in section 1 of the *HP-UX Reference*.
- Ensure that `inetd` correctly starts the servers on your local host by using `ftp` to the local host.

For example, if you just installed ARPA Services on the host `neptune`, do the following:

```
$ ftp neptune  
Connected to neptune.
```

If you successfully connect to the local host from the local host, `inetd` is correctly starting the `ftp` server. If this fails, ensure that `inetd` is running. To do this, issue the command:

```
/bin/ps -ef | fgrep inetd
```

6. Checking the ARPA Services Installation

If you do not see an entry for `inetd`, start it by typing `/etc/inetd` on the command line. Remember, you must be super-user to start `/etc/inetd`. Then try `ftp` again.

If `ftp` fails again, refer to the troubleshooting section of either the LAN or X.25 installation manual.

Administering ARPA Services

This chapter provides information for administering the ARPA Services product on your system. The information is presented in two sections:

- Daemons and Servers.
- System Log Files.

"Daemons and Servers" provides a quick reference list of the daemons and servers that are provided with the ARPA Services product. "System Log Files" describes how to configure and maintain log files for ARPA Services.

Daemons and Servers

This section describes the daemons and servers used by ARPA Services.

Daemons

The following daemons are used by the ARPA Services product:

<code>rwiod</code>	The daemon process that maintains the database used by the <code>rwio</code> and <code>ruptime</code> services. (<code>rwio</code> and <code>ruptime</code> are not supported on X.25 networks or networks using the PPL (SLIP) product.)
<code>sendmail</code>	The Simple Mail Transfer Protocol (SMTP) daemon.

Daemons and Servers

<code>inetd</code>	The internet daemon. It is provided with the link product.
<code>named</code>	The BIND name service daemon.
<code>gated</code>	The dynamic routing daemon.

Each TCP service has its own server and its own assigned port on your host. The service names and their associated port numbers are listed in the configuration file called `/etc/services`.

The `rwho` and `ruptime` services access the `rwhod` daemon (which uses UDP) and its assigned port number on your host. The entry in the `/etc/services` file has the service name `who`.

Note For a description of the `named` daemon, see Chapter 4, “Configuring and Maintaining the BIND Name Server.” For a description of the `gated` daemon, see Chapter 5, “Configuring Gated.” For a description of the `sendmail` daemon, see Chapter 6, “Internetwork Mail Routing.”

`rwhod` Daemon

The `rwhod` daemon broadcasts messages on the network and maintains a database for the `rwho` and `ruptime` commands.

The `rwhod` daemon is started by the ARPA Services initialization script, `/etc/netbsdsrc`. By uncommenting the lines that start `rwhod` in `/etc/netbsdsrc`, you can set up `rwhod` to start automatically whenever the node does. To run `rwhod` without rebooting your system, run `/etc/rwhod` as the super-user.

`rwhod` checks the state of your host and constructs status messages, which it broadcasts on the network. These status messages are generated by `rwhod` every 180 seconds. The daemon listens on the network for status messages broadcast by `rwhod` daemons on remote hosts. When these messages are received, `rwhod` records them in a database of files in `/usr/spool/rwho`. The files are named `whod.hostname`, where *hostname* is the name of the remote host from which the status information came.

Because UDP broadcasts do not go through gateways, `rwho` and `ruptime` do not report status for hosts that can only be reached through a gateway.

Status information collected by `rwhod` for the local host and from each remote host includes the following:

- System load average.
- Host name as returned by `gethostbyname`.
- Users logged-in.
- Time of last activity for logged-in users.

For more information on `rwhod`, see the `rwho` entry in the *HP-UX Reference*.

Daemons and Servers

inetd Daemon

The `inetd` daemon is a master server for all the ARPA/Berkeley Services TCP connections. It invokes the network servers upon demand, and therefore reduces the system load on your host and provides a more maintainable network configuration. When a connection to a particular service is requested, `inetd` creates a connection and invokes the server for that service.

The `inetd` daemon is started (with no options) by the link initialization script `/etc/netlinkrc` when the system is started. After `inetd` is started, it reads two configuration files: `/etc/inetd.conf` to determine what services to support, and `/usr/adm/inetd.sec` to determine which security checks to perform. (The `/usr/adm/inetd.sec` file is optional).

If you want to write your own service and want to tie it in to `inetd`, refer to the *Berkeley IPC Programmer's Guide*.

For more information on `inetd`, see the `inetd` entry in section 1M of the *HP-UX Reference*.

Note `inetd` is also used by the NFS Services product. Refer to *Using and Administering NFS Services* if you are using NFS Services.

Servers

The following servers are used by the ARPA Services product:

remshd	The remote shell server.
rlogind	The remote login server.
rexecd	The remote execution server.
fingerd	The Finger Protocol server.
telnetd	The Telnet Protocol server.
ftpd	The File Transfer Protocol (FTP) server.
tftpd	The Trivial File Transfer Protocol (TFTP) server.
bootpd	The Bootstrap Protocol (BOOTP) server.

For more information on tftpd and bootpd, see chapter 7, "Configuring TFTP and BOOTP Servers."

Daemons and Servers

Remote Shell Server

The remote shell server, called `remshd`, is the server for the `remsh` and `rcp` commands. It provides remote execution facilities.

`remshd` is called by `inetd` when a connection request is made for the port assigned to this service.

To use `remshd`, the following entry must be in `/etc/inetd.conf`:

```
shell stream tcp nowait root /etc/remshd remshd
```

This entry is already configured in the default `/etc/inetd.conf` file shipped with the link product.

For more information on `remshd`, see the `remshd` entry in section 1M of the *HP-UX Reference*.

Note `remshd` uses the `/etc/hosts.equiv` and `$HOME/.rhosts` files if they exist. Remember that these files create serious security risks. For more information about the `$HOME/.rhosts` file, see "Disabling Use of `$HOME/.rhosts`."

Remote Login Server

The remote login server, called `rlogind`, is the server for the `rlogin` command. It provides a remote login facility.

`rlogind` is called by `inetd` when a connection request is made for the port assigned to this service. To use `rlogind`, the following entry must be in `/etc/inetd.conf`:

```
login stream tcp nowait root /etc/rlogind rlogind
```

This entry is already configured in `/etc/inetd.conf`, the default file shipped with the link product.

When `rlogind` is invoked, it gets a pseudo terminal (`pty`) and sets it up as `stdin`, `stdout`, and `stderr` for the login process. The pseudo terminal files should be set up correctly on your host. If they are not, refer to the *HP-UX System Administration Tasks Manual* for information on how to correctly set up the `pty` device files.

Note

`rlogind` uses the `/etc/hosts.equiv` and `$HOME/.rhosts` files if they exist. Remember that these files create serious security risks. For more information about the `$HOME/.rhosts` file, see the next section, "Disabling Use of `$HOME/.rhosts`." For more information on `rlogind`, refer to section 1M of the *HP-UX Reference*.

Disabling Use of `$HOME/.rhosts`

Because of the possible security risk, you may wish to prevent users from using the `$HOME/.rhosts` file. To do this, configure `rlogind` and `remshd` with the `-l` option so that they start from `inetd`. For example:

```
login stream tcp nowait root /etc/rlogind rlogind -l
shell stream tcp nowait root /etc/remshd remshd -l
```

Daemons and Servers

Now any `$HOME/.rhosts` files are ignored, and only `/etc/hosts.equiv` is used to determine equivalence of remote hosts and users.

Remote Execution Server

The remote execution server, called `rexecd`, is the server for the `rexec` library routine. It provides remote execution facilities with access based on user name and password.

`rexecd` is called by `inetd` when a connection request is made for the port assigned to this library routine.

To use `rexecd`, the following entry must be in `/etc/inetd.conf`:

```
exec stream tcp nowait root /etc/rexecd rexecd
```

A user name can be specified instead of `root`, since `rexecd` does not require super-user privileges. However, make sure that the permissions of the user names are adequate for anything you would want `rexec` to access on the remote host.

This entry is already configured in the default `/etc/inetd.conf` file shipped with the link product.

For more information on `rexecd`, see the `rexecd` entry in section 1M of the *HP-UX Reference*.

Finger Protocol Server

The Finger protocol server, called `fingerd`, allows remote finger lookups with the `finger` command.

`fingerd` is called by `inetd` when a connection request is made for the port assigned to this service. To use `fingerd`, the following entry must be in `/etc/inetd.conf`:

```
finger stream tcp nowait bin /etc/fingerd fingerd
```

The entry is commented out in the default file `/etc/inetd.conf`, which is shipped with the link product. To configure the server, uncomment the entry.

For more information on `fingerd`, see the `fingerd` entry in section 1M of the *HP-UX Reference*.

Telnet Protocol Server

The Telnet server, called `telnetd`, is the server for the ARPA Telnet Protocol.

`telnetd` is called by `inetd` when a connection request is made for its port address, which is listed in `/etc/services`.

To use `telnetd`, the following entry must be in `/etc/inetd.conf`:

```
telnet stream tcp nowait root /etc/telnetd telnetd
```

This entry is already configured in the default `/etc/inetd.conf` file shipped with the link product.

When `telnetd` is invoked, it gets a pseudo terminal (`pty`) and sets it up as `stdin`, `stdout`, and `stderr` for the login process. The pseudo terminal files should be set up correctly on your host. If they are not, refer to *HP-UX System Administration Tasks Manual* for information on how to correctly set up the `pty` device files.

Daemons and Servers

For more information on `telnetd` see the `telnetd` entry in section 1M of the *HP-UX Reference*. Included in this entry is a complete list of error messages.

Internet File Transfer Protocol Server

The internet file transfer protocol server, called `ftpd`, is the server for `ftp`. It provides a file transfer facility with access based on user names and passwords.

`ftpd` is called by `inetd` when a connection request is made for the port assigned to this service.

To use `ftpd`, the following entry must be in `/etc/inetd.conf`:

```
ftp stream tcp nowait root /etc/ftpd ftpd
```

This entry is already configured in the default `/etc/inetd.conf` file shipped with the link product.

Note `ftpd` uses the `$HOME/.netrc` file if it exists. Remember that this file creates a serious security risk.

Altering the `ftpd` Timeout

`ftpd` times out an inactive session after *timeout* seconds. The default timeout value is 900 seconds.

To alter the `ftpd` timeout, add:

```
-t timeout
```

where *timeout* is an integer that indicates seconds after `ftpd` in the following line in the `/etc/inetd.conf` file:

```
ftp stream tcp nowait root /etc/ftpd ftpd
```

For example:

```
ftp stream tcp nowait root /etc/ftpd ftpd -t75
```

sets the ftpd timeout to 75 seconds.

For more information on ftpd see the ftpd entry in section 1M of the *HP-UX Reference*. Included in this entry is a complete list of error messages.

Enabling ftpd Logging

You can configure the ftp server ftpd to log messages about logins, login failures, and anonymous ftp activity. To enable this logging, add the `-l` option to ftpd in `/etc/inetd.conf`. For example:

```
ftp stream tcp nowait root /etc/ftpd ftpd -l
```

For more information on ftpd, see the ftpd entry in section 1M of the *HP-UX Reference*. Included in this entry is a complete list of error messages.

Anonymous ftp Account

If an anonymous ftp account is required on some hosts, you will need to set up the directories on these hosts to make this possible. An anonymous ftp account has the directory structure shown in Figure 3-1:

The system administrator can permit public access or anonymous ftp. If this has been set up, users can access the anonymous ftp account with the user name anonymous or ftp, and any non-null password (by convention, the client host's name). ftpd does a chroot (see chroot in the *HP-UX Reference*) to the home directory of the user ftp, thus limiting anonymous ftp users' access to the system.

Daemons and Servers

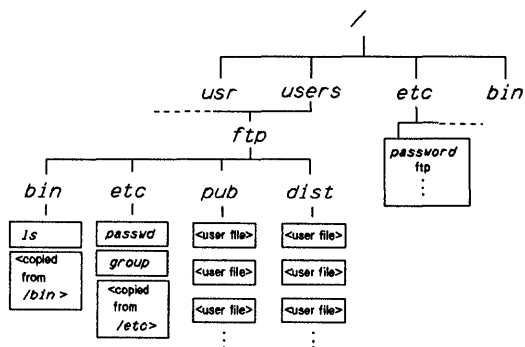


Figure 3-1. Directory Structure for Public ftp Account

Using SAM to Create an Anonymous ftp Account

You can use SAM to automatically create a public ("guest" or "anonymous") ftp account. SAM adds an ftp entry to the passwd database, creates the /users/ftp subdirectory structure, including bin and its contents, etc and its contents, pub (if desired), and dist (if desired). To maintain system security, SAM restricts anonymous ftp directory and file permissions as appropriate and removes non-essential information from the copies of the passwd and group databases placed in the /users/ftp/etc subdirectory.

Procedure

The following steps tell how to use SAM to automatically create a public ftp account:

1. At the HP-UX prompt, type:

```
sam
```

and wait for SAM's main menu to appear.

2. Select the Networking/Communications menu item.
3. Select Services Enable/Disable.

4. Select the Anonymous FTP menu item.
5. Select the Enable action.
6. Fill in the form according to the characteristics you want the public account to have. View the help screens for information about filling in the form.
7. Select OK to enter the change or Cancel to cancel when you are done with the screen.
8. Repeat steps 5 through 7 to allow other remote systems' access to a public ftp account on the local system.
9. Exit the Services Enable/Disable screen, by selecting Exit SAM to exit from SAM.

Verification

To view the public ftp account's directory structure, type at the HP-UX prompt:

```
ll -R /users/ftp
```

You should see subdirectories called bin, etc, and either or both of pub and dist.. To verify that a public login has been created for ftp access to your local host, type the following command at the HP-UX prompt:

```
grep '^ftp' /etc/passwd
```

This command should retrieve and display an entry for the ftp public login.

Undoing

The following steps tell how to remove part or all of the public ftp account:

1. At the HP-UX prompt, type:

```
sam
```

and wait for SAM's main menu to appear.

Daemons and Servers

2. Select the Networking/Communications menu item.
3. Select Services Enable/Disable.
4. Select the Anonymous FTP menu item.
5. Select the Disable action.
6. Fill in the form according to its instructions. Fill in the form according to the characteristics of the public account that you want to disable. View the help screens for information about filling in the form.
7. Select OK to enter the change or Cancel to cancel when you are done with the screen.
8. Exit the Services Enable/Disable screen, by selecting Exit SAM to exit from SAM.

Manually Creating an Anonymous ftp Account

To permit anonymous ftp, there must be an entry in the *passwd* (see *passwd* in the *HP-UX Reference*) database for an account named ftp. The password field should be "*", the group membership should be "guest", and the login shell should be `/bin/false`. For example:

```
ftp:*:500:guest:anonymous ftp:/users/ftp:/bin/false
```

The anonymous ftp directory should be set up as follows:

- `~ftp` The home directory of the ftp account should be owned by user ftp and mode 0555 (not writable). Since ftpd does a chroot to this directory, it must have the following subdirectories and files.
- `~ftp/bin` This directory must be owned by root and mode 0555 (not writable). It should contain a copy of `/bin/lis`, which is needed to support directory listing by ftpd. The command should be mode 0111 (executable only). If the home directory

Daemons and Servers

of the ftp account is on the same file system as /bin, the command can be a hard link, but it must not be a symbolic link because of the chroot. The /bin/lis command must be replaced when the system is updated.

~ftp/etc

This directory must be owned by root and mode 0555 (not writable). It should contain versions of the files passwd (see passwd) and group (see group). These files must be owned by root and mode 0444 (readable only). These are needed to map user and group ids in the ls command.

The passwd file should contain entries for the ftp user and any other users who own files under the anonymous ftp directory. Such entries should have "*" for passwords. Group ids should be listed in the anonymous ftp group file. The group file should have the group names associated with any group ids in the passwd file and any group ids of files in the anonymous ftp subdirectories.

Since the contents of /users/ftp/etc/passwd are often copied from /etc/passwd, it is possible for a user using the anonymous ftp account to determine which private accounts are not protected with a password. This risk can be alleviated by either requiring passwords for all accounts in /etc/passwd or deleting the unprotected accounts from the anonymous ftp /etc/passwd file.

~ftp/pub
(optional)

This directory would be used by anonymous ftp users to deposit files on the system. It should be owned by user ftp and should be mode 0777 (readable and writable by all).

~ftp/dist
(optional)

Directories used to make files available to anonymous ftp users should be mode 0555 (not writable), and any files to be

System Log Files

distributed should be owned by root and mode 0444 (readable only) so that they cannot be modified or removed by anonymous ftp users.

System Log Files

This section describes administrative tasks for ARPA service logging and configuration. It includes the following:

- Configuring `syslogd`.
- Maintaining system log files.
- Enabling the log file for `inetd` connection logging.
- Reconfiguring the log file for `inetd`.

Configuring `syslogd`

The ARPA daemons and servers log informational and error messages via `syslog`. You can monitor these messages by running `syslogd`, and determine the type and extent of monitoring in `syslogd`'s configuration file, `/etc/syslog.conf`

Each line in `/etc/syslog.conf` has a **selector** and an **action**. The selector tells which part of the system generated the message and what priority the message has. The action specifies where the message should be sent.

The part of the selector that tells where a message comes from is called the **facility**. All ARPA daemons and servers, except for `sendmail`, log messages to the daemon facility. `sendmail` logs messages to the `mail` facility. `syslogd` logs messages to the `syslog` facility. You may indicate all facilities in the configuration file with an asterisk (*).

System Log Files

The part of the selector that tells what priority a message has is called the **level**. Selector levels are debug, information, notice, warning, error, alert, emergency, and critical. A message must be at or above the level you specify in order to be logged.

The **action** allows you to specify where messages should be directed. You can have the messages directed to files, users, the console, or to a `syslogd` running on another host.

The following is the default configuration for `/etc/syslog.conf`:

```
mail.debug      /usr/spool/mqueue/syslog
*.info,mail.none /usr/adm/syslog
*.alert         /dev/console
*.alert         root
*.emerg         *
```

With this configuration all mail log messages at the debug level or higher are sent to `/usr/spool/mqueue/syslog`. Log messages from any facility at the information level or higher (but no mail messages) are sent to `/usr/adm/syslog`. Log messages from any facility at the alert level or higher are sent to the console and any terminal where the super-user is logged in. All messages at the emergency level or higher are sent to all users on the system.

For more information about `syslogd` and its configuration file, refer to the `syslog` entry in section 3C and the `syslogd` entry in section 1M of the *HP-UX Reference*.

Maintaining System Log Files

The log files specified in your `syslogd` configuration can fill up your disk if you do not monitor their size. To control the size of these files, do the following:

1. Remove or rename the log files in your `syslogd` configuration file.

For example, with the default configuration, you may wish to remove `/usr/adm/syslog` and `/usr/spool/mqueue/syslog`

System Log Files

2. Restart `syslogd` by sending it a `SIGHUP` (`kill -1`). For example:

```
kill -1 'cat/etc/syslogd.pid'
```

Enabling the Log File for `inetd` Connection Logging

`inetd` can log connection requests to configured servers. It logs successful connections at the `information` level and unsuccessful connection attempts at the `notice` level.

To enable `inetd` to log connections, add the `-l` option to the `inetd` command in `/etc/inetd.conf`. Then, when `inetd` is running, you can enable or disable connection logging with the following command:

```
/etc/inetd -l
```

Reconfiguring the Log File for `inetd`

If `inetd` is already active and you change the network configuration by changing `inetd.conf`, execute `inetd -c`. This option has the same effect as sending a `SIGHUP` signal (`kill -1`) to the `inetd` process. If the `inetd` daemon is not already running, an error message is returned to standard output.

To kill `inetd`, type the following command:

```
inetd -k
```

This option is the same as sending a `SIGTERM` signal (`kill -15`) to the `inetd` process.

Note

You should not use `kill -9` for killing `inetd`. Using `kill -9` produces the `inetd` error message “Unable to close semaphore,” (see the `inetd` entry of the *HP-UX Reference*) and may cause problems when you restart `inetd` without rebooting.

If you do kill `inetd` with `SIGKILL` (`kill -9`), execute an `inetd -k` command. This command resolves the semaphore problem and eliminates the need to reboot.

Configuring and Maintaining the BIND Name Server

The Berkeley Internet Name Domain (BIND) Name Server provides a distributed network information lookup service. It allows you to retrieve hostnames and internet addresses for any node on the network. It also provides mail routing capability via a list of hosts that will accept mail for another host.

This chapter includes the following sections:

- Key Terms.
- Overview of the BIND Name Service.
- Setting Up the BIND Name Service.
- Creating and Registering a New Domain.
- Name Server Data Files.
- Configuring Name Servers.
- Starting the Primary Server.
- Maintaining Network and Domain Data Files.
- Delegating a Subdomain.
- Configuring a Root Server.
- Using Other Options.
- Troubleshooting the BIND Name Server.

You cannot use SAM to configure a BIND name server.

Note

If you configure the BIND name server (either by itself or if you also configure the NFS Network Information Service (NIS), the name server is used to resolve all hostnames and internet addresses. If the name server is not configured and you have configured NIS, NIS is used for this purpose. If you have not configured NIS or the BIND name server, /etc/hosts is used. Figure 4-1 illustrates this algorithm.

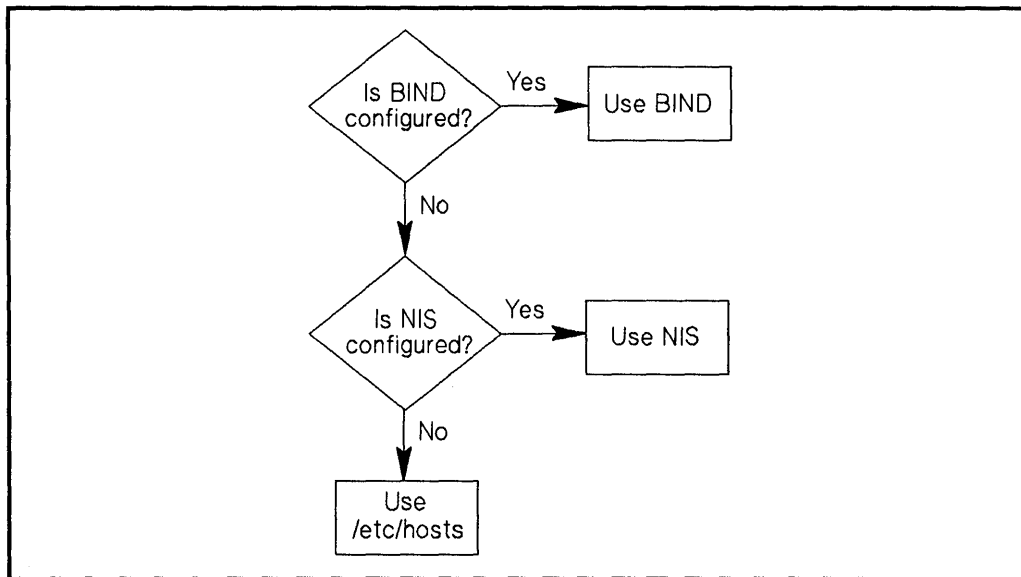
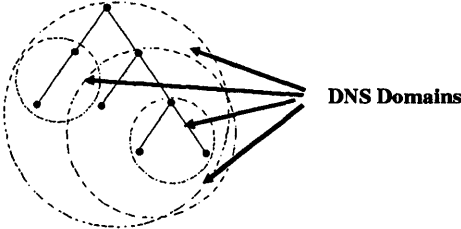


Figure 4-1. Resolution Algorithm

Key Terms

Key Terms	
Term	Definition
authority	For a BIND name server, “authority” means that the server has the most complete and accurate information for a domain.
BIND	Acronym for Berkeley Internet Name Domain. BIND is the Berkeley implementation of the Domain Name System (DNS).
bind	<p>1) The process by which a client locates a specific server and directs all requests for data to that serverbind.</p> <p>2) The process of establishing the address of a socket so other sockets may connect to it or send data to it.</p>
BIND resolver	Builds queries (for hostnames, internet addresses, or other information) and exchanges them with a name server. Resolver routines are in <code>libc.a</code> and are compiled into every program that uses <code>gethostbyname</code> or <code>gethostbyaddr</code> .
canonical name	The official hostname, as opposed to an alias.
Domain Name System (DNS)	The Domain Name System (DNS) has three components: a restructured name space and the data associated with the names, name servers, and resolvers. The DNS is fully described in RFC 1034 and RFC 1035. Compressed copies of these RFCs are located in <code>/usr/doc</code> .

Key Terms

Key Terms	
Term	Definition
DNS domain name	<p>Domain naming is an organization imposed on the DNS for administrative purposes. In the hierarchical domain name system, the DNS domain name of a node is the ordered list of labels on the path from that node to the root of the tree. The labels are separated by dots. An example is arthur.cs.purdue.edu. The top-level domain for educational institutions is edu; purdue is a subdomain of edu; cs is a subdomain of purdue; and arthur is the name of the host.</p>
DNS domain	<p>A DNS domain is a subtree of the hierarchical tree structure containing network resource names. DNS domains roughly correspond to organizational or geographic boundaries, and provide distributed management of host naming and addressing. A domain is identified by a domain name, and consists of that part of the domain name tree at or below the domain name. The following diagram illustrates the concept of DNS domains.</p> <div style="text-align: center;">  <p>The diagram shows a hierarchical tree structure of nodes and edges. Several nodes and their descendants are enclosed in dashed circles of varying sizes, representing different DNS domains. Arrows point from the text 'DNS Domains' to these circles.</p> </div>
caching only name server	<p>All name servers cache information by holding the data they receive for use until it expires, based on a “time to live” (<i>tll</i>) field attached to the data. A caching only name server, unlike other server types, is not authoritative for any domain. It services queries either directly from its cache (if it has recently received the same query) or by requesting information from servers with authority.</p>

Key Terms

Key Terms	
Term	Definition
fully qualified domain name	Also called “absolute” name, this is a domain name containing all the labels from the host to the root, terminated with a dot. <code>indigo.div.inc.com.</code> is an example.
hostname	The name assigned to a particular host with the <code>hostname</code> command.
master server	A name server that is authoritative for one or more domains. Two types of master servers exist: primary and secondary.
name server	A daemon, such as <code>named</code> , that answers queries from a BIND resolver or other name servers.
Network Information Service (NIS) domain	A logical grouping of NIS maps (databases) stored in one location. NIS domains are specific to the NIS network service (available through NFS) and are not associated with other network domains or the DNS.
node name	The Network Services (NS) product, as distinct from the ARPA/Berkeley services product, uses the node name assigned with the <code>nodename</code> command. This node name takes the form <i>node.domain.organization</i> . Note that this <i>domain</i> has nothing to do with the DNS.
primary name server	A name server authoritative for one or more domains. It loads its data from a file on its disk.
relative domain name	An incomplete domain name which is completed by local software using knowledge of the local domain. For example, <code>indigo</code> is a relative name that completes to <code>indigo.div.inc.com.</code>
remote name server	A configuration in which the local host uses a name server on a remote host.
root name server	A name server authoritative for the root domain.

Overview of the BIND Name Service

Key Terms	
Term	Definition
Requests for Comment (RFCs)	These documents are the “working notes” of the Internet research and development community. They describe computer communication-related topics, and may contain anything from meeting reports to standard specifications.
secondary name server	A name server authoritative for one or more domains. It receives its data for a domain over the network from another master server. At boot time, the secondary server either reads zone data from its disk (if it has backed up the data), or requests the data for the given zone from a master server.
subdomain	A domain wholly contained within a larger domain. For example, <code>div.inc.com</code> is a subdomain of <code>inc.com</code> .
zone	A zone is any subset of the hierarchical name tree which is administered under a single authority. Each zone is at least one level deep in the hierarchical tree and may span several levels. The top of the zone is the domain. The bottom of the zone is the bottom of the tree or where a subdomain is delegated. A zone defines limits of authority and defines the data transferred to a secondary name server during a zone transfer.

Overview of the BIND Name Service

This section presents an overview of the components, benefits, and functionality of the BIND name service. The BIND name service is an implementation of the Domain Name System (DNS). The DNS consists of three parts: the name space itself, which is abstract and conceptual; and the name server and the resolver, which are the software components that allow you to use the DNS.

Overview of the BIND Name Service

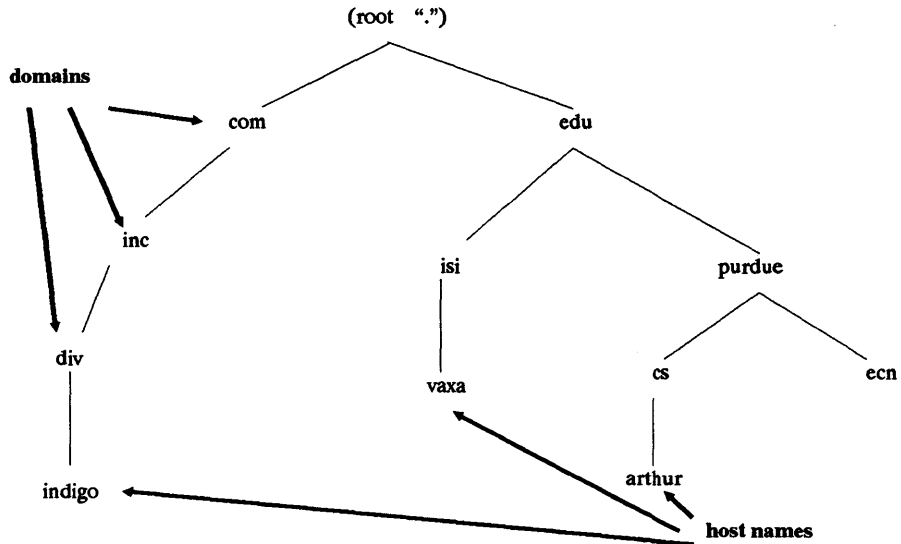


Figure 4-2. The Domain Name System

Domain Name System (DNS)

The DNS name space is hierarchical and conceptually organized as a tree. Each node of the tree is called a domain, and each domain has a label. Top-level domains on the public internet include com (companies and businesses), edu (educational institutions), gov (government agencies), mil (military and defense), net (network-related organizations), org (other entities); two-letter country codes such as au (Australia) and uk (Great Britain); and others. A top-level domain such as edu has subdomains like purdue, ukans, cu, and berkeley. In turn, the purdue subdomain may be divided into ecn, cs, and cc subdomains. Domain names differ from string-type or flat hostnames, such as indivhost, by naming paths through the hierarchy. The name of a particular domain or node is formed by concatenating all the labels of the domains from the current domain or node to the root and separating each label with dots. For example, a node arthur in the cs subdomain of Purdue University has the domain name arthur.cs.purdue.edu. Figure 4-2 illustrates the domain naming concept.

Overview of the BIND Name Service

Refer to RFC 1034 and RFC 1035 for a complete explanation of the DNS database format and domain name structure. These RFCs are in `/usr/doc`.

Name Servers

BIND name servers provide users of the DNS with access to the host information in the domain database. Primary and secondary name servers are authoritative for a particular subset of the data. However, they can resolve names outside that subset. To resolve outside names, they need only the internet address of the root name server. The root server can then provide the names and IP addresses of name servers authoritative for other subsets.

Name servers are daemon processes that answer queries from BIND resolvers and other name servers. They perform three primary functions:

1. Mapping hostnames to internet addresses.
2. Mapping internet addresses to hostnames.
3. Facilitating internet mail routing.

Resolvers

BIND resolvers are the programmatic interface to the name server. They consist of a set of library routines that query name servers for data. If a name server is not running locally, resolvers read the internet addresses of the name servers for their domain from a configuration file, `/etc/resolv.conf`. Resolvers perform general lookups.

BIND Name Server versus Network Information Service

The BIND name server and the NFS Network Information Service (NIS) provide similar services. Table 4-1 compares what each has to offer.

Overview of the BIND Name Service

Table 4-1. BIND versus NIS	
BIND Name Server	Network Information Service
Allows distributed maintenance of information.	Requires central maintenance of all databases from one NIS master server.
Spans networks.	Uses local LAN unless explicitly set otherwise.
Manages host names, IP addresses, mail exchanger data, and other unique data.	Manages many databases, such as the /etc files (group, hosts, netgroup, networks, passwd, protocols, rpc, and services).
Does not broadcast. Instead, queries name servers in a static list until answer received or timeout.	Makes queries by broadcasting to find server to bind to. Re-binds if server goes down.
Lookups are not case sensitive.	Lookups are case sensitive.
Master servers are polled by secondary servers to see if secondary needs to load new data.	Master servers push data to secondary (slave) servers.
Uses in-memory hash table.	Uses hash table stored on disk.

BIND Name Server versus /etc/hosts

The BIND name server also provides a mapping between names and addresses similar to what /etc/hosts does. The following table summarizes the major differences.

Table 4-2. BIND versus /etc/hosts	
/etc/hosts	BIND Name Server
Requires central maintenance of host table to ensure a complete, updated hostname list.	Allows each entity within the network to maintain its own portion of the hostname database.
Requires that the complete host table be replicated on every host.	Allows host information to be distributed over many servers.

Overview of the BIND Name Service

Table 4-2. BIND versus /etc/hosts	
/etc/hosts	BIND Name Server
Searches host table sequentially for requested name or address.	Answers from cache or queries other name servers to resolve requested name or address.
Does not require hierarchical names.	Requires hierarchical names.

Benefits of Using the BIND Name Server

The following table summarizes the advantages of the BIND name server.

Table 4-3. Benefits of the BIND Name Server	
Without the BIND Name Server	With the BIND Name Server
Host tables may become too large to manage easily.	Administration is distributed across many domains. Network administrators need only maintain their own small section of the name and address data.
Host tables must be replicated on every host.	Host information can be distributed over many hosts.
Hostname collisions may occur.	Hostname collisions are limited to the local domain and can easily be avoided.

Sites connected to the ARPA Internet and sites with large networks benefit the most from the BIND name service. The ARPA Internet is migrating to name servers, and name server use is well-established. Sites with computers that have limited disk space and processing speed may also wish to use the BIND name service to avoid having to store and access a large /etc/hosts file.

How the BIND Name Server Works

Name servers in the domain name system have data for only their local portion of the tree. Hence, they must query other name servers for data outside that portion.

Overview of the BIND Name Service

Given the internet addresses of the root name servers (the name servers authoritative for the root domain), a name server can look up any name in the tree. This is possible because root name servers, in response to a query, can direct the name server to query another name server that is either authoritative for the data or knows how to reach another authoritative server. When you look up a hostname, the resolver builds the query, sends it to a name server, receives the response, and returns the answer. In turn, the name server receives a request from the resolver, searches the DNS database for the answer (perhaps querying one or more remote servers in the process), and returns the result of the search. The following diagrams illustrate how queries are handled. Note that the resolver and the local name server need not be on the same host.

In this local query,

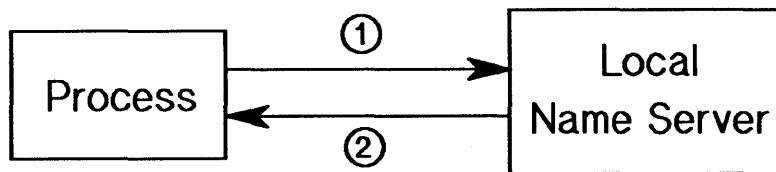


Figure 4-2. Local Query

1. A process, such as ftp, asks for the internet address of a host. The resolver sends this query to the local name server.
2. The local name server receives the request. Because it is authoritative for the name requested, it returns the answer to the resolver directly.

In this remote query,

1. A process, such as ftp, asks for the internet address of a host. The resolver sends this query to the local name server.
2. The local name server receives the request. In this case, it does not have the requested address within its database, so it queries the root server to find a server that is authoritative.
3. The root server returns the name and address of an authoritative remote server to the local server.
4. The local server then queries that remote server for the required data.

Overview of the BIND Name Service

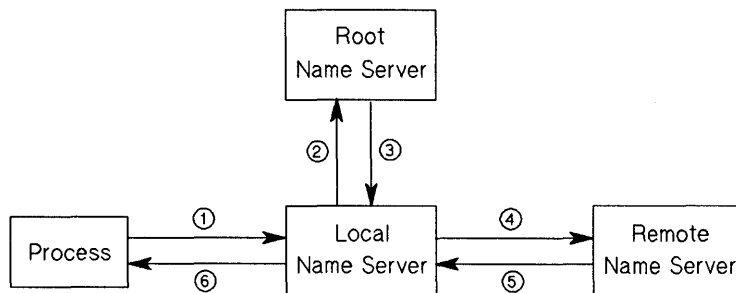


Figure 4-3. Remote Query

5. The remote server sends the data to the local server.
6. The local server now returns the answer to the resolver.

Note that caching data speeds up the above process. The next query for a different host in the same remote domain skips steps 2 and 3, because the local server has cached the names and internet addresses of the remote servers.

Root servers typically do not have access to the addresses of servers more than two levels below them. Hence, if the local server queries the root server for the address of `indigo.div.inc.com`, the root server can return only the address of the `inc.com` domain server. The local server must then query the `inc.com` server, which returns the name and address of the `div.inc.com` server. Finally, the local server queries the `div.inc.com` server and receives the address of `indigo.div.inc.com`. (Essentially, the name server repeats steps 4 and 5 above until the name is resolved.)

Resolver Search Algorithm

Because complete domain names can be cumbersome to type, the resolver provides a way for you to define personal aliases. The resolver also executes a search algorithm to resolve names that are not fully qualified (i.e., do not contain every label from the host to the root and end with a dot). The following section describes how the name server resolves hostnames, depending on their structure.

- If the input hostname consists of a single component (contains no dot), and if the environment variable `HOSTALIASES` is set to the name of a file, that file is searched for a string that matches the input hostname. The

Overview of the BIND Name Service

string consists of a hostname alias and its corresponding complete name. If a match is found, the complete name is substituted for the alias. Then the address of the substituted name is looked up. For example, *m* is found to be an alias for `myhost.div.inc.com`. Then the address of `myhost.div.inc.com` is looked up. Following is an example alias file:

```
m myhost.div.inc.com
a anyhost.div.inc.com
```

The first field is the user-defined alias for the complete name in the second field.

- If the input hostname ends with a dot (is an absolute name), the name is looked up as is. No domains are appended.
- If the input hostname does not end with a dot, it is looked up in the local domain and its parent domains. Lookups continue until either a match is found or fewer than two components of the local domain remain. If the name has not been matched by appending the local and parent domains, and the name contains a dot, it is looked up without modification. For example, in the domain `cs.berkeley.edu`, the hostname `lithium.cchem` is checked first as `lithium.cchem.cs.berkeley.edu` and then as `lithium.cchem.berkeley.edu`. However, `lithium.cchem.edu` is not tried, because only one component remains from the local domain. If none of the previous attempts matched, the name is looked up as `lithium.cchem` without any domains appended.

Note

It is always correct to use a name that contains all of the labels from the host to the root and does not end with a dot. Names that end in a dot are not allowed in the following places: mail addresses, *hostname*, and network-related files. (See the section titled “Updating Network-Related Files.”) Names that contain all of the name components and end in a dot are used primarily with commands such as `nslookup`, `ping`, and `telnet`, to facilitate the lookup process.

Setting Up the BIND Name Service

The following is an overview of the tasks you must perform to configure the BIND name service on your network. Each step is described in the pages following this list.

- Create and Register a New Domain (for first-time setup only).
- Create Domain Data Files.
- Configure the Name Servers.
- Set the Local Domain.
- Start the Name Servers.
- Maintain Network and Domain Data Files.

Optional steps include: delegating a subdomain, configuring a root server, and using other options.

Note

After configuring the BIND name service on your network, the following failures may occur: (1) `rcp` and `remsh` may fail with “permission denied” messages. (2) `rlogin` may prompt you for a password. These problems are the result of switching to domain names. To correct these problems, you will need to update other network files. Instructions are given in the section titled “Updating Network-Related Files.” After you configure the BIND name service, `sendmail` will use the name server’s MX (mail exchanger) records for mail routing. Refer to the “Inter-network Mail Routing” chapter for details on `sendmail`.

Creating and Registering a New Domain

This section is intended only for users planning to set up a new domain. If you are interested in adding servers to an existing domain, skip to the section titled “Name Server Data Files.”

To set up a new domain:

- If your organization already has a domain on a public network, the person in charge of the domain should be able to set up a subdomain for you.
- If your organization does not have a pre-existing domain and you want to set up a new domain on a public network, the site administrator should request a domain registration form from the organization in charge of that network. Public network contacts are listed in Appendix A. Note that if your organization belongs to several networks, it should register with only one.
- If your organization has a private network, you may set up domains without registering them. However, we suggest that you follow ARPA naming conventions in case you later decide to join a public network.

ARPA naming conventions:

- Use only letters (A-Z), digits (0-9), and hyphens.
- Make individual labels 63 characters or less, and complete names 255 characters or less. (A label is a component of a complete name, such as arthur or cs.) For ease of administration, we recommend avoiding individual labels longer than 12 characters.
- If a host connects to more than one network, it should have the same name on each.
- No distinction is made between upper and lower case letters.
- Do not use blanks.

Name Server Information

- Do not use dots except to separate components of domain names.
- Do not use NIC or other well-known acronyms as leftmost (most specific) labels in a name.

Once you have registered your domain, you may delegate any number of subdomains within it. These subdomains do not need to be registered with the parent network. Control of a subdomain is usually delegated, but does not have to be. To create a subdomain, refer to the section titled “Delegating a Subdomain.”

Name Server Information

This section contains general information about server types and considerations.

Name servers may be of three types: primary master, secondary master, or caching only. A primary master server is the authority for its domain, and contains all data corresponding to its domain. It reads its information from a master file on disk. A secondary is also the authority for its domain and contains that domain’s data, but gets its data over the network from another master server. A caching only server is not authoritative for any domain, and gets its data from an authoritative server or from its cache.

A fourth configuration, called a “remote” server, is simply a local host that uses a server on a remote machine instead. By default, the resolver is initialized to use the local name server. For more information, refer to the `resolver` entry in the *HP-UX Reference*.

No strict rules exist to determine which server configuration should be used on each host. Following are some suggestions for configuration:

- Timeshare machines or cluster servers should be **primary** or **secondary** servers.

Name Server Information

- If you want the benefits of a name server but do not want to maintain authoritative data, you may want to set up a **caching only server**.
- PCs, workstations that do not want to maintain a server, and other small networked systems should use a **remote server**. In addition, cluster nodes should use the cluster server as their remote server.

Information on configuring each of these server types appears in the section titled “Configuring Name Servers.”

Considerations for Selecting Name Servers

Follow these guidelines when selecting a name server:

- Choose systems that are as independent as possible for redundancy. These may include different power sources or cables.
- Choose systems that have the best Internet connectivity, such as their gateway connections.
- You must have at least two master servers per domain: a primary master and one or more secondary masters for redundancy. One server may be master for multiple domains: primary for some, secondary for others.
- Note that name servers for a particular zone need not physically reside within that domain. In general, zones are more accessible to the rest of the Internet if their name servers are widely distributed instead of on the premises of the organization that manages the zone.

Name Server Data Files

For operation, each primary name server requires a boot file and certain standard data files. Other name server types require a boot file plus some or all of these standard data files. The section titled “Configuring Name Servers” lists the data files necessary for configuring each server.

The location of the standard data files is specified in the boot file, `/etc/named.boot`, as illustrated in Figure 4-4.

- The **boot file**, `/etc/named.boot`, is read first when the name server, `/etc/named`, starts up. The boot file tells the name server what type of server it is, what zones it is authoritative for, and where it should get its initial data. Examples of boot files for each server type appear under the instructions for configuring that type.
- Four types of **standard data files** specify the data for a domain: `db.cache`, `db.127.0.0`, `db.[domain]`, and `db.[net]`. Naming the domain data files `db.[name]` is a Hewlett-Packard convention. These data files provide each name server with three specific types of information:
 - Information about the hosts in its zone (`db.[domain]`, `db.[net]`, `db.127.0.0`)
 - The names and internet addresses of name servers for the zone directly below it (`db.[domain]`)
 - The names and internet addresses of the root name servers (`db.cache`)

Refer to Appendix B, “Standard Data Files and Experimental Resource Records,” for a complete explanation and examples of the boot file and standard data files.

Note that case is preserved when names and data fields are loaded into the server, but all comparisons and lookups in the name server data base are case insensitive.

You may put only data that you are authoritative for in your standard data files. Do not add entries for domains other than your own, except in the case of “glue” resource records, described in the section titled “Delegating a Subdomain.”

Resource Record Format

The standard data files consist of resource records (RRs) that specify information such as name server addresses, canonical names, and mail exchangers. RRs use the standard resource record format explained below:

[name] [ttl] [class] Record Type Record Specific Data

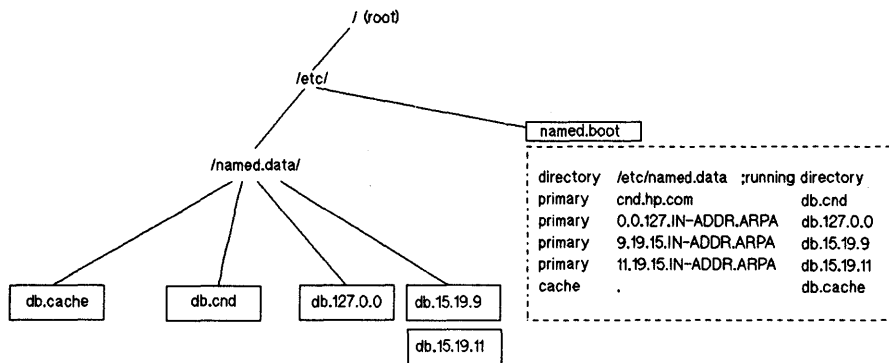


Figure 4-4. Standard Data Files

- The *name* field (optional) tells what domain name applies to the given RR. If the entry begins with a blank, it defaults to the name in the previous RR.
- The optional *time to live* field (ttl) indicates how long, in seconds, a server may cache the data it receives in answer to a query. If *time to live* is left blank, it defaults to the minimum time specified in the Start of Authority RR.
- The optional *class* field specifies the protocol group. IN, indicating Internet addresses, is the most common address class, although others

Name Server Data Files

exist. If left blank, it defaults to the last class specified.

- The *record type* field specifies what type of data is in the RR, such as canonical name, address, or mail exchanger information.
- The *record specific data* varies, depending on the RR type and class.

Standard Resource Records

The resource records (RRs) described in this section are the most common ones. Other RRs exist, many of them experimental. These are described in Appendix B, “Standard Data Files and Experimental Resource Records.” Following is an explanation of the most common RRs, including their format, contents, and use. The following characters have special meanings when used in RRs:

Character	Meaning
.	Refers to the root domain.
@	Denotes the current origin.
()	Indicates no line break, used to group data that crosses a line (only implemented for SOA and WKS records).
;	Indicates the start of a comment; the rest of the line is ignored.
*	Indicates wildcarding.
\X	Where X is any character other than a digit (0-9), quotes that character so it is not interpreted with its special meaning.

Note The current origin is appended to names in resource records that do not end with a dot. Hence, if a name is not in the domain for which you are creating the data file, end the name with a dot.

Name Server Data Files

A - Address

```
:[name] [ttl] [class] A      address
incdiv          IN      A      134.23.0.4
                IN      A      12.0.0.64
```

The Address record, A, lists an internet address of a given host, and is used to map a hostname to an address. `gethostbyname` looks for data of type A.

The name field is the host name and the address field is the internet address. As in this example, one machine may have multiple A records.

CNAME - Canonical Name

```
:alias          [ttl] [class] CNAME  canonical name
incindigo       IN      CNAME  indigo
```

The Canonical Name record, CNAME, specifies an alias for a canonical name (the host's official name). If an alias name is looked up, it is replaced with the canonical name and data for the canonical name is looked up. Aliases should be unique, and all other resource records should use the canonical name instead of the alias. If a CNAME RR is present at a node, no other data should be present; this ensures that the data for a canonical name and its aliases cannot be different.

HINFO - Host Information

```
:[name] [ttl] [class] HINFO  hardware OS
incnode          IN      HINFO  HP9000/840 HPUX
```

The Host Information record, HINFO, lists the hardware and operating system of the host. Currently, this type is informational only. It can be used to determine the CPU or operating system of the host being connected to. The service used or manner of interaction may change according to this information.

Name Server Data Files

Note that at least one space separates the hardware and operating system information. To include a space in the hardware name, put the name in quotes.

MX - Mail Exchanger

;	[name]	[ttl]	[class]	MX	preference	mail exchanger
	muz.oz.au.		IN	MX	5	ocean.nws.gov.
	*.il.		IN	MX	5	relay.cs.net.
	x.y.z.		IN	MX	10	x.y.z.
				MX	20	mailhub.y.z.

Mail Exchanger records, MX, specify a weighted list of hosts to try when mailing to a destination. This type is used for internet mail. `sendmail` checks for MX records when attempting to send mail to a host. The MX data may indicate an alternate host to send mail to, or may provide a list of hosts that accept mail for the target host. MX records can also be used to mail to a host that is not directly accessible by using an accessible host on the list as an intermediary. *Preference* specifies the order a mailer should follow if there is more than one mail exchanger for a given host. A low preference value indicates a higher precedence for the mail exchanger.

In the first example, `ocean.nws.gov` is a mail gateway that can deliver mail to `muz.oz.au` through a private connection or different transport medium.

In the second example, a wildcard name (containing an asterisk) is used for mail routing. Hence, all mail for hosts in the `il` domain is routed through `relay.cs.net`.

In the third example, mail to `x.y.z` first attempts delivery directly to the host itself. If that fails, because the host is down or is not directly accessible, mail is sent to `mailhub.y.z`.

NS - Name Server

Name Server Data Files

```
:[name]      [ttl]  [class] NS      name server's name
div.inc.com.          IN    NS      turtle.div.inc.com.
```

The Name Server record, NS, lists the name servers authoritative for a given domain. This type is used for subdomain delegation. The name field lists the domain that the name server is authoritative for.

PTR - Pointer

```
:[name]      [ttl]          [class] PTR      real name
7.8.19.15.in-addr.arpa.          IN    PTR      mr.div.inc.com.
```

A Domain Name Pointer record, PTR, allows special names to point to another location in the domain. PTR records are normally associated with a name in the `in-addr.arpa` domain and point back to the canonical name of the host owning the IP address. (See the section titled “Standard Data File Format” in Appendix B for an explanation of the `in-addr.arpa` domain.) PTR names should be canonical names, not aliases, and should be unique to the zone. `gethostbyaddr` looks up data of type PTR. In this example, `7.8.19.15.in-addr.arpa` points to the canonical name `mr.div.inc.com`.

SOA-Start of Authority

```
:[name]  [ttl]  [class] SOA      origin          person in charge
@                IN    SOA      mr.div.inc.com  jrs.dv.inc.com (
1                ; Serial
10800            ; Refresh after 3 hours
3600             ; Retry after 1 hour
604800          ; Expire after 1 week
86400 )         ; Minimum ttl of 1 day
```

The SOA record designates the start of a domain, and indicates that this server is authoritative for the data in the domain. This type is used during name server operation. Only one SOA record per domain should exist. The SOA is interpreted as follows:

Name Server Data Files

The *name* field is the subdomain's name. *Origin* is the name of the host this data file was created on. *Person in charge* is the mailing address of the person responsible for the name server. *Serial number* is the version number of this file, incremented whenever the data is changed. *Refresh* indicates (in seconds) how often a secondary name server should try to update its data from a master server. *Retry* indicates the time (in seconds) when a secondary server should retry after an attempted refresh fails. *Expire* indicates (in seconds) how long the secondary name server can use the data before it expires for lack of a refresh. *Minimum ttl* is the minimum number of seconds for the time to live field on resource records.

TXT - Text Data

```
:[name] [ttl] [class] TXT text  
  
host1 IN TXT "LOCATION: Machine Room"
```

TXT records are text strings, used for information associated with a host. This type is currently not used, but can be used to store information encoded in a string. Interpretation of string contents is service-dependent.

WKS - Well Known Services

```
:[name] [ttl] [class] WKS address protocol services  
  
host1 IN WKS 138.42.0.10 UDP syslog route  
domain IN WKS 138.42.0.10 TCP ( telnet smtp f  
tp shell domain )
```

The Well Known Services record, WKS, describes the services supported by a particular protocol at a specified address. It can be used to determine if a service is provided by a host. Currently this type is informational only. The list of services comes from the list specified in `/etc/services`. There should be only one WKS record per protocol per address.

Configuring Name Servers

This section provides instructions for configuring each type of name server.

To create the name server data files and the boot file for the primary server, you must convert the information contained in `/etc/hosts` into name server format. Once you have created the boot file and data files for the primary server, you can use them to set up other server types. Simply copy the files to another host and modify accordingly.

You have two options for converting this information and creating the four types of domain data files described in Appendix B.

1. You can use the utility `hosts_to_named` (explained below) that automatically converts `/etc/hosts` host table format to name server format and creates boot files and other data files.
2. You can do this conversion and create the boot file and data files manually.

Note

If you perform manual conversion, you must convert hostnames into fully qualified domain names (names containing all labels from the host to the root, terminated with a dot: i.e. `indigo.div.inc.com.`). You should also use `/etc/named.boot` as the name of your boot file, as that is the name expected by the name server.

Configuring a Primary Name Server

The primary name server loads its data base from the master data files. The primary server also uses all of the standard files that contain domain data, as described in Appendix B. To set up a primary server, run `hosts_to_named` or

Configuring Name Servers

create the boot and data files manually. An example boot file for a primary server authoritative for the `div.inc.com` domain and for networks 15.19.8 and 15.19.13 follows.

Example Boot File for Primary Name Server

```
;  
;type      domain          source file ← Comment lines.  
;  
  
directory /etc/named.data ; running directory for named ← Directory  
                        containing standard  
                        data files listed.  
  
primary   div.inc.com      db.div  
  
primary   0.0.127.IN-ADDR.ARPA db.127.0.0  
  
primary   8.19.15.IN-ADDR.ARPA db.15.19.8 ← "Primary" lines designate  
                        server as primary for zone  
                        in second field. Third field  
                        is the file from which data  
                        is read.  
  
primary   13.19.15.IN-ADDR.ARPA db.15.19.13  
  
cache     .                db.cache ← This file contains the  
                        addresses of network root  
                        name servers.
```

Using `hosts_to_named`

`hosts_to_named` automatically converts host table entries to name server format and creates a primary boot file. It also creates the standard data files, `db.cache`, `db.127.0.0`, `db.[domain]`, and `db.[net]`. However, it does not fill in the cache file, `db.cache`, with the root server names and addresses. You must do this manually. The section titled "Standard Data File Format" in Appendix B lists current root servers for the ARPA Internet.

Note

`hosts_to_named` completely rebuilds the `db.[domain]` and `db.[net]` files. Manual modifications to these files (with the exception of the SOA values) will be lost.

Configuring Name Servers

Once the host table has been translated, the name server files can be maintained directly, or the translation can be repeated after each change to the host table. To run `hosts_to_named`, you need to know the name of the domain and the network numbers of networks in the domain. The syntax of `hosts_to_named` is as follows:

```
hosts_to_named -d domain -n network-number [options]
```

- `-d` specifies the domain for which data is to be created.
- `-n` specifies the network number for which data is to be created.

The following example creates name server data for networks 15.19.13 and 15.19.8 in the domain `div.inc.com`.

```
hosts_to_named -d div.inc.com -n 15.19.13 -n 15.19.8
```

Note

For the name server to run, the boot file, `named.boot`, must be in the `/etc` directory. Because `hosts_to_named` places the files it creates in the directory you are in when you execute the command, you must do one of the following:

- (1) Use the `-b` option of `hosts_to_named` to specify the boot file as `/etc/named.boot`, or
- (2) Move `named.boot` to the `/etc` directory after running `hosts_to_named` elsewhere.

Refer to the `hosts_to_named` entry in the *HP-UX Reference* for a complete description of this command and available options.

Configuring a Secondary Name Server

A secondary server loads its information over the network from a primary server or another secondary server. It can use the same `db.cache` and `db.127.0.0` files as the primary server, but requires a different `named.boot` file. The `named.boot` file lists the IP addresses of servers from which the secondary server gets its

Configuring Name Servers

information. You can create the `named.boot` file by running `hosts_to_named`, or by creating the file manually.

Creating the Boot File with `hosts_to_named`

When used with the `-Z` and/or `-z` options, `hosts_to_named` uses the primary server's `named.boot` file to create two secondary boot files: `boot.sec.save` and `boot.sec`. With `boot.sec.save`, the server saves a backup copy of the information loaded over the network. With `boot.sec`, the server does not save a backup copy. The `-Z` and `-z` options are used to list the IP addresses of servers from which the secondary server gets its information.

To configure the secondary server, copy `db.cache`, `db.127.0.0`, and either `boot.sec.save` or `boot.sec` to the host that will run the secondary server. Then rename `boot.sec.save` or `boot.sec` to `/etc/named.boot`.

Creating the Boot File Manually

If you need more control over the contents of the secondary boot file, you can create the `named.boot` file manually. To do this, copy the files `named.boot`, `db.cache`, and `db.127.0.0` from the primary server. Then edit `named.boot` as follows:

In every primary line except the one containing `db.127.0.0`, replace the word “primary” with the word “secondary.” Leave the second field as is, and in the third field add the internet address of a primary server after the domain. After the internet address, delete the fourth field if you do not want to back up domain data to this file. (The back-up file is optional.)

For example, the following line is from the primary name server's boot file.

```
primary 8.19.15.IN-ADDR.ARPA db.15.19.8
```

Here is the same line, modified for the secondary name server's boot file.

```
secondary 8.19.15.IN-ADDR.ARPA 15.19.8.119 db.15.19.8
```

Configuring Name Servers

No other file modification is necessary. You may configure the secondary server in one of two ways:

1. Configure it so that it initially reads the authoritative data from a file on its disk, and does not have to rely on loading data from a master server. Eventually it will check with the master server to be sure this data is still up-to-date. To do this you must use a backup file to save domain data on the local disk. Specify the backup file in the secondary's boot file, as described above.
2. Configure it so that it does not have authoritative data stored on its disk, and must contact another master for its initial data. The secondary server then loads the data across the network. Do not use a backup file to do this.

The previous example and the example below show the first configuration using a backup file. The secondary server gets its data across the network from one of the listed servers, trying them in order until it successfully receives data.

Example Boot File for Secondary Server

```
;  
;type          domain          server address  backup file  
;  
  
directory /etc/named.data ; running directory for named  
  
secondary      div.inc.com        15.19.8.119    db.div  
primary        0.0.127.IN-ADDR.ARPA    db.127.0.0  
secondary      8.19.15.IN-ADDR.ARPA    15.19.8.119    db.15.19.8  
secondary      13.19.15.IN-ADDR.ARPA  15.19.8.119    db.15.19.13  
  
cache          .                          db.cache
```

“Secondary” lines designate server as secondary for zone in second field. The third field contains up to 10 addresses for servers authoritative for the zone.

The files in the fourth field are used to save the domain data on the local disk.

Configuring Name Servers

Configuring a Caching Only Name Server

No special line in the boot file is required to designate a server as caching only. Rather, caching only is indicated by the **absence** of primary or secondary lines in the boot file. The only primary line indicates the `in-addr.arpa` domain (the loopback interface), for which all name servers must be authoritative.

To create a caching only server, do not run `hosts_to_named` again. Instead, copy `boot.cacheonly`, `db.127.0.0`, and `db.cache` from the primary server. Rename `boot.cacheonly` to `/etc/named.boot`.

Example Boot File for Caching Only Server

```

;
;type          domain          source file
;

directory /etc/named.data      ; running directory for named
primary       0.0.127.in-addr.arpa  db.127.0.0
cache        .                   db.cache
```

Setting Up a Remote Name Server

If you want the resolver routines to use a remote server instead of a local server, you need to change the default values used to initialize the resolver. To change the resolver's default values so that it uses a remote name server for lookups, you need to create a configuration file called `/etc/resolv.conf`. This file designates which name servers on the network should be sent queries. If you are running a local server, creating this file is not advisable, as it is read every time `gethostbyname` or `gethostbyaddr` is called--though several applications (like HP Vue) do not work without `/etc/resolv.conf` when running a name server.

`/etc/resolv.conf` has three configuration options:

Configuring Name Servers

1. `domain` followed by the default domain name. The domain entry is needed only when the hostname (*hostname*) of the local system is not a domain name.
2. `nameserver` followed by the Internet address (in dot notation) of a name server that the resolver should query. Up to three name servers may be listed here.
3. `search` followed by up to six domains separated by spaces or tabs. The first domain in the search list must be the local domain. The `domain` and `search` keywords are mutually exclusive.

The following is an example of `/etc/resolv.conf`:

```
domain Berkeley.Edu
nameserver 132.22.0.4
nameserver 132.22.0.12
```

Refer to the resolver entry in the *HP-UX Reference* for more detail.

Note

When the name server is unavailable, NIS or `/etc/hosts` is used for hostname and address resolution. However, certain remote server configurations may not detect that the remote name server host(s) is unavailable and thus will not use NIS or `/etc/hosts` for lookups. The following configurations may exhibit this behavior:

- (1) The remote server host is down.
 - (2) More than one remote server is listed in `/etc/resolv.conf` and all the remote server hosts are unreachable.
-

Setting the Local Domain

When you use domain names that are not fully qualified, the resolver completes the names with the local domain. To do this, the resolver must know what the local domain is.

There are two ways in which you can set the local domain:

1. Set the hostname of the local system (*hostname*) to be a domain name without a trailing dot. For example, you would set the hostname in `/etc/rc` for the machine `indigo` in the domain `div.inc.com` as follows:

```
hostname indigo.div.inc.com
```

2. Create `/etc/resolv.conf` and add a domain or search entry. For example, create an `/etc/resolv.conf` file that contains the following line:

```
domain div.inc.com
```

Starting the Primary Server

After you have created the primary name server's boot file and domain data files, you are ready to start the primary server and begin using the BIND name service. Before you start the name server, make sure `syslogd` is running. `syslogd` logs informational and error messages. (To configure `syslogd`, refer to chapter 3 of this manual.)

To start the name server, type the following:

```
/etc/named
```

If you enter `/etc/named` without any arguments, it reads the default boot file `/etc/named.boot`, reads any initial data, and listens for queries. If the name server

Starting the Primary Server

boot file is in the `/etc` directory, the `/etc/netbsdsrc` file will automatically start the name server at boot time.

The procedure for starting other types of servers is the same as above. Refer to the named entry in the *HP-UX Reference* for information on additional options.

Verifying the Name Server

This section describes how to tell if you have correctly configured the BIND name servers.

To verify that the server has started successfully, do the following:

- If you are running `syslogd`, check the file(s) in `/etc/syslog.conf` for error messages. If error messages are recorded, refer to the section titled “Troubleshooting the BIND Name Server.”
- Use `nslookup` to look up a hostname to verify that lookups are using the name server and completing successfully. In the following example, `nslookup` is used to find host `charlie`, and completes successfully using the name server.

```
$ nslookup charlie

Name Server: indigo.div.inc.com
Addresses: 15.19.14.100, 15.19.15.100

Name: charlie.div.inc.com
Address: 15.19.9.100
```

Refer to the `nslookup` entry in the *HP-UX Reference* for complete instructions on how to use this command.

Maintaining Network and Domain Data Files

This section describes how to update and maintain network and domain data files.

Updating Network-Related Files

Once the name server is running, you may need to update network-related files to contain domain names. Flat or string-type hostnames which are not hosts in the local domain (such as `incdivhost`) must be converted to domain names in the following files: all `$HOME/.rhosts` files, all `$HOME/.netrc` files, `/etc/hosts`, `/etc/hosts.equiv`, and `/usr/adm/inetd.sec`.

When you convert the flat names to domain names, you must be sure that the domain name contains every label from the host to the top-level domain. However, do not end the domain name with a dot. For example, in an `.rhosts` file, do not use:

```
indigo.div.inc.com. fred
```

Instead, use:

```
indigo.div.inc.com fred
```

Note To provide an alternate means of lookup if the name server is down, you should maintain a minimal `/etc/hosts` file. It should only contain the hostnames and the internet addresses of all hosts in your local domain.

A utility is available to convert the hostnames in `.rhosts` and `hosts.equiv` to fully qualified domain names automatically. The shell script `convert_rhosts`, found in `/etc/newconfig/bind`, accepts input conforming to the syntax in `hosts.equiv` and converts it to fully qualified domain names. Instructions for using this utility are in the comments at the beginning of the script itself.

Updating Domain Data Files

As the composition of your network changes, you may need to add or remove hosts.

Note After modifying the domain data files, the name server must be restarted so that the files are reread. You can use either `kill -1 process-id` or `sig_named restart` for this purpose. Both signal the name server to reload its database. Refer to the `sig_named` entry in the *HP-UX Reference*.

Adding a Host

There are two ways to add a host.

1. Add the host to `/etc/hosts` and run `hosts_to_named` again.
2. Add the host manually:
 - a. Edit `db.[domain]`. Add an Address (A) resource record for each address of the new host. Add CNAME, HINFO, WKS, and MX resource records as necessary. Increment the serial number in the SOA resource record.
 - b. Edit `db.[net]`. Add a PTR resource record for each host address. Increment the serial number in the SOA resource record.

Deleting a Host

There are two ways to delete a host:

1. Delete the host from `/etc/hosts` and run `hosts_to_named` again.
2. Delete the host manually:
 - a. Edit `db.[domain]`. Delete all A, CNAME, HINFO, WKS, and MX resource records associated with the host. Increment the serial number in the SOA resource record.
 - b. Edit `db.[net]`. Delete all PTR resource records for the host. Increment the serial number in the SOA resource record.

Delegating a Subdomain

Within your own domain, you may delegate any number and level of subdomains to distribute control and management responsibility. These subdomains need not be registered with the parent network. The organization that owns a zone or subdomain is responsible for maintaining the data and insuring that up-to-date data is available from multiple, redundant servers.

To add a subdomain:

1. Set up the name servers for the subdomain.
2. Edit the existing zone file, `db.[domain]`, (or `spcl.[domain]` if you are using `hosts_to_named` to build `db.[domain]`) on the name server for the parent domain:
 - a. Add an NS resource record for each server of the new domain.
 - b. Add the necessary “glue” resource records. These records contain the internet addresses of the name servers listed in the NS records and are needed because the NS records give only the names of the name servers. The “glue” RR is an RR of type Address. For example:

ISI.EDU.	86400	IN	NS	A.ISI.EDU.	NS resource records
	86400	IN	NS	Venera.ISI.EDU.	
A.ISI.EDU.	86400	IN	A	128.9.0.107	“Glue” resource records
	86400	IN	A	26.3.0.103	
Venera.ISI.EDU.	86400	IN	A	128.9.0.32	
	86400	IN	A	10.1.0.52	

Configuring a Root Server

The BIND name service requires servers for the root domain. If you are connected to the ARPA Internet, you must use the root servers already available (listed under “/etc/named.data/db.cache” in Appendix B.) However, if you are on an isolated network, you must set up your own root servers.

A root server does not have a cache line in its boot file. Instead, it has a line like this:

```
primary . db.root
```

which indicates that the server is primary for the root domain. The db.root file typically contains only NS and A resource records for the authoritative name space tree. Following is an example of the root zone file, db.root.

```

@           IN           SOA         rabbit.div.inc.com. root.moon.div.inc.com. (
                                3           ; Serial
                                10800        ; Refresh every 3 hours
                                3600         ; Retry every hour
                                604800       ; Expire after a week
                                86400 ) ; Minimum ttl of 1 day
                                IN NS       rabbit.div.inc.com.
                                IN NS       denny.dept.inc.com.
                                IN NS       sally.doc.inc.com.
rabbit.div.inc.com. 86400 IN A       15.19.8.119
denny.dept.inc.com. 86400 IN A       15.19.15.33
sally.doc.inc.com. 86400 IN A       15.19.9.17
; Set ttl to 3 days
inc.com.           259200 IN NS       eduardo.inc.com.
                   259200 IN NS       labs.inc.com.
15.in-addr.arpa.  259200 IN NS       eduardo.inc.com.
                   259200 IN NS       labs.inc.com.

eduardo.inc.com.  259200 IN A       15.19.11.2
labs.inc.com.     259200 IN A       15.19.13.7

```

Name servers
authoritative
for root
domain

Name servers
authoritative for
delegated
subdomain
inc.com.

Using Other Options

Using Other Options

Once you have completed the basic configuration of the BIND name service and have successfully started the name server, you may wish to use other, more complex, options.

Wildcarding

The wildcard character (*) should be used with extreme caution. It is useful primarily in mail exchanger records as a mail relay for any host in that domain. For example, this mail exchange record

```
*.il IN MX 5 relay.cs.net
```

causes all mail for hosts in the `il` domain to be routed through `relay.cs.net`.

Note that the wildcard only matches names for which the name server does not already have data. If an MX record exists for `xyz.il`, a lookup of `xyz.il` would return the data for `xyz.il` and not the data associated with the wildcard.

Boot File Options

Boot file options include *forwarders*, *slave*, and *sortlist*. The following example shows their inclusion in the boot file:

```
;
; type          domain          source file
;
directory      /etc/named.data ;running directory for named
primary        div.inc.com          db.div
primary        0.0.127.IN-ADDR.ARPA  db.127.0.0
primary        8.19.15.IN-ADDR.ARPA  db.15.19.8
primary        13.19.15.IN-ADDR.ARPA db.15.19.13
cache          .                  db.cache
forwarders     15.19.13.3 15.19.8.64
slave
sortlist       10.0.0.0 26 .0.0.0
```

| Option lines

Forwarding Option

The *forwarders* line specifies the internet addresses of sitewide servers that accept queries from other servers. If many servers use the same forwarder, network traffic to outside servers is reduced, as a large sitewide cache is generated on the forwarder. The *forwarders* line works as follows:

1. If one or more forwarders are specified, the server sends all queries for data not in its cache to the forwarders first.
2. Each forwarder is queried in turn until an answer is received or the list of forwarders is exhausted.
3. If no answer is received from a forwarder, the server continues as it would have without the forwarder line, unless it is in slave mode. If it is in slave mode, it returns a temporary failure.

Slave Option

The *slave* option is used to put the server in slave mode. In this mode, the server only makes queries to forwarders. Note that this option makes the name server dependent on the operation of the forwarders.

Sort Option

The *sortlist* line is used to indicate networks that are to be preferred over other networks not listed. If a host has multiple addresses, *sortlist* controls the order in which they will be returned from a lookup of that host's name. Address sorting occurs only when the host making the query is on the same network as the local name server. The order of address preference is the following:

1. Local network addresses.
2. Addresses on the sort list.
3. All other addresses.

Troubleshooting the BIND Name Server

For more details on the above options, refer to the named entry in the *HP-UX Reference*.

Troubleshooting the BIND Name Server

If the BIND name server is not operating correctly, use this section to identify and correct the problem.

Troubleshooting Tools and Techniques

Because the BIND name service is distributed, failures can be difficult to diagnose. This section describes the available tools for general troubleshooting of the BIND name server and what they can tell you. More complex problems can be handled with these same tools and techniques, but require a more thorough knowledge of the Domain Name System (DNS) and name server operation.

Standard networking commands

Networking commands, such as ping or telnet, provide a way to test whether a specific hostname can be looked up.

```
$ telnet hostname
```

```
$ ping hostname
```

If hostname lookups are failing, use ping with an IP address to test network connectivity.

```
$ ping IP_address
```

nslookup

This tool allows you to send queries to name servers and display the contents of their responses. It also tells you whether the name server, NIS, or the host table is being used for name lookups, and which server your host is using. For more information, refer to the `nslookup` entry in the *HP-UX Reference*.

For example,

```
$ nslookup
Default Name Server: indigo.div.inc.com
Addresses: 15.19.8.100, 15.19.15.104
```

syslogd

Informational and error messages relating to `named` are logged using `syslogd`. `syslogd` reads and logs messages into a set of files listed in the configuration file `/etc/syslog.conf`. Refer to Chapter Three of this manual for information on configuring `syslogd`.

Name Server Debugging

The name server has debugging messages that can be turned on and off. You can use these debugging messages to help track down problems. Many of the messages are low level and are useful for discovering coding errors. Other messages show information about queries and responses. You can use these messages for detecting operational problems.

You can start the name server with the debugging option, or you can send signals to a running name server to turn debugging on and off. The commands `sig_named` or `kill` can be used to turn on debugging if the name server is already running. (See the sections titled “Understanding Name Server Debugging Output,” “Sending Signals to the Name Server,” and “Name Server Debug Levels.”) You can also start the name server using the `named` command with the `-d` option to turn on debugging. For more information, refer to the `named` entry in the *HP-UX Reference*.

Dump the name server database

A signal causes the name server to dump its database, which can then be analyzed for the cause of a problem. Use `sig_named` or `kill` to send the signal. For more information, see the section titled “Sending Signals to the Name Server” later in this chapter.

Problem Symptoms

This section describes symptoms of common name server problems, and lists possible problems to check for. A description of the problems appears in the next section, “Name Server Problems.” The most likely problems are listed first.

1. After configuring the primary server for the first time, names in the local domain cannot be found. Check:
 - Problem 2, Syntax Errors
 - Problem 1, Incorrect `hosts_to_named` Parameters
 - Problem 8, Local Domain Not Set
2. After configuring the primary server for the first time, names in the local domain can be found, but names in remote domains fail. Check:
 - Problem 3, Missing Cache Information
 - Problem 5, Network Connectivity
 - Problem 7, Incorrect Delegation of Subdomain
3. After configuring the local host to use a remote server, all name lookups fail, or only names in the NIS database or `/etc/hosts` are found. The server on the remote host is configured properly. Check:
 - Problem 4, Syntax Errors in `/etc/resolv.conf`
 - Problem 8, Local Domain Not Set

Troubleshooting the BIND Name Server

4. A remote name lookup now fails that has completed successfully before.

Check:

- Problem 5, Network Connectivity
- Problem 2, Syntax Errors
- Problem 4, Syntax Errors in `/etc/resolv.conf`

5. A local name lookup now fails that has completed successfully before.

Check:

- Problem 2, Syntax Errors
- Problem 6, Secondary Master Unable to Load from Another Master
- Problem 4, Syntax Errors in `/etc/resolv.conf`
- Problem 5, Network Connectivity

6. Names in the local and remote domains are looked up successfully. However, other servers not in your domain cannot look up names within your domain. Check:

- Problem 7, Incorrect Delegation of Subdomain

Name Server Problems

This section explains the problems that may cause the symptoms listed above, and suggests ways to solve the problems. Each of the tools described in the “Tools and Techniques” section is applied to the problems below. For specific problems, some tools or techniques are more effective than others. These are listed first.

1. Incorrect parameters supplied to `hosts_to_named`.

For this problem, none of the standard debugging tools are useful. Instead, you need to check the domain data files to be sure they contain resource records for the hosts in your domain. If `localhost` is the only host listed, you may have supplied incorrect domain names or network numbers to `hosts_to_named`.

2. Syntax error in the boot file or a data file.

Troubleshooting the BIND Name Server

a. `syslogd`

Syntax error messages are logged indicating the file name and line number. The actual message may or may not be useful.

b. Name server debugging output

Start the name server at debug level 1. Check for syntax error messages in `/usr/tmp/named.run` indicating the file name and line number.

c. `nslookup`

Depending on the error, the name server may exit or run in a partially usable state. If `nslookup` indicates it is using NIS or `/etc/hosts`, the server has exited. If `nslookup` starts but lookups indicate “`servfail`,” there is probably a syntax error in a data file.

d. `ping hostname`

If `ping` indicates that the host is unknown and the local name server should be authoritative for that name, the syntax error is probably in the file that maps hostnames to internet addresses, `db.[domain]`.

e. Dumping the name server database

In this case this does not provide useful information.

3. Missing cache information about the root servers. Without information about the root servers, names outside of the local domain cannot be looked up because the local server relies on the root servers to direct it to servers for other domains.

a. `syslogd`

Queries for names outside of the local domain cause the following message to be logged with `syslogd`: “No root name servers for class 1.” (Class 1 is the IN class.)

b. `nslookup`

May fail to look up the local host’s name on startup and give a “`servfail`” message. To check root server information, execute:

```
$ nslookup  
> set type=NS
```

Troubleshooting the BIND Name Server

> .

This asks for the NS records for the root. If no records for root servers are present, it returns “Can't find “.:” : Server failed.”

c. ping *hostname*

Names in the local domain are found, while names in remote domains are not found.

d. Name server debugging output

Set debugging to level 1. ping a host name not in the local domain. The debugging output in `/usr/tmp/named.run` contains: No root name servers for class 1. (Class 1 is the IN class.)

e. Dumping the name server database

No root server data appears in the “Hints” section at the end of the file `/usr/tmp/named_dump.db`.

4. Syntax errors in `/etc/resolv.conf` (for remote server configuration only). This assumes that the server on the remote host is configured properly.

Note

Errors in `/etc/resolv.conf` are silently ignored by the resolver code.

a. ns lookup

This indicates that NIS or the host table is being used for lookups.

b. ping *IP_address* or
ping *hostname*

Only names in the NIS database or host table can be looked up. ping the remote server's address to verify connectivity.

c. Name server debugging output

Troubleshooting the BIND Name Server

Turn on debugging on the remote server. Check that it is receiving queries from the local host. If queries are not being received, check the “name server” entries in `/etc/resolv.conf` and check network connectivity to the remote server.

d. Dumping the name server database

Because the server is on the remote host and we assume it is configured correctly, dumping the database is unnecessary.

e. `syslogd`

Checking `syslogd` output is unnecessary if the remote server is configured correctly.

5. Network connectivity problems may cause certain lookups to fail. This failure occurs in normal operation. Before name servers, the address for a name would be found, but a failure could occur when a connection was attempted. With name servers, a failure can occur sooner—when looking up the address for a name. See the *Installing and Administering LAN/9000* manual (Series 300 or 800) for information on troubleshooting network connectivity.

a. Name server debugging output

Turn on debug level 1. ping the hostname. Check the name server debugging output in `/usr/tmp/named.run` for lines like this:

```
req: found 'cucard.med.columbia.edu' as 'columbia.edu'
resend(addr=1 n=0) -> 128.59.32.1 6 (53) nsid=18 id=1 0ms
resend(addr=2 n=0) -> 128.59.40.130 6 (53) nsid=18 id=1 0ms
resend(addr=3 n=0) -> 128.103.1.1 6 (53) nsid=18 id=1 764ms
```

In this case the name server is trying to contact the `columbia.edu` name servers, but is not getting a response. Check network connectivity by pinging the addresses the server is trying to contact.

If the addresses being tried are the root name servers, either the host does not have connectivity to these machines, or the root server addresses are wrong.

b. `nslookup`

Troubleshooting the BIND Name Server

`nslookup` times out while trying to look up the name.

c. `ping hostname`

A message is returned saying that the host is unknown.

d. Dumping the name server database

In this case this does not provide useful information.

e. `syslogd`

In this case this does not provide useful information.

6. Secondary master is unable to load from another master. This may be caused by a configuration error or problems with network connectivity. Check that the domain being loaded and the address of the remote server are correct in the boot file.

a. `syslogd`

An error message is logged indicating the master server for the secondary zone is unreachable.

b. Name Server debugging output

Start the secondary server at debugging level 2 or 3. Watch for error messages in the debug output. These could show that the other server is unreachable, the other server is not authoritative for the domain, or that the local SOA serial number is higher than the remote SOA serial number for this zone.

c. `ping IP_address`

Verify connectivity to the server the secondary is trying to load from. If the host is temporarily unreachable, the secondary server will load when it is reachable.

d. `nslookup`

Use `nslookup` and set the name server to the master the secondary is trying to load from.

Troubleshooting the BIND Name Server

```
$ nslookup
```

```
> server server_name or IP_address
```

```
> ls domain
```

The `ls` command initiates a zone transfer. If the error message is "No response from server," then no server is running on the remote host. If the `ls` command succeeds, the secondary should be able to load the data from this server.

e. Dumping the name server database

In this case this does not provide any useful information.

7. Incorrect subdomain delegation may be caused by missing or incorrect NS or "glue" records in the parent server for the subdomain.

a. `nslookup`

Use `nslookup` to query the parent server for delegation information.

Execute:

```
$ nslookup
```

```
> server parent_server_name or IP_address
```

```
> set type=NS
```

```
> subdomain_name
```

This should show you the NS and glue records for the subdomain servers, as seen in the example below. In the example, the subdomain is delegated correctly.

```
hershey.div.inc.com:rootk> nslookup
Default Name Server: hershey.div.inc.com
Addresses: 15.19.14.100, 15.19.15.100
```

```
> server eduardo.doc.inc.com.
Default Name Server: eduardo.doc.inc.com
Address: 15.19.11.2
```

```
> set type=ns
> div.inc.com
```

"Hershey" is the default name server for this host.

Set the default name server to be this subdomain's parent server, "eduardo."

Set query type to "nameserver." Look up the div.inc.com domain.

Troubleshooting the BIND Name Server

Name Server: eduardo.doc.inc.com
Address: 15.19.11.2

Non-authoritative answer:

```
div.inc.com      nameserver = walleye.div.inc.com
div.inc.com      nameserver = friday.div.inc.com
Authoritative answers can be found from:
walleye.div.inc.com  inet address = 15.19.13.197
friday.div.inc.com   inet address = 15.19.10.74
```

Name server records for div.inc.com, the delegated subdomain.

"Glue" records with the addresses of the name servers for div.inc.com.

b. Dumping the name server database

Because the name server caches information, a database dump can be searched for the NS and "glue" records for the subdomain. If no NS and "glue" records exist, the parent server for the subdomain or the root servers are not reachable. If NS and "glue" records exist, check their correctness, if possible. Then try pinging the addresses of the name servers to see if they are reachable.

c. Name server debugging output

Turn on debugging to level 1 and try to look up a name in the domain. Check the debug output for name server retransmissions. This will indicate which servers are not responding. Check that the servers and their addresses are correct, if possible.

d. `nslookup`

In this case this does not provide any useful information.

e. `ping hostname`

In this case this does not provide any useful information.

8. The local domain is not set. The local domain is used to complete names that do not end with a dot. To set the local domain, either set the host name (*hostname*) of the local system to a domain name (without a trailing dot), or add a *domain* entry to `/etc/resolv.conf`. For examples of setting the domain, see "Setting the Local Domain" earlier in this chapter.

a. `nslookup`

`nslookup` gives a warning that the local domain is not set.

Troubleshooting the BIND Name Server

b. Name server debugging output

The debug output at level 1 shows names being looked up that are not domain names.

c. ping *hostname*

Hostname is only found when it is a completely specified domain name (with or without a trailing dot).

d. Dumping the name server database

In this case this does not provide any useful information.

e. syslogd

In this case this does not provide any useful information.

Understanding Name Server Debugging Output

To diagnose problems in the debugging output of the name server, you need to know what output from a successful query looks like. The following two examples show output from successful host name lookups. The first example does not involve any retransmissions, while the second example does. Note that debugging output looks the same whether it comes from a primary, secondary, or caching only server.

Example 1: No Retransmissions

Debug turned ON, Level 1

```
datagram from 15.19.10.14 port 4258, fd 6, len 35
req: nlookup(john.dept.inc.com) id 1 type=1
req: found 'john.dept.inc.com' as 'inc.com' (cname=0)
forw: forw -> 192.67.67.53 6 (53) nsid=29 id=1 0ms retry 4 sec
```

A query is received for john.dept.inc.com. The query is forwarded to a root server, ns.nic.ddn.mil at address 192.67.67.53.

```
datagram from 192.67.67.53 port 53, fd 6, len 166
resp: nlookup(john.dept.inc.com) type=1
resp: found 'john.dept.inc.com' as 'inc.com' (cname=0)
resp: forw -> 15.19.11.2 6 (53) nsid=32 id=1 0ms
```

ns.nic.ddn.mil responded with NS records for inc.com.

Troubleshooting the BIND Name Server

```
datagram from 15.19.11.2 port 53, fd 6, len 119
resp: nlookup(john.dept.inc.com) type=1
resp: found 'john.dept.inc.com' as 'dept.inc.com' (cname=0)
resp: forw -> 15.19.15.15 6 (53) nsid=33 id=1 0ms
```

The inc.com server responded with NS records for dept.inc.com.

```
datagram from 15.19.15.15 port 53, fd 6, len 51
send_msg -> 15.19.10.14 (UDP 7 4258) id=1
Debug turned OFF, Level 1
```

The dept.inc.com server responded with the address of john. The local server responds with the answer to 15.19.10.14.

Following are detailed explanations of certain lines from the above example.

Debug turned ON, Level 1

The name server was already running. The first level of debugging was turned on with `sig_named debug 1`.

```
datagram from 15.19.10.14 port 4258, fd 6, len 35
```

This line shows the IP address of the host that generated the query, the port that the request comes from, the file descriptor that the name server received the query on, and the length of the query.

```
req: nlookup(john.dept.inc.com) ID 1 type=1
```

This message was logged from the routine that handles requests. Shown are the name looked up, the packet ID (used to determine duplicate requests), and the type (as defined in `/usr/include/arpa/nameser.h`). Type 1 is an address query.

```
req: found 'john.dept.inc.com' as 'inc.com' (cname=0)
```

Since the server is authoritative for `div.inc.com`, it has an entry for `inc.com` in its database. The only data at `inc.com` is the subdomain entry for `div`. This line does not indicate what was found at `inc.com`. Since the server sent the next query to a root name server, we conclude that there were no NS records for `inc.com`. For more information, including the domain for which the queried server is authoritative, check Debug level 3. This is described in “Name Server Debug Levels” later in this chapter.

```
forw: forw -> 192.67.67.53 6 (53) nsid=29 id=1 0ms retry 4 sec
```

The query was forwarded to 192.67.67.53. The name server tags each query it sends out so that it can detect duplicate responses. Here the assigned ID is 29. The original ID was 1. The query will be retried in four seconds.

```
resp: found 'john.dept.inc.com' as 'inc.com' (cname=0)
```

After the response from the root server, the database is searched again. Inc.com is found once again. The next query goes to an inc.com server, so this time there were NS records.

```
datagram from 15.19.11.2 port 53, fd 6, len 119
```

This datagram is from another name server since it is from port 53. Since our server sent a query to 15.19.11.2, we assume this is the response.

```
send_msg -> 15.19.10.14 (UDP 7 4258) id=1
```

The response was sent back to host 15.19.10.14 on port 4258.

Example 2: Retransmissions

The next example shows a successful lookup which involved retransmissions. Retransmissions take place from the resolver and the name server. The resolver retransmits to the local name server, and the local name server retransmits to remote name servers during the process of looking up a name. When the local server receives the resolver retransmissions, it discards them as duplicates if it is still processing the first request.

Following are detailed explanations of certain lines from this example.

```
req: nlookup(cucard.med.columbia.edu) id 1 type=1
```

This message was logged from the routine that handles requests. Shown are the name looked up, the packet ID (used to determine duplicate requests), and the type (as defined in /usr/include/arpa/nameser.h). Type 1 is an address query.

```
resend(addr=1 n=0) -> 128.59.32.1 6 (53) nsid=18 id=1 0ms
```

Troubleshooting the BIND Name Server

```
datagram from 15.19.10.14 port 4253, fd 6, len 41
req: nlookup(cucard.med.columbia.edu) id 1 type=1
req: found 'cucard.med.columbia.edu' as 'edu' (cname=0)
forw: forw -> 128.9.0.107 6 (53) nsid=17 id=1 1478ms retry 4 sec
```

A query is received for cucard.med.columbia.edu. The query is forwarded to a root server, a.isi.edu at address 128.9.0.107.

```
datagram from 128.9.0.107 port 53, fd 6, len 212
resp: nlookup(cucard.med.columbia.edu) type=1
resp: found 'cucard.med.columbia.edu' as 'columbia.edu' (cname=0)
resp: forw -> 128.59.16.1 6 (53) nsid=18 id=1 0ms
```

NS records for columbia.edu are returned. Name server trying columbia.edu

```
datagram from 15.19.10.14 port 4253, fd 6, len 41
req: nlookup(cucard.med.columbia.edu) id 1 type=1
req: found 'cucard.med.columbia.edu' as 'columbia.edu' (cname=0)
resend(addr=1 n=0) -> 128.59.32.1 6 (53) nsid=18 id=1 0ms
resend(addr=2 n=0) -> 128.59.40.130 6 (53) nsid=18 id=1 0ms
```

Retransmission from resolver (discarded).

Retransmitting to other columbia.edu servers.

```
datagram from 15.19.10.14 port 4253, fd 6, len 41
req: nlookup(cucard.med.columbia.edu) id 1 type=1
req: found 'cucard.med.columbia.edu' as 'columbia.edu' (cname=0)
resend(addr=3 n=0) -> 128.103.1.1 6 (53) nsid=18 id=1 764ms
```

Retransmission from resolver (discarded).

Trying another columbia.edu server.

```
datagram from 128.103.1.1 port 53, fd 6, len 57
send_msg -> 15.19.10.14 (UDP 7 4253) ID=1
```

A columbia.edu server responded with the address of cucard. The local server responds with the answer to 15.19.10.14.

Since no response came from 128.59.16.1, the query with *nsid* 18 was resent to other servers.

```
datagram from 15.19.10.14 port 4253, fd 6, len 41
req: nlookup(cucard.med.columbia.edu) id 1 type=1
```

Note that this came from the same IP address and port and has the same length and ID as the preceding datagram. It is a duplicate and thus *forw* discards it. These two lines are repeated three times throughout this trace. The queries came from the same IP address and port, and have the same ID and length in each case. Thus, these are all the same query. The resolver sent the query three times because the name server didn't respond. The name server detects that the second and third are duplicates and discards them. (We can tell because the duplicates did not get to the *forw* line.)

Troubleshooting the BIND Name Server

Sending Signals to the Name Server

The name server responds to the signals listed in Table 4-5, which includes the `sig_named` mnemonics to send the signal.

Signal	sig_named	Effect
SIGHUP (1)	restart	Causes the name server to read the boot file and reload its database.
SIGINT (2)	dump	Dumps the current database and cache to <code>/usr/tmp/named_dump.db</code> .
SIGIOT (6)	stats	Dumps the statistics data into <code>/usr/tmp/named.stats</code> . Statistics data are appended to the file.
SIGUSR1 (16)	debug <level>	Turns on debugging; each following SIGUSR1 increments the debug level.
SIGUSR2 (17)	debug 0	Turns off debugging completely.

These signals can either be sent by `kill` or `sig_named`.

For example, to dump the name server database execute:

```
$ kill -2 process_id
```

OR

```
$ sig_named dump
```

Name server process ids can be found in `/etc/named.pid`.

To turn on debugging and increment to level 3, execute:

```
$ kill -16 process_id
```

```
$ kill -16 process_id
```

Troubleshooting the BIND Name Server

```
$ kill -16 process_id
```

OR

```
$ sig_named debug 3
```

To turn off debugging, execute:

```
$ kill -17 process_id
```

OR

```
$ sig_named debug 0
```

Name Server Debug Levels

The debugging output from the name server goes to `/usr/tmp/named.run`. Much of the debug output is not useful unless you have the source code to trace. However, the output does contain enough information to diagnose most problems. Debug levels range from 1 to 10; the most useful information at each level is explained below.

At certain levels, the actual packets are displayed. See RFC 1035, pages 25-30, for the format of DNS packets. This RFC is in `/usr/doc`.

- 1** This is the most useful debug level. It contains information about transactions being processed, although the messages are rather cryptic. This level logs the IP address of the sender, the name looked up, and the IP addresses of other servers queried.
- 2** The list of IP addresses about to be queried and their current round trip time calculations are displayed. A secondary server displays information about each zone it is maintaining when it contacts a master to see if a zone is up to date.
- 3** More detailed information about internal operation is given, most of it not useful. This level tells you when a resolver retransmission is dropped, what name servers were found for a remote domain, and how many addresses were found for each server. When a secondary server checks

Troubleshooting the BIND Name Server

with the primary to see if the secondary's data is up to date, an SOA query is made. The SOA responses are displayed at this level.

- 4 The initial query packet and the response packets from other remote servers are displayed.
- 5 More internal operation information, most of it not helpful.
- 10 Shows the packet sent to other servers during name lookup. Also shows the packet the local server sent back to the querying process.

Name Server Statistics

The name server keeps track of various statistics. You can print these statistics to the file `/usr/tmp/named.stats` by sending the name server a SIGIOT signal. To send a SIGIOT signal, use the `sig_namedstats` command. The statistics look similar to the following:

```
1273431  time since boot (secs)
29082    time since reset (secs)
326031   input packets
327165   output packets
284353   queries
0        iqueries
214     duplicate queries
50109   responses
70      duplicate responses
220220  OK answers
63919   FAIL answers
0       FORMERR answers
23      system queries
4       prime cache calls
4       check_ns calls
0       bad responses dropped
0       martian responses
0       Unknown query types
47921   A querys
2054    CNAME querys
8216    SOA querys
35906   PTR querys
```

Troubleshooting the BIND Name Server

```
10569    MX queries
424      AXFR queries
179263   ANY queries
```

The first two lines print out the number of seconds that the name server has been running, and the number of seconds since the last restart caused by a SIGHUP signal. To convert these values to days, divide by 86,400 (the number of seconds in a day.)

input packets are the number of datagrams received by the name server. The datagrams come from the resolver code compiled into the services, and from queries and responses from other name servers.

output packets are the number of datagrams sent by the name server. These datagrams are responses to resolver queries, responses to queries from other name servers, and system queries. Because queries to other name servers may not be answered, there will probably be more output packets than input packets.

queries are the number of queries received by this name server. Because the name server can handle datagram and stream connections, there can be more queries than input packets. The total number of queries is the sum of all the counts of different query types listed in this statistics dump, starting with unknown query types.

inverse queries are the number of inverse queries. Inverse queries can be used to map a host address to a domain name, although PTR queries (discussed below) are the normal method. Some versions of `nslookup` send inverse queries when they are starting up.

duplicate queries are retransmitted queries for pending lookups that the resolver sends to the name server. The name server detects the duplicate queries and discards them.

responses are the number of response packets that the name server receives from queries to other name servers.

Troubleshooting the BIND Name Server

duplicate responses are response packets from remote name servers for queries that are no longer pending. The name server retransmits queries to remote name servers. If the remote server responds to the original query and responds to the retransmitted query, the local name server discards the second response as a duplicate.

OK answers are the number of responses to queries that contain some information.

FAIL answers are the number of responses indicating either that the name does not exist, or that there is no data of the requested type for this name.

FORMERR answers are the number of malformed response packets from other name servers. A message is sent to the `syslog` daemon listing the sender of the malformed response packet.

system queries are queries generated by the name server. These usually occur when the name server detects another name server listed for a domain for which there is no address data. The system query is an attempt to find the address data for that name server. System queries are also used to keep up-to-date information about the name servers for the root domain.

prime cache calls are calls to update the information about the name servers for the root domain.

check_ns calls are calls to check the state of the information about the name servers for the root domain.

bad responses dropped are responses from remote name servers that are dropped. These occur most often when the remote name server responds with `SERVFAIL`, indicating a problem with the server's domain data.

martian responses are responses from unexpected addresses. The name server keeps track of how long it takes for a remote name server to respond. If the remote name server is a multi-homed host, a query to one of the addresses may

Troubleshooting the BIND Name Server

result in a response from another of its addresses. If the local server does not know about this other address, the response is counted as a martian response.

unknown query types are queries for data types unknown to this server.

A queries are queries for the host address for a domain name. The `gethostbyname` library routine generates these address queries.

CNAME queries are queries for the canonical name for a domain name. Some versions of `sendmail` query for CNAME records during name canonicalization from `[$]` tokens in `/usr/lib/sendmail.cf`.

SOA queries are queries for the start of authority records. These queries are most often made by secondary servers over a stream connection to check if their domain data is current.

PTR queries are queries for the domain name for a host address. The `gethostbyaddr` library routine generates these queries.

MX queries are mail exchanger queries made by `sendmail` during the delivery of electronic mail.

AXFR queries are the number of zone transfers done by secondary servers. A secondary server first makes an SOA query and will follow that with an AXFR query if new domain data should be transferred.

ANY queries are queries for any data associated with the domain name. Some versions of `sendmail` make queries for *ANY* data during name canonicalization from `[$]` tokens in `/usr/lib/sendmail.cf`.

Configuring gated

This chapter contains information about how to configure and use gated (pronounced *gate-d*), a daemon that dynamically determines packet routing. It includes the following sections:

- Overview.
- Protocols and Metrics.
- Configuration Options.
- Configuring gated.
- Sample Configurations.
- Starting gated.
- Troubleshooting gated.

You cannot use SAM to configure gated.

Overview

The gated daemon dynamically determines packet routing between internet gateways. Developed at Cornell University, gated handles the RIP, EGP, and HELLO routing protocols, or any combination of the three.

Overview

Advantages

Using `gated` offers these advantages:

- Dynamic routing eliminates the need to reset routes manually. The internet system can automatically route around network failures.
- Dynamic routing makes it easier to add and administer nodes.
- Dynamic routing lowers the cost of operating complex internet systems.
- `gated` translates among three protocols, passing information between domains that are using different routing protocols. **Domain** is used here to refer to a group of connected nodes that are running a given protocol.
- `gated` gives the system administrator more flexibility in setting up and controlling network routing. For example, `gated` can listen to network traffic at specified gateways, determine available routes, and update local routing tables accordingly.

When to Use `gated`

`gated` is most often used in large networks, or small networks connected to larger wide-area networks. In large local networks, there are often multiple paths to other parts of the local network. `gated` can be used to maintain near optimal routing to the other parts of the local network, and to recover from occasional losses of certain links in the path. When connected to wide-area networks, `gated` can be used to inject local routing information into the wide-area network's routing table.

`gated` should be run on gateways so its routing information can be sent to other gateways and to passive listeners on the LAN. Hosts with only one LAN interface may run `gated` to passively listen to routing information when there is more than one gateway on the LAN. If there is only one gateway on the LAN (leaving only one path off the local LAN), you may prefer to configure a static route to that gateway in `/etc/netlinkrc` instead of running `gated`.

Protocols and Metrics

gated uses three standard protocols to maintain current routing tables:

- RIP (Routing Information Protocol) is the most widely used routing protocol within UNIX environments. A de facto industry standard, it is also used by routed, a service distributed by Berkeley. RIP is not intended for use in WAN applications.
- EGP (External Gateway Protocol) is known as a reachability protocol primarily because it permits a node on the NSFNET backbone to exchange information with other backbone nodes about whether a destination can be reached. Use EGP to communicate with other domains.
- HELLO was designed to work with routers called “Fuzzballs.” Most installations use RIP instead of HELLO.

Routing protocols such as RIP and HELLO are designed to find a path between network nodes. If multiple paths exist, they usually pick one of the shorter paths. Each protocol has a metric that it applies to each path. In most cases, the lower the metric for a given path, the more likely a protocol will choose it.

Protocols have different types of metrics. The HELLO metric, for example, uses milliseconds, while the RIP metric uses hopcounts. **Hopcounts** are the number of gateways the packet must pass through to reach its destination. If a path is directly connected, it receives the lowest metric of one. If the path passes through a single gateway, the metric goes to two. The metric continues to increase up to RIP’s limit of sixteen. Sixteen is known as RIP’s **infinity metric** because it indicates a node that cannot be reached or has been specified as unreachable.

If gated encounters an infinity metric or a node that has been designated as unreachable, it goes into **Holddown Mode**. **Holddown Mode** stops a node from propagating routing information until the other nodes it is communicating with stabilize their routing information.

Configuration Options

Sometimes you may not want traffic to take a certain path because it incurs an unacceptable cost or security risk. In these cases gated allows you to assign a metric to each interface. This allows you to select or bypass a path, regardless of its length or speed.

Configuration Options

This section contains reference information about options you can specify in a gated configuration file (/etc/gated.conf). The configuration options are grouped under five subheadings. The contents of each subsection are summarized in Table 5-1.

Subheading	Contents
Configuring Routing Protocols.	How to configure the RIP routing protocol. For information about configuring HELLO or EGP, see the gated entry in the <i>HP-UX Reference</i> .
Customizing Routes.	How to specify default gateways, static routes, passive interfaces, and interface metrics.
Limiting Routing Information That gated Sends Out.	How to specify interfaces that do not send out routing information, and gateways that can receive outgoing information. Also, how to limit outgoing information about certain networks and hosts.
Limiting Routing Information That gated Takes In.	How to specify interfaces that do not process routing information, and gateways that can send incoming information. Also, how to limit incoming information about certain networks and hosts.
Specifying Tracing Output.	Briefly described. For more information, see the "Starting gated" section of this chapter.

The next section, "Configuring gated," describes how to use these options while creating a gated configuration file. The section after that, "Sample Configurations," shows some sample configuration files.

Note It is best to use IP addresses in dot notation (for example, a.b.c.d.) anytime you are specifying an address for a configuration option such as a gateway, host, or interface. Hostnames that have multiple IP addresses associated with them will be considered an error.

Configuring Routing Protocols

When gated starts, it reads its configuration file to find out how each protocol should be used to manage routing. This subsection lists the configuration options for the RIP protocol. For more detailed descriptions of the RIP options, as well as descriptions of the HELLO and EGP options, refer to the gated entry of the *HP-UX Reference*.

RIP

```
rip {yes| no| on| off| supplier| pointopoint| quiet}
```

This clause tells gated how to perform the RIP routing protocol. Only one argument is allowed after the keyword `rip`.

`yes` or `on` tells gated to enable the RIP protocol at this node and process all RIP packets coming in from other nodes. If gated finds fewer than two network interfaces, the node only listens to RIP information. If gated finds two or more network interfaces, the node both listens to and broadcasts RIP information. If you do not specify a `rip` line in your configuration file, `rip yes` is assumed.

`no` or `off` tells gated to disable the RIP protocol at this node.

Configuration Options

`supplier` tells `gated` to enable the RIP protocol at this node and process all RIP packets coming in from other nodes. `supplier` forces `gated` to send out RIP packets every thirty seconds no matter how many network interfaces it finds at this node. Use `supplier` if you want this node to both listen to and broadcast RIP information.

`pointpoint` is similar to `supplier`. It differs in that RIP information is not sent out in a broadcast packet but is sent directly to the gateways listed in the `sourcegateway` option in the `rip` protocol statement.

`quiet` tells `gated` to process all RIP packets coming in from other nodes, but not to supply any RIP information no matter how many network interfaces it finds. Use `quiet` if you want this node to only listen to RIP information.

EGP and HELLO

EGP and HELLO are used less frequently than RIP, and their configuration is not discussed here. For information about EGP and HELLO configuration options, refer to the *HP-UX Reference*.

Customizing Routes

This subsection describes options for setting up customized routes in the `gated` configuration file. It includes options for specifying default gateways, static routes, passive interfaces, and routing metrics for interfaces.

Specifying a Default Gateway

To specify a default gateway in the kernel routing tables during initialization and to reinstall it whenever information about the default route is lost, use the following static clause:

```
static {
    default gateway gw_addr preference 255 ;
};
propagate proto rip metric 15 {
    proto static {
```

```
        announce default;  
    };  
}
```

If you specify the propagate clause above for the default route, gated installs the default route with a time delay equivalent to an RIP metric of 15. A high metric of 15 prevents the route from being propagated more than one hop beyond the default gateway.

If only the static clause is specified and not the propagate clause, then the default route will not be passed on as a route to other gateways. This is considered a passive default route and is used only by the host that this gated is running on. The preference value of 255 allows any other default route learned from another protocol to replace this one. This route will be reinstalled when the other default routing options are removed.

Installing Static Routes

A static route provides a specific destination for network packets. These options install a static route to a network address or host address (*dest_addr*) through a gateway (*gw_addr*). This route is installed in the kernel's routing table and is never affected by any other gateway's RIP or HELLO announcements. Following is an example:

```
static {  
    dest_addr gateway gw_addr;  
};
```

Setting Interface Metrics or States

```
interface intf_addr [ intf_addr... ] [passive] [metric mnumber ];
```

gated times out routes that pass through interfaces that are not receiving any RIP, HELLO, or EGP packets. The *passive* option in the interface statement allows you to set the list of interfaces whose routes will not time out if protocol traffic stops. If an interface's routes do time out, the routes are reinstalled when the interface begins to receive routing packets again. *It is recommended that you use this option for all interfaces in HP-UX machines.*

Configuration Options

The `metric` option allows you to add the metric *mnumber* to the true metric of each route that comes in via routing information from the listed interface. *Mnumber* is also added to the true metric of any information sent out via the listed interface.

Adding *mnumber* makes a route metric appear to be higher than it really is. When gated encounters the higher metric, it forces routing to a different path with a lower metric, if available. Use this option to avoid routes you do not want to use such as costly or insecure routes.

Limiting Routing Information that gated Sends Out

This subsection describes configuration options for limiting routing information sent out by gated from the node. The first two options are located within the `rip` protocol statement. They concern interfaces and gateways and limit where the node can send routing information. The last group of options, located within the `control` section of the configuration file, concern networks and hosts. They limit the extent of routing information that is sent out. Use these options to hide all or part of your network from other networks or to limit network traffic.

Specifying Interfaces that Cannot Send Information Out

```
rip supplier {
    interface intf_addr [ intf_addr ... ] noripout ;
};
```

gated does not send any RIP information through the listed interfaces. A similar option for the HELLO protocol is described in the `gated-configentry` of the *HP-UX Reference*.

Specifying Gateways that Receive Outgoing Information

```
rip supplier {
    sourcegateways gw_addr [ gw_addr ... ] ;
};
```

gated sends RIP information directly to specified gateways. If the `rip` option was changed from `supplier` to `pointpoint`, then only the `sourcegateways` will receive

RIP packets from this host (no RIP packets will be broadcasted). A similar option for the HELLO protocol is described in the `gated-config` entry of the *HP-UX Reference*.

Limiting Information Sent Out About Networks and Hosts

These statements restrict the networks and hosts that are announced by the specified protocols and interfaces. If these configuration statements are not used, all known networks and hosts are propagated. The format of a propagate statement for RIP follows this pattern:

```
propagate proto rip [ interface intf_addr [intf_addr ... ] ] [ metric mnumber
] {
    proto { rip | hello | direct | static | default } [ interface intf_addr
[intf_addr ...] [metric mnumber ] {
        [ announce dest_addr [ metric mnumber ]; ]
        [ noannounce dest_addr ; ]
    };
};
```

The first line (representing the outer layer curly braces) will be called the propagate line. The second line (representing the inner curly braces) will be called the proto line. Both are denoted by their starting keyword: `propagate` or `proto`. The propagate line provides the protocol that these routes will [or will not] be sent out over. An interface option on this line can limit which interfaces these routes will be sent out over with this protocol. The proto line specifies the source of where the routes to be propagated were learned.

`announce` tells gated to announce *only* the specified networks or hosts via the specified protocol. `noannounce` tells gated to announce everything *except* the specified networks or hosts. This allows a choice of announcing only what is on the announce lists, or everything except the networks on the noannounce list on a per interface or protocol basis. The announce and noannounce clauses cannot be used together with the same proto statement.

The metric to use for propagating the route can be specified at various levels in the propagate statement. Each outer level, will be the default to the next inner

Configuration Options

layer. In other words, if no metric is specified on an announce line, it will use the first metric specified on the proto line, propagate line, or the default RIP metric (in that order).

The restrictions set by the propagate statement apply only if the interface learns of the network or host through one of the routing protocols. If a restricted network suddenly becomes unreachable and goes away, it will not be announced again until the interface learns of it again.

For further information and examples, refer to the `gated-conf` entry of the *HP-UX Reference*.

Limiting Routing Information That gated Takes In

This subsection describes options for limiting routing information that arrives at the node. The first two options concern interfaces and gateways. They limit the sources from which the node can receive routing information. The last group of options, located in the control section of the configuration file, concern networks and hosts. They limit the extent of routing information that is received. Use these options to ensure that the routing information that your network receives is from trusted sources.

Specifying Interfaces That Cannot Receive Information

```
rip supplier {
    interface intf_addr [ intf_addr ... ] noripin ;
};
```

`gated` prevents any RIP information coming into the listed interfaces from being processed. A similar option for the HELLO protocol is described in the `gated-conf` entry of the *HP-UX Reference*.

Specifying Gateways that Send Incoming Information

```
rip supplier {
    trustedgateways gw_addr [ gw_addr ... ] ;
};
```

Configuration Options

`gated` listens to RIP information only from specified RIP gateways. The `gw_addr` should be an IP host address on a connected network. A similar option for the HELLO protocol is described in the `gated-conf` entry of the *HP-UX Reference*.

Limiting Information Received About Networks and Hosts

These statements restrict the networks and the hosts that `gated` will accept from the interfaces and routing protocols. If these configuration statements are not used, then all networks and hosts will be listened to by `gated`.

```
accept proto rip [interface intf_addr [intf_addr ...] |gateway gw_addr [gw_addr...]]{
    nolisten src_addr;
};
```

`gated` ignores information regarding `src_addr` that comes in via specified protocols from specified interfaces or gateways. Use the keyword `all` after the keyword `interface` to specify all interfaces on a host.

```
accept proto rip[interface intf_addr [intf_addr...] |gateway gw_addr[gw_addr...]]{
    listen src_addr;
    [listen src_addr:]
};
```

`gated` listens only to information about the network `src_addr` that arrives via the specified protocol, interface, or gateway.

Specifying Tracing Output

```
tracoptions traceflag [traceflag...];
tracefile fname [replace];
```

This clause specifies the desired level of tracing output from `gated`. Tracing output provides useful system information for setting up a node on the network. You specify tracing either on the command line or in the configuration file. Trace information is appended to the trace file unless you specify `replace`. For more information about tracing options, see “Starting `gated`” later in this chapter. Command line options are useful for tracing events in `gated` prior to the reading of the configuration file.

Configuring gated

This section describes how to create a gated configuration file. The file includes configuration options described in the previous section. The order in which these options appear *does* matter. Besides the traceoptions, the following four classes of statements must be specified in this order: definition, protocol, route, and control. See the gated-config entry of the *HP-UX Reference* to determine which statements belong to which class. Creating the configuration file is usually the responsibility of the system administrator.

To configure gated:

1. Create the gated configuration file `/etc/gated.conf`. Decide how you want to configure each routing protocol, then add a line for each to `/etc/gated.conf`.

If the protocols are not explicitly specified, gated assumes the following:

```
rip yes;  
hello no;  
egp no;
```

For more information about configuring protocols, see “Configuration Options” earlier in this chapter and the gated-config entry of the *HP-UX Reference*.

Configuring gated

2. Insert a line in the file for each passive interface address and the default gateway for this node. For example, you might enter the following lines for the configuration shown in Figure 5-1. Within the rip protocol statement [protocol class] the passive interfaces are specified:

```
rip yes {  
    interface 130.0.5.1 190.5.0.1 passive ;  
};
```

A default route is specified using a static statement [route class]:

```
static {  
    default gateway 190.5.0.2 preference 255 ;  
};
```

If an `/etc/gated.conf` file does not contain these statements, existing routes in the node's routing table may time out.

3. Add lines as needed for any additional configuration options. These are described in "Configuration Options" earlier in this chapter.
4. Set up gated so that it will start automatically whenever the node does. Do this by uncommenting the line that starts gated in the `/etc/netbsdsrc` file. If you do not find the line in this file, look in the `/etc/newconfig/netbsdsrc` file for an example of how to start gated.

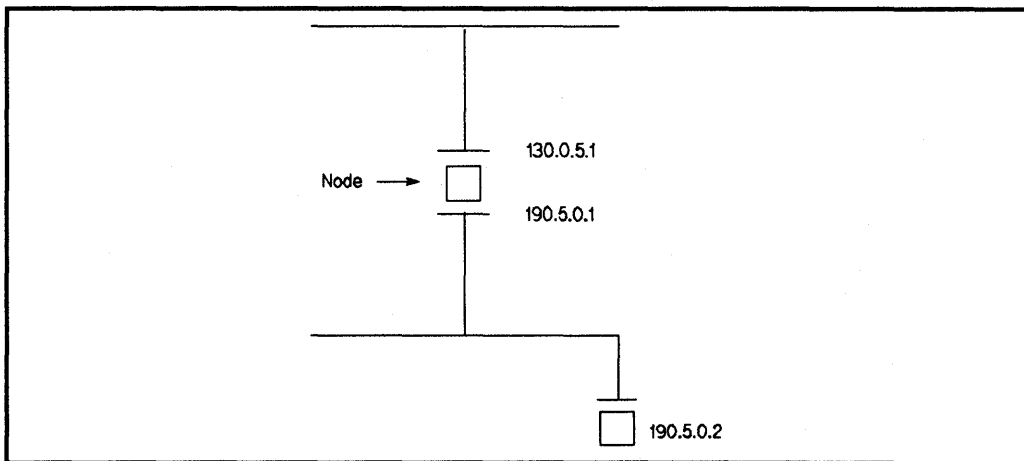


Figure 5-1. Configuring gated

Sample Configurations

Examples of complete gated configuration files are in the next section, "Sample Configurations" and in the `/etc/newconfig/gated/conf` directory.

Sample Configurations

Figure 5-2 and accompanying text describe several examples of how gated might be configured in each node within a networked system.

A. Cluster node (or isolated node)

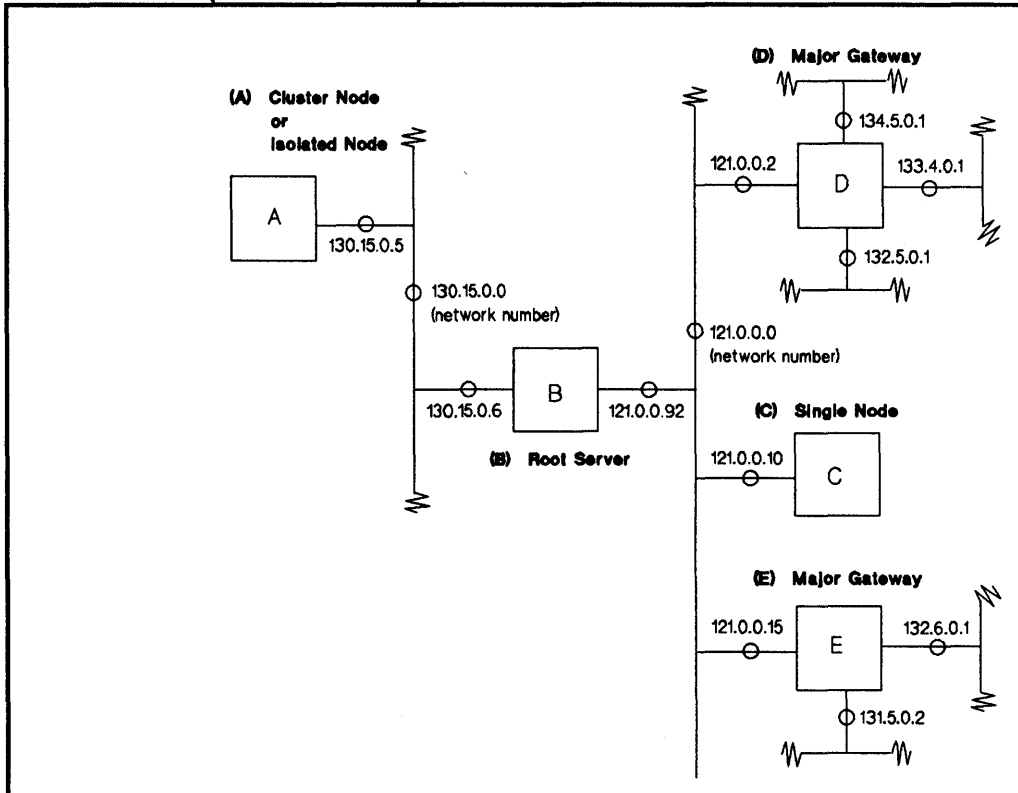


Figure 5-2. Internet Using Gated

Sample Configurations

There is no need to run gated at this node since it is on a LAN with only one gateway. Set a static default route to the cluster server (B) in the `/etc/netlinkrc` file as follows:

```
/etc/route add default 130.15.0.6 1
```

B. Cluster (or root) server node

Run gated to get routing information about the 121.0.0.0 network. Set up `/etc/gated.conf` as follows:

```
rip yes {
    interface 130.15.0.6 121.0.0.92 passive ;
    interface 130.15.0.6 noripout ;
};
hello no;
egp no;

static {
    default gateway 121.0.0.2 preference 255 ;
};
```

In this case, setting rip to yes is like setting rip to supplier. Either argument tells the node to send out RIP packets because the node has at least two interfaces. To reduce traffic on the 130.15.0.0 LAN, use a noripout option on this interface. This prevents RIP from sending packets on the 130.15.0.0 network.

To isolate the 130.15.0.0 LAN, use the following:

```
propagate proto rip interface 121.0.0.92 {
    proto direct {
        noannounce 130.15.0.0 ;
    };
};
```

To further isolate the LAN from the 121.0.0.0 LAN, do not specify any static routes that say you can reach the LAN through B.

Sample Configurations

Always specify the `passive` option with the interface's IP address. This clause tells `gated` to maintain routes even if no other nodes on the 121.0.0.0 network are using RIP. Without this clause, the interface times out the routes when RIP packets stop coming from other nodes. The static default route adds the specified default to the kernel routing table. Setting the preference to 255 will allow this route to be replaced whenever another default route is learned via one of the protocols.

C. Single node on a LAN with multiple gateways

Set up `/etc/gated.conf` as follows:

```
rip yes {
    interface 121.0.0.10 passive ;
};
hello no;
egp no;

static {
    default gateway 121.0.0.10 preference 255 ;
};
```

In this case, setting `rip to yes` is equivalent to setting `rip to quiet` because the C node has only one interface. With one interface, C can listen to RIP traffic on the network but will not broadcast. Listening allows C to update its routing table.

D. Major gateway

Set up `/etc/gated.conf` as follows:

```
rip yes {
    interface 121.0.0.2 132.5.0.1 133.4.0.1 134.5.0.1 passive;
};
hello no;
egp no;
```

This runs RIP on all attached networks.

E. Major gateway

Set up `/etc/gated.conf` as follows:

```
rip yes {  
    interface 121.0.0.15 131.5.0.2 132.6.0.1 passive;  
};
```

F. General Notes

B, D, and E pass routing information among themselves and update their routes accordingly. C listens to the RIP conversation among B, D, and E, and updates its routes accordingly. If gateways D and E can both provide a path to a network but the path through gateway D is shorter, nodes B, C, and E will use gateway D when routing packets to that network. If D goes down, E becomes the new gateway to that network for nodes B, C, and E.

Starting gated

To start gated:

```
gated [-c] [-n] [-t[AickrmtPuRHCpB]] [-f conffile ] [tracefile]
```

Table 5-2 lists the commonly used command line options for gated.

Starting gated

Table 5-2. Command Line Options for gated

Flag	Effect
-t	When used alone, -t causes gated to log all error messages and route changes. It turns on the i, e, and r trace flags automatically. When -t is followed by one or more traceflags, only those flags will be turned on. The -t flag always must immediately precede the other flags, however.
i	Logs all internal errors and interior routing errors.
e	Logs all external errors due to EGP, exterior routing errors, and EGP state changes.
r	Logs all routing changes.
p	Traces all EGP packets sent and received.
u	Displays the entire contents of routing packets sent and received (when used with p, R, or H).
R	Traces all RIP packets received.
H	Traces all HELLO packets received.
-c	Specifies that the configuration file will be parsed for syntax errors, and then gated will exit.
-n	Specifies that gated will not modify the kernel's routing tables.

For information about other traceflags that you can specify on the command line, refer to the `gated` entry of the *HP-UX Reference*.

Table 5-3 shows the valid traceflags for gated configuration files. Use traceflags if you are setting up a node and want a certain type of tracing sent to a log file. For example, if you want to see all RIP packets sent and received by a node, add the following lines to `/etc/gated.conf`

```
tracoptions rip update;  
tracefile "/tmp/logfile";
```

Table 5-3. Traceflags for gated Configuration Files	
Flag	Effect
internal	Logs all internal errors and interior routing errors.
external	Logs all external errors due to EGP, exterior routing errors, and EGP state changes.
route	Logs all routing changes.
egp	Traces all EGP packets sent and received.
update	Logs all routing updates sent.
rip	Traces all RIP packets received.
hello	Traces all HELLO packets received.
icmp	Traces all ICMP redirect packets received.
nostamp	will not print a timestamp to the log file every ten minutes.
general	A combination of internal, external, route and egp.
all	Enables all of the above tracing flags.

To find out what other traceflags are available within the configuration file, see the gated-config entry in the HP-UX Reference.

To Find Out if gated is Running

To find out if gated is running:

```
ps -p'cat /etc/gated.pid'
```

This command reports the process identification (PID), current time, and the command invoked (gated).

Troubleshooting gated

If `gated` is not operating properly, use this section to identify and correct the problem.

Troubleshooting Tools and Techniques

This section describes the available tools for general troubleshooting of `gated`.

syslog Output

`gated` logs informational and error conditions with `syslogd`. For example, `syslogd` logs syntax errors in the configuration file `syslog`. The facility used for logging is `LOG_DAEMON`. The messages range from `LOG_NOTICE` to `LOG_EMERG`.

To enable `syslogd` logging, you must configure `/etc/syslog.conf`. For details on how to do this, see the “System Log Files” section in chapter 3 of this manual.

gated Tracing

`gated` prints information about its activity in the form of tracing output. This information includes routes that `gated` reads, adds, and deletes from the kernel routing table, as well as packets sent and received.

You can specify tracing either with `gated` command line options or with the `traceoptions` options in `/etc/gated.conf`. Using any of the following combinations, you can determine where the tracing output is printed and whether tracing is even done:

- If you use command line `traceflags` options and no log file, tracing output is printed on the display where `gated` was started.
- If you use command line `traceflags` options and a log file, tracing output is printed to the log file.
- If you do not use specify `traceflag` options or a log file on the command

line but specify only traceoptions, in the config file, then no tracing takes place.

Once tracing is started to a log file, the log file can be rotated. A SIGUSR1 signal causes gated to stop tracing and closes the log file. The log file can then be moved out of the way. Another SIGUSR1 signal starts tracing again to a new log file. If the traceflags are changed before tracing is started up again, the new traceflags will take effect.

Dump the gated routing table

Sending gated a SIGINT signal causes gated to write out its information in /usr/tmp/gated_dump. The information includes the interface configurations, EGP neighbor status, and the routing tables.

ripquery

ripquery is a support tool that can be used to query gated for RIP routing information. You can also use ripquery to query other non-gated RIP gateways. To do so you may need to use the -p option, since the default query sent by ripquery may not be supported by all RIP gateways.

The information returned when using -p may not be a complete list of routes because of the poison reverse and split horizon techniques. (Poison reverse and split horizon are explained in the RIP RFC.)

Problems

This section covers typical problems with gated operation.

1. gated does not do what you thought you had configured it to do.

First, check the syslogd output for any syntax errors that may have been flagged.

To detect incorrect configuration commands, use gated tracing and/or the information dumping techniques described earlier.

Troubleshooting gated

Configuration commands are easily found in the trace output. For example, the interface statement with the passive option is printed in the trace output.

If the parse and lex trace flags are turned on, the interface statement: `interface 17.13.119.5 passive;` would be seen as follows in the tracefile:

```
lex: /etc/gated.conf:4 KEYWORD: interface
parse: /etc/gated.conf:4 octet: 17
parse: /etc/gated.conf:4 octet: 13
parse: /etc/gated.conf:4 octet: 119
parse: /etc/gated.conf:4 octet: 5
parse: /etc/gated.conf:4 IP_ADDR: 17.13.119.5
parse: /etc/gated.conf:4 INTERFACE: 17.13.119.5 (lan0)
lex: /etc/gated.conf:4 KEYWORD: passive
parse: interface lan0: up addr 17.13.119.5 metric 0 index 3 preference 0
parse: interface lan0: broadaddr 17.13.119.255
parse: interface lan0: net 17.0.0.0 netmask 255.0.0.0
parse: interface lan0: subnet 17.13.119.0 subnetmask 255.255.255.0
parse: interface lan0: flags Broadcast Subnet Interface NoAge
```

The NoAge flag in the last line of the trace output shows that it has been set to passive.

The results of this same command can also be found in the `gated_dump` file, although not as easily. In the following segment of a `gated_dump` file, the interface is listed as passive.

```
lan0 17.13.119.5 Index: 3 Preference: 0 Metric: 0
Up-down transitions: 0 Flags: Broadcast Subnet Interface NoAge
Broadcast Address: 17.13.119.255
Net Number: 17.0.0.0 Net Mask: 255.0.0.0
Subnet Number: 15.13.119.0 Subnet Mask: 255.255.255.0
```

A common mistake is to always expect gated to send out RIP packets when you specify `rip yes` in a configuration file. gated will only be an active RIP participant if the host can be a gateway (the host has more than one network interface).

You can detect this mistake by tracing gated activity. Check whether gated is only receiving packets, or whether the following line appears in the tracefile:

```
rip_init: Acting as RIP supplier to our direct nets
```

2. gated removes the local interface route.

gated deletes the local interface route if it has not received an EGP, HELLO, or RIP packet from that interface for a period of time (usually several minutes). The determination is that the interface must not be working properly.

On some hosts, broadcast packets sent out from an interface are not received by the local host. If there is no other active gated activity on the network, the local host may erroneously remove the route. Use the `passive` option with the interface in the statement for these hosts to prevent the interface route from being deleted.

To correct this problem, run `ifconfig` again for the interface. This adds the interface route back to the routing table.

3. gated deletes routes from the routing table that existed before gated was started.

gated maintains a complete routing table in the user space, and keeps the kernel routing table in sync with this table. gated starts out reading the routes in the kernel. If no routing activity suggests that the route be kept, it is removed from both the gated and kernel routing tables. If you want to maintain this route and gated will not hear about this route over any of the routing protocols, you should install the route from the configuration file as follows:

```
static {  
    2.0.0.0 gateway 1.1.1.6 metric 1 ;  
};
```

You may want to put `noannounce` clauses in the `propagate` statements to keep these extra routes from being advertised. If the route is the default route, use the following:

Troubleshooting gated

```
static {  
    default gateway 1.1.1.6;  
};
```

4. gated adds routes that appear to be incorrect.

Start by looking at the routing table maintained by gated. Send gated a SIGUSR1, and look at the information output in `/usr/tmp/gated_dump`. Look for the entry of the route in question. The entry will show the protocol that this route was heard over and the first hop gateway. The first hop gateway is likely to be the immediate source of the information.

If the route was learned over RIP, use `ripquery` to query the first hop gateway for the route. That gateway may claim to have heard the route from a gateway further on. If the first hop gateway is another host running gated, have that host's gated dump its routing table to find out where it learned about the route. You may have to repeat this process several times to track down the original source of the route. If the problem is that you expect the route to go through a different router, turn on gated tracing. The tracing tells you which routers are advertising this route and the values attached to those routes.

5. gated does not add routes that you think it should.

Tracking down this problem is much like the last problem. You expect one or more routers to advertise the route. Turn on gated tracing to verify that gated is receiving packets of the type of routing protocol you expect. If these packets do not contain a route you expect to be there, trace packets on the router you expect to advertise the route.

Internetwork Mail Routing

This chapter describes `sendmail`, the ARPA Services' internetwork mail routing facility. `sendmail` provides a modular mail routing system that relays incoming and outgoing mail to the appropriate programs for delivery or further routing. In addition, `sendmail` enables your system to send mail to and receive mail from other hosts on a local area network or through a gateway. `sendmail` can be configured to operate with many transport protocols.

Because installing `sendmail` is not essential for the operation of the rest of the ARPA Services product, this chapter includes information to help you decide whether to install and operate `sendmail` on your system.

Note `sendmail` does not run on the Applications Execution Environment. You must install the Programming Environment to use `sendmail`.

Chapter Overview

This chapter includes the following:

- Key terms used in internetwork mail routing.
- An overview of `sendmail`.
- A detailed description of how `sendmail` works.
- Guidelines for deciding whether to install `sendmail`.

Key Terms

- Instructions for installing sendmail.
- Instructions for operating sendmail.
- A description of the default routing provided by the supplied sendmail configuration file.
- Troubleshooting information for sendmail.

Key Terms

The terms in Table 6-1 are used in this chapter and are consistent with widely known and accepted networking terminology.

Term	Definition
alias	An alternate name for a recipient or a list of recipients (a mailing list).
delivery agent	A program that accepts messages from a routing facility and delivers them to a final, local destination, or passes the message, via a communications medium, to a receiving agent for further routing or remote delivery. Also called mailer.
domain (DNS)	A subtree of the hierarchical tree structure containing network resource names. Domain Name System (DNS) domains roughly correspond to organizational or geographic structure
envelope	The information needed for routing and delivering a message and for returning error information (the addresses of senders and recipients).

Table 6-1. Key Terms	
Term	Definition
mailer	A term commonly used to refer to either a user agent or a delivery agent. In this chapter, mailer usually means delivery agent.
message	The information unit transferred by a message transfer system. A message is composed of an envelope, a message header and a message body.
message address	uniquely identifies a recipient, either by specifying a mailbox and a domain (e.g., ddm@HP.COM), or by specifying a mailbox and a route from the originator to the recipient (e.g., node1!ddm), or just a mailbox (e.g. ddm on the local host).
message body	The principal information the originator transmits to the recipients (text).
message header	Auxiliary message information consisting of a collection of fields. Each field is one line of text that includes a field designator (e.g., To:, From:, Date:, Subject:, or In-Reply-To:), which is followed by appropriate information. RFC 822 contains detailed information about headers and their formats.
message transfer system	A system that relays messages from an originator to a recipient. The system is composed of user agents, receiving and delivery agents and routing facilities.
originator	A user or a process that sends a message (the sender).
recipient	A user, file or process to which a message is sent.
receiving agent	A process that receives messages from a remote delivery agent (mailer) and passes it to a routing facility. The <code>sendmail</code> daemon is an example of a receiving agent. It receives messages from a peer delivery agent via a local area network using the SMTP protocol. Once the <code>sendmail</code> daemon collects the message, it passes it to <code>sendmail</code> for further processing and routing.

sendmail Overview

Term	Definition
routing facility	A program that receives messages from user agents, receiving agents or message originators, determines routing requirements and invokes the appropriate delivery agent to deliver the message. <code>sendmail</code> is a routing facility.
sender	The originator of a message.
SMTP	SMTP (Simple Mail Transfer Protocol) is a protocol used for transmitting messages through the ARPA Internet and local area networks. <code>sendmail</code> implements both an SMTP receiving agent and an SMTP delivery agent. RFC 821 is the specification for SMTP.
transport agent	A term referring either to a receiving or delivery agent, depending on the context in which it is used.
user agent	A program or set of programs that provides a human interface for creating, inspecting and managing messages. Once a message is created, the user agent passes the message to a routing facility for processing. HP-UX user agents are <code>mailx</code> , <code>mail</code> , and <code>elm</code> .

sendmail Overview

Function

`sendmail` acts as a “post office” to which all messages can be submitted for routing. Address interpretation is controlled by a production system that interprets both Internet-style addressing (that is, `user@domain`) and UUCP-style addressing (that is, `host!user`). This production system is defined by the contents of the `sendmail` configuration file and is powerful enough to rewrite message header addresses to conform to standards on many common target networks. `sendmail` can be called directly by a user, by a user agent, or by a receiving agent.

Features

sendmail has or can be configured for the following capabilities:

- System-wide mail address aliasing.
- User-maintainable mailing lists.
- User-controlled mail forwarding.
- Automatic routing to network gateways.
- Message queueing for re-attempting failed mail deliveries.
- Delivery of error message transcripts to senders from any unsuccessful mail transfers.
- An SMTP server for receiving network mail transfers.
- Access to the Domain Name System (BIND).
- Access to external name server programs.

How sendmail Works

How sendmail Works

sendmail performs its task in two phases, first collecting messages and then routing them. A message has three parts: an envelope, a message header and a message body.

- The **envelope** consists of the sender address, recipient address, and routing information shared by the programs that create, route, and deliver the message. It is usually not seen directly by either the sender or recipients of the message.
- The **message header** consists of a series of standard text lines used to incorporate address, routing, date, and other information into the message. Header lines may be part of the original message, and may also be added or modified by the various mail programs that process the message. Header lines may or may not be used by these programs as envelope information.
- By default, the first blank line in the message terminates the message header. Everything that follows is the **message body**, and is passed uninterpreted from the sender to the recipient.

sendmail can be invoked by any of the following:

- A user agent that calls sendmail to route a piece of mail.
- A receiving agent that calls sendmail to route a piece of mail received from the network.
- A user that calls sendmail directly.
- The sendmail daemon.

Once sendmail collects the message, it routes the information. To route the message, sendmail does the following:

- Rewrites the recipient and sender addresses given to it to conform to the standards of the target network.

How sendmail Works

- If necessary, adds lines to the message header so that the recipient is able to reply.
- Passes the mail to one of several specialized delivery agents for delivery.

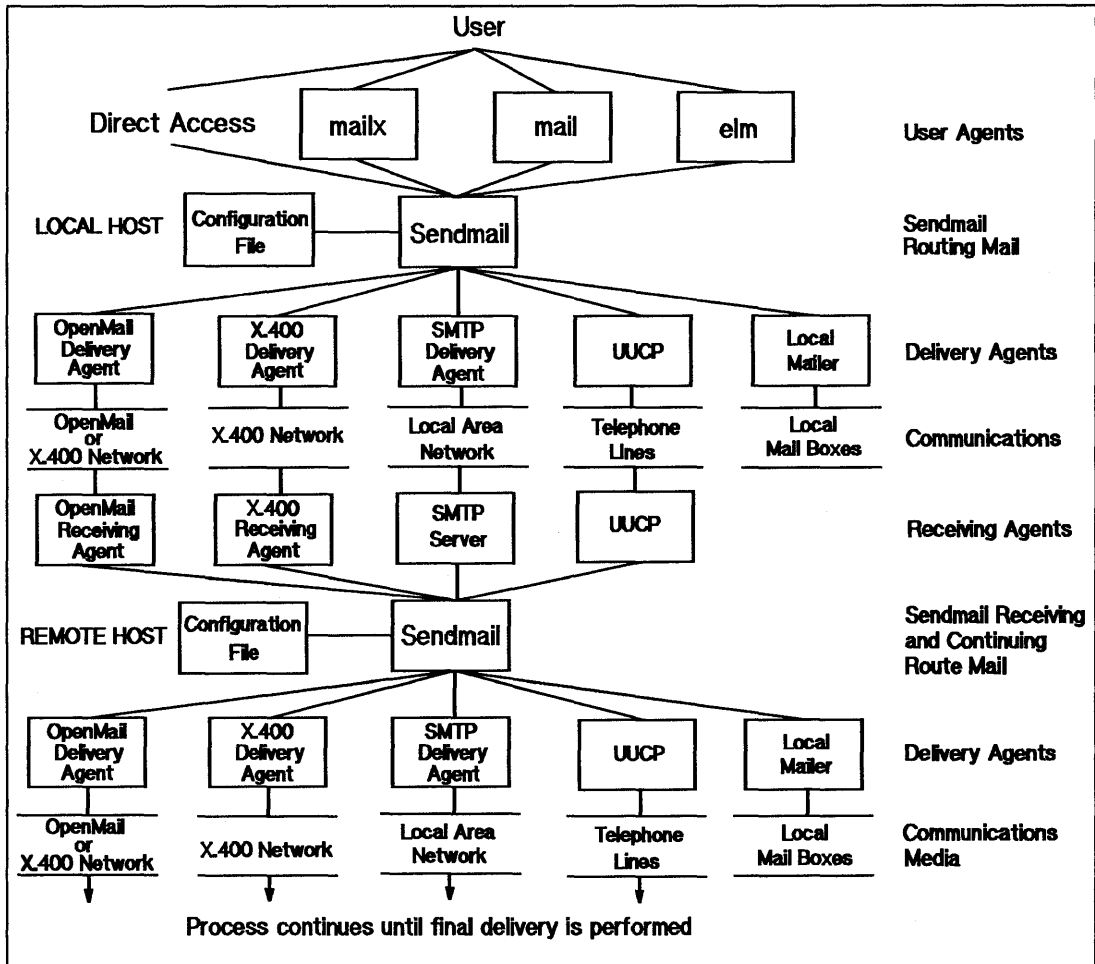


Figure 6-1. Flow of Mail Through sendmail

How sendmail Works

When a receiving agent receives incoming mail, it passes the mail to `sendmail` for routing in the same way that a user agent invokes `sendmail`. Figure 6-1 outlines the flow of messages through `sendmail`.

Collecting Messages

`sendmail` collects messages in three ways:

- From the command line's argument vector and `stdin`.
- Via the SMTP protocol.
- From the mail queue.

`sendmail`'s three message collection methods are described in the following sections.

From the Argument Vector and Standard Input

When `sendmail` collects messages from the argument vector and `stdin`, it is called with the recipient list following the program name on the command line. For example, a user, user agent (e.g., `mailx`), or receiving agent (for example, `UUCP`) might issue the command:

```
/usr/lib/sendmail -f node0!mickie dan@node1 node2!david  
(program name)          (argument vector)
```

First, `sendmail` determines the originator's identity (`node0!mickie`). If the message originates locally, the sender is the user on whose behalf `sendmail` is being executed. If the message is from `UUCP` as in the example above, the sender is identified by the `-f` flag on the command line.

Second, `sendmail` collects the recipient addresses from the command line (`dan@node1` and `node2!david`). It eventually interprets these addresses and determine how to route the message to these recipients according to a set of rules in the configuration file, `/usr/lib/sendmail.cf`.

The information collected so far comprises the envelope.

Finally, `sendmail` reads the header lines and the message body from standard input. The header lines are immediately stored in memory, since `sendmail` may need to do some processing on them before routing the message. By default, the first blank line in the input terminates the header. Everything following the header is treated as the message body and is stored uninterpreted in a temporary file.

Using the Simple Mail Transfer Protocol (SMTP)

`sendmail` implements a client and a server for the SMTP specified in RFC 821. `sendmail` and other mail handling programs use this protocol to transfer messages over TCP/IP networks. The `sendmail` daemon listens for connection requests on the SMTP TCP port, and when it receives such a request, forks an SMTP server to communicate with the sending program (perhaps another `sendmail`). Local user agent programs can also communicate with `sendmail` using SMTP over pipes, by running `/usr/lib/sendmail -bs`.

SMTP is an interactive protocol. The envelope information is passed to `sendmail` as a series of commands. `sendmail` acknowledges these commands with reply codes, whose meaning is defined in the SMTP specification. The following is an example of an SMTP delivery agent (S) sending a message to an SMTP receiving agent (R).

```
(R) 220 gik.cnd.hp.com HP sendmail ready at Sun, 9 Apr 90
    18:27:47 mdt
(S) HELO bip.cnd.hp.com
(R) 250 gik.cnd.hp.com Hello bip.cnd.hp.com, pleased to meet you
(S) MAIL FROM:<fred@bip.cnd.hp.com>
(R) 250 <fred@bip.cnd.hp.com>...Sender ok
(S) RCPT TO: <ernie@gik.cnd.hp.com>
(R) 250 <ernie@gik.cnd.hp.com>... Recipient ok
(S) RCPT TO: <louise@enk.cnd.hp.com>
(R) 250 <louise@enk.cnd.hp.com>... Recipient ok
(S) DATA
(R) 354 Enter mail, end with "." on a line by itself
(S) Date: Sun, 9 Apr 90 18:19:38 mdt
```

How sendmail Works

```
(S) From: Fred Schmelp <fred@bip.cnd.hp.com>
(S) To: Ernie Bork <ernie@gik.cnd.hp.com>,
(S) Louise Debris <louise@enk.cnd.hp.com>
(S) Subject: SMTP example
(S)
(S) This is an example of how the SMTP protocol is used to transfer mail.
(S) .
(R) 250 Ok
(S) QUIT
(R) 221 gik.cnd.hp.com closing connection
```

Note that the envelope information is transferred with the HELO, MAIL, and RCPT commands, and the header and message body are transferred as DATA.

From the Queue

sendmail also collects messages from the mail queue. If a message is temporarily undeliverable, sendmail saves the message components in two files created in the mail queue directory. The message body is saved in a "data" file, and the envelope information, the header lines, and the name of the data file are saved in a "queue control" file.

Typically, the sendmail daemon is run with the `-q time_interval` option. For example:

```
/usr/lib/sendmail -bd -q30m
```

In this case, every 30 minutes the daemon forks a child to process any messages currently in the queue. sendmail also processes the mail queue once immediately if called with:

```
/usr/lib/sendmail -q
```

sendmail reads the queue control file to re-collect the pre-processed envelope information, the header lines, and the name of the data file containing the message body. sendmail then proceeds to process the message just as it did when it was originally collected.

The format of the queue files is described in the “Operating sendmail” section of this chapter.

Routing Messages

Once sendmail collects a message via any of the three collection methods, it routes the message to each of the specified recipient addresses. In order to route a message to a particular address, sendmail must resolve that address to a *{delivery-agent, host, user}* triple. This resolution is based on rules defined in the sendmail configuration file, `/usr/lib/sendmail.cf`. Sendmail keeps one copy of the message body (in a file) and one copy of the message headers (internally) and processes these separately for each delivery agent it invokes to route the message.

Each delivery agent is invoked separately for each host it is being used to route to. Some delivery agents can accept multiple users in a given invocation; others must be invoked separately for each recipient.

Routing to Remote Systems

The interface between sendmail and its delivery agents (mailers) is similar to that between sendmail and the user agents or receiving agents that invoke it.

To invoke a delivery agent, sendmail normally sets up a pipe and forks. The child execs the delivery agent with a command line constructed according to a template in the configuration file. The parent writes the message header and body to the pipe. If the command line template includes `$u`, sendmail passes the recipient address to the delivery agent on the command line. Otherwise, sendmail opens a bi-directional pipe to its child, and uses SMTP to transmit the envelope and the message.

If the delivery agent is specified as `[IPC]`, sendmail does not exec an external delivery agent but instead opens a TCP/IP connection to the SMTP server on the specified host and transmits the message using SMTP.

How sendmail Works

Routing to Local Destinations

If an address resolves to the local mailer, `sendmail` looks up the address in its alias database and expands it appropriately if found. The aliasing facility may be used to route mail to programs and to files (`sendmail` does not mail directly to programs or files).

Mail to programs is piped to the prog mailer (`/bin/sh -c`), executing a command specified as the right-hand side of the alias. Mail to a file is directly appended to the file by `sendmail` if certain conditions of ownership and permission are met. Users may also alias themselves by means of `.forward` files in their home directories. `sendmail`'s aliasing facility and `.forward` files are described in detail in the "Operating sendmail" section in this chapter.

After all alias expansion is complete, mail that is addressed to a local user name is routed to the local mailer (`/bin/rmail`), which deposits the message in the user's mailbox.

Error Handling

By default `sendmail` immediately reports to standard output any errors that occur during the routing or delivery of a message. Examples of this would be if `sendmail` could not resolve a recipient address or the delivery agent immediately reports a failure.

If a delivery failure is "temporary," `sendmail` leaves the message in the mail queue and tries to deliver it again later. For more information about temporary failures, see the "Mail Queue" section of "Operating sendmail" in this chapter.

In most cases, if message delivery fails permanently on a remote system, mail that includes a transcript of the failed delivery attempt and the undelivered message is returned to the sender. This transcript includes any standard error output from the delivery agent that failed. Mail is also returned, if `sendmail` has been unable to deliver it for the "queue timeout period," which is normally 3 days. For more

information about permanent failures, see the “Mail Error Handling” section of “Operating sendmail” in this chapter.

Deciding Whether to Install sendmail

Installing sendmail is optional. If sendmail is not installed, only local and UUCP mail are supported.

sendmail must be installed in order to do the following:

- Deliver mail to other hosts via SMTP over a LAN or WAN.
- Route X.400 mail via the X.400/9000 delivery agent.
- Route OpenMail or X.400 mail via the OpenMail product.

The supplied sendmail configuration file and installation instructions are intended to make operating sendmail relatively simple and trouble free.

Installing sendmail

There are two ways to install and start sendmail:

- Use SAM with the supplied sendmail configuration file, which provides basic connectivity appropriate for many installations. SAM (System Administration Manager) is a menu-driven utility for performing system administration tasks.
- Install and start it manually, as explained in the “Manually Installing sendmail” section of this chapter.

Installing sendmail

Using SAM to Install sendmail

SAM installs the default sendmail configuration file, makes sendmail executable, creates the system-wide mail alias database files from the aliases source file, and invokes the daemon.

The following steps tell how to use SAM to install sendmail:

1. At the HP-UX prompt, type: `sam`
and wait for SAM's main menu to appear.
2. Select the Networking/Communications menu item.
3. Select Services Enable/Disable.
4. Select the sendmail menu item.
5. Select the Enable action.
6. Select OK. View the help screens if you need additional information.
7. Exit the Services Enable/Disable screen by selecting Exit from the List menu. At the Networking/Communications screen, select either Previous Level to get to a previous level, or select Exit SAM to exit from SAM.

Verification

To verify that sendmail has been properly installed and is properly working, see the "Examples for Verifying sendmail" section of this chapter.

Manually Installing sendmail

When you installed the ARPA Services product, all files and directories needed to use sendmail were installed in the correct directories with the proper permissions except the alias file (`/usr/lib/aliases`), the sendmail configuration file (`/usr/lib/sendmail.cf`), and the sendmail executable file (`/usr/lib/sendmail`).

To install sendmail, you must do the following:

- Install a sendmail configuration file as `/usr/lib/sendmail.cf`.
- Make sendmail executable.
- Create system-wide mail aliases.
- Invoke the sendmail daemon.

1. Installing a Configuration File

The sendmail configuration file, `/usr/lib/sendmail.cf`, sets a number of operational parameters for sendmail and determines how sendmail routes messages based on their recipient addresses.

The supplied configuration file, `/etc/newconfig/sendmail.cf` is appropriate for many installations without modification. In particular, when initially installing sendmail, simply copying the supplied configuration file into place provides basic connectivity and permits you to integrate your system into your mail environment without immediately having to edit the configuration file. The default routing implemented in the supplied configuration file is described in the next section, “Default Routing.”

In addition, a number of localizations and routing options are provided. Descriptions of these options and detailed editing instructions are provided in the supplied configuration file itself.

Note	HP does provide support for the supplied configuration file with no modifications or with the modifications described under “Localizations” and “Routing Options” in the supplied configuration file itself. HP does not provide support for configuration files modified in other ways.
-------------	--

Installing sendmail

To install the supplied configuration file unmodified, copy the file `/etc/newconfig/sendmail.cf` to `/usr/lib/sendmail.cf`.

To make any of the supported modifications, copy `/etc/newconfig/sendmail.cf` to `/usr/lib/sendmail.cf` and edit `/usr/lib/sendmail.cf` according to the instructions in the file itself.

To use some other sendmail configuration file, copy it to `/usr/lib/sendmail.cf`.

Default Routing

The supplied configuration file, if installed unmodified, routes mail depending on the syntax of the recipient addresses as described in the following sections.

Local Addresses

The following forms, where *localhost* is the local host name, are recognized as local addresses and delivered locally:

```
user
user@localhost
user@localhost.localdomain
user@alias
user@alias.localdomain
user@[local.host's.internet.address]
localhost!user
localhost!localhost!user
user@localhost.uucp
```

UUCP Addresses

Where host is not the local host name, the addresses of the forms

```
host!user  
host!host!user  
user@host.uucp
```

are recognized as UUCP addresses. If your host has a direct UUCP connection to the next host in the path, the mail is delivered to that host via UUCP. If not, the message is returned with an error. The supplied configuration file provides detailed instructions for arranging to relay such mail through hosts to which you can connect.

SMTP Addresses

RFC 822-style addresses in any of the forms:

```
user@host  
user@host.domain  
<@host,@host2,@host3:user@host4> (source route)  
user@[remote.host's.internet.address] (domain literal)
```

where host is not the local host name, are routed via SMTP over TCP/IP.

If the name server is in use, sendmail requests MX (mail exchanger) records for the remote host; if there are any, it attempts to deliver the mail to each of them, in preference order, until delivery succeeds.

Otherwise, sendmail connects directly to the recipient host and delivers the message.

Installing sendmail

Mixed Addresses

The supplied configuration file interprets address operators with the precedence:

@ . ! , %

This means that recipient addresses using mixtures of these operators are resolved as shown Table 6-2.

Address	{ Mailer	Host	User }	Recipient
user%hostA@hostB	tcp	hostB	user%hostA@hostB	user@hostA
user!hostA@hostB	tcp	hostB	hostA!user@hostB	hostA!user
hostA!user%hostB	uucp	hostA	user@hostB	user@hostB

2. Making sendmail Executable

The user agent programs use `sendmail` to route mail only if `sendmail` is executable.

To make `sendmail` executable, as superuser, issue the command:

```
chmod 555 /usr/lib/sendmail
```

`sendmail` is normally run `setuid` to root (mode 5555). The default configuration is believed to be safe. However, it is possible to misconfigure `sendmail` so that it inappropriately promotes the privilege of ordinary users.

Installing sendmail

If `sendmail` does not run `setuid` to root, this risk is eliminated. Note that this causes `sendmail` to ignore the 'S' mailer flag (not specified in the mailers defined in the default configuration file) and the values of the 'u' and 'g' configuration options, since it is unable to `setuid` to these users when executing mailers.

`sendmail` can be run non-`setuid` (mode 1555) if the following changes to the default configuration are made:

- The queue directory (by default `/usr/spool/mqueue`) must be writable by all (mode 0777).
- The alias database (by default `/usr/lib/aliases.dir` and `/usr/lib/aliases.pag`) must be writable by all (mode 0666). If these files do not already exist, the superuser must first create them and then `chmod` them:

```
/usr/lib/sendmail -bi
chmod 666 /usr/lib/aliases.dir /usr/lib/aliases.pag
```

- It is necessary to start and kill the `sendmail` daemon as superuser.

Making these changes creates some security risk. Anyone will be able to delete mail from the mail queue. However, no one will be able to read other people's mail in the mail queue.

3. Creating System-wide Mail Aliases

The `/etc/newconfig/aliases` file is an example alias file that contains default aliases needed by `sendmail`. To create system-wide mail aliases, copy `/etc/newconfig/aliases` to `/usr/lib/aliases`.

You can add any aliases that are appropriate for your system by editing the `/usr/lib/aliases` file. RFC 822 requires that a "postmaster" address be defined on each host. HP suggests that the person designated as postmaster be the person that is responsible for handling problems that occur with the mail system.

Installing sendmail

sendmail uses a database form of the `/usr/lib/aliases` file and does not recognize the aliases until the alias database has been initialized. Once sendmail is executable issue the following command:

```
newaliases
```

This creates the alias database files, `/usr/lib/aliases.dir` and `/usr/lib/aliases.pag`.

4. Invoking the sendmail Daemon

To receive mail from the network, and to ensure that queued messages are retransmitted, a sendmail daemon must be running.

To start the sendmail daemon, as superuser, issue the following command:

```
/usr/lib/sendmail -bd -q30m
```

The `-bd` mode initializes the sendmail daemon to receive mail from the network. The `-q30m` flag causes sendmail to process the mail queue every 30 minutes.

Whenever your system is rebooted, if sendmail is executable, the `/etc/netbsdsrc` script does the following:

- Starts the sendmail daemon to accept SMTP connections from the network and to process the mail queue every 30 minutes.
- Logs the restart of the sendmail daemon in the mail log, usually `/usr/spool/mqueue/syslog`.

5. Verifying sendmail Examples

You can verify that sendmail has been properly installed and is properly working by the following:

- Mailing to a local user.
- Mailing to a remote user via the UUCP transport (if you are using it).
- Mailing to a remote user via the SMTP transport (if you are using it).

Local Mailing

To check your local mailer or user agent, mail a message to a local user (for example, Joe) on your system:

```
date | mailx -s "Local sendmail Test" joe
```

This should result in a message similar to the following being sent to Joe:

```
From joe Wed Aug 6 09:18 MDT 1986
Received: by node2; Wed, 6 Aug 86 09:18:53 mdt
Date: Wed, 6 Aug 86 09:18:53 mdt
From: Joe User <joe>
Return-Path: <joe>
To: joe
Subject: Local sendmail Test
```

```
Wed Aug 6 09:18:49 MDT 1986
```

An entry in your `/usr/spool/mqueue/syslog` file should have been logged for the local message transaction. It should look similar to this:

```
Aug 6 09:18:54 node2 sendmail[7790] AA07790: from=joe, size=64, class=0
Aug 6 09:18:55 node2 sendmail[7790] AA07790: to=joe, delay=00:00:02,
stat=Sent, mailer=local
```


Installing sendmail

Remote Mailing via the UUCP Transport

For this test, mail a message to a remote user via the UUCP transport by using a *host/user* address, where *host* is a system to which your local host has a direct UUCP connection (see the `uname` entry in section 1 of the *HP-UX Reference*).

To verify both inbound and outbound UUCP connections, set up the remote user to forward the message back to your system or mail the message in a loop (for example, *remote_host/my_host/user*). For example, if you try,

```
date | mailx -s "UUCP Test" node1!node2!joe
```

and node2 is your local host, you should receive a message similar to this:

```
From node1!node2!joe Wed Aug 6 09:48 MDT 1986
Received: by node2; Wed, 6 Aug 86 09:48:09 mdt
Return-Path: <node1!node2!joe>
Received: from node1.UUCP; Wed, 6 Aug 86 09:30:16
Received: by node1; Wed, 6 Aug 86 09:30:16 mdt
Received: from node2.UUCP; Wed, 6 Aug 86 09:26:18
Received: by node2; Wed, 6 Aug 86 09:26:18 mdt
Date: Wed, 6 Aug 86 09:26:18 mdt
From: Joe User <node1!node2!joe>
To: node1!node2!joe
Subject: UUCP Test
```

```
Wed Aug 6 09:26:15 MDT 1986
```

An entry in your `/usr/spool/mqueue/syslog` file should have been logged for the UUCP mail transaction. It should look similar to this:

```
Aug 6 09:26:18 node2 sendmail[7925] AA07925: from=joe, size=69, class=0
Aug 6 09:26:20 node2 sendmail[7925] AA07925: to=node1!node2!joe, delay=00:00:02,
stat=Sent, mailer=UUCP, host=node1
Aug 6 09:48:10 node2 sendmail[8233] AA08233: from=node1!node2!joe, size=339,
class=0
Aug 6 09:48:11 node2 sendmail[8233] AA08233: to=joe, delay=00:00:02, stat=Sent,
mailer=local
```

Note

In this example, if you mail to yourself and if the remote system is running `sendmail`, be sure the `m` option is set in the configuration file on the remote system; the remote system's configuration file should contain a line beginning with `0m`. If such a line is not in the remote host's configuration file, `sendmail` on the remote host notices that the sender is the same as the recipient and your address is removed from the recipient list.

Remote Mailing Via The SMTP Transport

For this test, mail a message to a remote user via the SMTP transport with a `user@host` address, where `host` is a system that provides an SMTP server (for example, the `sendmail` daemon).

To verify both inbound and outbound SMTP connections, set up the remote user to forward the message back to your system or mail the message in a loop (e.g., `joe%node2@node1`, where `node2` is your local host). For example, if you try,

```
date | mailx -s "Round Robin SMTP" joe%node2@node1
```

you should receive a message similar to:

```
From joe@node2 Wed Aug 6 14:22 MDT 1986
Received: from node1 by node2; Wed, 6 Aug 86 14:22:56 mdt
Return-Path: <joe@node2>
Received: from node2 by node1; Wed, 6 Aug 86 14:25:04 mdt
Received: by node2; Wed, 6 Aug 86 14:22:31 mdt
Date: Wed, 6 Aug 86 14:22:31 mdt
From: Joe User <joe@node2>
To: joe%node2@node1
Subject: Round Robin SMTP
```

```
Wed Aug 6 14:22:28 MDT 1986
```

Operating sendmail

An entry in your `/usr/spool/mqueue/syslog` file should have been logged for the SMTP mail transaction. It should look similar to this:

```
Aug 6 14:22:31 node2 sendmail[11279] AA11279: from=joe, size=76, class=0
Aug 6 14:22:44 node2 sendmail[11279] AA11279: to=joe%node2@node1, delay=00:00:13
, stat=Sent, mailer=tcp, host=node1
Aug 6 14:22:57 node2 sendmail[11289] AA11289: from=<joe@node2>, size=247, class=
0
Aug 6 14:22:58 node2 sendmail[11289] AA11289: to=<joe@node2>, delay=00:00:02, st
at=Sent, mailer=local
```

The first two entries show the outbound SMTP transaction and the last two entries show the inbound SMTP transaction.

Note In this example, if you mail to yourself and if the remote system is running `sendmail`, be sure the `m` option is set in the configuration file on the remote system; the remote system's configuration file should contain a line beginning with `Om`. If such a line is not in the remote host's configuration file, `sendmail` on the remote host notices that the sender is the same as the recipient and your address is removed from the recipient list.

Operating sendmail

The sendmail Daemon

The `sendmail` daemon is run for two purposes:

- To receive incoming mail from the network, the `sendmail` daemon (`-bd` mode) listens for connection requests on the well-known SMTP port, and forks SMTP servers to accept these connections.

- The sendmail queue-processing daemon (*-qinterval* flag) awakens at the specified interval and processes the mail queue.

Ordinarily the script `/etc/netbsdsrc` starts the sendmail daemon to do both of these things when the system reboots with the command:

```
/usr/lib/sendmail -bd -q30m
```

If the sendmail configuration file or frozen configuration changes (see the section “Freezing the Configuration File”), you should kill and restart the sendmail daemon so that it re-reads the configuration file. The command

```
/usr/lib/sendmail -bk
```

run by the superuser, kills a sendmail daemon started with `-bd`, whether or not it was started with the *-q interval* flag. It does not kill a sendmail daemon started only with *-q interval*. You must kill this type of daemon by finding its process ID (for example, with `ps (1)` and `grep (1)`) and killing it explicitly.

Note

Do not kill sendmail with `kill -9`. This may cause sendmail to corrupt the alias database. Use `kill -15` instead.

You can run multiple queue processing daemons at once using *-q interval*, but you can only run one sendmail daemon at a time in `-bd` mode.

The `X` configuration option controls the operation of the sendmail daemon, for load limiting purposes. If the five-minute system load average is greater than the value set with the `X` option, the sendmail daemon will not accept connections from the network.

If `X` is not set, the default value is 12. Disable the `X` option by setting it to an absurdly high value, such as 100.

Operating sendmail

Freezing the Configuration

Each time you invoke the sendmail program, it reads and interprets the sendmail configuration file, /usr/lib/sendmail.cf (for “configuration file”). To minimize execution time, you can freeze the configuration.

When freezing the configuration, sendmail reads and interprets the configuration file and then saves an image of its data space as a file, /usr/lib/sendmail.fc (for “frozen configuration”). If this file is present, a future invocation of sendmail simply reads it into memory and continues as if it had read and processed the text version of the configuration file.

To freeze the configuration file, issue one of the following commands as superuser:

```
/usr/lib/sendmail -bz
```

or

```
/etc/freeze
```

For security reasons, only the superuser can freeze the sendmail configuration.

Note that you should re-freeze the configuration and kill and restart the sendmail daemon when any of the following change:

- The sendmail configuration file, /usr/lib/sendmail.cf.
- The UUCP configuration, as reflected in the output of uuname.
- The contents of a file or the output of a program from which a file class is defined.

As superuser, do the following:

```
/etc/freeze           # freeze the configuration
/usr/lib/sendmail -bk # kill the sendmail daemon
/usr/lib/sendmail -bd -q30m # restart the sendmail daemon
```

To use SAM to re-freeze the configuration and kill and restart the `sendmail` daemon, do the following:

1. At the HP-UX prompt, type:

```
sam
```

and wait for SAM's main menu to appear.
2. Select the Networking/Communications menu item.
3. Select Services Enable/Disable.
4. Select the `sendmail` menu item.
5. Select the Restart action.
6. Fill in the form according to its instructions or answer the prompt in the window. View the help screens for information about filling in the form.
7. Exit the Services Enable/Disable screen to a previous level by selecting Exit from the List menu. Then, to exit the Networking/Communications screen, select either Previous Level to exit to a previous level or Exit SAM to exit from SAM.
8. Select apply to enter additional names of systems to be configured (use apply as a shortcut to remain in the add screen). Then, press OK when you are done with the screen.

sendmail on a Cluster

On a cluster, since there is a single file system, `sendmail` is configured to behave as if the cluster was a single host with the cluster server's host name.

This model requires you to run `sendmail` slightly differently on a cluster than you would on a stand-alone system. The differences are these:

- The `sendmail` daemon is run only on the server node.

Operating sendmail

- `sendmail` on the client cnodes is configured so that outgoing mail appears as if it originated on the server. This causes replies to messages from the clients to be directed to the server.

By default, mail from remote systems addressed to users at the client cnodes will fail, since they do not run the `sendmail` daemon. Such mail should be addressed to the user at the server node instead.

On a cluster, the frozen configuration file, `/usr/lib/sendmail.fc`, is a context dependent file (see `cdf`), since Series 300 and 800 frozen configuration files are incompatible.

The configuration is frozen separately on each cnode in the cluster by the command:

```
/etc/freeze
```

which in effect runs:

```
/usr/lib/sendmail -bz -oMw'cnodes -r'
```

on each cnode. This sets `sendmail`'s idea of the local host name to the name of the cluster server.

If `/etc/freeze` is run on a stand-alone system, it simply does:

```
/usr/lib/sendmail -bz
```

The Mail Queue

`sendmail` distinguishes between two kinds of mail delivery failures:

- Permanent failures are mail transactions that are unlikely to succeed without some intervention on the part of the sender or a system administrator. For example, mailing to an unknown user is a permanent failure, and so is a delivery failure of the local mailer because the file

system is full.

- Temporary failures are mail transactions that might succeed if retried later. For example, “connection refused” when attempting to connect to a remote SMTP server is a temporary failure, since it probably means that the server is temporarily not running on the remote host.

sendmail’s handling of permanent failures is described the “Mail Error Handling” section later in this chapter.

When a temporary failure occurs, sendmail saves the message and envelope information in a mail queue. By default, this mail queue is stored in the directory `/usr/spool/mqueue`. sendmail “processes the queue,” which means it attempts to deliver the saved messages again, when invoked with the `-q` flag. If invoked as

```
/usr/lib/sendmail -q
```

it immediately processes the queue once. If invoked as

```
/usr/lib/sendmail -q interval
```

it starts a queue-processing daemon that awakens regularly at the specified interval to process the queue. This interval is usually set between five minutes to one hour. Normally the queue-processing daemon is started with the same command that starts the SMTP server daemon. (See the previous section “The sendmail Daemon.”)

Note The command `/usr/lib/sendmail -bk` kills a sendmail daemon started with both `-bd` and `-qinterval`. It does not kill a sendmail daemon started only with `-qinterval`.

When processing the queue, sendmail first creates and sorts a list of the messages in the queue. It then attempts to deliver each queued message in order. Since it is possible for multiple queue-processing daemons to run at once, sendmail locks

Operating sendmail

each message before starting the delivery, instead of locking all the messages at once. If a queue-processing daemon discovers a locked message, it simply goes on to the next message in its list. It logs the fact in the system log file, if the logging level is 6 or higher.

If sendmail detects, from the time stamp in a queued message, that the message has been in the mail queue longer than the queue timeout, it returns the message to the sender. The queue timeout is set with the T configuration option and, by default, is three days.

Queue Priorities

A message is given an initial priority value based on its size, its precedence, and the number of recipients. The priority value and the creation time of the message are used to order the queue. *A high priority value means that the message is of low priority.*

The initial priority of a message is calculated as

$$\text{priority} = \text{size} - (\text{precedence} * z) + (\text{recipients} * y)$$

In computing the priority, small messages are preferred over large messages. Users can influence the priority of their messages with the "Precedence:" header field. Since the number of recipients affects the load a message presents to the system, messages with large numbers of recipients are given a higher priority value (lower priority).

The weight given to the number of recipients in the priority calculation is controlled with the y option in the configuration file. The weight given to the message precedence in the priority calculation is controlled with the z option. The default values for the y and z options are 1000 and 1800, respectively.

The priority of a queued message is adjusted each time it is processed, on the grounds that messages which have failed many times will tend to fail again in the future. The value set with the Z option in the configuration file is added to the

priority value of a message each time an attempt is made to deliver it. The default value is 9000.

You can look at the priority of queued messages by running `sendmail` or `mailq` in verbose mode (`-v`). Refer to the “Printing the Mail Queue” section.

Load Limiting

The `x` option defines a load average limit so that `sendmail` queues, rather than delivers, low priority messages if the system load average is too high. When the five-minute load average exceeds the value of the `x` option, messages are queued if the queue factor (set with the `q` option) divided by the difference (plus one) between the current load average and the load average limit exceeds the priority of the message.

In other words, if the load average is greater than the load average limit (`x`), then deliver the message only if the priority value is less than:

$$\text{queue factor} / (\text{load average} - \text{load average limit} + 1)$$

This means that when the load average reaches the load average limit, the `q` option defines the maximum priority value (minimum priority) of messages that will be delivered. The default value of the `q` option is 10000. The default value of the `x` option is 8.

Mail Queue File Format

The files used by `sendmail` to store and synchronize operations on the mail queue all have names of the form `zzAAnnnnn`, where `zz` is the type of the queue file (see Table 6-3) and `AA` is an identifier used to distinguish separate queue entries that happen to have the same process ID. `sendmail` starts with `AA` and loops through `AA`, `AB`, `AC`, etc., until it is able to form a unique ID. The five-digit number (`nnnnn`) is the process ID of the process creating the queue entry.

Operating sendmail

Type	Defines
df	The data file. The message body, excluding the header, is kept in this file.
qf	The queue-control file which contains the information necessary to process the job. Queue-control file formats are explained in detail below.
tf	A temporary file that is an image of the queue-control file while it is being rebuilt. You can expect this to be renamed to a queue-control file very quickly.
xf	The transcript file. This file is normally empty while a piece of mail is in the queue. If a failure occurs, a transcript of the failed mail transaction is generated in this file.
af	A temporary file used when rebuilding the alias database (<code>sendmail -bi</code>). These files are removed when the initialization of the alias database is complete.

The queue-control file (type qf) is structured as a series of lines each beginning with a letter that defines the content of the line. Definition letters used in queue-control files are described in Table 6-4.

Table 6-4. Definition Letters for Lines in the Queue-Control File

Letter	Meaning
C	The controlling user for message delivery. This line always precedes a recipient line (R) that specifies the name of a file or program name. This line contains the user name which sendmail should run as when it is delivering a message into a file or a program's <i>stdin</i> .
D	The name of the data file. There can be only one D line in the queue-control file.
E	An error address. If any such lines exist, they represent the addresses that should receive error messages.
H	A header definition. There can be many H lines in the queue-control file. The order of the headers in the final message is the same as the order of the H lines. Header definitions follow the header definition syntax in the configuration file.
P	The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority decreases (that is, the number grows) as the message sits in the queue. The initial priority depends on the message precedence, the number of recipients, and the size of the message.
M	A message. This line is printed by the <code>mailq</code> command, and is generally used to store status information (i.e., the reason the message was deferred). It can contain any text.
R	A recipient address. Normally this has already been completely aliased, but is actually re-aliased when the queue is processed. There is one line for each recipient.
S	The sender address. There can be only one sender address line.
T	The job creation time (in seconds since January, 1970). This is used to determine when to time out the job.

Operating sendmail

The following example is a queue-control file named qfAA00186. The sender is david and the recipient is the local user carolyn. The current priority of the message is 17, the time in seconds since January, 1970 is 515 961 566. The last seven lines describe the header lines that appear on the message.

```
P17
T515961566
DdfAA00186
Sdavid
Rcarolyn
Hreceived: by lab; Thu, 8 May 86 12:39:26 mdt
Hdate: Thu, 8 May 86 12:39:26 mdt
Hfrom: David <david>
Hfull-name: David
Hreturn-path: <david>
Hmessage-id: <8605081839.AA00186@lab.HP>
Happarently-to: carolyn
```

Printing the Mail Queue

The current contents of the mail queue can be printed with either:

```
/usr/lib/sendmail -bp
```

or

```
mailq
```

The output looks similar to this example:

```

                                Mail Queue (3 requests)
--QID--  ---Size--  ---Q-Time---  -----Sender/Recipient-----
AA15841      86  Wed Feb 9 07:08      janet
                                (Deferred: Connection refused by med.hub.com)
                                ees@vetmed.umd.edu
                                ebs@surv.ob.com
AA15794     1482  Wed Feb 9 07:57      carole
                                bja@edp.cloq.potlatch.com
                                vls@ee.cmu.edu
AA15792     10169  Wed Feb 9 07:57      chuck
                                hrm@per.stmarys.com
```

Operating sendmail

sys6!sysloc!njm
v1s@ce.umd.edu

The first entry is a message with queue ID AA15841 and a size of 86 bytes. The message arrived in the queue on Wednesday, February 9 at 7:08 a.m. The sender was janet. She sent a message to the recipients ees@vetmed.umd.edu and ebs@surv.ob.com. sendmail has already attempted to route the message, but the message remains in the queue because its SMTP connection was refused. This usually means that the SMTP server is temporarily not running on the remote host, but it also occurs if the remote host never runs an SMTP server. sendmail attempts to deliver this message the next time the mail queue is processed. There are also two other messages in the queue that are routed for delivery the next time the mail queue is processed.

If sendmail or mailq is run in verbose mode (with the -v option), then when it prints the queue, it will also show the priority of each queued message.

The System Log

sendmail logs its mail messages through the syslogd logging facility.

Note In HP-UX releases before 8.0, sendmail writes messages directly to /usr/spool/mqueue/syslog. In the 8.0 HP-UX release, syslogd must be configured and running so that sendmail logging is recorded in /usr/spool/mqueue/syslog..

The syslogd configuration on diskless clients should forward all logging that results from sending mail to the cluster root server. On a system whose root server is named hp-root.hp.com, you can do this by adding the following line in /etc/syslog.conf:

```
mail.debug @hp-root.hp.com
```

Operating sendmail

The `syslogd` configuration on cluster servers and standalone systems should write mail logging to the file `/usr/spool/mqueue/syslog`. On a system whose root server is named `hp-root.hp.com`, you can do this by adding the following line in `/etc/syslog.conf`:

```
mail.debug /usr/spool/mqueue/syslog
```

For more information about configuring `syslogd` for ARPA services, see the “Administering ARPA Services” chapter earlier in the manual.

Setting Log Levels

You can set the log level with the `-oL` option on the command line or on the `OL` line in the configuration file. At the lowest level no logging is done. At the highest level, even the most mundane events are recorded. As a convention, log levels eleven and lower are considered useful. Log levels above eleven are normally used only for debugging purposes. It is recommended that you configure `syslogd` to log mail messages with a priority level of debug and higher. `sendmail`'s behavior at each log level is described in Table 6-5.

Level	Type of Information Logged
0	No logging.
1	Major problems only.
2	Message collections and failed deliveries.
3	Successful deliveries.
4	Messages being deferred (due to a host being down, etc.).
5	Messages being added to the queue in routine circumstances.
6	Unusual but benign incidents, such as trying to process a locked queue file.
9	Log internal queue ID to external message ID mappings. This can be useful for tracing a message as it travels between several hosts.

Table 6-5. Log Levels	
Level	Type of Information Logged
10	The name of the mailer used, the host (if non-local), and the user name passed to the mailer are logged. If the log level is 10 or higher, sendmail also reports this information in -bv (verify) mode.
11	For successful deliveries to IPC mailers, the MX (mail exchanger) host delivered to (if any) and the internet address used for the connection are logged.
12	Several messages that are basically only of interest when debugging.
16	Verbose information regarding the queue.

At typical logging levels, every piece of mail passing through sendmail adds two or three lines to the mail log. A script to manage the growth of the mail log could be run nightly, at midnight, with an entry in root's crontab file such as the following:

```
0 0 * * * /usr/adm/newsyslog
```

The following example shows what the script /usr/adm/newsyslog might contain. Note that the script assumes that syslog is configured to direct mail logging to /usr/spool/mqueue/syslog.

```
#!/bin/sh
#
# NEWSYSLOG: save only the last week's worth of sendmail
# logging

cd /usr/spool/mqueue
mv syslog.6 syslog.7
mv syslog.5 syslog.6
mv syslog.4 syslog.5
mv syslog.3 syslog.4
mv syslog.2 syslog.3
mvn syslog.1 syslog.2
cp syslog syslog.1
kill -1 'cat /etc/syslog.pid'
```


Operating sendmail

Aliasing and the Alias Database

sendmail maintains a hashed alias database based on a text file, by default `/usr/lib/aliases`. This database is described in the section “The Alias Database.”

When routing mail, sendmail looks up each local address (unless preceded by a backslash, ‘\’) in this alias database and, if found, expands it by replacing it with the specified list of addresses. This is done recursively; in other words, if a local name resolves to a list, each name in the list is looked up and expanded.

Alias expansion is not done if the alias database files do not exist or if sendmail is run with the `-n` flag.

Normally the sender is not included in any alias expansions. For example, if “joe” sends to “group,” and the expansion of “group” includes “joe,” then the letter is not delivered to “joe.” You can override this with the ‘m’ (MeToo) configuration option.

The alias text file can contain comments, alias definitions, and continuation lines. Blank lines and lines beginning with ‘#’ are comments. Each alias definition must be of the form:

```
alias : mailing_list
```

Aliases must be local names. The alias definition:

```
user@somewhere.else : user@here, anybody@anywhere
```

is invalid because `user@somewhere.else` is not a local address.

Alias definitions may be continued onto multiple lines; lines beginning with a blank or a tab are continuation lines.

A mailing list is a comma-separated list of one or more of the following:

- user_name** Local user names occurring in alias expansions are looked up in the alias database unless they are preceded by a backslash(\).
- remote_address** The remote address syntax understood by `sendmail` is configured in the `sendmail` configuration file, and normally includes RFC 822 style *user@domain* and UUCP style *host/user*.
- filename** This must be an absolute pathname. `sendmail` appends a message to the file only if the directory in which it resides is readable and searchable by all, and only if the file already exists, is not executable, and is writable by all.
- "| command line"** `sendmail` pipes the message as standard input to the specified command. Note that the double quotes(") are necessary to protect blanks in the command line.
- :include: filename** `sendmail` reads *filename* for a list of recipient addresses and forwards the message to each.

Note `sendmail` does not mail directly to file names, command lines, or `:include:` specifications. It sends to these only if they occur on the right-hand side of an alias definition in the alias database, in an `:include:` specification, or in a `.forward` file. These forms are described in more detail in the sections that follow.

Operating sendmail

If an error occurs on sending to a certain address, say `x`, `sendmail` searches for an alias of the form `owner-x` to receive the errors. This is useful for a mailing list where a user mailing to the list has no control over the maintenance of the list itself; in this case the list maintainer is the owner of the list. For example, if `dan@node1` owned the following mailing list:

```
unix-wizards: dan@node1, david@node2, nosuchuser,  
             mickie@node3  
owner-unix-wizards: dan@node1
```

`dan@node1` receives the error message when anyone attempts to send to `unix-wizards` because of the inclusion of `nosuchuser` in the list.

Filename Aliases

If the destination of an alias is a file, the file must already exist, must not be executable and must be writable by all users. The message is appended to the file.

The following example specifies an alias that mails to a file.

```
# Examples of aliases for files:  
public : /tmp/publicfile  
terminal : /dev/tty
```

If `public` appears as a recipient address, the message is appended to the file `/tmp/publicfile`. Mail to `terminal` appears on the sender's terminal.

Command Line Aliases

You can also alias to a command line. You might use this, for example, to implement a news facility, collect statistics from remote systems, or implement a junk mail filter. The following is an example of aliasing to a command line:

```
# Example of an alias for a command line:  
prog : "| /bin/cat | /bin/sed 's/Z/z/g' > /tmp/outputfile"
```

Mail addressed to `prog` is saved in `/tmp/outputfile`, with all capital `Z`'s changed to lower-case `z`'s.

Operating sendmail

The first `|` character informs `sendmail` that the recipient address is a command line. Note that the double-quotes are required to protect the command line from interpretation by `sendmail`.

Files should be referred to using absolute pathnames, since the command line is executed with `sendmail`'s queue directory as the working directory.

`sendmail` disconnects `stdout` and `stderr` from the terminal, so if the `stdout` or `stderr` of a command is not redirected, it disappears. However, if the exit status of the command is non-zero, anything it outputs to `stderr` becomes part of the error transcript that is returned by `sendmail`.

The command is executed by the prog mailer defined in the configuration file. In the supplied configuration file, the prog mailer is `"sh -c"`. The command is executed as either the user executing `sendmail`, or the user and group set with the `u` and `g` configuration options. The path is `/bin:/usr/bin`, so it may be necessary to refer to commands by absolute pathnames.

:include: Specifications

The alias database may either be writable by all, or writable by the superuser only. A publicly writable alias database permits users to maintain mailing lists and other public aliases in a central location, making distribution of mailing list updates unnecessary. However, this is a security risk; users can add themselves to any mailing list or take another user's mail. If the alias database can be updated only by the superuser, security can be more easily controlled, but maintaining the database becomes the responsibility of the system administrator.

Users can also administer their own mailing lists by using the inclusion facility. An alias of the form,

```
alias: :include:/file_pathname
```

causes mail addressed to *alias* to be sent to all the addresses in the file `/file_pathname`, that must be an absolute path, (that is, begin with a `"/"`). The

Operating sendmail

syntax for the `:include:` file is exactly the same as for the alias file. (See the beginning of this section, "Aliasing and the Aliasing Database.")

User-Controlled Aliasing: `.forward`

A user may redirect his or her own mail with a `.forward` file in the user's home directory. If such a file exists and is owned by the user, `sendmail` redirects mail for that user to the list of addresses in the `.forward` file. The syntax of this file is exactly the same as that for the alias file (see previous section).

For example, if user Ernie has a `.forward` file in his home directory on his workstation, `ernie`, containing the following:

```
ernie@bert, \ernie
bert@bert
```

then mail for Ernie is forwarded to users `ernie` and `bert` on host `bert`, and is also deposited in Ernie's local mailbox on `ernie`. If he had:

```
ernie@bert, ernie@ernie
bert@bert
```

the result would be the same.

Note

The alias database is examined before a user's `.forward` file. This means that the `.forward` file is only examined if the user's name does not appear as an alias, or if an alias expands to the user's name.

Aliasing Loops

You should avoid creating aliasing loops. This can occur either locally or remotely. For example:

```
# Example of a local aliasing loop:  
first : second  
second : first
```

When rebuilding the alias database, *newaliases* does not notice a loop like the one shown in the previous example. However, after the alias database is rebuilt, mail addressed to either `first` or `second` is not sent. If the only recipients for the message are in local alias loops, the message is returned with the error message “All recipients suppressed.”

For example, if mail is addressed to `first`, `first` expands to `second`, which expands to `first`. This causes `sendmail` to remove `first` from the recipient list as a duplicate.

```
# Example of a remote aliasing loop:  
dave : dave@remote
```

If `dave@remote` is aliased on the remote system to `dave` on the local system, by means of a `sendmail` alias database, `.forward` file, etc., mail sent to `dave` at either host bounces between the two systems. `sendmail` adds a tracing header line (Received:) with each hop. When 30 tracing header lines have been added, `sendmail` recognizes the aliasing loop and aborts the delivery with an error message.

The Alias Database

To improve performance, `sendmail` looks up aliases in a hashed database created from the alias file. Whenever the alias file is changed, the alias database must be rebuilt with the command:

```
newaliases  
or
```

Operating sendmail

```
/usr/lib/sendmail -bi
```

By default, this creates the database files, `/usr/lib/aliases.dir` and `/usr/lib/aliases.pag`, from the alias file `/usr/lib/aliases`. (For more information, see `ndbm` in section 3 of the *HP-UX Reference*.)

If the `D` option is set in the configuration file or set with `-oD` on the command line, and if the alias database files are writable by all, `sendmail` rebuilds the alias database automatically if it is out of date.

If the alias database files do not exist, or if `sendmail` is run with the `-n` flag, no aliasing is done.

If the `n` option (do not confuse with the `-n` flag) is set when rebuilding the alias database, `sendmail` tries to resolve all the addresses on the right-hand sides of alias definitions. It reports errors if it is unable to resolve any of them. This way errors in the alias file are discovered when the database is being built, rather than when someone tries to mail to one of the aliases. However, a large alias database may take a long time to rebuild.

By default, the `n` option is disabled. In this case, `sendmail` stores the right-hand sides of aliases without examining them for correctness. The disadvantage is that an error is discovered only when someone tries to mail to the alias with the error. The sender may not be in a position to fix the error. The advantage is that, if only a small part of the alias file has changed since it was last rebuilt with the `n` option set, it is much faster to rebuild the alias database with the `n` option disabled. Verify the new aliases manually with the following:

```
/usr/lib/sendmail -bv -v alias1, alias2, alias3
```

You can use the `A` option to make `sendmail` build or use an alias database based on an alias file other than the default.

`sendmail` avoids the possibility of more than one `sendmail` process rebuilding the alias database at the same time by locking the alias file `/usr/lib/aliases` during

the rebuild process. While `/usr/lib/aliases` is locked, no other sendmail processes will rebuild the alias database.

Problems may occur if a sendmail process accesses an alias database that is only partially rebuilt. Partial rebuilding occurs when the process that is rebuilding the database dies. This can happen if the rebuilding process is killed, or the system reboots before completing the rebuild. To avoid these problems, sendmail uses two techniques:

- sendmail ignores interrupts while rebuilding the database. This technique allows you to avoid terminating the process and leaving a partially rebuilt database.
- At the end of a rebuilt alias database, sendmail adds an alias of the form

`@: @`

Before sendmail accesses the database, it checks to ensure that this entry exists. If the `a` option is set in the configuration file, sendmail waits for the `@: @` entry to appear. If the `@: @` entry does not appear within the number of minutes specified with the `a` option, sendmail either forces an alias database rebuild (if the `D` option is set), or issues a warning and continues.

Operating sendmail

MX Records

The BIND nameserver, if it is in use on your host, provides MX (Mail Exchanger) records. These can be used to inform `sendmail` that mail for a particular host can be relayed by another host, if the addressed host is temporarily down or otherwise inaccessible. Configuring the name server is discussed in more detail in the BIND chapter.

How sendmail Uses MX Records

MX records are only used if a message address resolves to an [IPC] mailer; that is, one that uses SMTP over sockets to perform delivery. Instead of attempting to connect directly to the recipient host, `sendmail` first queries the nameserver, if it is running, for MX records for that host. If the nameserver returns any, `sendmail` sorts them in preference order, highest preference (lowest number) first. If the local host appears in the list, it and any MX hosts with lower preference (higher numbers) are removed from the list. If any MX hosts remain, `sendmail` then tries to connect to each MX host in the list in order, and it delivers the message to the first MX host to which it succeeds in connecting. If that MX host is not the final destination for the message, it is expected that the host will relay the message to its final destination.

Otherwise, if the nameserver returns no MX records, or if the nameserver is not running, or if the local host is the highest preference (lowest number) mail exchanger in the list, `sendmail` will simply try to connect to the host to which the message is addressed.

At log level 11 and above, `sendmail` logs in the system log the name and internet address of the MX host (if any) to which it delivered (or attempted to deliver) a message.

How to Use MX Records for Mail Relaying

MX records are used for two main purposes:

- To arrange that one host “back up” another by receiving mail for it when it is down.

For example, the nameserver serving the domain `paf.edu` might have the following MX records for the host `bling`:

[name]	[ttl]	[class]	MX	preference	mail exchanger
bling		IN	MX	10	bling.paf.edu.
		IN	MX	20	wheo.paf.edu.
		IN	MX	30	munch.paf.edu.

Ordinarily mail for `bling` will go directly to `bling`. However, if `bling` is down, or if the sending host cannot connect to `bling`, `sendmail` will route mail for it to `wheo`; if `wheo` is also down or unreachable, `sendmail` will route the mail to `munch`. Naturally, for this to be useful, `wheo` and `munch` must be able to route mail to `bling`.

Assuming that the host and its mail exchangers see the same MX data from the nameserver, each host that has MX records should have an MX record for itself, and the preference on its own record should be the highest (i.e. the lowest number) in the list.

- To arrange that mail addressed to remote networks be relayed through the appropriate gateways.

For example:

[name]	[ttl]	[class]	MX	preference	mail exchanger
*.nz.		IN	MX	0	gw.dcc.nz.

Messages addressed to hosts in the `nz` domain will be relayed to the host `gw.dcc.nz`. Courtesy suggests that you seek permission from the administrators of hosts not under your own control before relaying mail through them.

Operating sendmail

MX Failures

There are several possible failures associated with MX configuration:

- The nameserver query for MX records fails.

If this fails because there are no MX records for the target host or because the nameserver is not in use, `sendmail` will attempt to connect directly to the target host. However, if the `I` configuration option is set and if the nameserver is not running, `sendmail` will defer the message.

If this fails temporarily (i.e. `h_errno` is set to `TRY_AGAIN`) the message will be deferred. The possible values of `h_errno` are documented in the header file `/usr/include/netdb.h`.

- Connection attempts to the hosts in the MX list all fail.

`sendmail` reports the failure attempting to connect to the last MX host (i.e. the highest preference value) in the list that it tried. For example, with mail exchangers configured as in the `paf.edu` example above, if the attempts to connect to `bling` and `wheo` result in temporary failures, but the attempt to connect to `munch` fails permanently, the message will be returned as an error. If the attempts to connect to `bling` and `wheo` result in permanent failures, but the attempt to connect to `munch` fails temporarily, the message will be deferred.

- A host cannot deliver a message to another host for which it is a mail exchanger.

This failure is handled as a normal delivery failure, either by the mail exchanger host or by the host sending to the mail exchanger.

Mail Error Handling

Mail deliveries can fail for a variety of reasons. `sendmail` distinguishes between “temporary failures,” where the delivery might succeed if retried later, and “permanent failures,” where the delivery will not succeed unless something is modified, either in the message envelope or in the mail configuration.

Messages that fail temporarily are saved in the mail queue and retried later. See the “Mail Queue” section of “Operating sendmail” in this chapter.

Permanent failures include the following:

- Temporary failures that have remained in the mail queue for the queue time out period (set with the `T` configuration option).
- Local recipient user unknown.
- The recipient address cannot be resolved by `ruleset 0` (see the “Address Rewriting” section of “The sendmail Configuration File” later in this chapter).
- Permanent delivery agent (mailer) failures.
- Inability to find an internet address for a remote host.
- A remote SMTP server reports during the SMTP transaction that an address is undeliverable.

If permanent delivery failures are detected locally when the delivery is first attempted, errors are reported as specified by the `e` configuration option. See the “Option Specifications” section of “The sendmail Configuration File” later in this chapter.

If the failure is detected on a host other than the one where the message originated, or on a retry from the mail queue, or the `e` option is set to `m` (or `w` and the sender is not logged in), `sendmail` will attempt to return the message, along with an error transcript, by mail. The error transcript will contain a message

Operating sendmail

provided by `sendmail` explaining why delivery failed, as well as any error output from the delivery agent or SMTP transaction.

If delivery failed on an alias, if there is an owner for that alias, `sendmail` will return the message and transcript to the alias owner. See the “Aliasing” section of “Operating sendmail” in this chapter.

If there is an `Errors-To:` header line in the message header, `sendmail` will return the message and transcript to the address on the `Errors-To:` line instead of to the sender.

Otherwise `sendmail` will return the message and transcript to the sender of the message. If the `PostMasterCopy` option (option `P`) is set to a valid address, then a copy of the transcript and failed message (with the message body deleted) will also be sent to the `PostMasterCopy` address.

If the attempt to return the failed message itself fails, `sendmail` will return the message and transcript to the address “`postmaster`” on the local system. RFC 822 requires that `postmaster` be a valid address on every system supporting electronic mail. HP suggests that `postmaster` be an alias for the mail administrator for the system, and that the `P` option, if used, refer to `postmaster`.

If `sendmail` is unable to return the message to any of the addresses described above, as a last resort it will append the error transcript and return message to the file `/usr/tmp/dead.letter`.

Finally, if this fails, `sendmail` logs the failure and leaves the original failed message in the mail queue so that a future queue processing daemon will try to send it, fail, and try again to return an error message.

The sendmail Configuration File

Operation of the sendmail program is controlled by its configuration file, `/usr/lib/sendmail.cf`. This file does the following:

- Defines certain names and formats, such as the name of the sender for error messages (MAILER-DAEMON), the banner displayed by the SMTP server on startup, and the default header field formats.
- Sets values of operational parameters, such as timeout values and logging level.
- Specifies how mail will be routed. In other words, it specifies how recipient addresses are to be interpreted.
- Defines the delivery agents (mailers) to be used for delivering the mail.
- Specifies how sendmail should rewrite addresses in the header, if necessary, so that the message address can be understood by the receiving host. The address rewriting process is controlled by sets of address rewriting rules called **rulesets**.

The configuration file is organized as a series of several distinct types of lines. The first character of each line defines the type of the line. Lines beginning with a space or a tab are continuation lines. Blank lines and lines beginning with a sharp sign (**#**) are comments.

Table 6-6 is a list of the syntax definition characters.

The sendmail Configuration File

Table 6-6. Syntax Definition Characters

Letter(s)	Define(s)
C or F	Classes
D	Macros
H	Header line formats
M	Mailers (delivery agents)
O	Options
P	Precedence classes for messages
T	Trusted users
S	Start of rulesets
R	Rewriting rules

The following sections define the syntax and semantics of the various lines.

C or F Lines – Class Definitions

A class identifies a set of elements. Classes are used in the address rewriting process. Specifically, a rewriting rule may be selected if a given token is (or is not) a member of a class. Classes can be defined in three ways:

- Directly in the configuration file.
- Read in from another file.
- Read from the output of a program.

Class names should only be upper-case letters. Lower-case letters and characters are reserved for system use.

Literal Class Definition

The syntax for defining a class directly in the configuration file is:

```
Cdword1 word2...
```

The name *d* is assigned to the class consisting of the specified *words*. *Words* can be split among multiple lines. For example, both cases below define class *H* to contain *mickie* and *doug*.

```
CHmickie doug
and
CHmickie
CHdoug
```

The effect of multiple definitions of a class is to add the new members to it. Members cannot be deleted from a class.

File Class Definition

The syntax for defining a class from a file is:

```
Fdfile [format]
```

where *d* is the single letter name of the class, *file* is the name of the file containing the members of the class, and *format* is a *scanf* format that should produce a single string from each line in the file. If the format is omitted, "%s" is used by default.

For example, if *sendmail*'s configuration file contains the lines:

```
#file class "S" defines known SMTP connections
FS/etc/hosts.SMTP %*[0-9.] %s
```

The line beginning with a sharp sign (#) is a comment line. The second line defines *S* as the class of all names contained in the */etc/hosts.SMTP* file. The format that

The sendmail Configuration File

is applied (`%*[0-9.] %s`) skips the first word in each line and reads the second word as the class member.

If the file `/etc/hosts.SMTP` contains

```
192.0.2.93    node1
192.0.2.101  node2
192.0.2.112  node3
```

the numerical address in each line (read as a single string) is skipped and the host name is defined as a member of class S.

File Class Definition from Program Output

The syntax for defining a class from the output of a program is:

```
Fd|program [format ]
```

where *d* is the name of the class, *program* is the absolute pathname of an executable program that returns the desired output, and *format* is a `scanf` format as previously described. The program name must be a single word with no embedded spaces because each word after the first is interpreted as part of the format. If you wish to define a class from a program with options or arguments you must enclose it in an executable shell script, and define the class from the shell script.

Program is run when the configuration file is read or frozen. If the output of *program* changes, the configuration should be refrozen.

For example, a cluster's client nodes do not ordinarily run the `sendmail` daemon, since they share a file system with the server node. If you wanted `sendmail` on the server to recognize the client nodes' host names as local, you could add their canonical host names to the class `w`, which is defined internally from the aliases for the local host name.

Create a shell script `/usr/lib/clients`:

The sendmail Configuration File

```
#!/bin/sh
# /usr/lib/clients : canonical names of the cluster's client
# nodes

for c1 in 'cnodes -A' : do
    nslookup $c1
done | awk '/^Name:/ { print $2 }'
```

and define class w:

```
Fw|/usr/lib/clients %s
```

D Lines – Macro Definitions

Macros can be thought of as variables whose values are defined for a particular execution of `sendmail`; the value of a macro will be substituted wherever the macro is used following the macro definition in the configuration file.

Macros have single character names. User-defined macros should be selected from the set of upper-case letters only. `sendmail` uses lower-case letters and special symbols internally.

The syntax for macro definitions is:

```
Dxval
```

where `x` is the name of the macro and `val` is the value that will be substituted for `x`.

Macros and classes can use the same letters for their names. For example, both a macro `B` and a class `B` can exist without conflict.

Macro Interpolation

Macros are interpolated using the construct:

```
$x
```

The sendmail Configuration File

where *x* is the name of the macro to be interpolated. Note that the current value of a macro is interpolated when the configuration file is read. If a macro has not been defined, it has no value. Conditionals are specified using the syntax:

```
$ ?x text1 $| text2 $.
```

This interpolates *text1* if the macro *\$x* is set, and *text2* otherwise. The *\$.* terminates the conditional clause. The else clause (*\$|*) can be omitted.

Required Macro Definitions

Each of the following macros must be defined in a **D** line of the configuration file because each time that `sendmail` is invoked, these macros are used as constants. The configuration file provided with this product defines these macros as described in Table 6-7.

Macro ID	Interpretation
e	SMTP banner. The \$e macro is the first output when the SMTP server is invoked. The first word of macro e must be the \$j macro, which should be your host's canonical name.
j	Canonical or official name of this host, as returned by <code>gethostbyname</code> .
l	The format of the UNIX <code>From_</code> line.
n	The name of the sender (MAILER-DAEMON) for error messages.
o	The set of operators or delimiters in addresses. The \$o macro consists of a list of characters that are considered tokens and which separate tokens when parsing occurs. For example, if r is in the \$o macro, then the input <code>address</code> is scanned as three tokens: <code>add</code> , <code>r</code> , and <code>ess</code> .
q	The default format of the sender address. This is output in the From: header field and by the VERFY command in SMTP mode.

Macros Defined at Run Time

Some macros are defined by sendmail each time it runs for use internally by the program and in the rewriting rules. These macros are described in Table 6-8.

Table 6-8. Macros Defined During the Execution of sendmail	
Macro ID	Interpretation
a	The origination date in ARPANET format. \$a is the time as extracted from the Date: line of the message. If no Date: line exists in the incoming message, \$a is set to the current date.
b	The current date and time in ARPANET format. This is used for time stamps.
c	The hop count, or the number of times this message has been processed. This can be set with the -h flag on the command line or by counting the time stamps in the message.
d	The equivalent of the \$a macro in UNIX (ctime) format.
f	The address of the sender. This is used to set the From: address, if sendmail needs to add one.
g	The sender address, relative to the recipient, when mailing to a specific host.
h	The recipient host. This is set from the \$@ part of ruleset 0.
i	The queue ID (zzAAnnnnn) of the message on this host. If \$i is put into the Message-Id: or Received: header lines, it can be extremely useful for tracking messages.
p	sendmail's process ID (PID). This macro is used to create unique strings (e.g., for the Message-Id: field).
r	The protocol used to receive the message.
s	The sender host name.
t	A numeric representation of the current Greenwich Mean time which is used to create unique strings. Often this is used with the process ID (PID) to create unique message IDs.

The sendmail Configuration File

Macro ID	Interpretation
u	The recipient user. When an address is resolved in ruleset 0, this is set from the \$: part of the right side of the rule that resolved the address.
v	The version number of sendmail. This normally appears in the SMTP banner.
w	<p>The canonical name of the local host used to recognize the local host name in the configuration file. By default, sendmail calls gethostbyname with the host name returned by gethostname, and defines the macro w as the canonical host name returned by gethostbyname().</p> <p>The macro values are part of the frozen configuration. If a frozen configuration file is present, the value of macro w in the frozen configuration is used, and gethostname() is not called.</p> <p>Both of these can be overridden with the M option. For example, with /usr/lib/sendmail -oMwsome_other_name, sendmail will call gethostbyname() with the name defined by -oMw.</p> <p>The name defined as macro w and any host name aliases returned by gethostbyname() are also automatically added to the class w. Note that if the nameserver is in use, gethostbyname() does not return any host name aliases.</p>
x	The full name of the sender. The sender's full name can be determined in four ways: It can be passed with the -F command line flag to sendmail; it can be the value of the Full-name: line in the header if it exists; or it can be the comment field of a From: line. If none of these are present, and if the message originates locally, sendmail searches the passwd database for the sender's full name.

Macro ID	Interpretation
y	The ID of the sender's terminal (tty). Note that some systems store this information in the network-style From: line.
z	The home directory of the recipient, if local.

Example of Macro Definitions

The example below shows a section of a configuration file that defines macros.

```
De$j Sendmail $v ready at $b
DnMAILER-DAEMON
DlFrom $g $d
Do.:%@!^=/
Dq$?x $x <$g>$| $g $.
```

The SMTP banner (e) is defined as the local domain name (\$j) followed by Sendmail, Sendmail's version number (\$v) and the current network-style date and time (\$b).

The sender of error messages (n) is identified as MAILER-DAEMON.

The format of the UNIX From_ line is defined to contain the word From followed by the sender's address (\$g) and the current UNIX time in ctime format (\$d).

The set of operators is defined as period (.), colon (:), percent (%), at (@), exclamation point (!), circumflex (^), equals (=) and slash (/).

The macro q is defined as the full name of the sender (\$x), if it is defined, followed by the sender's address in angle brackets (<\$g>); otherwise (\$ |) it is the sender address (\$g).

The sendmail Configuration File

H Lines – Header Definitions

H lines define the format for certain header lines required by RFC 822 (default header lines) or by particular delivery agents (conditional header lines). These formats are used by `sendmail` to generate header lines, if they did not appear in the input message.

The H line syntax for default header lines is:

Hfield: template

The H line syntax for conditional header lines is:

H?mflags?field: template

Field is the name of the header field (for example, **Date** or **From**). The field name is recognized in any mixture of capital and lower-case letters, but is always output with an initial capital letter.

Template is a template for constructing the line, if necessary. The template may contain absolute text, macros, conditionals, and white space. RFC 822 defines the names and required format of certain header fields and, to a limited extent, permits mail handling programs to define additional header fields.

In conditional headers, *mflags* is a set of mailer flags. If the *Flags* field of a mailer definition (see the “Mailer Definitions” section) contains any of the *mflags* associated with a particular header, `sendmail` will generate that header, if necessary, in messages routed via that mailer.

Default Processing of Header Lines

sendmail's handling of message headers is controlled by the configuration file and by header flags compiled into the program. In addition, certain header lines are handled individually by special routines. These are described in the "Special Header Lines" section.

A header line in the input message is preserved in the output message unless H_ACHECK is set for that header line. H_ACHECK is described Table 6-10.

A header line that is defined unconditionally in the configuration file (default header line) is generated in the output message, unless that header line is in the input message already, or H_ACHECK is set for that header line.

A header line that is defined conditionally in the configuration file is generated if the mailer for that particular instance of the message requires it, providing the header line is not already in the input message. For example, if the following is defined in the configuration file,

```
H?E?Example-Header:   for example
.
.
.
Mlocal, P=/bin/rmail, F=lsDFPmE, S=10, R=20 A=rmail -d $u
```

and the particular message is to be delivered by the local mailer, the message will have an Example-Header: line added, unless the message already had one.

The sendmail Configuration File

Special Processing of Standard Header Lines

Table 6-9 shows standard header lines and the header flags associated with each header line.

Header Line	Header Flags
Resent-From:	H_FROM, H_RESENT
Sender:	H_FROM
From:	H_FROM, H_ACHECK
Return-Receipt-To:	H_FROM
Errors-To:	H_FROM
Return-Path:	H_FROM, H_ACHECK
To:	H_RCPT
Resent-To:	H_RCPT, H_RESENT
Cc:	H_RCPT
Resent-Cc:	H_RCPT, H_RESENT
Bcc:	H_RCPT, H_ACHECK
Resent-Bcc:	H_RCPT, H_ACHECK, H_RESENT
Apparently-To:	H_RCPT
Resent-Date:	H_RESENT
Resent-Message-Id:	H_RESENT
Message:	H_EOH
Text:	H_EOH
Received:	H_TRACE, H_FORCE
Via:	H_TRACE, H_FORCE
Mail-From:	H_TRACE, H_FORCE

Table 6-10 describes the meaning of each header flag and the special processing done to header lines that have that flag.

Table 6-10. Header Flags	
Header Flag	How Sendmail Handles the Header Line
H_FROM	If H_FROM flag is set in a header line, the header line is treated as an originator header line. In -ba mode, in order to have a return address for errors, sendmail identifies the sender from the first of these lines that it finds in the message header (sought in the order shown in Table 6-9). Sender address rewriting is performed on addresses in these header lines.
H_RCPT	If H_RCPT flag is set in a header line, the header line is treated as a recipient header line. When called with the -t flag, sendmail sends the message to addresses in these lines. Recipient address rewriting is performed on addresses in these header lines.
H_ACHECK	Normally, sendmail includes any header lines from the input message in the output message, whether the mailer requires those header lines or not. However, if the H_ACHECK flag is set for a header line, sendmail deletes that header line unless the header definition in the configuration file is conditional (i.e., the header line is defined "H?*?Header-Name: ...,," where * is some mailer flag), and that header line is required by the mailer.
H_RESENT	<p>If header lines with this flag appear in the input message, the message is marked resent. Header lines that have the H_RESENT flag are treated like other header lines, except that they will be output only if the message has been marked resent.</p> <p>In other words, if there is at least one Resent header line in the input message, all the default Resent header lines defined in the configuration file will be added.</p>

The sendmail Configuration File

Header Flag	How Sendmail Handles the Header Line
H_EOH	Header lines with this flag set are interpreted as the end of the message header; everything following the first of these lines in a message is treated as part of the message body.
H_TRACE	These header lines are interpreted as tracing header lines. If sendmail finds more than 30 H_TRACE lines, it assumes the message is caught in an aliasing loop, and returns it to the sender as an error.
H_FORCE	Unless H_FORCE is set, sendmail does not add a particular header line if one already is in the message. This may be used for each system handling a message to add a tracing header line.

Special Header Lines

Table 6-11 describes header lines that are specially handled by sendmail.

Special Header Line	How sendmail Handles the Header Line
Full-Name:	sendmail defines the macro <code>x</code> from a Full-Name: header line if one is present.
Return-Receipt-To :	If the message contains this header line, and if the mailer for the current recipient has the <code>I</code> flag (local delivery) set in its mailer definition, acknowledgment is sent to any specified addresses when the final delivery is complete.

Table 6-11. Special Header Lines

Special Header Line	How sendmail Handles the Header Line
Errors-To:	If errors occur during processing, this header line causes error messages to be routed to the listed addresses rather than to the sender. This is intended for mailing lists.
Apparently-To:	If sendmail receives a message without any standard recipient lines (To: , Cc: , or Bcc: header lines), it adds this header line for any recipients in the envelope. This header line is used rather than a standard recipient line to warn recipients that the list may not be complete. At least one recipient line is required by RFC 822.
Precedence:	<p>If this header line is present, sendmail sets the precedence of the message to the specified value. The value may be either a numeric value or one to the precedence classes defined on a P line in the configuration file.</p> <p>See the “Precedence” subsection of the “sendmail Configuration File” section for how to set precedence classes, and the “Mail Queue” subsection of the “Operating sendmail” section for how precedence is used by sendmail.</p>

M Lines – Mailer Definitions

In this context, mailers are delivery agents. The **M** line is used to define sendmail’s interface to a mailer. The syntax is:

Mname, field=value [,field=value,...]

where *name* is the name *sendmail* uses to refer to the mailer and the *field=value* pairs define attributes of the mailer. Only the first character of each *field* is checked. Available mailer fields are described in Table 6-12:

The sendmail Configuration File

Table 6-12. Mailer Fields

Field	Description
Argv	A template for the argument vector to pass to this mailer.
Eol	The end-of-line string for this mailer.
Flags	Special flags for this mailer.
Maxsize	The maximum message length to this mailer.
Path	The full pathname of the program that implements this mailer.
Recipient	A mailer-dependent rewriting ruleset for recipient addresses in the message header.
Sender	A mailer-dependent rewriting ruleset for sender addresses in the message header.

Mailer Name

Each delivery agent must have an internal name. This can be arbitrary, except that the `local` and `prog` mailers must be defined.

The Path Field

The `Path` field of the mailer definition must be specified as the pathname of the executable mailer program. If this mailer will use `sendmail`'s internal SMTP client code over a socket, use the string “[IPC]” instead.

The Flags Field

The flags described in Table 6-13 can be specified in the *flags* field of mailer definitions. These flags either define some aspect of `sendmail`'s interface with the mailer or indicate that the mailer requires a particular header field. Any unassigned flags can be used freely to indicate particular header fields required by the mailer. The interaction between mailer flags and conditional header fields is described in the “Header Definitions” section.

Table 6-13. Mailer Flags

Flag	Description
A	This is an ARPANET-compatible mailer, and all appropriate modes should be set.
C	<p>If mail is received from a mailer with this flag set, any addresses in the header that do not have an at sign (@) after being rewritten by ruleset 3 will have the clause “@domain” from the sender tacked on. This causes mail with headers of the form:</p> <pre>From: dan@node1 To: david@node2, mickie</pre> <p>to be rewritten automatically as:</p> <pre>From: dan@node1 To: david@node2, mickie@node1</pre>
D	This mailer expects a Date: header line.
e	This mailer is expensive to connect to, so avoid connecting immediately; wait until the next queue run.
E	If mail is sent to this mailer, <code>sendmail</code> should escape message lines that begin with "From" with the > character.
F	This mailer expects a UNIX-style From_ header line.
f	The mailer expects a -f flag, but only if this is a network forward operation (i.e., the mailer will give an error if the executing user does not have special permissions).
h	Upper-case should be preserved in host names for this mailer.
I	This mailer will be speaking SMTP to another <code>sendmail</code> and, as such, it can use special protocol features. This option is not required. (If this option is omitted the transmission will still operate successfully, although not as efficiently as possible).
L	Limits the line lengths as specified in RFC 821.
l	This is a local mailer (i.e., final delivery will be performed).

The sendmail Configuration File

Table 6-13. Mailer Flags

Flag	Description
m	Allows this mailer to send to multiple users on the same host in one transaction. When a \$u macro occurs in the argv part of the mailer definition, that field is repeated as necessary for all qualifying recipients.
M	This mailer expects a Message-Id: header line.
n	Do not insert UNIX-style From_ line on the front of messages to this mailer.
P	This mailer expects a Return_Path: header line. RFC 822 specifies that Return_Path: should only be added for delivery agents that are performing final delivery, i.e., those that have the l mailer flag set.
p	Outputs the return path, in the form of a source route, instead of the sender's address in the SMTP MAIL From: command to this mailer. Although required by RFC 822, p is not the default because many hosts do not process source routes properly.
r	This is the same as f but sends a -r flag.
S	Causes <code>sendmail</code> not to reset the user ID before calling the mailer. This could be used to avoid forged addresses. This flag is suppressed if specified from an unsafe environment, such as a configuration file specified with the -C flag, or if <code>sendmail</code> is not running <code>setuid</code> to root.
s	Strips quote characters from the address before calling the mailer.
U	This mailer expects UNIX-style From_ lines with the UUCP-style "remote from" on the end.
u	Upper-case should be preserved in user names for this mailer.
X	This mailer uses the hidden-dot algorithm as specified in RFC 821; any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This insures that lines in the message containing a dot do not terminate the message prematurely.
x	This mailer expects a Full-Name: header line.

The sendmail Configuration File

If the delivery agent does not require the **r** or **f** mailer flag to be specified in the **F** field, **\$g** can be used in the **argv** template to pass the name of the sender on the command line. If the **r** or **f** mailer flag is specified in the **F** field, the sender will only be passed on the command line if it was passed to `sendmail` this way.

If the mailer must be called as **root**, the **S** mailer flag should be specified; this will not reset the user ID before calling the mailer. Otherwise, before calling a mailer, `sendmail` resets its **uid** to the calling user's **uid**. If `sendmail` is running as a daemon, it sets its **uid** to the default defined with the **u** option.

If the delivery agent is local (i.e., it performs final delivery to a user's mailbox rather than another network hop), the **l** mailer flag should be specified. The **ruleset 0** lines that resolve to this mailer must omit the **\$\$@host**. `sendmail` will also generate a **Return-Receipt-To:** message, if requested, if delivery occurred to a mailer marked local.

If the **s** mailer flag is specified, quote characters (backslashes (****) and quotation marks (**"**)) will be stripped from addresses. If the **s** flag is not specified, the quote characters will be passed through to the mailer.

If the mailer is capable of sending to more than one user on the same host in a single transaction, the **m** mailer flag should be specified. If this flag is on, then **\$u** will be repeated in the **argv** template for each unique user on a given host.

The **e** mailer flag will mark the mailer as being expensive, which will cause `sendmail` to defer connection until a queue run. The **c** configuration option must also be specified for this to be effective.

An unusual case is the **C** mailer flag. This flag applies to the mailer that the message is **received** from, rather than the mailer to which the message is being sent. (Actually, this is determined by rewriting the sender address with **ruleset 0** and determining which mailer would be used.) If set, the domain specification of the sender (i.e., **@domain**) is saved and is appended to any addresses in the message that do not already contain a domain specification. For example, if a message received by `node2` had the header lines,

The sendmail Configuration File

From: joe@node1
To: wddy@node2, doug

and the C flag is defined for the mailer to which joe@node1 resolves, sendmail would rewrite doug as doug@node1:

From: joe@node1
To: wddy@node2, doug@node1

This is done because the address doug in a message originating from node1 almost certainly refers to a local mailbox on node1. If @node1 were not added, a user agent replying to this message would incorrectly interpret doug as local on node2.

Actually, node1 should have rewritten doug as doug@node1, as described in the “Address Rewriting” section.

The Sender and Recipient Fields

The S and R fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient addresses respectively. These are applied after the sending domain is appended and **rulesets 1 and 2** are applied, but before the final output rewrite (**ruleset 4**) is applied. A typical use is to append the current domain to addresses that do not already have a domain. For example, a header of the form

From: dan

might be changed to the following:

From: dan@node1

or

From: node1!dan

depending on the domain into which it is being sent. These sets can also be used to do special purpose output rewriting in cooperation with **ruleset 4**.

The Eol Field

The **E** field defines the string to use to terminate lines passed to the mailer. A string containing only newline is the default. The backslash escapes `\r` (carriage return), `\n` (line feed), `\f` (form feed), and `\b` (backspace) can be used.

The Argv Field

Finally, an argv template is given as the **A** field. It may have embedded spaces. If there is no argv with the `$u` macro in it, `sendmail` will speak SMTP to the mailer. If the pathname for this mailer is IPC, the argv should be:

```
IPC $h [port]
```

where port is the optional port number to connect to. By default, `sendmail` will use the SMTP TCP port returned by `getservbyname`.

Example Delivery Agents Definitions

The following example specifies a delivery agent that performs local delivery:

```
Mlocal, P=/bin/rmail, F=lsDFm, S=10, R=20, A=rmail -d $u
```

This delivery agent is called `local` (`Mlocal`). It is located in the file `/bin/rmail` (`P=/bin/rmail`). It expects quotes to be stripped from addresses, expects a **Date:** and a **From:** line, and multiple users can be delivered to at once (`F=lsDFm`). **Ruleset 10** should be applied to sender addresses in the message (`S=10`) and **ruleset 20** should be applied to recipient addresses (`R=20`). The argv used when exec'ing the mailer to send a message will be the words `rmail -d` (`A=rmail -d`) followed by the recipient names (`$u`).

The following example specifies a delivery agent that performs Ethernet delivery:

```
Mtcp, P=[IPC], F=meC, S=11, R=21, A=IPC $h, M=100000
```

This delivery agent is called `tcp` (`Mtcp`). It uses an IPC connection to deliver the mail (`P=[IPC]`). It can handle multiple users at once, is considered an **expensive** mailer and expects any domain from the sender address to be appended to any

The sendmail Configuration File

receiver name without a domain (F=meC). Sender addresses are processed by **ruleset 11** (S=11) and recipient addresses are processed by **ruleset 21** (R=21). Since this mailer uses an IPC connection, the argv field contains the word IPC followed by the name of the host, and since no port is specified, the default port will be used (A=IPC \$h). A 100,000-byte limit on messages passed through this delivery agent is also specified (M=100000).

O Lines – Option Specifications

A number of processing options can be set for sendmail from the configuration file. Options are represented by single characters. The syntax of an O line is:

Ovalue

This sets option o to value. Depending on the option, value may be a string, an integer, a boolean (only the first character is examined; legal values are t, T, f or F; the default is T), or a time interval.

Before actually modifying the options in your configuration file, you may want to test how sendmail behaves with the changed option. Options in the configuration file or frozen configuration can be overridden using the **-o** flag in the command line.

For example,

```
/usr/lib/sendmail -oT2m
```

sets the **T** (queue time out) option to two minutes for this run only.

All time intervals are set using a scaled syntax. For example, **10m** represents ten minutes and **2h30m** represents two and a half hours. The full set of scales is:

The sendmail Configuration File

Letter	Interpretation
s	seconds
m	minutes
h	hours
d	days
w	weeks

The options in Table 6-15 can be modified to reflect local needs or preferences.

Option	Description
dx	<p>Sets the delivery mode to mode x. Legal modes are:</p> <ul style="list-style-type: none">i Deliver interactively (synchronously)b Deliver in background (asynchronously)q Queue the message (deliver during next queue run) <p>In mode i, <code>sendmail</code> completes delivery or fails before returning to the calling program or user. Mode i passes the maximum amount of information to the sender, but is rarely necessary.</p> <p>In mode b, <code>sendmail</code> forks a child to perform delivery; the parent process returns immediately to the calling program user. Mode b is usually a good compromise. However, this mode can generate large numbers of processes if you have a mailer that takes a long time to deliver a message.</p> <p>In q mode, <code>sendmail</code> puts any messages in the queue and returns immediately. Messages are delivered when the queue is processed next. This puts the minimum load on your system, but mail delivery will be delayed for up to the queue interval.</p>

The sendmail Configuration File

Table 6-15. Configurable Sendmail Configuration File Options	
Option	Description
ex	<p>Disposes of errors using mode x. The values are:</p> <ul style="list-style-type: none"> m Mail errors to sender p Print error messages to standard output (default) q No messages, but give exit status w Write errors to sender's terminal; mail if user not logged in <p>If the message text is not mailed back by the m or w modes and the sender is local to the host, a copy of the message is appended to the <code>.l</code> letter file in the sender's home directory.</p> <p>The <code>sendmail</code> daemon (<code>-bd</code> mode or <code>-q</code> flag) runs in error mode m regardless of the value of this option.</p>
I	<p>If this host is using the domain nameserver, the I option should be set. In most cases, <code>sendmail</code> can detect whether the nameserver is in use and will automatically do the right thing. However, in some cases it will incorrectly appear to <code>sendmail</code> as though the nameserver is not being used; for example, the nameserver may not have finished loading its data and is not yet responding to queries. If this host normally uses the nameserver, messages should be deferred when the nameserver is not running.</p> <p>If the I option is set: If an MX record lookup or <code>gethostbyname()</code> call fails because the nameserver is not running, <code>sendmail</code> will defer the message.</p> <p>If the I option is not set: If an MX record lookup fails, <code>sendmail</code> assumes that the name server is not being used and that there are no MX records, and tries to connect directly to the host. If a host name lookup fails, <code>sendmail</code> assumes that there is no entry for the host and returns an error.</p>
L<i>n</i>	Sets the default log level to <i>n</i> .

Table 6-15. Configurable Sendmail Configuration File Options	
Option	Description
<i>rtimeout</i>	Times out SMTP reads after <i>timeout</i> interval.
<i>Ttime</i>	<p>Sets the timeout on message in the queue to <i>time</i>. After this interval, messages that have not been successfully sent are returned to the sender.</p> <p>The queue-control file contains the time of submission, rather than the amount of time left until time out. As a result, you can remove messages that have been in the queue for a short period by processing the queue with a short message time out. For example,</p> <pre style="text-align: center;">/usr/lib/sendmail -oT1d -q</pre> <p>processes the queue and removes anything that is one day old.</p>
<i>Paddress</i>	When an undeliverable message is mailed back, a copy of the message header (but not the message body) is delivered to this address. It is suggested that this address be "Postmaster" since RFC 822 requires that there be a valid Postmaster address on all mail systems, and since sendmail also attempts to deliver undeliverable error return messages to Postmaster. Postmaster should be aliased locally to the (local or remote) address of the person responsible for mail system administration.

Table 6-16 describes options that HP recommends **not** be changed from the defaults in the supplied configuration file.

The sendmail Configuration File

Table 6-16. Options Not Recommended for Modification

Option	Description
<i>Afile</i>	Uses the named file as the alias file. If <i>A</i> is specified with no file, a file named <code>aliases</code> in the current directory is used.
<i>aN</i>	If set, <code>sendmail</code> waits up to <i>N</i> minutes for an <code>@: @</code> entry to appear in the alias database before proceeding with message delivery. If the entry does not appear in <i>N</i> minutes, it rebuilds the database (if the <i>D</i> option is also set), or issues a warning.
<i>Bc</i>	Sets the blank substitution character to <i>c</i> . This character replaces unquoted spaces in addresses.
<i>c</i>	Causes <code>sendmail</code> to delay connecting to any outgoing mailers that are marked “expensive”. Instead, the mail will be routed the next time that the mail queue is processed.
<i>D</i>	If set, the alias database is rebuilt automatically if it is out of date and the database files are writable by all. If this option is not set, <code>sendmail</code> never rebuilds the alias database unless it is run in <code>-bi</code> mode (<code>newaliases</code>).
<i>Fmode</i>	Sets the mode for queue files, in octal. A good choice is <code>600</code> , since it does not permit users to read other people’s mail.
<i>f</i>	Saves UNIX-style <code>From_</code> lines at the front of headers. Otherwise, they are assumed redundant and discarded.
<i>gN</i>	Sets the default group ID for mailers to <i>N</i> (also see option <i>u</i>).
<i>Hfile</i>	Specifies the SMTP help file.
<i>i</i>	Causes <code>sendmail</code> not to interpret a period (<code>.</code>) on a line by itself as a message terminator.
<i>Mxvalue</i>	Sets the macro <i>x</i> to <i>value</i> .
<i>m</i>	Sends the message to the sender also if the sender is in an alias expansion. Note that if this option is not set and <code>sendmail</code> determines that a recipient is the same as the sender, the message is not sent to that recipient.

Table 6-16. Options Not Recommended for Modification

Option	Description
n	Parses the right-hand sides of alias definitions when initializing the alias database. If this option is set, an error in a mailing list will be reported when initializing the alias database. However, since each address must be parsed, it may take a long time to initialize a large alias database. If this option is not set, store the right-hand sides unexamined and an error in a mailing list will be reported whenever an attempt is made to mail to it.
o	Assumes that the headers may be in old format, (i.e., spaces delimit names). This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it is assumed that the list is already comma-separated. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.
Q <i>queuedir</i>	Uses the named <i>queuedir</i> as the queue directory. The default queue directory is <i>/usr/spool/mqueue</i> .
q <i>N</i>	If the current load average exceeds the load average limit set with the x option, then a priority limit is found by dividing <i>N</i> by the difference (plus one) between the current load average and the load average limit. Messages with a priority value exceeding the priority limit (i.e. of lower priority) will be queued. The default value of <i>N</i> is 10000.
S <i>file</i>	Specifies the file in which to save statistics. The default file is <i>/usr/lib/sendmail.st</i> .
s	Always creates the queue control file, even in circumstances where it may not be necessary (i.e., interactive delivery mode).

The sendmail Configuration File

Table 6-16. Options Not Recommended for Modification

Option	Description
uN	<p>Sets the default user ID for mailers to <i>N</i>.</p> <p>For security reasons, before <code>sendmail</code> exec's a mailer when it is running as a daemon, it resets both its real and its effective user ID and groupid to the default set by the <code>u</code> and <code>g</code> options.</p> <p>For a mailer that is trusted and can be run as root, resetting the user ID and groupid can be overridden, if necessary, by setting the <code>S</code> mailer flag. The <code>S</code> mailer flag is not required by any HP-UX mailers.</p> <p>When running in daemon mode (<code>-bd</code>), <code>sendmail</code> resets its real (but not its effective) user ID and groupid to these defaults.</p>
v	Runs <code>sendmail</code> in verbose mode.
XN	If the current five minute load average is greater than <i>N</i> , the <code>sendmail</code> daemon will not accept connections. If <code>X</code> is not explicitly set, the default value is 12.
xN	If the current five minute load average is greater than <i>N</i> , <code>sendmail</code> queues low priority messages rather than delivering them immediately. If <code>x</code> is not explicitly set, the default value is 8.
Y	Setting this option will cause <code>sendmail</code> to process each queued message in a separate process. If <code>Y</code> is not set, <code>sendmail</code> keeps track of hosts that are down during the queue processing.
yN	The value <i>N</i> is added to the priority of a message (thus lowering its priority) for each recipient. This penalizes messages with large numbers of recipients. The default value of <i>N</i> is 1000.
ZN	Each time a message is processed from the queue, the value <i>N</i> is added to the priority of the message, thus penalizing queued messages that are undeliverable for long periods of time. The default value of <i>N</i> is 9000.

Table 6-16. Options Not Recommended for Modification

Option	Description
zN	The result of multiplying the message precedence by the value <i>N</i> is subtracted from the message priority, thus favoring higher precedence messages. The default value of <i>N</i> is 1800.

P Lines — Precedence Definitions

Precedence is used to define a rank, if specified, to classes of messages. The syntax of this field is:

```
Pmessage_class=num
```

When the `message_class` is specified in a precedence field in the message header, the precedence of the `message_class` is set to `num`. Higher numbers define a higher precedence. Numbers less than zero will suppress the delivery of error messages. The default precedence is zero. For example, if the following is set in the **P** lines of the configuration file,

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

and Precedence: `first-class` appears in the message header, the message will be routed with a precedence of zero, the default precedence. If the message header contains the line Precedence: `junk`, the message will be routed with a precedence of -100 and, since a negative value is given to the message class, the delivery of error messages for that message will be suppressed.

See the “Mail Queue” subsection of the “Operating sendmail” section for details of how precedence is used.

The sendmail Configuration File

T Lines – Trusted User Definitions

Trusted users are permitted to define the sender address using the **-f** flag. These typically are root, UUCP and network. However, it may be convenient to extend this list to include other users, for example, to support a separate UUCP login for each host. The syntax of this line is:

```
T user1 user2...
```

Multiple trusted user lines can exist in the configuration file.

S and R Lines – Address Rewriting

Address rewriting rulesets are ordered sets of address rewriting rules defined in the configuration file. These rulesets perform two major functions:

- They tell sendmail how to route mail by specifying how sendmail should interpret and resolve recipient addresses to {delivery_agent, host, user} triples.
- They tell sendmail how to rewrite sender and recipient addresses which are passed in the message as envelope or header information. This ensures that this information can be correctly understood at the destination.

The syntax for specifying the start of a ruleset is:

```
Sn
```

This sets the current ruleset being collected to *n*, where *n* is the number of the ruleset that is being defined. If a ruleset is begun more than once, the old definition is deleted. It is an error to have an **R** line before the first **S** line.

The syntax for specifying a rule is:

```
Rleft-side right-side [optional comment]
```

The sendmail Configuration File

The rule becomes part of the most recently defined ruleset. The left-side is a pattern that is applied to the input. If it matches, the input is rewritten as specified by the right-side.

Since the left-side and right-side may contain embedded blanks, they must be separated by one or more tabs. Any comment must also be separated from the right-side by one or more tabs.

`sendmail` rewrites addresses by applying whole rulesets to them. It applies a ruleset by trying each rule in the ruleset on the address. If the rule matches, `sendmail` rewrites the address. If all rules in a given ruleset fail, the input address is returned with no modifications.

`sendmail` uses rulesets 0, 1, 2, 3, and 4 in internally defined predetermined ways, as indicated in Figure 6-2. Additional rulesets can be defined as “subroutines” that are called by other rulesets or defined as mailer-specific header rewriting rulesets.

The sendmail Configuration File

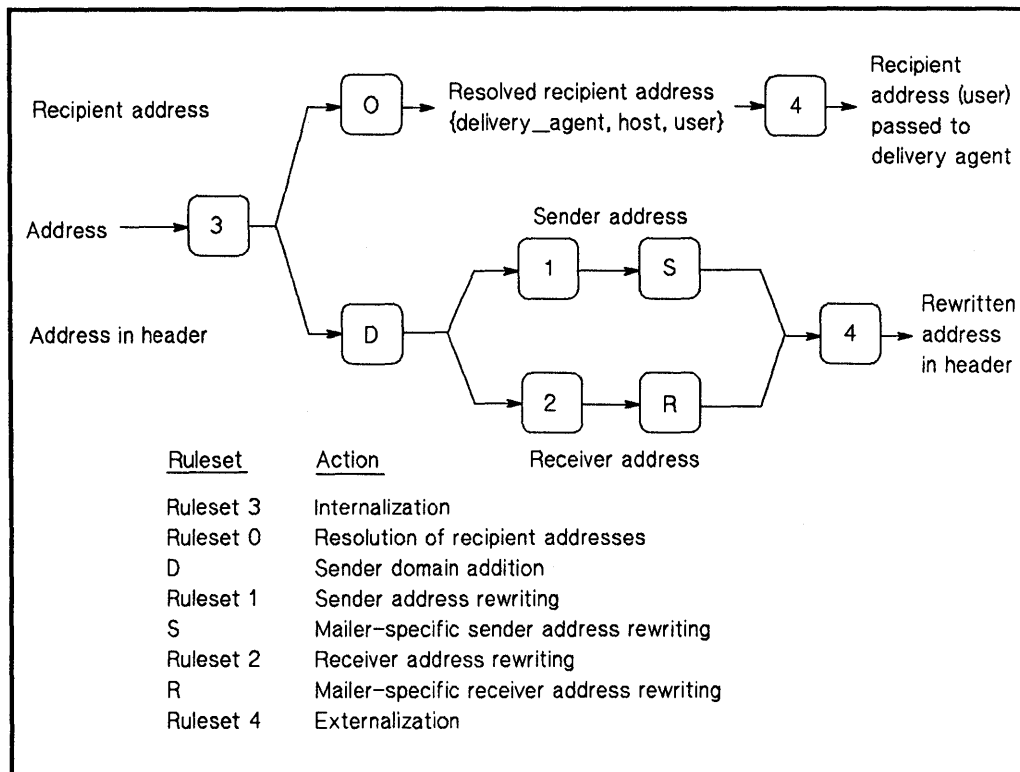


Figure 6-2. Rulesets

Recipient Address Rewriting

Sendmail uses **rulesets 3, 0, and 4** to determine how to route recipient addresses. `sendmail` resolves a recipient address to a {delivery-agent, host, user} triple. The message will be passed to the specified delivery-agent (mailer) for delivery to the specified host, and the recipient address passed to the delivery agent will be user.

1. Internalization

Ruleset 3 is always applied to addresses before any other rewriting. **Ruleset 3** is intended to rewrite the address in an **internal form** that makes it easy for later rulesets to distinguish the important features of the address. One widely used

The sendmail Configuration File

internal form is to rewrite addresses with angle brackets enclosing an @ sign followed by the host part of the address. For example:

```
user@host
```

might be rewritten by **ruleset 3**:

```
user<@host>
```

This internalization should later be undone by **ruleset 4**.

2. Resolution

Ruleset 0 is where the actual routing decisions are made. The selection of a delivery agent is based on the syntax of the address or on the recipient host name. For example, **ruleset 0** might resolve the address fred@jungle to the tcp delivery agent, the host jungle, and the user fred<@jungle>. **Ruleset 0** can also reject certain addresses as undeliverable.

Note

Once **ruleset 0** has resolved an address, sendmail will attempt delivery. If delivery fails, sendmail **will not** resume rewriting the recipient address in an attempt to find another mailer to try.

The \$< external nameserver facility could be used to query a mailer before resolving to it.

3. Externalization

The user part of the **ruleset 0** result is translated back to external form for the delivery agent by **ruleset 4**. For example, **ruleset 4** might externalize user<@host> as user@host.

The sendmail Configuration File

Recipient address rewriting is also applied to the sender address from the envelope, so that sendmail can associate a mailer and a host with the sender address.

Header Address Rewriting

Once the recipient and sender addresses from the envelope have been resolved, sender or recipient header line addresses are rewritten appropriately for the receiving mail system. For example, route-based addresses such as those used by UUCP must be rewritten from the point of view of the receiving host, and local addresses sent to a remote system must be rewritten “@” the sending host’s domain.

1. Internalization

Both sender and recipient header addresses are rewritten in a configuration-defined internal form by **ruleset 3**. The internal form for any non-local address must contain an @ sign so that sender domain addition will work.

2. Sender Domain Addition

Under some circumstances, sendmail will add the sender domain to certain header addresses. When the C flag is set in the mailer that sendmail associates with the sender and the envelope sender address contains an @ sign, then sendmail will add the @ sign and everything following it from the sender address to any addresses in the header that do not contain @ signs. sendmail does this because addresses that are local to the sending host are not local to the receiving host. The sending host should have (but may not have) already added “@” its own domain.

3. Sender and Receiver Rewriting

sendmail rewrites addresses in “sender” header lines separately from addresses in “recipient” header lines. It distinguishes sender from recipient header lines according to an internal table described in the “Header Definition” section.

Sender header addresses are first rewritten by **ruleset 1**, then by the **S** ruleset (if any), defined for the recipient mailer (see the “Mailer Definition” section).

Recipient header addresses are first rewritten by **ruleset 2**, then by the **R** ruleset (if any), defined for the recipient mailer.

4. Externalization

The header addresses are externalized by **ruleset 4**.

Rewriting Rules

Each rewriting rule has a left side and a right side. The left side is a pattern, consisting of fields separated by delimiters.

Any address field that matches the pattern in the left side of the rule is rewritten according to the format on the right side of the rule. Then, the same rule is repeatedly re-tried on the output of the rewriting until the pattern no longer matches.

Then, the next rule in the ruleset is tried. This is repeated until a rule terminates the ruleset with **\$@** (see Table 6-18, Token Metasymbols for the Right Side) or the end of the ruleset is reached.

The output of the ruleset is the address as it has been rewritten by any of the rules in the set that were applied. If no rule matches, the address is not changed.

The sendmail Configuration File

The Left Side

The fields of the left side of the rule are either literal text, delimiters, or metasymbols. The delimiters are characters from the set defined by the `o` macro. For example, `o` typically includes `.`, `:`, `!`, `@`, and `/`.

`sendmail` begins the address rewriting process by parsing the address into tokens. For example, if `"/` is a delimiter, the address

`a/b/c`

will be parsed into five tokens: `a`, `/`, `b`, `/` and `c`.

Once the address has been parsed into tokens, the tokens are matched to patterns which are defined in the left side of the rule. A token can match a pattern by matching literal text or by matching metasymbols; metasymbols are described below. A token matches a delimiter by matching the literal delimiter character.

Table 6-17. Token Metasymbols for the Left Side

Metasymbol	Meaning
<code>\$*</code>	Match zero or more tokens
<code>\$+</code>	Match one or more tokens
<code>\$-</code>	Match exactly one token
<code>\$=x</code>	Match any token or concatenation of tokens in class <code>x</code>
<code>\$~x</code>	Match any single token not in class <code>x</code>

A token (or tokens) that matches a metasymbol on the left side of the rewriting rule is referred to as `$n` on the right side of the rule, where `n` is the index of the metasymbol on the left side. For example, if the left side of the rewriting rule contains:

`$-:$+`

The sendmail Configuration File

and this is applied to the address:

```
HP:dvs
```

First, the colon (:) delimiter is identified. Then, the rewriting rule searches for exactly one token (\$-) for \$1. Since HP is the first token before the colon (:) delimiter, it is assigned to \$1. Next, the rewriting rule searches for one or more tokens to assign to \$2. Since dvs qualifies as one or more tokens, it is assigned to \$2. The final matches would be represented as:

```
$1 = HP  
literal match = :  
$2 = dvs
```

Consider a more complex example. If the left side of the rule is:

```
<@$->:$+
```

and it is applied to:

```
<@HP>:david%node1
```

the following would occur.

First, all tokens and delimiters are identified. Then, the rule searches for a literal match for < before an @ delimiter. The literal match is made. Next, the rule searches for exactly one token followed by >:. HP qualifies as that token and is assigned to \$1. Finally, the rule searches for zero or more tokens after a : delimiter. David, the % delimiter, and node1 qualify as the tokens. So the rule matches, and the right side of the rule would be applied to the following tokens:

```
literal match = @  
$1 = HP  
literal match = :  
$2 = david%node1
```

The sendmail Configuration File

The Right Side

When a token on the left side of a rewriting rule matches the pattern specified in the rule, the action specified by the right side is performed with the values determined from the left side. Tokens are copied directly from the right side unless they begin with a \$. Metasymbols are described in Table 6-18.

Meta-symbol	Interpretation
\$n	Where n is a number, inserts tokens into the rewritten address. The token or tokens that matched the nth occurrence of any of \$+, \$-, \$*, \$=, or \$~ on the left side of the rule are substituted for \$n.
\$>n	Calls ruleset n. This syntax causes the remainder of the line to be substituted as usual and then be rewritten by ruleset n. The output of ruleset n then becomes the output for this rule.
\$<program	<p>Calls program to rewrite the address. This allows access to external nameservers or address resolvers.</p> <p>Program is the name of a macro that must be defined as the full pathname of an executable program. The macro must be defined as a single word, with no arguments. The address that matched the left side of the rule, rewritten by the remainder of the right side of the rule, is passed as the single argument to program.</p> <p>If program returns an exit status of 0, the address is rewritten as the first line of program's standard output. Any standard error output is treated as a warning and logged.</p> <p>If program returns other than 0, the address remains as rewritten before being passed to program and the standard output of program is ignored. In this case, any standard error output of program is treated as an error message.</p>

Table 6-18. Token Metasymbols for the Right Side

Meta-symbol	Interpretation
<p>\$<program (cont.)</p>	<p>For example, the following lines cause <code>sendmail</code> to call the program <code>/usr/bin/uupath</code> if the left side of the rewriting rule matches the address being rewritten:</p> <pre># pathalias address resolver - see uupath, pathalias DP/usr/bin/uupath . . . # attempt to resolve address via uupath. R\$+<@\$-> \$:\$<P\$1@\$2 #pathalias routing</pre> <p>The first two lines define macro <code>P</code> as <code>/usr/bin/uupath</code>, the <code>pathalias</code> address resolver.</p> <p>The last two lines specify that <code>/usr/bin/uupath</code> should be invoked with <code>\$1@\$2</code> as its argument whenever the left side of the rule matches. Specifically, if the rule in the example above is applied to <code>joe<@sys6></code>, <code>joe</code> matches <code>\$+</code> and <code>sys6</code> matches <code>\$-</code>. Since the rule matches, <code>/usr/bin/uupath</code> is invoked with argument <code>joe@sys6</code>. The output of <code>uupath</code> is substituted for the address and rewriting continues. Note that <code>\$:</code> is used to avoid repeated application of this rule.</p>

The sendmail Configuration File

Table 6-18. Token Metasymbols for the Right Side

Meta-symbol	Interpretation
\$[host\$]	<p>Canonicalizes host. Host can be one or more tokens and can be either an internet address (within square brackets []) or a host name.</p> <p>If the nameserver is in use (configuration option I is true), it will be called to replace the internet address or name with the official fully domain-qualified name of the host. Host names will be looked up using the search algorithm described in <code>hostname</code>. Note that if the nameserver is in use, if recipient hostnames are <i>not</i> canonicalized in Ruleset 0, all recipient host names destined for [IPC] mailers must be presented to <code>sendmail</code> in canonical form, or else <code>sendmail</code> will not be able to connect to the remote host. The supplied configuration file canonicalizes all recipient host names destined for the <code>tcp</code> mailer with <code>[\$]</code>.</p> <p>If the nameserver is not in use, <code>gethostbyaddr()</code> or <code>gethostbyname()</code> will be called to canonicalize the hostname. See <code>gethostent</code>.</p> <p>In either case, if the lookup fails, host is unchanged. For example, the rule:</p> <pre>R\$*<@\$+>\$* \$:\$1<@[\$2\$]>\$3</pre> <p>would match the address:</p> <pre>dmh<@hpcndr></pre> <p>and rewrite it as:</p> <pre>dmh<@hpcndr.cnd.hp.com></pre> <p>It would also match:</p> <pre>jmc<@[15.2.112.119]></pre> <p>and rewrite it as:</p> <pre>jmc<@hpcndr.cnd.hp.com></pre>

Table 6-18. Token Metasymbols for the Right Side

Meta-symbol	Interpretation
\$#mailer	<p>Resolves to mailer. This delivery agent must be one of the defined mailers. \$# must only be used in ruleset zero because it causes evaluation of the ruleset to terminate immediately, and signals to sendmail that the address has completely resolved.</p> <p>The complete syntax is:</p> <pre>#mailer \$@host \$:user</pre> <p>This specifies the {delivery_agent, host, user} triple necessary to route the message. If the mailer is local (i.e., has the l flag defined), \$@host must be omitted.</p> <p>The mailer with the name error can be used to generate error numbers and text for user errors. The host field, which is optional in this case, is a numeric exit status to be returned to the sender of the message, and the user field is a message to be printed. For example, if the address user<@planet> matches the rule</p> <pre>R\$+<@\$+> \$#error\$@68\$:host\$2unknown</pre> <p>sendmail issues the error message:</p> <pre>sendmail: user@planet... host planet unknown</pre> <p>and sets its exit status to 68.</p>

The sendmail Configuration File

Meta-symbol	Interpretation
\$@	<p>Occurs in two contexts. When \$@host follows ##mail er in the right side of a rule in ruleset 0, the token following \$@ is the host part of the {delivery_agent, host, user} triple to which the address resolves.</p> <p>When \$@ begins the right side of a rule, the rule is applied once and the ruleset terminates, returning the address as rewritten once by the rest of the right side of the rule.</p>
\$:	<p>Occurs in two contexts. When :\$user follows ##mail er and \$@host in the right side of a rule in ruleset 0, the token following \$: is the user part of the {delivery_agent, host, user} triple to which the address resolves.</p> <p>When \$: begins the right side of a rule, the rule is applied once but the ruleset continues. This syntax is used to avoid continued application of a rule.</p>

The **\$@** and **\$:** prefixes can precede a **\$>** or **\$<**. For example, in the rule:

```
R$+  $:.$>7$1
```

the **\$+** matches anything, the entire address (**\$1**) is passed to **ruleset 7** once, and the current ruleset continues. Since **\$+** will match anything **ruleset 7** returns, the **\$:** is necessary to avoid an infinite loop.

Testing Address Rewriting Rulesets

sendmail provides an address test mode (**-bt**) that allows you to observe how addresses are rewritten by specified rulesets. For example:

```
/usr/lib/sendmail -bt -Ctest.cf
```

could be used to examine the behavior of a sendmail configuration file named `test.cf`.

Each line of input to **sendmail -bt** must be of the form:

```
rulesets addresses
```

where `rulesets` is either a single ruleset number or a list of rulesets separated by commas but no blanks, which will be applied in the order specified. `Addresses` is either a single mail address or a comma-separated list of addresses.

First, sendmail breaks the address into tokens. Then, it rewrites the address with **ruleset 3**. (If necessary, this can be disabled by commenting out **ruleset 3** in the configuration file.) Finally, it rewrites the address with the rulesets specified. It outputs each token with double quotes, and reports address resolution with the internal form of the `$#`, `$@`, and `$:` metasympols.

Exit address test mode by typing the **<EOF>** character (usually `^D`).

The sendmail Configuration File

The following examples use the supplied configuration file.

Example 1. Recipient Address Rewriting

```
$ /usr/lib/sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
>0,4 ghe@ghi
rewrite: ruleset 3  input: "ghe" "@" "ghi"
rewrite: ruleset 3  returns: "ghe" "<" "@" "ghi" ">"
rewrite: ruleset 0  input: "ghe" "<" "@" "ghi" ">"
rewrite: ruleset 6  input: "ghe" "<" "@" "ghi" ">"
rewrite: ruleset 6  returns: "ghe" "<" "@" "ghi" "." "ghu" "." "edu" ">"
rewrite: ruleset 0  returns: "^V" "tcp" ^W" "ghi" "." "ghu" "." "edu" ^X" "g
he" "<" "@" "ghi" "." "ghu" "." "edu" ">"
rewrite: ruleset 4  input: "^V" "tcp" ^W" "ghi" "." "ghu" "." "edu" ^X" "g
he" "<" "@" "ghi" "." "ghu" "." "edu" ">"
rewrite: ruleset 4  returns: "^V" "tcp" ^W" "ghi" "." "ghu" "." "edu" ^X" "g
he" "@" "ghi" "." "ghu" "." "edu"
> ^D$
```

Note the internalization done by **ruleset 3**. Then, **ruleset 0** calls **ruleset 6** to determine whether the address is local and to canonicalize the host name. Next, **ruleset 0** resolves the address to the tcp mailer, the host ghi.ghu.edu, and the user ghe<@ghi.ghu.edu>. Finally, **Ruleset 4** restores the user to external form.

A simpler way to test recipient address resolution is to do:

```
$ /usr/lib/sendmail -bv -oL10 ghe@ghi
ghe@ghi... deliverable, mailer=tcp, host=ghi.ghu.edu, user=ghe@ghi.ghu.edu
```

Example 2. Sender Header Address Rewriting

```
$ /usr/lib/sendmail -bt
  ADDRESS TEST MODE
Enter <ruleset> <address>
> 1,13,4 gup!guest
rewrite: ruleset 3: input: "gup" "!" "guest"
rewrite: ruleset 3 returns: "guest" "<" "@" "gup" "." "UUX" ">"
rewrite: ruleset 1 input: "guest" "<" "@" "gup" "." "UUX" ">"
rewrite: ruleset 6 input: "guest" "<" "@" "gup" "." "UUX" ">"
rewrite: ruleset 6 returns: "guest" "<" "@" "gup" "." "UUX" ">"
rewrite: ruleset 1 returns: "guest" "<" "@" "gup" "." "UUX" ">"
rewrite: ruleset 13 input: "guest" "<" "@" "gup" "." "UUX" ">"
rewrite: ruleset 13 returns: "gup" "!" "guest" "<" "@" "myhost" "." "UUX" ">"
rewrite: ruleset 4 input: "gup" "!" "guest" "<" "@" "myhost" "." "UUX" ">"
rewrite: ruleset 4 returns: "myhost" "!" "gup" "!" "guest"
> ^D$
```

This is an example of the sender header address rewriting done for messages routed via the uucp mailer. Note that the internal form of `gup!guest` is `guest<@gup.UUX>`. **Ruleset 1**, the default sender header ruleset, calls **ruleset 6**, but does no other rewriting on this type of address. **Ruleset 13**, the sender header ruleset for the uucp mailer, adds the local host to the path for sender addresses to facilitate replies. **Ruleset 4** restores `gup!guest<@myhost.UUX>` to the external form `myhost!gup!guest`.

The sendmail Configuration File

Example 3. Recipient Header Address Rewriting

```
$ /usr/lib/sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
> 2,21,4 me@myhost
rewrite: ruleset 3 input: "me" "@" "myhost"
rewrite: ruleset 3 returns: "me" "<" "@" "myhost" ">"
rewrite: ruleset 2 input: "me" "<" "@" "myhost" ">"
rewrite: ruleset 6 input: "me" "<" "@" "myhost" ">"
rewrite: ruleset 6 input: "me" "<" "@" "myhost" "." "mydomain" ">"
rewrite: ruleset 3 input: "me"
rewrite: ruleset 3 returns: "me"
rewrite: ruleset 6 returns: "me"
rewrite: ruleset 6 returns: "me"
rewrite: ruleset 2 returns: "me"
rewrite: ruleset 21 input: "me"
rewrite: ruleset 21 returns: "me" "<" "@" "myhost" "." "mydomain" ">"
rewrite: ruleset 4 input: "me" "<" "@" "myhost" "." "mydomain" ">"
rewrite: ruleset 4 returns: "me" "@" "myhost" "." "mydomain"
> ^D$
```

This is an example of the recipient header address rewriting done for messages routed via the tcp mailer. **Ruleset 2**, the default recipient header ruleset, calls **ruleset 6**, which recognizes me<@myhost> as a local address and rewrites it as me. Notice that **ruleset 6** calls itself recursively. **Ruleset 21**, the recipient header ruleset for the tcp mailer, adds @ and the canonical local host name to the address, since the recipient host is remote. **Ruleset 4** restores the address to external form.

Troubleshooting sendmail

Superuser

Almost all sendmail troubleshooting must be done as superuser.

Consistency

One common source of surprising sendmail behavior is inconsistencies between current configuration files and the configuration actually in use by sendmail.

The frozen configuration file must be up-to-date with any recent changes to the sendmail configuration file, local host name, nameserver data regarding the local host, and local UUCP configuration:

```
/etc/freeze # freeze the sendmail configuration
```

The sendmail daemon (on a stand-alone system or cluster server) must be using an up to date configuration:

```
/usr/lib/sendmail -bk # kill the sendmail daemon  
/usr/lib/sendmail -bd -q30m # restart the sendmail daemon
```

The alias database must be rebuilt if changes have been made to the alias text file:

```
newaliases # rebuild the alias database
```

Troubleshooting sendmail

Address Resolution and Aliasing

In order to deliver a message, `sendmail` must first resolve the recipient addresses appropriately. To determine how `sendmail` would route mail to a particular address, do:

```
/usr/lib/sendmail -bv -v -oL10 address [ address ... ]
```

Table 6-19. sendmail Address Resolution

Where:	Action:
-bv (verify mode)	Verifies addresses only. Do not collect or send a message.
v (verbose flag)	Reports alias expansion and duplicate suppression.
-oL10 (log level option)	Sets log level to 10. At this log level and above, sendmail -bv reports the mailer and host to which it resolves recipient addresses.

For hosts that resolve to [IPC] mailers, MX hosts are not reported when using verify mode, since MX records are not collected until delivery is actually attempted. (See the “Message Delivery” section and the “MX Records” subsection of “Operating `sendmail`”.)

If the address is not being resolved as you expect, you may have to modify:

- The `sendmail` configuration file.
- The files or programs from which file classes are generated.
- The nameserver configuration.
- The UUCP configuration.

More detailed information about how the configuration file is rewriting the recipient addresses is provided by address test mode:

```
/usr/lib/sendmail -bt
```

For more information about address test mode, see the discussion and examples in the “Sendmail Configuration File” section.

Message Delivery

You can observe sendmail’s interaction with the delivery agents to which it routes a message by delivering the message in verbose mode:

```
/usr/lib/sendmail -v address [ address ... ]
```

sendmail has interfaces to three types of delivery agents. In verbose mode, sendmail reports its interactions with them as follows:

- Mailers that use SMTP to a remote host over a TCP/IP connection ([IPC] mailers):

In verbose mode, sendmail reports the name of the mailer used, each MX host (if any) to which it tries to connect, and each internet address it tries for each host. Once a connection succeeds, the SMTP transaction is reported in detail.

- Mailers that run SMTP (locally) over pipes:

The name of the mailer used and the command line passed to `exec()` are reported. Then the SMTP transaction is reported in detail. If the mailer returns an abnormal error status, that is also reported.

- Mailers that expect envelope information in the argument vector and expect message headers and body from standard input:

The name of the mailer used and the command line passed to `exec()` are reported. If the mailer returns an abnormal error status, that is also reported.

Troubleshooting sendmail

Contacting the sendmail Daemon

It is possible to talk to the sendmail daemon and other SMTP servers directly with the command:

```
telnet host 25
```

This can be used to determine whether an SMTP server is running on host. If not, your connection attempt will return, "Connection refused."

The SMTP VRFY command can be used to determine whether the server can route to a particular address. For example:

```
telnet furschlugginer 25
220 furschlugginer.bftxp.edu SMTP server ready
vrfy aen
250 Alfred E. Newman <aen@axolotl.bftxp.edu>
vrfy blemph@morb.poot
554 blemph@morb.poot: unable to route to domain morb.poot
quit
221 furschlugginer.bftxp.edu SMTP server shutting down
```

Note

Not all SMTP servers support the VRFY and EXPN commands.

sendmail Logging

sendmail logs the following:

- Failures beyond its control (SYSERR).
- Administrative activities (for example, rebuilding the alias database, and killing and restarting the daemon).
- Events associated with mail transactions.

Log entries marked SYSERR indicate either system failures or configuration errors, and may require the attention of the system administrator. The system log is described in the “Operating sendmail” section.

Each system log entry for a mail transaction has a Queue ID associated with it. All log entries for the same input message have the same Queue ID. Log level is normally set to 10 in the configuration file. At this level, the following information is logged for each delivery:

- message-id=** If a message had a Message ID header line when it was input to sendmail, this is logged. sendmail can also be configured to add a Message ID header line if none is present. This ID uniquely identifies a message, and can be used to trace the progress of a message through mail relays.
- from =** The sender of the message and the message size are logged.
- to =** The recipient of the message. One message may have multiple recipients. sendmail logs a separate entry for each separate delivery attempt it makes, so multiple recipients on the same host may appear on the same line, but multiple recipients on different hosts will appear on different lines. The delivery status of the message (whether delivery succeeded, failed, or was deferred), the mailer, and the host used are logged.

Queued messages and SYSERRs are also logged.

Troubleshooting sendmail

At log level 11, for messages routed to [IPC] mailers, sendmail also logs the MX host to which the message was forwarded (if any) and the internet address to which the connection was made.

Configuring TFTP and BOOTP Servers

The Trivial File Transfer Protocol (TFTP) is a simple protocol used to read and write files to or from a remote system.

The Bootstrap Protocol (BOOTP) allows certain systems to discover network configuration information (such as an IP address and a subnet mask) and boot information automatically.

Together, TFTP and BOOTP allow a system to provide boot information for systems that support BOOTP, such as HP's 700/X terminal. These protocols are implemented on top of the Internet User Datagram protocol (UDP), so they can be used across networks that support UDP.

This chapter explains how to configure BOOTP and TFTP servers for your network either automatically from SAM, the on-line configuration interface, or manually from the shell prompt. Examples are provided to help you configure the servers. A troubleshooting section is also provided to help you recover from problems that may occur while using the BOOTP and TFTP servers.

Note

BOOTP is not supported over the X.25 link product or networks using the PPL (SLIP) product.

Chapter Overview

The topics covered in this chapter include:

- Key terms.
- How BOOTP works.
- Configuring TFTP and BOOTP servers using SAM.
- Configuring the TFTP server manually.
- Configuring the BOOTP server manually.
- Adding a BOOTP client manually.
- Command options for using TFTP.
- Troubleshooting BOOTP and TFTP servers.

Key Terms

Table 7-1. Key Terms for TFTP and BOOTP	
Term	Definition
ARP	The Address Resolution Protocol that dynamically binds an Internet address to a physical hardware address.
BOOTP	A protocol (named the Bootstrap Protocol) that allows a client to find its own IP address, the address of a TFTP server host, and the boot file to be loaded into memory and executed. BOOTP is fully described in RFC 951, RFC 1048, and RFC 1084.
bootpd	The name of the BOOTP server implementation.
bootpquery	A diagnostic tool which acts as a BOOTP client used to check the configuration of bootpd.
bootreply	The name of the BOOTP packet sent in response to a client's bootrequest. If a BOOTP server is configured to respond to a client's bootrequest, it supplies all of the known configuration parameters for that client within the bootreply.
bootrequest	The name of the BOOTP packet broadcast by BOOTP clients requesting boot information. The request usually contains only the hardware address and hardware type of the client.
TFTP	A protocol (named the Trivial File Transfer Protocol) that allows TFTP clients to read files from a remote server.
tftp	The name of the TFTP client implementation.
tftpd	The name of the TFTP server implementation.

How BOOTP Works

How BOOTP Works

The Bootstrap Protocol (BOOTP) allows a client system to discover its own IP address, the address of a server host, and the name of a file to be loaded into memory and executed.

The bootstrap operation happens in two phases. In the first phase, address determination and bootfile selection occur. This phase uses the BOOTP server, bootpd. After the address and file name information is obtained, control passes to the second phase of the bootstrap where a file transfer occurs. This phase uses the TFTP server, tftpd.

The first phase involves a packet exchange between the client and server as follows:

1. The client broadcasts a bootrequest packet. The bootrequest contains the client's hardware address and, if known, its IP address.
2. The server answers the bootrequest with a bootreply packet. In the case of clients who do not know their IP addresses, the server has a database relating a hardware address to an IP address.
3. The client IP address is then placed into a field in the bootreply. The bootreply may also contain a file name of a boot file which the client should load in the second phase of the protocol, the file transfer via TFTP. Other information possibly included in the bootreply are the client's subnet mask, the addresses of nameservers, and the addresses of gateways.

The server replies to clients that do not know their IP address in *one* of the following ways:

- *Option 1:* The server creates an ARP address cache entry using both the hardware address from the client's bootrequest and the IP address that it is placing in the bootreply. It then sends the bootreply to the client's IP address.

Configuring TFTP and BOOTP Servers Using SAM

- *Option 2*: The server sends the bootreply to the IP broadcast address on the appropriate interface.

The BOOTP server lets you choose either option of sending a bootreply. By default, bootpd will use *Option 1* and constructs the ARP cache entry. You can configure bootpd to reply to a client using *Option 2* (see *ba* tag in Table 7-2). However, since *Option 2* uses broadcasts, HP recommends using it only when you have the need to intercept the server's bootreply (for example, when you use the bootpquery diagnostic tool).

The second phase, the file transfer using TFTP, is optional. Some BOOTP clients utilize BOOTP only for IP address resolution and do not use TFTP.

Configuring TFTP and BOOTP Servers Using SAM

SAM stands for **System Administration Manager**, a menu-driven utility for performing system administration tasks, including configuration of networking software. When you configure bootpd (the BOOTP Server) in SAM, you may notice that tftpd (the TFTP Server) configuration is not included as an ARPA Services Configuration menu item. However, tftpd configurations are done “behind the scenes” when you configure bootpd.

Note You must be superuser to use SAM.

Procedure for Configuring tftpd and bootpd with SAM

The following steps take you to the ARPA Services Configuration menu where you can configure your system to be or not be a BOOTP server. At the ARPA Services Configuration menu, you can select forms that allow you to add, modify, view or remove a client's boot information.

Configuring TFTP and BOOTP Servers Using SAM

This example explains how to set up your local system to act as a BOOTP server and how to add new clients' boot information:

Note SAM configures `tftpd` as part of this task.

1. At the HP-UX prompt, type:

```
sam
```

Wait for SAM's main menu to appear.

2. Select the Networking/Communications menu item.
3. Select Device Connectivity menu item.
4. Select the Boot Protocol menu item.
5. Select the Add action.
6. Fill in the form according to its instructions. View the help screens for information about filling in the form.
7. Select apply to enter additional names of systems to be configured (use apply as a shortcut to remain in the add screen). Then, select OK to enter the change or Cancel to cancel when you are done with the screen.
8. Exit the Bootp screen by selecting exit from the List menu. From the Device Connectivity screen, select Exit SAM to exit from SAM.

Configuring the TFTP Server Manually

To manually configure the TFTP server, `tftpd`, you need to add entries to the files `/etc/services`, `/etc/inetd.conf`, and `/etc/passwd`. When you use SAM to do the configuration as specified in the section “Configuring TFTP and BOOTP Servers Using SAM,” entries are made to the appropriate files automatically. The following sections explain the manual method for configuring and verifying `tftpd`.

Note You must be superuser to configure the TFTP server.

Procedure for Configuring `tftpd`

Configuring `tftpd` on your system allows you to make files available to remote clients that support TFTP.

1. Add the TFTP protocol to `/etc/services`:

```
tftp 69/udp # ARPA Trivial file transfer protocol
```

2. Add the following entry to `/etc/inetd.conf`:

```
tftp dgram udp wait root /etc/tftpd tftpd
```

3. Reconfigure `/etc/inetd`:

```
/etc/inetd -c
```

4. Add the user `tftp` to `/etc/passwd`. The home directory of this user is the directory containing files available via TFTP. For example:

```
tftp:*:510:10:tftp directory: /usr/tftpdir: /bin/false
```

In this example, only the files in `/usr/tftpdir` are available via TFTP.

Configuring the TFTP Server Manually

5. Create the home directory for the user `tftp`. Make the directory owned by the user `tftp`, and ensure the directory gives the user `tftp` read, write, and execute permissions. For example:

```
$ mkdir /usr/tftpd
$ chown tftp /usr/tftpd
$ chgrp guest /usr/tftpd
$ chmod 700 /usr/tftpd
```

6. Place files that you want to make available for TFTP in the home directory of the user `tftp`. If you want to give remote systems permission to retrieve a file via TFTP, the file must not only exist in the home directory of the user `tftp`, but also be readable by the user `tftp`.

If you want to give remote systems permission to transmit a file to your system via TFTP, the file must already exist in the home directory of the user `tftp` and must be writeable by the user `tftp`.

Caution

Implementations of TFTP present a security problem since no user authorization mechanism exists within the protocol. HP's combination of the home directory and the file access permissions enforces strict security on what remote systems can read and write to your system via TFTP. Furthermore, implementing these features through the `/etc/passwd` entry forces the system administrator to make a conscious decision about what files are accessible with TFTP. This prevents the administrator from accidentally configuring the TFTP server in an unsecured manner.

Verify Your tftpd Installation

To verify your tftpd installation, create a file and use the tftp program to perform a file transfer:

1. Place a file in the TFTP home directory, `/usr/tftpd/`. Ensure that the file is readable by the user tftp. For example:

```
$ echo "Hello, this is a test." > /usr/tftpd/testfile
$ chown tftp /usr/tftpd/testfile
$ chmod 400 /usr/tftpd/testfile
```

2. Using the TFTP client, try to retrieve the file:

```
$ tftp localhost
tftp> get testfile
Received 24 bytes in 0.6 seconds
tftp> quit
```

If this step fails, see the section “Troubleshooting TFTP and BOOTP.”

3. Compare the files:

```
$ diff testfile /usr/tftpd/testfile
$
```

4. Remove the test file once you have verified the installation.

Configuring the BOOTP Server Manually

Configuring the BOOTP Server Manually

To manually configure the BOOTP server, `bootpd`, you need to add entries to the files `/etc/services` and `/etc/inetd.conf`. When you use SAM to do the configuration as specified in the section “Configuring TFTP and BOOTP Using SAM,” entries are made to the appropriate files automatically. The following sections explain the manual method for configuring and verifying `bootpd`.

Note You must be superuser to configure the BOOTP server.

Procedure for Configuring `bootpd`

Configuring `bootpd` sets up your local system to act as a server of boot information for remote clients.

1. Add the BOOTP server and client protocols to `/etc/services`:

```
bootps  67/udp          # Bootstrap protocol server
bootpc  68/udp          # Bootstrap protocol client
```

2. Add the following entry to `/etc/inetd.conf`:

```
bootps  dgram  udp  wait  root  /etc/bootpd  bootpd
```

3. Reconfigure `/etc/inetd`:

```
/etc/inetd -c
```

You are now ready to add information about individual BOOTP clients to the configuration file `/etc/bootptab`. This step is discussed in the section “Adding a BOOTP Client Manually.” If you wish to verify your `bootpd` installation, continue to the next section.

Verify Your bootpd Installation

The verification step only ensures that bootpd is started by inetd. To test whether you have correctly configured bootpd to handle boot requests, perform the following steps:

1. On the host where you configured bootpd, use bootpquery to send a boot request to the server. (See *bootpquery(1M)* in the *HP-UX Reference*.) For example, if you configured bootpd on a system named myhost, when you enter:

```
/etc/bootpquery 001122334455 -s myhost
```

A bootrequest is sent to the server, requesting a bootreply for the client with hardware address 001122334455. The BOOTP server will not respond to this request, so you will see the message:

```
bootpquery:Bootp servers not responding!
```

2. To see if the BOOTP server was started, on myhost enter the command:

```
ps -e | grep bootpd
```

You should see a bootpd entry.

3. If your system is configured to use syslogd (see *syslogd(1M)* in the *HP-UX Reference*), bootpd logs informative messages to the daemon facility. In the default configuration where syslogd sends daemon information messages to /usr/adm/syslog, you should see messages similar to the following.

```
Dec 13 13:32:22 myhost bootpd[13381]: reading "/etc/bootptab"  
Dec 13 13:32:22 myhost bootpd[13381]: read 0 entries from "/etc/bootptab"  
Dec 13 13:32:22 myhost bootpd[13381]: hardware address not found: 001122334455
```

These messages tell you that bootpd was able to read the empty configuration file /etc/bootptab and that it correctly rejected the test boot request that you sent with bootpquery.

Adding a BOOTP Client Manually

Having verified that bootpd is configured to start from inetd, you should add to the configuration file any BOOTP clients that you want to serve. The next section, “Adding a BOOTP Client Manually,” describes how to add client information and how to verify that the server will respond to the client.

Adding a BOOTP Client Manually

To allow a client to boot from your local system, you must add information about the client in your `/etc/bootptab` file. bootpd uses the `/etc/bootptab` file as the database for a client’s hardware address, hardware type, and configuration parameters (such as the IP address, boot file, and subnet mask).

Collect Information About the Client

You need to collect the following information about the client so you can make an entry for the client in the `/etc/bootptab` file:

- Host name—name of the client’s system.
- Hardware type—type of network interface (IEEE 802.3 or Ethernet).
- Link level address—client’s hardware address.
- IP address—client’s assigned address.
- Subnet mask—the mask (IP address) identifying the network where the client resides.
- Boot file—file name that the client will retrieve via tftp.

Understanding Boot File Configurations

A configuration entry is a single line with the following format:

hostname:tag=value:tag=value:...tag=value

Adding a BOOTP Client Manually

Each client parameter is defined with a two-character case-sensitive tag followed by the equals sign and the tag's client-specific *value*. A value consists of a hardware address, an IP address, a list of IP addresses, a numeric value, or a string of characters representing a host name or a file name. A colon separates each *tag=value* parameter definition. `bootpd` uses these tags and values to recognize a client's bootrequest and to supply parameters in the bootreply to the client.

For example, parameters for the BOOTP client `hp-xterm` are represented with the following entry in `/etc/bootptab`:

```
hp-xterm:ha=080009030166:ht=ether:ip=15.19.8.2:sm=255.255.248.0:bf=/hp-xterm
```

This line tells `bootpd` the following information:

- `hp-xterm`'s hardware address is 080009030166.
- `hp-xterm` has an Ethernet network interface.
- `hp-xterm`'s IP address is 15.19.8.2.
- `hp-xterm`'s subnet mask is 255.255.248.0.
- `hp-xterm` should get the file `/hp-xterm` via TFTP. (Since `tftpd "chroot's"` to the `tftp` user's home directory `/usr/tftpd/r`, the file `/hp-xterm` is actually relative to that directory.)

Some tags can be used as "boolean" tags where no equals sign or value is specified. Such tags represent parameters that `bootpd` can determine automatically. For example, the boot file size tag, `bs`, defines the number of 512K blocks in the client's boot file (an optional parameter in the bootreply). When a tag is used as a boolean tag, `bootpd` automatically determines the number of blocks in the client's boot file and uses this value in the bootreply.

You may enter tags in any order, with the following exceptions:

- The client's *hostname* must be the first field of an entry.
- The *hardware type* must precede the *hardware address*.

Adding a BOOTP Client Manually

Other points to know when adding an entry in `/etc/bootptab` include:

- IP addresses listed for a single tag must be separated by a space.
- A single client entry can be extended over multiple lines if you use a backslash (`\`) at the end of each line.
- Blank lines and lines that begin with the pound sign (`#`) are ignored.
- A legal host name consists of a letter followed by any number of letters, digits, periods, or hyphens.

Parameter Tags and Descriptions

Table 7-2 shows you the most commonly tags used to define the client parameters. For more information on these tags and the other tags available, see *bootpd(1M)* in the *HP-UX Reference*.

Tag	Description
<code>ba</code>	Forces bootpd to broadcast the bootreply to the client.
<code>bf</code>	Boot file name that the client down loads via TFTP.
<code>bs</code>	Boot file size in 512-byte blocks
<code>ds</code>	IP address(es) of the RFC 1034 domain name server(s).
<code>gw</code>	IP address(es) of the gateway(s) for the client's subnet.
<code>ha</code>	Client's hardware address. This tag must be preceded by the <code>ht</code> tag.
<code>hd</code>	Boot file's home directory. The default home directory is <code>/</code> .
<code>hn</code>	Send the host name in the bootreply. This tag is strictly boolean; it does not need an equals sign or an assigned value.
<code>ht</code>	Client's hardware type (see Assigned Numbers RFC). May be assigned the value <code>ieee</code> or <code>ether</code> .

Table 7-2. bootpd Configuration Options	
Tag	Description
ip	BOOTP Client's IP address. This tag only takes one IP address.
lg	IP address(es) of the UDP log server(s).
sm	The subnet mask for the client's network.
tc	Table continuation. Allows several client entries to share common tag values.
T <i>nnn</i>	Generic tag where <i>nnn</i> is an RFC 1048 vendor field tag number. Allows bootpd to take immediate advantage of future extensions to RFC 1048. These tags can be used for vendor specific extensions to the BOOTP tags.
to	Client's time zone offset in seconds from UTC.

Examples of Adding BOOTP Clients

This section shows two examples of adding entries to the `/etc/bootptab` file. In the first example, you will see how to configure a server for an HP 700/X terminal. In the second example, you will see how to configure the BOOTP server for a gateway.

Example 1: Adding an HP 700/X Terminal as a Client

As superuser, Sarah needs to configure the bootpd server on her system so that it can boot an HP 700/X terminal. To accomplish this, Sarah will perform the following steps:

1. Collect necessary information about the 700/X terminal:
 - Link level address.
 - IP address.
 - Host name.
 - Subnet mask.

Adding a BOOTP Client Manually

2. Collect optional information about the 700/X terminal:
 - Gateway address.
 - Name server address.
 - Boot file name.
3. Add the appropriate entry to `/etc/bootptab`.
4. Verify the entry with `/etc/bootpquery`.

Collect Necessary and Optional Information about the 700/X Terminal. Sarah collected the following information for the HP 700/X terminal:

- The hardware or link level address is 080009030165.
- The host name assigned to the 700/X is `xterm01`.
- The IP address assigned to the 700/X is 15.19.8.37.
- The subnet mask for the network to which the 700/X is connected is 255.255.248.0.
- The IP address of the gateway that the 700/X must use is 15.19.8.1.
- The IP address of the network's name server is 15.19.8.119.
- The boot file that the 700/X must load is named `xterminal`. The file is already located in the `tftp` directory `/usr/tftpdir`.

Add the Appropriate Entry to `/etc/bootptab`. The information that Sarah gathered is represented in `/etc/bootptab` with the following tag assignments:

- The link level address is represented with the tags `ht=ether` and `ha=080009030165`.
- The host name is represented by `xterm01` in the first field and by the `hn` tag that places the host name in the `bootreply`.
- The IP address is represented with the tag `ip=15.19.8.37`.
- The subnet mask is represented with the tag `sm=255.255.248.0`.

Adding a BOOTP Client Manually

- The gateway address is represented with the tag `gw=15.19.8.1`.
- The name server address is represented with the tag `ds=15.19.8.119`.
- The boot file is represented with the tag `bf=/xterminal`.

Sarah now adds the following lines to `/etc/bootptab` (she uses the line continuation character “\” so that the entry is readable):

```
xterm01:    hn: ht=ether: ha=080009030165: \  
            ip=15.19.8.37: sm=255.255.248.0: \  
            gw=15.19.8.1: ds=15.19.8.119: bf=/xterminal
```

Verify the Entry with `/etc/bootpquery`. To verify the new `/etc/bootptab` entry with `/etc/bootpquery`, Sarah must force `bootpd` to broadcast responses to boot requests for `xterm01` by adding the `ba` (broadcast address) tag to the entry (so that the boot response is not sent directly to `xterm01`). This allows the diagnostic tool `/etc/bootpquery` to intercept the `bootreply` packets for `xterm01`. Once she has verified how the server will respond to boot requests for `xterm01`, she will remove the `ba` tag (so that boot replies are sent directly to `xterm01`).

First, she adds the `ba` tag to the entry in `/etc/bootptab` as shown:

```
xterm01:    hn: ht=ether: ha=080009030165: \  
            ip=15.19.8.37: sm=255.255.248.0: \  
            gw=15.19.8.1: ds=15.19.8.119: bf=/xterminal: ba
```

Next, as superuser, Sarah uses `/etc/bootpquery` to see how `bootpd` on her system, `hpserver`, responds to a request from `xterm01`. When she enters the following command:

```
/etc/bootpquery 080009030165 -s hpserver
```

She sees the output:

```
Received BOOTREPLY from hpserver.hp.com (15.19.8.119)
```

```
Hardware Address:    08:00:09:03:01:65  
Hardware Type:      ethernet
```

Adding a BOOTP Client Manually

```
IP Address:      15.19.8.37
Boot file:      /xterminal
```

RFC 1048 Vendor Information:

```
Subnet Mask:    255.255.248.0
Gateway:       15.19.8.1
Domain Name Server: 15.19.8.119
Host Name:     xterm01.hp.com
```

This shows that `hpservers.hp.com` responded with information that corresponds to the entry Sarah added to `/etc/bootptab`. Now she can remove the `ba` tag from the `/etc/bootptab` file. The following example shows the `/etc/bootptab` file with the `ba` tag removed:

```
xterm01:  hn: ht=ether: ha=080009030165: \
          ip=15.19.8.37: sm=255.255.248.0: \
          gw=15.19.8.1: ds=15.19.8.119: bf=/xterminal
```

Example 2: Adding a Gateway as a Client

In this example, Sarah will add entries to `/etc/bootptab` so that a gateway can use BOOTP to configure both of its interfaces. Sarah will again perform the following tasks:

1. Collect necessary information about the gateway:

- Link level addresses of each LAN interface.
- IP addresses assigned to each interface.
- Host name.
- Subnet mask for each interface.

2. Collect optional information:

- Name server address.
- Log server address.

Adding a BOOTP Client Manually

3. Add the appropriate entries to `/etc/bootptab`. Since the gateway has two LAN interfaces, it requires two entries in `/etc/bootptab`.
4. Verify the entries with `/etc/bootpquery`.

Collect Necessary Information about the Gateway. For the first interface of the gateway, Sarah collected the following information:

- The hardware or link level address is 02608CEE018E.
- The IP address assigned to this interface is 15.19.15.91.
- The subnet mask for the network to which this interface is connected is 255.255.248.0.

For the second interface of the gateway:

- The hardware or link level address is 02608CEE0144.
- The IP address assigned to this interface is 15.19.8.248.
- The subnet mask for the network to which this interface is connected is 255.255.248.0.

Sarah has also decided that the name server and log server for the gateway will be `hpserver.hp.com`, which has the IP address 15.19.15.119.

Add the Appropriate Entry to `/etc/bootptab`. The configuration file requires two entries in `/etc/bootptab`, one for each interface in the gateway. The information for the first interface is represented in `/etc/bootptab` with the following tag assignments:

- The link level address is represented with the tags `ht=ether` and `ha=02608CEE018E`.
- The host name is represented by `hp-gateway` in the first field. Since this particular gateway will use the name server to find its host name and since no host name field is necessary in the bootreply, the `hn` tag is not needed.
- The IP address is represented with the tag `ip=15.19.15.91`.

Adding a BOOTP Client Manually

- The subnet mask is represented with the tag `sm=255.255.248.0`.
- The name server address is represented with the tag `ds=15.19.15.119`.
- The log server address is represented with the tag `lg=15.19.15.119`.

The information for the second interface is represented in `/etc/bootptab` with the following tag assignments:

- The link level address is represented with the tags `ht=ether` and `ha=02608CEE0144`.
- The host name is represented by `hp-gateway` in the first field.
- The IP address is represented with the tag `ip=15.19.8.248`.
- The subnet mask is represented with the tag `sm=255.255.248.0`.
- The name server address is represented with the tag `ds=15.19.15.119`.
- The log server address is represented with the tag `lg=15.19.15.119`.

Sarah now adds the following lines to `/etc/bootptab`:

```
hp-gateway.hp.com: ht=ether: ha=02608CEE018E: \  
                    ip=15.19.15.91: sm=255.255.248.0:\  
                    ds=15.19.15.119: lg=15.19.15.119  
hp-gateway.hp.com: ht=ether: ha=02608CEE0144: \  
                    ip=15.19.8.248: sm=255.255.248.0:\  
                    ds=15.19.15.119: lg=15.19.15.119
```

Verify the Entry with `/etc/bootpquery`. To verify the new `/etc/bootptab` entry with `/etc/bootpquery`, Sarah must force the bootpd to broadcast responses to boot requests for the `hp-gateway` interfaces by adding the `ba` tag to each `hp-gateway` entry in `/etc/bootptab`:

```
hp-gateway.hp.com: ht=ether: ha=02608CEE018E: \  
                    ip=15.19.15.91: sm=255.255.248.0:\  
                    ds=15.19.15.119: lg=15.19.15.119: ba  
hp-gateway.hp.com: ht=ether: ha=02608CEE0144: \  
                    ip=15.19.8.248: sm=255.255.248.0:\  
                    ds=15.19.15.119: lg=15.19.15.119: ba
```

Adding a BOOTP Client Manually

```
ip=15.19.8.248: sm=255.255.248.0:\  
ds=15.19.15.119: lg=15.19.15.119: ba
```

As superuser, Sarah issues a request for each interface with `/etc/bootpquery`.
When she enters the following command:

```
/etc/bootpquery 02608CEE018E -s hpserver
```

She sees the output:

```
Received BOOTREPLY from hpserver.hp.com (15.19.15.119)
```

```
Hardware Address:    02:60:8c:ee:01:8e  
Hardware Type:      ethernet  
IP Address:         15.19.15.91  
Boot file:          (None)
```

```
RFC 1048 Vendor Information:
```

```
Subnet Mask:        255.255.248.0  
Domain Name Server: 15.19.15.119  
Log Server:         15.19.15.119
```

Next, Sarah enters the command:

```
/etc/bootpquery 02608CEE0144 -s hpserver
```

And sees the output:

```
Received BOOTREPLY from hpserver.hp.com (15.19.15.119)
```

```
Hardware Address:    02:60:8c:ee:01:44  
Hardware Type:      ethernet  
IP Address:         15.19.8.248  
Boot file:          (None)
```

```
RFC 1048 Vendor Information:
```

```
Subnet Mask:        255.255.248.0  
Domain Name Server: 15.19.15.119  
Log Server:         15.19.15.119
```

Command Options for Using TFTP

This shows that `hpserver.hp.com` responded with information that corresponds to the entry Sarah added to `/etc/bootptab`. She can now remove the `ba` tags from each `hp-gateway` entry in `/etc/bootptab`. The following example shows the `/etc/bootptab` file with the `ba` tag removed:

```
hp-gateway.hp.com: ht=ether: ha=02608CEE018E: \  
                    ip=15.19.15.91: sm=255.255.248.0:\   
                    ds=15.19.15.119: lg=15.19.15.119  
  
hp-gateway.hp.com: ht=ether: ha=080009030165: \  
                    ip=15.19.8.248: sm=255.255.248.0:\   
                    ds=15.19.15.119: lg=15.19.15.119
```

Command Options for Using TFTP

ARPA Services includes a TFTP client implementation, `/usr/bin/tftp`. You can use this client to verify that your TFTP server is working correctly.

Table 7-3 describes the most common `tftp` commands you can use when transferring files. For information on the other `tftp` options, refer to *tftp(1M)* in the *HP-UX Reference*.

Command Options for Using TFTP

Option	Description
ascii	Sets the TFTP file transfer type to ascii. This is the default type.
binary	Sets the TFTP file transfer type to binary.
get <i>remote-file</i> [<i>local-file</i>]	Copy <i>remote-file</i> to <i>local-file</i> . If <i>local-file</i> is unspecified, tftpd uses the specified <i>remote-file</i> name as the <i>local-file</i> name. If <i>local-file</i> is specified as "-", the remote file is copied to standard output.
put <i>local-file</i> [<i>remote-file</i>]	Copy <i>local-file</i> to <i>remote-file</i> . If <i>remote-file</i> is unspecified, tftpd assigns the <i>local-file</i> name to the <i>remote-file</i> name.
verbose	When verbose is on, tftpd displays responses from the server host. When verbose is on and a file transfer completes, tftpd reports information about the efficiency of the transfer. Enter the verbose command at the tftpd> prompt to turn the verbose setting on or off.

Troubleshooting BOOTP and TFTP Servers

This section outlines techniques that can help you diagnose and correct common problems with the BOOTP and TFTP servers.

When a BOOTP client experiences a boot failure, you need to decide whether the failure occurred with `bootpd` or `tftpd`. Since BOOTP involves two phases (see the section "How BOOTP Works"), you need to determine which phase of the protocol is failing as follows:

- If your BOOTP client system cannot determine simple configuration information, such as its IP address, host name, or boot file name, then it is experiencing a `bootpd` failure.
- If your BOOTP client system can determine an IP address but cannot load its boot file, then it is experiencing a `tftpd` failure.

Helpful Configuration Changes

To make troubleshooting easier, configure your system as follows:

- Ensure `syslogd` is configured to log daemon information messages to the file `/usr/adm/syslog`. To check this configuration, make sure `/etc/syslog.conf` includes *one* of the following lines:

```
*.info      /usr/adm/syslog
```

OR

```
daemon.info /usr/adm/syslog
```

- Configure `bootpd` to start with debug logging set to level 2. This logging level causes `bootpd` to log useful debugging messages about how it is replying to BOOTP clients. To set the debug log level:
 - Add the `-d 2` option to the `bootpd` line in `/etc/inetd.conf`:

```
bootps dgram udp wait root /etc/bootpd bootpd -d 2
```
 - Reconfigure `inetd` with the following command:

Troubleshooting BOOTP and TFTP Servers

```
inetd -c
```

- Kill any bootpd daemon that is still running on your system. For example:

```
$ ps -e | grep bootpd
429 ? 0:00 bootpd
$ kill 429
```

Common bootpd Problems

Some of the problems in this section have more than one *possible cause* listed under them.

Problem #1: The BOOTP client does not seem to receive responses from the BOOTP server.

1a) Possible Cause The server is not started or does not receive a bootrequest from the client.

1a) Symptom The system log file `/usr/adm/sys log` does not contain any log messages from `/etc/bootpd` showing that the server started. A `ps -ef` listing does not show a running `/etc/bootpd`.

1a) Hints

- Ensure that `/etc/inetd.conf` is configured correctly as documented earlier in this chapter.
- Ensure that you have reconfigured `inetd` with the command `inetd -c`. Check `inetd`'s logging in `/usr/adm/sys log` to ensure `inetd` is configured to start `bootpd`.
- As documented in "Configuring the BOOTP Server Manually," verify that the server will start by using the `bootpquery` command.
- Check whether the client is on the same network as the BOOTP server. If the client is not on the same network, ensure that intervening gateways are configured to relay bootrequest broadcasts.

Troubleshooting BOOTP and TFTP Servers

1b) Possible Cause The BOOTP server does not have any information associated with the clients hardware address.

1b) Symptom The system log `/usr/adm/syslog` contains *one* of the following messages:

hardware address not found: *hardware_address*

IP address not found: *ip_address*

This means that bootpd does not have an entry in `/etc/bootptab` for this client's hardware address or IP address.

- 1b) Hints**
- Check the system log for any indication of syntax errors for the client's configuration entry. Correct the entry in `/etc/bootptab` and reboot the BOOTP client.
 - Ensure that the hardware address you specified for the `ha=` tag matches the hardware address that `/etc/bootpd` said it could not find. Correct the tag and reboot the BOOTP client.
 - Ensure the hardware type tag `ht=` has the correct value for the client. For example, if you have specified `ether` but the client is reporting `ieee` in its bootrequest, bootpd will reject the request. Correct the tag and reboot the BOOTP client.

1c) Possible Cause bootpd cannot locate the boot file that the client requested in its bootrequest.

1c) Symptom The system log `/usr/adm/syslog` contains a message that looks like:

requested file not found: *file_name*

This means that the client specified *file_name* as the boot file in its bootrequest, but bootpd could not find the file in the `tftp` directory.

Troubleshooting BOOTP and TFTP Servers

- 1c) Hints**
- Ensure that you have configured `tftpd` with the entry in `/etc/passwd` for the user `tftp`.
 - Ensure that the requested file is present in the `tftp` directory, which is usually `/usr/tftpdir`. If it is not, place the file in the `tftp` directory and reboot the BOOTP client.
 - If the requested file exists in the `tftp` directory, be sure it is readable by the TFTP server. (See the section "Common tftpd Problems.")

1d) Possible Cause The BOOTP server cannot send a reply to the BOOTP client.

1d) Symptom The system log `/usr/adm/sys log` contains the message:

cannot route reply to *client's_IP_address*

This means that the IP address you have specified for the client is one which the server's system cannot reach directly.

- 1d) Hints**
- Ensure you have specified the correct IP address for the client in `/etc/bootptab`. Correct the entry and reboot the BOOTP client.
 - If the server is to reply directly to the client, it must reside on the same network or subnet as the client. Ensure the IP address you have chosen for the client is a valid IP address for the server's network. If the client resides on another network, ensure intervening gateways are configured to relay bootrequests (in which case, `bootpd` replies through the gateways).

1e) Possible Cause There are typos or other errors in the configuration file `/etc/bootptab`.

1e) Symptom The system log contains *one or more* of the following error messages:

duplicate hardware address: *link address*

bad host name: *hostname*

syntax error in entry for host *hostname*

Troubleshooting BOOTP and TFTP Servers

unknown symbol in entry for host *hostname*
bad IP address for host *hostname*
bad subnet mask for host *hostname*
bad time offset for host *hostname*
bad vendor magic cookie for host *hostname*
bad reply broadcast address for host *hostname*

Any of these error messages means there are errors in the configuration file entry for the client.

1e) Hints:

- Refer to the "Error Logging" section for an explanation of the error message. Correct the appropriate field for the entry in `/etc/bootptab` and reboot the BOOTP client.
- Use `bootpquery` to send a bootrequest to `/etc/bootpd` for the client whose entry you have corrected. Ensure the server is replying to the client. You can do this by performing *one* of the following tasks:

- Enable a broadcast bootreply for the client by adding the `ba` tag to the client's `/etc/bootptab` entry. Use `bootpquery` to emulate the client's bootrequest with the command:

```
bootpquery client's_hardware_address -s server_name
```

`bootpquery` prints the reply it receives from the server, which allows you to examine the information supplied to the client. Remove the `ba` tag from the configuration entry once you've verified the correctness of the bootreply.

- Use `bootpquery` to emulate the client's bootrequest with the command:

```
bootpquery client's_hardware_address -s server_name
```

Then check the system log `/usr/adm/syslog` to see if the server replies. At debug level 2 (see "Helpful Configuration Changes" section), `bootpd` logs the following sequence of messages when it

Troubleshooting BOOTP and TFTP Servers

responds to a bootrequest:

```
request from hardware address link_address
found ip_address hostname
vendor magic field is magic_cookie
sending RFC1048-style reply
```

- Problem #2:** **The BOOTP client received a reply from the BOOTP server, but some of the configuration information is missing or is incorrect.**
- 2a) Possible Cause** You have entered incorrect information for the client's configuration entry in `/etc/bootptab`.
- 2a) Symptom** The client is configured with invalid information. For example, it has an incorrect subnet mask.
- 2a) Hints** ■ Review the entry for the client in `/etc/bootptab` and ensure all the specified information is correct.
- Enable a broadcast bootreply by using the `ba` tag in the client's `/etc/bootptab` entry. Use `bootpquery` to emulate the client's bootrequest with the command:
- ```
bootpquery client's_hardware_address -s server_name
```
- `bootpquery` prints the reply it receives from the server, which allows you to examine the information supplied to the client. Remove the `ba` tag from the configuration entry once you've verified the correctness of the bootreply.
- Reboot the BOOTP client and verify the correctness of client configuration.

## Troubleshooting BOOTP and TFTP Servers

- 2b) **Possible Cause** You have specified too many RFC-1048 options for the client's configuration entry in `/etc/bootptab`.
- 2b) **Symptom** The client did not appear to receive configuration information for one or more of the tags listed under "Hints."
- 2b) **Hints**
- The BOOTP protocol allows only 64 bytes of "vendor extension" information. The following tags are among the tags included in the RFC 1048 vendor information:

bs = boot file size  
ds = domain nameserver addresses  
gw = gateway addresses  
hn = host name  
lg = log server addresses  
sm = subnet mask  
to = time offset  
*nnnn* = generic information

When such extended information is included in the bootreply, bootpd must also add a 4 byte vendor magic cookie to the bootreply, a 1 byte tag indicating the end of the vendor information, and a 1 or 2 byte tag for each field (depending on the format of the field) along with the value of the tag itself. The total size of the extended information you list for a client must not exceed 64 bytes.

- Ensure the configuration contains only the necessary information to boot the client. Check the documentation for the BOOTP client to find out which tags are necessary for configuration and which tags are supported.

For example, if the client only supports one nameserver address, there is no need to list three nameserver addresses with the `ds` tag. If the client does not support configuring its host name with the `hn` tag, there is no reason to include it.

## Troubleshooting BOOTP and TFTP Servers

To view the information that `bootpd` places in the bootreply, enable a broadcast bootreply by adding the `ba` tag to the client's `/etc/bootptab` entry. Use `bootpquery` to emulate the client's bootrequest with the command:

```
bootpquery client's_hardware_address -s server_name
```

`bootpquery` prints the reply it receives from the server, which allows you to examine the vendor information supplied to the client. Remove the `ba` tag from the configuration entry once you've verified the correctness of the bootreply.

### Common tftpd Problems

Some of the problems in this section have more than one *possible cause* listed under them.

**Problem #1:**            **The file transfer "timed out".**

**1a) Possible Cause**    The TFTP server, `tftpd`, is not starting.

**1a) Symptom**            `inetd` connection logging is enabled (with the `inetd -l` command) but did not show any connection to the TFTP server.

- 1a) Hints**
- Ensure `/etc/inetd.conf` is configured correctly as documented earlier in this chapter.
  - Ensure you have reconfigured `inetd` with the command `inetd -c`.
  - As documented in "Configuring the BOOTP Server Manually," verify the server is working by using `tftp` to transfer a small file. It might be helpful to try the transfer from another node on your network rather than from the server node itself.
  - If the server still fails to start when the client attempts the file transfer, then you probably have a connectivity problem. For information about troubleshooting connectivity, refer to the link product manual (e.g., HP 9000 LAN Link installation) or the



## Troubleshooting BOOTP and TFTP Servers

BOOTP client manual (for example, HP 700/X documentation).

**1b) Possible Cause** The TFTP server, `tftpd`, is exiting prematurely.

**1b) Symptom** The system log contains *one* of the following messages:

User tftp unknown

*system call: error*

- 1b) Hints**
- The User tftp unknown message means the password database entry for the user tftp is either missing or is incorrect. Verify the entry exists and is correct, then try the transfer again.
  - If `tftpd` experiences a system call failure that causes it to exit, it will log the name of the system call and the reason for the system call failure. For more information about the reason why it failed, refer to the system call in the *HP-UX Reference*.

**Problem #2:** **The transfer failed with the message "File not found" or "No such file or directory."**

**2) Symptom** The client displays *one* of these messages indicating that the file does not exist. It may also display a message indicating "TFTP Error Code 1."

- 2) Hints**
- The message means that the file the client is attempting to read from or write to the server did not exist within the home directory of the user tftp.
  - Ensure the full path name that the client is requesting from the server exists within the tftp directory.

For example, if the tftp directory is `/usr/tftpd` and the TFTP client is requesting the file `/usr/lib/X11/700X/C2300A`, the file must exist as:

## Troubleshooting BOOTP and TFTP Servers

/usr/tftpdir/usr/lib/X11/700X/C2300A

**Problem #3:**           **The transfer failed with the message "Access violation" or "Permission denied."**

**3) Cause**               tftpd does not have permission to read the file.

**3) Symptom**           The client displays *one* of the messages indicating the permission problem. It may also display a message indicating "TFTP Error Code 2".

**3) Hints**               ■ If the transfer is a get operation where the client is attempting to read the file from the server, then the server does not have read permissions on the file that it is trying to send. Ensure the file that the client is reading has read permissions for the user tftp.

For example, if the client was attempting to read the file named `xterminal`, `/usr/tftpdir/xterminal` should be mode `0400` and owned by the user `tftp`:

```
$ ll /usr/tftpdir/xterminal
-r----- 1 tftp guest 438 May 10 1989 xterminal
```

■ If the transfer is a put operation (which is *not* something a BOOTP client will be doing as part of the BOOTP protocol), then this message means that the file did not have sufficient write permissions for the server to write to the file. If the server is to receive a file, it must already exist and be writeable by the user `tftp`.

For example, if a `tftp` client is sending the file named `fontlist`, the file must exist as `/usr/tftpdir/fontlist` and must be mode `0600` and owned by `tftp`:

```
$ ll /usr/tftpdir/fontlist
-rw----- 1 tftp guest 0 May 10 1989 fontlist
```

## Troubleshooting BOOTP and TFTP Servers

### Error Logging

This section explains the error messages that bootpd logs via syslogd. The three levels of error logging documented in this section include:

- Information log level.
- Notice log level.
- Error log level.

The bootpd debug level must be set for some of these messages to be logged. Set the debug level using the `-d` option to bootpd.

#### Information Log Level

The following messages are logged at the syslogd information log level.

|                |                                                                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Message</b> | exiting after <i>time</i> minutes of inactivity                                                                                                         |
| <b>Cause</b>   | If bootpd hasn't received a bootrequest within <i>time</i> minutes (the timeout set with the <code>-t</code> option), it issues this message and exits. |

---

|                |                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------|
| <b>Message</b> | reading <i>configuration_file</i><br>reading new <i>configuration_file</i>                              |
| <b>Cause</b>   | bootpd is reading or rereading configuration information from the indicated <i>configuration_file</i> . |

---

|                |                                                                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Message</b> | read <i>number</i> entries from <i>configuration_file</i>                                                                                                     |
| <b>Cause</b>   | Shows that bootpd successfully read <i>number</i> configuration entries, including table continuation entries, from the indicated <i>configuration_file</i> . |

---

## Troubleshooting BOOTP and TFTP Servers

|                |                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Message</b> | request from hardware address <i>hardware_address</i>                                                                                                                             |
| <b>Cause</b>   | bootpd received a bootrequest from a client with the indicated <i>hardware_address</i> . This message is logged at debug level 1.                                                 |
| <hr/>          |                                                                                                                                                                                   |
| <b>Message</b> | request from IP addr <i>ip_address</i>                                                                                                                                            |
| <b>Cause</b>   | bootpd received a bootrequest from a client with the indicated <i>ip_address</i> . This message is logged at debug level 1.                                                       |
| <hr/>          |                                                                                                                                                                                   |
| <b>Message</b> | found <i>ip_address hostname</i>                                                                                                                                                  |
| <b>Cause</b>   | bootpd located information for the specified client in its configuration database. This message is logged at debug level 1.                                                       |
| <hr/>          |                                                                                                                                                                                   |
| <b>Message</b> | broadcasting reply on <i>ip_address</i>                                                                                                                                           |
| <b>Cause</b>   | Shows the broadcast address that bootpd uses to reply to a client whose configuration entry has the ba flag. This message is logged at debug level 2.                             |
| <hr/>          |                                                                                                                                                                                   |
| <b>Message</b> | vendor magic field is <i>magic_cookie</i><br>sending CMU-style reply<br>sending RFC1048-style reply                                                                               |
| <b>Cause</b>   | Shows which vendor magic cookie was sent in the client's bootrequest and the corresponding vendor magic cookie used in the bootreply. These messages are logged at debug level 2. |
| <hr/>          |                                                                                                                                                                                   |
| <b>Message</b> | bootptab mtime is <i>time</i>                                                                                                                                                     |
| <b>Cause</b>   | bootpd uses the indicated modification time to determine if the configuration file has been modified and should be reread. This message is logged at debug level 3.               |
| <hr/>          |                                                                                                                                                                                   |

## Troubleshooting BOOTP and TFTP Servers

### Notice Log Level

There may be cases where bootpd receives a bootrequest, but does not send a bootreply. The reason is given in one of the following messages and logged at the notice log level.

**Message** hardware address not found: *hardware\_address*

**Cause** bootpd could not find a configuration entry for the client with the indicated *hardware\_address*. If bootpd should know about the client that is booting, ensure that you have correctly specified the client's hardware address in the configuration file.

---

**Message** IP address not found: *ip\_address*

**Cause** bootpd could not find a configuration entry for the client with the indicated *ip\_address*. If bootpd should know about the client that is booting, ensure that you have correctly specified the client's IP address in the appropriate configuration file entry.

---

**Message** requested file not found: *file\_name*

**Cause** The client requested the boot file *file\_name*, but bootpd could not locate it. Ensure that the boot file the client is requesting is located in the tftp directory on the server system.

---

**Message** cannot route reply to *ip\_address*

**Cause** The IP address to which bootpd must send the bootreply is for a client or gateway that is not on a directly connected network. Ensure that you have specified a valid IP address for the client or gateway.

---

## Troubleshooting BOOTP and TFTP Servers

### Error Log Level

The following errors indicate problems with the configuration file. They are logged at the error log level. If you see any of these messages, you should correct the indicated configuration entry in `/etc/bootptab` and try to reboot the BOOTP client.

**Message** duplicate hardware address: *hardware\_address*

**Cause** More than one configuration entry was specified for the client with the indicated *hardware\_address*.

---

**Message** bad hostname: *hostname*

**Cause** The name given in the `hostname` field was not a valid host name. A valid host name consists a letter followed by any number of letters, digits, periods, or hyphens.

---

**Message** syntax error in entry for host *hostname*

**Cause** The configuration entry for the indicated host *hostname* is incorrectly formatted.

---

**Message** unknown symbol in entry for host *hostname*

**Cause** The configuration entry contains an unknown tag or invalid character.

---

**Message** bad IP address for host *hostname*

**Cause** One of the IP addresses listed for the `ip` tag or any tag requiring a list of IP addresses is incorrectly formatted in the configuration file entry for *hostname*.

---

## Troubleshooting BOOTP and TFTP Servers

**Message** bad subnet mask for host *hostname*

**Cause** The value for the subnet mask tag *sm* was incorrectly formatted in the configuration file entry for *hostname*.

---

**Message** bad time offset for host *hostname*

**Cause** The value for the *to=* tag was not a valid number.

---

**Message** bad vendor magic cookie for host *hostname*

**Cause** The vendor magic cookie was incorrectly formatted.

---

**Message** bad reply broadcast address for host *hostname*

**Cause** The address given for the *ba* tag was invalid or incorrectly formatted.

---

**Message** can't find *tc=label*

**Cause** bootpd could not find a table continuation configuration entry with the host field *label*.

---

## Troubleshooting ARPA Services

---

Troubleshooting data communications problems may require you to investigate many hardware and software components. Some problems can be quickly identified and resolved. These include invalid software installation, version incompatibilities, insufficient HP-UX resources, corrupt configuration shell scripts, and programming or command errors. Other problems require more investigation.

Once identified, most problems can be resolved by the programmer, user, or node manager, using the suggestions in this chapter or the error messages documented in the link installation manuals. However, there may be problems that you should report to your Hewlett-Packard support contact. This chapter includes guidelines for submitting an HP Service Request (SR).

---

### Chapter Overview

The strategy and tools to use while investigating the software and hardware components are provided in this chapter.

This chapter contains the following sections:

- Characterizing the Problem.
- Diagnostic Tools Summary.
- Diagnosing Repeater and Gateway Problems.



## Characterizing the Problem

- Flowchart Format.
- Troubleshooting the ARPA/Berkeley Services.
- Reporting Problems to Your Hewlett-Packard Support Contact.

Troubleshooting information for DDFA is documented in the *DTC Device File Access Utilities manual*.

---

## Characterizing the Problem

It is important to ask questions when you are trying to characterize a problem. Start with global questions and gradually get more specific. Depending on the response, ask another series of questions until you have enough information to understand exactly what happened. Key questions to ask are:

1. Does the problem seem isolated to one user or program? Can the problem be reproduced? Did the problem occur under any of the following circumstances:
  - When running a program?
  - When issuing a command?
  - When using a nodal management utility?
  - When transmitting data?
2. Does the problem affect all users? The entire node? Has anything changed recently? The possibilities are:
  - New software and hardware installation?
  - Same hardware but changes to the software. Has the configuration file been modified? Has the HP-UX configuration been changed?
  - Same software but changes to the hardware. Do you suspect hardware or software?

## Diagnostic Tools Summary

It is often difficult to determine whether the problem is hardware or software related. The symptoms of the problem which mean you should suspect the hardware are:

- Intermittent errors.
- Network-wide problems after no change in software.
- Link level errors, from logging subsystem, logged to the console.
- Data corruption—link level trace that shows that data is sent without error but is corrupt or lost at the receiver.
- Red light on the LAN card is lit, or yellow light on the X.25/800 card is lit.

These are symptoms that would lead you to suspect the software:

- Network services errors returned to users or programs.
- Data corruption.
- Logging messages at the console.

Knowing what has recently changed on your network may also indicate whether the problem is software or hardware related.

---

## Diagnostic Tools Summary

The most frequently used diagnostic tools are listed below. These tools are documented in the link installation manuals.

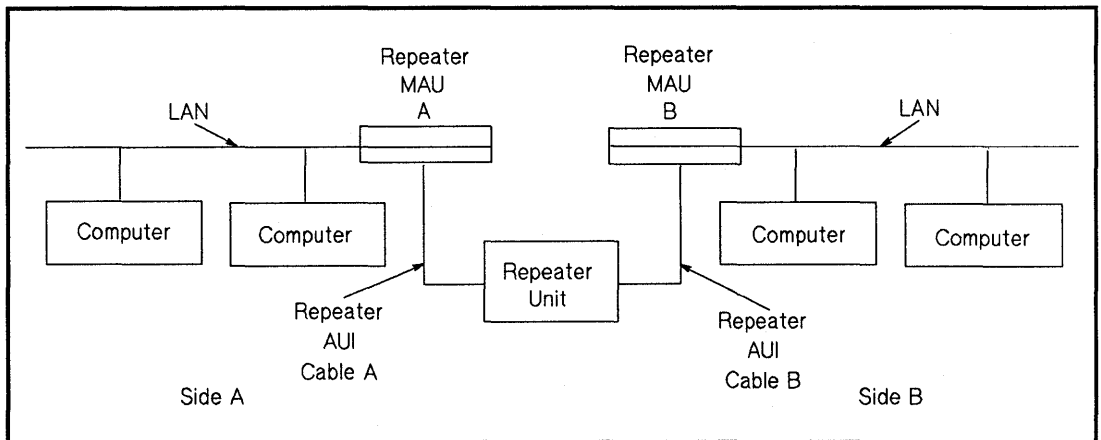
| <b>Tool</b> | <b>Function</b>                                                                          |
|-------------|------------------------------------------------------------------------------------------|
| netstat     | A nodal management command which returns statistical information regarding your network. |

## Diagnostic Tools Summary

| <b>Table 8-1. Diagnostic Tools</b> |                                                                                                                                                                                                                                                                                                                                                      |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Tool</b>                        | <b>Function</b>                                                                                                                                                                                                                                                                                                                                      |
| landiag                            | A diagnostic program that tests LAN connections between HP 9000 computers.                                                                                                                                                                                                                                                                           |
| linkloop                           | A diagnostic program that runs link-level loopback tests between HP 9000 systems. Linkloop uses IEEE 802.3 link-level test frames to check physical connectivity with the LAN. This diagnostic tool is different from the loopback capability of landiag because it tests only the link-level connectivity and not the transport-level connectivity. |
| ping                               | A diagnostic program that verifies the physical connection to a remote host and reports the round-trip communications time between the local and remote hosts. (See ping in the <i>HP-UX Reference</i> .)                                                                                                                                            |
| psidad                             | A utility under DUI that can help to identify problems on the PSI/800 board/card.                                                                                                                                                                                                                                                                    |
| r1b                                | A diagnostic program which tests LAN connections to other HP 9000 computers. r1b does not test a connection to an HP 1000 computer.                                                                                                                                                                                                                  |
| x25check<br>x25server              | These two work in tandem. x25server runs on the logically remote host (could be same physical host) and echoes packets sent to it over the X.25 network by x25check.                                                                                                                                                                                 |
| x25stat                            | A nodal management command that returns status and information of the X.25 device/card. It provides interface status configuration information and virtual circuit statistics.                                                                                                                                                                       |
| x25upload                          | This is used to upload the firmware in case of problems with the firmware on the board.                                                                                                                                                                                                                                                              |
| Event Logging                      | A utility that sends informational messages regarding network activity to the system console or to a file.                                                                                                                                                                                                                                           |
| Network Tracing                    | A utility that traces link-level traffic to and from a node. HP recommends that you enable tracing only when troubleshooting a problem unsolved by other means.                                                                                                                                                                                      |

## Diagnosing Repeater and Gateway Problems

If you are using a repeater and hosts on either side of the repeater are having difficulty communicating with each other, a repeater subsystem failure may have occurred. In the illustration below, all of the systems on side A are able to communicate with one another. All the systems on side B are able to communicate with each other. If communication is cut from side A to side B, the repeater subsystem is suspect for causing the fault, since it is the medium by which side A



**Figure 8-1. Troubleshooting Networks That Use Repeaters**  
and side B communicate.

## Diagnosing Repeater and Gateway Problems

The same concept holds for communication through a gateway. If you suspect a gateway problem, try the following procedures:

- To determine if you are set up to communicate with the desired node, execute:

```
netstat -r
```

- To obtain routing statistics, execute:

```
netstat -rs
```

The statistics could indicate a bad route, suggesting a problem with a gateway node. If so:

- Check with the node manager of the gateway node to ascertain proper operation of the gateway.
- You can detect problems with the X.25 line by the number of errors shown when you execute:

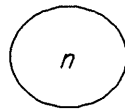
```
x25stat -f -d /devicefile
```

For more information on troubleshooting gateways, refer to the appropriate link manual. For information on repeaters, refer to the *HP-PB LAN Interface Controller (LANIC) Installation Manual*.

---

## Flowchart Format

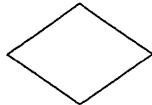
The flowcharts in this chapter each have a corresponding set of labeled explanations. You can follow the flowcharts alone or follow the flowcharts and read the explanations for more detail. The explanations are on the pages which follow each flowchart.



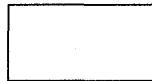
Start of flowchart *n*; re-enter current flowchart



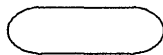
Go to and enter flowchart *n*



Make a decision



Perform an action



Exit flowchart

### Troubleshooting the ARPA/Berkeley Services

When troubleshooting problems with the ARPA/Berkeley Services, you need a reference point to work from. For example, does the problem exist on the remote system or on the local system? However, the terms “local” and “remote” are limited in their description of complex communications, such as when a local system logs onto a remote system and then the remote system logs back onto the local system. At that point, which is the local system and which is the remote system?

A better solution is to use the terms **client** and **server**. The term **client** refers to a process that is requesting a service from another process. The term **server** refers to a process or host that performs operations requested by local or remote hosts that are running client processes.

HP has implemented a “super-server” known as the internet daemon, `inetd`. This program acts like a switchboard; that is, it listens for any request and activates the appropriate server based on the request.

A typical network service consists of two cooperating programs. The client program runs on the requesting system. The server program runs on the system with which you want your system to communicate. The client program initiates requests to communicate. The server program accepts requests for communication. For example, the network service `rlogin` is the client program that requests a login to a remote HP-UX or UNIX system. When the request to login is received on the remote host by `inetd`, `inetd` invokes the server program for `rlogin` called `rlogind` to handle the service request.

### Error Messages

The error messages generated by a service as seen on the client can be generated by the client or the server. Error messages from the client occur before a connection is completely established. Error messages from the server occur once a connection is completely established.

## Troubleshooting the ARPA/Berkeley Services

Whenever you receive an error message, follow the corrective action supplied in the section of the *HP-UX Reference* for that service. The error message is preceded by the name of the service. Table 8-2 shows the appropriate section of the *HP-UX Reference* to refer to for a description of the error messages:

| <b>Service</b> | <b>Client</b>    | <b>Server</b> |
|----------------|------------------|---------------|
| telnet         | telnet           | telnetd       |
| ftp            | ftp              | ftpd          |
| rlogin         | rlogin           | rlogind       |
| remsh          | remsh            | remshd        |
| rcp            | rcp              | remshd        |
| ruptime        | ruptime          | rwhod         |
| rwho           | rwho             | rwhod         |
| ddfa           | user application | ocd           |

If the server or the client is not an HP 9000 computer, refer to the appropriate user's manual or system administration manual for that system. There is not a standard naming convention for servers or processes that activate the servers; however, you should be able to find the information in the system's documentation.

### Services Checklist

1. Did you answer the questions in the troubleshooting checklist at the beginning of this chapter?
2. Run the service to your own node. To do this, your node name and internet address must be in the `/etc/hosts` file. If the server is successful, then the client and the server halves of the service operate correctly. This provides a starting point to determine where problems are occurring.



## Troubleshooting the ARPA/Berkeley Services

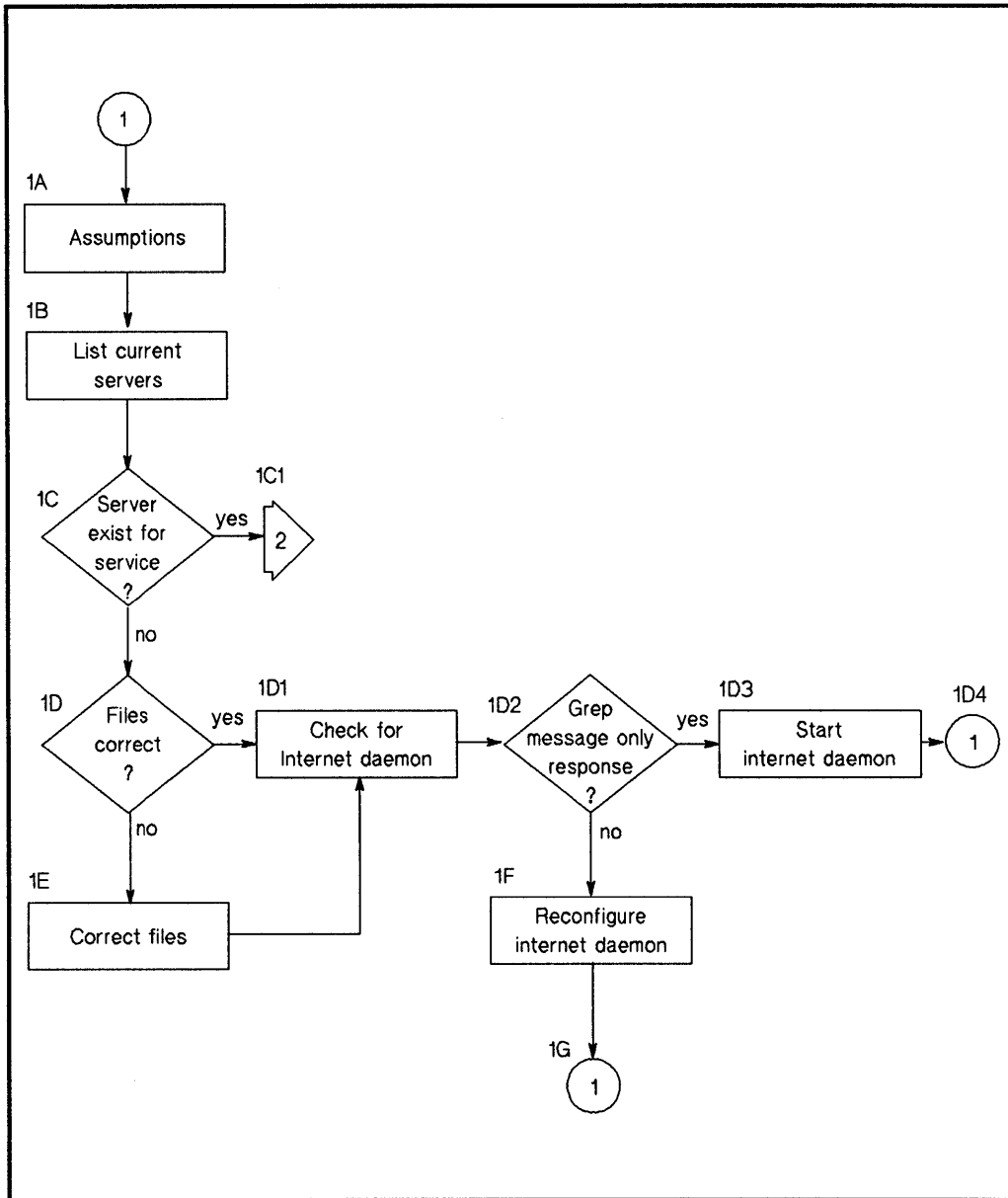


Figure 8-1. Flowchart 1. Checking for a Server

### Flowchart 1. Checking for a Server

Follow this flowchart for all services and servers, and replace the words “service” and “server” with the appropriate service name or server name.

1A. **Assumptions.** Before you begin Flowchart 1, you should have verified local node operations and verified connectivity with ping (see the troubleshooting section of *Installing and Administering LAN*).

1B. **List current servers.** List the servers currently running on your system by executing:

```
netstat -a
```

| Local Address | Client/Request | TCP State |
|---------------|----------------|-----------|
| *.ftp         | ftp            | LISTEN    |
| *.telnet      | telnet         | LISTEN    |
| *.login       | rlogin         | LISTEN    |
| *.shell       | remsh, rcp     | LISTEN    |
| *.exec        | rexec library  | LISTEN    |
| *.who         | rwho, ruptime, |           |
| *.smtp        | sendmail SMTP  | LISTEN    |
| *.tftp        | tftp           | LISTEN    |
| *.bootps      | bootpd         | LISTEN    |
| *.finger      | fingerd        | LISTEN    |

Note that UDP-based protocols are datagram driven so they do not show a TCP LISTEN status.

## Troubleshooting the ARPA/Berkeley Services

- 1C. **Server exists for service?** If the server does not exist for the requested service, continue with 1D to determine why. If the server does exist for the server, continue with 1C1.
- 1C1. **Go to Flowchart 2.** Go to the next flowchart to begin troubleshooting the security of the ARPA/Berkeley Services.
- 1D. **Files correct?** Is there an entry for the servers/services in the `/etc/inetd.conf` or `/etc/services` files?

The following entries are required in the `/etc/inetd.conf` file:

| <b>Service Requested</b> | <b>Entry</b>                                       |
|--------------------------|----------------------------------------------------|
| ftp                      | ftp stream tcp nowait root /etc/ftpd ftpd          |
| telnet                   | telnet stream tcp nowait root /etc/telnetd telnetd |
| rlogin                   | login stream tcp nowait root /etc/rlogind rlogind  |
| remsh, rcp               | shell stream tcp nowait root /etc/remshd remshd    |
| rexec library            | exec stream tcp nowait root /etc/rexecd rexecd     |
| tftp                     | tftp dgram udp nowait root /etc/tftpd tftpd        |
| bootpd                   | bootps dgram udp wait root /etc/bootpd bootpd      |
| fingerd                  | finger stream tcp nowait bin /etc/fingerd fingerd  |

Check the permissions on the files in the `/etc` directory. The file servers `ftpd`, `bootpd`, `telnetd`, `rlogind`, `remshd`, `rexecd`, `rwhod`, and the daemon `inetd`, must be owned and executable by `root` only. `fingerd` should be owned and executed by `bin` only. No other user should have permission to write them, although all users can read them.

## Troubleshooting the ARPA/Berkeley Services

The following entries are required in the `/etc/services` file:

| <b>Service Requested</b> | <b>Entry</b>                   |
|--------------------------|--------------------------------|
| ftp                      | ftp 21/tcp                     |
| telnet                   | telnet 23/tcp                  |
| sendmail/SMTP            | smtp 25/tcp                    |
| rexec library            | exec 512/tcp                   |
| rlogin                   | login 513/tcp                  |
| remsh, rcp               | shell 514/tcp                  |
| rwho, ruptime            | who 513/tcp                    |
| tftp                     | tftp 69/udp                    |
| bootpd                   | bootps 67/udp<br>bootpc 68/udp |
| fingerd                  | finger 79/tcp                  |

If the file entries or permissions are not correct, continue with 1E.

**1D1. Check for internet daemon.** To see if the `inetd` is active on the server node, login to the server node and execute:

```
ps -ef | grep inetd
```

**1D2. Grep message only response?** If the `grep` message is the only response, `inetd` is not active. If this is true, continue with 1D3.

**1D3. Start internet daemon.** To start `inetd`, execute one of the following as super-user:

```
inetd
```

## Troubleshooting the ARPA/Berkeley Services

or, if you want `inetd` to log connections, execute:

```
inetd -l
```

The `/etc/netbsdsrc` shell script usually starts `inetd`. Refer to the "inet Daemon" section of chapter 3 for more information

- 1D4.** Go to 1B. Once `inetd` is running, repeat this flowchart beginning with 1B.
- 1E.** **Correct files.** If there was an incorrect entry or no entry in the `/etc/inetd.conf` or `/etc/services` files, enter the correct information and continue with 1D1.
- 1F.** **Reconfigure internet daemon.** To reconfigure `inetd`, execute as superuser:
- ```
inetd -c
```
- and continue with 1G.
- 1F.** Go to 1B. Repeat flowchart from 1B to check if the server exists.

Flowchart 2. Security for ARPA Services

Even though a server exists for a service, the server may not accept connections due to the security that has been implemented for the server.

Note The corrections suggested in 2B1, 2C1 and 2F1 must be done by the super-user. Also, except for the “anonymous” user ID, ftp requires non-null passwords on remote user accounts.

- 2A. **Determine number of existing connections.** If `inetd` was started with the `-l` option, the system log may list the number of connections. If these messages do not appear in the system log, continue with 2C, or enable the connection logging with `inetd -l`
- 2B. **Maximum number of connections?** The maximum number of simultaneous connections is specified in the optional file, `/usr/adm/inetd.sec`. When `inetd` is configured, it checks this file to determine the number of allowable incoming connections. Look at this file to determine how many connections are allowed; the default is 1000.
- 2B1. **See node manager.** If the maximum number of connections has been reached, the node manager can change this value in the `/usr/adm/inetd.sec` file.
- 2C. **Access to the server?** The `/usr/adm/inetd.sec` file also contains a list of systems that may not access the server. If `inetd` was started with the `-l` option, the system log may list the connections that are refused access to the server. Check this log file, if it exists, or ask the node manager to verify whether you have access to the server. If you find that you do not have access to the server, continue with 2D.
- 2C1. **Using an ARPA service?** There are additional security files that exist for the ARPA services that must be checked. If you are using `ftp` or `telnet` go to 2C2; otherwise, go to 2E.

Troubleshooting the ARPA/Berkeley Services

- 2C2. **Using ftp?** Are you attempting to use ftp? If you are, go to 2C3; otherwise, go to 2F.
- 2C3. **Access to ftp?** If the user you are logging in as is listed in the `/etc/ftpusers` file on the server system, you may not use ftp to that system. If you do not have access to ftp, go to 2G.
- 2C4. **`$HOME/.netrc` file incorrect or non-existent?** If this file is incorrect or non-existent, it is not used for the connection attempt. In particular, if the file exists, check its mode bits, owner ID, and syntax. See the `netrc` reference page in section 4 of the *HP-UX Reference* for more information. If it is correct, go to 2H.
- 2C5. **Fix `$HOME/.netrc`.** If the file is incorrect, make corrections and go to 2C6.
- 2C6. Once the corrections are made, repeat this flowchart beginning with 2A.
- 2D. **See node manager.** If your system was denied access to the server system by the `/usr/adm/inetd.sec` file but you wish to use the server, contact the node manager of the server system and request access.
- 2E. **Go to Flowchart 3.** If you are using the Berkeley Services, go to Flowchart 3 to begin troubleshooting the security for those services.
- 2F. **Telnet should work.** If you have reached this point in the flowchart, the telnet server exists and you have access to the system. If you are using correct syntax, if the login password you are using exists on the server system, and if none of the error messages have solved the problem, report the problem to your Hewlett-Packard support contact.
- 2G. **See node manager.** You are not allowed to use ftp to access the server system. Check with the node manager of the server system and request that the appropriate user name be removed from the `/etc/ftpusers` file.
- 2H. **Ftp should work.** If you have reached this point in the flowchart, the ftp server exists and you have access to the system. If you are using correct

Troubleshooting the ARPA/Berkeley Services

syntax and none of the error messages have solved the problem, report the problem to your Hewlett-Packard support contact.

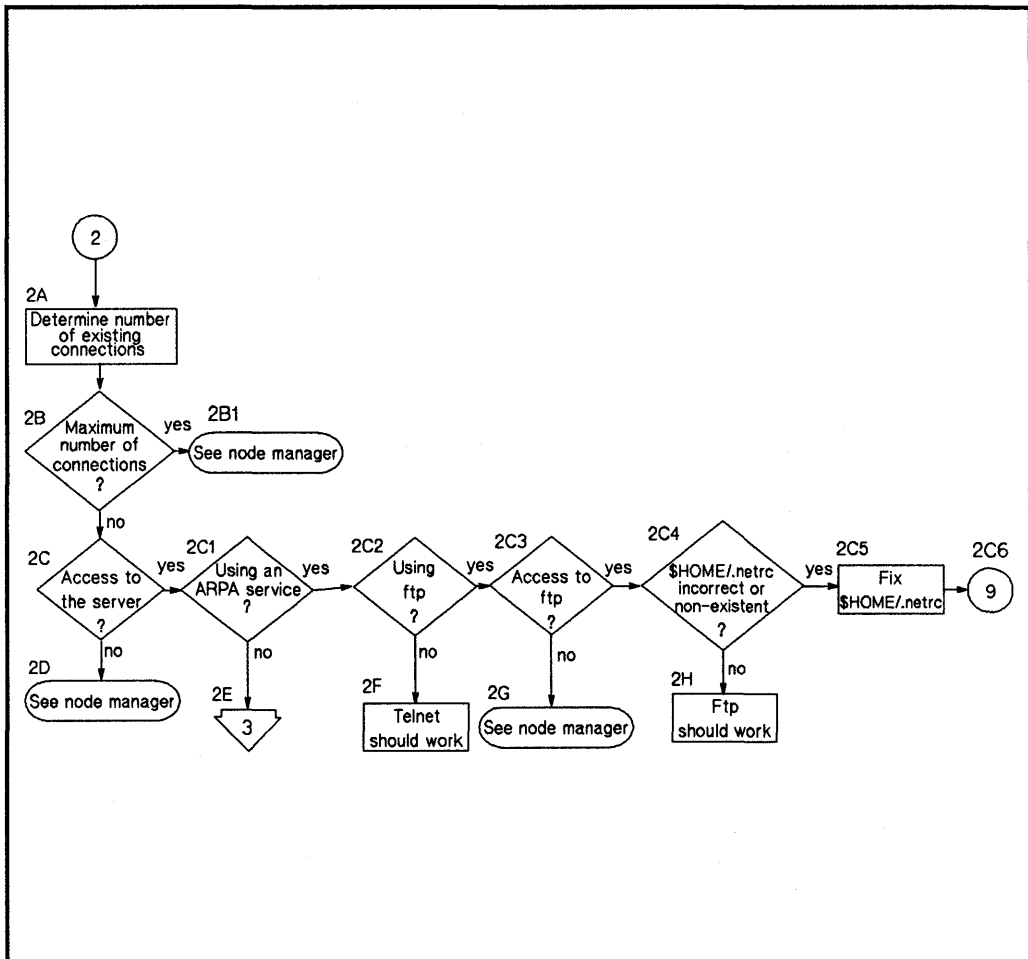


Figure 8-2. Flowchart 2. Security for ARPA Services

Troubleshooting the ARPA/Berkeley Services

Flowchart 3. Security for Berkeley Services

The following information assumes an account has a password. If it does not, the security checks are not performed.

- 3A. **User name exists on server host?** Does the `user_name` that you want to log in as exist on the server host? You can specify another user's name by using the `-l` option with `rlogin`. If the desired `user_name` does not exist on the server host, continue with 3B.
- 3A1. **Accessing server system as yourself?** If not, go to 3D.
- 3A2. **Are you superuser?** If you are, go to 3D; otherwise continue with 3C.
- 3B. **Cannot access.** Since your user name or the `user_name` that you want to use to login does not exist on the remote system, you cannot log in to the remote system unless the remote system's node manager creates an account for you.
- 3C. **Entry in server's `/etc/hosts.equiv` file?** Does the server system have your official host name entered in its `/etc/hosts.equiv` file? If so, you should be logged into the remote system without a password prompt. If you can do this, continue with 3C1; otherwise go to 3D.
- 3C1. **OK.** If you are using the `rlogin` service, you are automatically logged in. If you are using another Berkeley service, permission is granted for the operation.
- 3D. **`$HOME/.rhosts` file exists and has entry for you?** Does the user name that you want to become on the server system have a `.rhosts` file in that user's `$HOME` directory? If it does, does it have your local host and user name(s) listed properly? If the `$HOME/.rhosts` file does not exist on the server system or if it does not have an entry for you, continue with 3F; otherwise continue with 3C1.
- 3E. **Using `rlogin`.** If you are using the `rlogin` service go to 3E1. If you are not using `rlogin`, go to 3F.

Troubleshooting the ARPA/Berkeley Services

- 3E1. **Password prompt.** You will receive a password prompt. Enter the password for your remote user name.
- 3F. **Permission denied.** You do not have permission to access the user's account. Ask the user to add your local host and user name(s) to his or her `.rhosts` file.

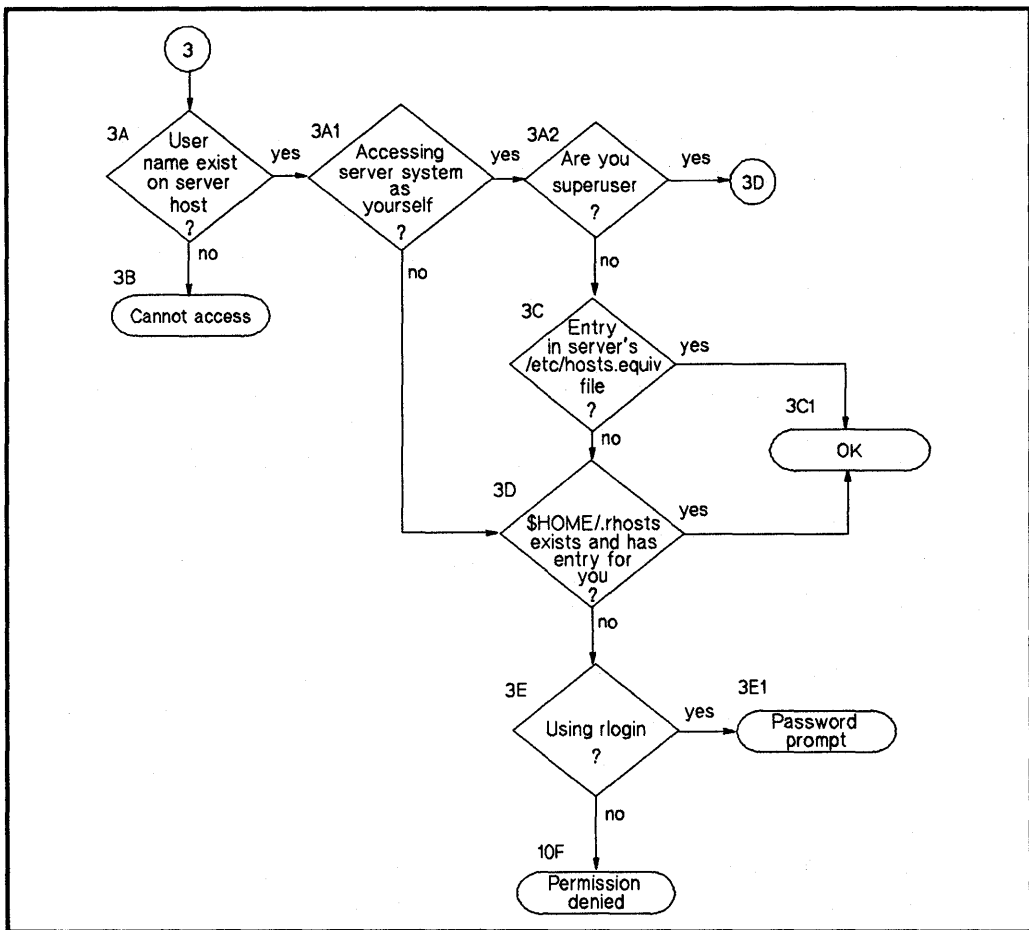


Figure 8-3. Flowchart 3. Security: Berkeley Services

Reporting Problems to Your Hewlett-Packard Support Contact

Note

For C2 Security, refer to *A Beginner's Guide to HP-UX* and the *HP-UX System Security Manual*.

Reporting Problems to Your Hewlett-Packard Support Contact

If you do not have a service contract with HP, you may follow the procedure described below but you will be billed accordingly for time and materials.

If you have a service contract with HP, document the problem as a Service Request (SR) and forward it to your Hewlett-Packard support contact. Include the following information where applicable:

- A characterization of the problem. Describe the events leading up to and including the problem. Attempt to describe the source of the problem. Describe the symptoms of the problem and what led up to the problem.

Your characterization should include: HP-UX commands, communication subsystem commands, job streams, result codes and messages, and data that can reproduce the problem.

Illustrate as clearly as possible the context of any message(s). Prepare copies of information displayed at the system console and user terminal.

- Obtain the version, update, and fix information for all software.

To check your ARPA Services version, execute the `what service_name` command, where `service_name` is a network service specific to the networking product such as `ftp` for ARPA Services.

To check the version of your kernel, execute `uname -r`.

Reporting Problems to Your Hewlett-Packard Support Contact

This allows your support contact to determine if the problem is already known, and if the correct software is installed at your site.

- Record all error messages and numbers that appear at the user terminal and the system console.
- Save all network log files.

Prepare the formatted output and a copy of the log file for your Hewlett-Packard support contact to further analyze.

- Prepare a listing of the HP-UX I/O configuration you are using for your Hewlett-Packard support contact to further analyze.
- Try to determine the general area within the software where you think the problem exists. Refer to the appropriate reference manual and follow the guidelines on gathering information for problems. For ARPA Services, refer to *Using ARPA Services* for information about the services.
- Document your interim or “workaround” solution. The cause of the problem can sometimes be found by comparing the circumstances in which it occurs with the circumstances in which it does not occur.
- Create copies of any ARPA Services or other trace files that were active when the problem occurred for your Hewlett-Packard support contact to further analyze.
- In the event of a system failure, a full memory dump must be taken. Use the HP-UX utility `/etc/savecore` to save a core dump. See the *HP-UX System Administration Tasks Manual* for details. Send the output to your Hewlett-Packard support contact.

Public Network Contacts

This appendix is for use in setting up a new domain for the BIND name server. If you are a member of one of these networks, contact the address or phone number given for information on setting up a new domain.

If your site uses ..	Contact
ARPA Internet	<p>hostmaster@nic.ddn.mil or 1-800-235-3155.</p> <p>You may also wish to join the BIND mailing list, a mail group for people running BIND name servers on the ARPA Internet. The group discusses future design decisions, operational problems, and related topics. To be added to this mailing list, contact: bind-request@ucbarpa.berkeley.edu.</p>
CSNET	cic@sh.cs.net or (617) 873-2777.
BITNET	domains@bitnic
UUCP	uunet!postmaster or (703) 876-5050.

If you are not a member of one of these networks and you need an application form for establishing or joining a domain, get a copy of NETINFO:DOMAIN-TEMPLATE.TXT from nic.ddn.mil.

Note

Files from `nic.ddn.mil` can be obtained by either of two methods:

- 1) Use anonymous ftp to get the specified file.
- 2) Use electronic mail. Send electronic mail to `service@nic.ddn.mil`, using the "subject" line of the message header to specify the file you want. For example:

```
To: SERVICE@NIC.DDN.MIL
Subject:          NETINFO:DOMAIN-INFO.TXT.
```

Choosing a BIND Domain

The parent network to which you belong will define your choices for a domain. Typical choices may include:

- A second-level domain under the top-level generic domains `com`, `edu`, `mil`, `net`, or `org`.
- A second-level domain under an existing top-level country domain, such as `us`, `uk` (Great Britain), `au` (Australia), or `ca` (Canada).
- A third-level (or lower) domain under an existing domain.

Other useful information for domain choice includes:

1. A list of existing top- and second-level domains; get a copy of the file `NETINFO:DOMAIN-INFO.TXT` from `nic.ddn.mil`.
2. Information on who is in charge of a particular domain; send an electronic mail message to `service@nic.ddn.mil`. In the "subject" field, type the following:

```
whois domain domain_name
```

You will receive the requested information via electronic mail. Or you can get a copy of the file `NETINFO:DOMAIN-CONTACTS.TXT` from `nic.ddn.mil`.

Standard Data Files and Experimental Resource Records

This appendix describes the format and contents of the standard domain data files associated with the BIND name server. It also describes certain resource records that are available for experimental use only. These experimental record types may change or become obsolete in the future.

Domain Data Files

This section explains the boot file and standard data files required to configure BIND name servers.

Boot File

`/etc/named` (the name server daemon) is started from `/etc/netbsdsrc` automatically at boot up when the boot file, `/etc/named.boot`, exists. When `/etc/named` starts up, it reads the boot file first. The boot file tells the name server what type of server it is, what zones it is authoritative for, and where it should get its initial data. Examples of boot files for each server type appear under the instructions for configuring that type, in the chapter titled “Configuring and Maintaining the BIND Name Server.”

Standard Data File Format

This section describes the contents and format of the standard data files. The data in the files consists of resource records (RRs), as described in the chapter titled

Domain Data Files

“Configuring and Maintaining the BIND Name Server.” Note that each host in a domain must have at least two RRs: one address record (A), in db.[domain], and one pointer record (PTR), in db.[net]. db.[domain] may also contain other records for hosts, such as canonical name (CNAME) or mail exchanger (MX) records.

```
/etc/named.data/db.cache
```

Every name server must have a db.cache file. When the name server boots, it reads the list of root name servers from db.cache. The name server cycles through the list, querying each root server for the current list of root servers until it receives a response. The up-to-date root server information is cached and the original list is kept. The original list is used only if the up-to-date information times out and is discarded.

The following example lists the current root name servers for the ARPA Internet. (This list can also be obtained from nic.ddn.mil via anonymous ftp, in the file NETINFO:ROOT-SERVERS.TXT.) If you are connected to the ARPA Internet, use this list or an updated version of it. If you are on an isolated network, fill in the cache file with the root servers for your isolated network. (Refer to the section titled “Configuring a Root Server” in the chapter titled “Configuring and Maintaining the BIND Name Server.”)

```
;  
; list of root domain servers  
;  
      86400      IN      NS      NS.NIC.DDN.MIL.  
      86400      IN      NS      AOS.BRL.MIL.  
      86400      IN      NS      A.ISI.EDU.  
      86400      IN      NS      GUNTER-ADAM.AF.MIL.  
      86400      IN      NS      C.NYSER.NET.  
      86400      IN      NS      TERP.UMD.EDU.  
      86400      IN      NS      NS.NASA.GOV.  
  
; addresses for these root servers  
NS.NIC.DDN.MIL.      86400      IN      A      192.67.67.53  
AOS.BRL.MIL.        86400      IN      A      128.20.1.2  
AOS.BRL.MIL.        86400      IN      A      192.5.25.82
```

Domain Data Files

```
A.ISI.EDU.          86400  IN    A      26.3.0.103
A.ISI.EDU.          86400  IN    A      128.9.0.103
GUNTER-ADAM.AF.MIL. 86400  IN    A      26.1.0.13
C.NYSER.NET.       86400  IN    A      192.33.4.12
TERP.UMD.EDU.      86400  IN    A      128.8.10.90
NS.NASA.GOV.       86400  IN    A      128.102.16.10
NS.NASA.GOV.       86400  IN    A      192.52.195.10
```

/etc/named.data/db.127.0.0

Each name server must have a db.127.0.0 file. By convention, hosts running Berkeley networking use 127.0.0.1 as the address of the loopback interface. Since the network number 127.0.0 is not assigned to any one site but is used by all hosts running Berkeley networking, each name server must be authoritative for network 127.0.0. The file db.127.0.0 contains the resource record that maps 127.0.0.1 to the name of the loopback address, usually "localhost".

In data files, @ refers to the current origin. The current origin is also appended to names that do not end with a dot. For example, the 1 in the PTR line would have the origin 0.0.127.in-addr.arpa appended to it.

An example follows:

```
@           IN    SOA    rabbit.div.inc.com. root.moon.div.inc.com.(
           1           ;Serial
           10800      ; Refresh every 3 hours
           3600       ; Retry every hour
           604800     ; Expire after a week
           86400     ) ; Minimum ttl of 1 day
           IN    NS    rabbit.div.inc.com.
1           IN    PTR    localhost.
```

/etc/named.data/db.[domain]

Only primary servers have db.[domain] files, one for each domain they are authoritative for. [domain] is the first part of the domain specified in the command line when using hosts_to_named. This file should contain an A

Domain Data Files

(address) resource record for every host in the zone, plus all other data about hosts in the zone (except for PTR records, described under db.[net].)

In data files, @ refers to the current origin. The current origin is also appended to names that do not end with a dot. For example, indigo has div.inc.com appended to it.

```

;
; db.div
;
@           IN      SOA      rabbit.div.inc.com. root.moon.div.inc.com. (
                1          ; Serial
                10800       ; Refresh every 3 hours
                3600        ; Retry every hour
                604800      ; Expire after a week
                86400      ) ; Minimum ttl of 1 day
                IN      NS      rabbit.div.inc.com.
localhost   IN      NS      indigo.div.inc.com.
localhost   IN      A        127.0.0.1
indigo      IN      A        15.19.8.197
            IN      A        15.19.13.197
            IN      HINFO    HP9000/840 HPUX
incindigo   IN      CNAME    indigo
cheetah     IN      A        15.19.8.64
            IN      HINFO    HP9000/850 HPUX
            IN      WKS      15.19.8.64 UDP syslog domain route
            IN      WKS      15.19.8.64 TCP ( telnet smtp ftp
                                shell domain )
rabbit      IN      MX        5 rabbit.div.inc.com.
            IN      MX        10 indigo.div.inc.com.
rabbit      IN      A        15.19.8.119
```

/etc/named.data/db.[net]

Only primary servers have db.[net] files, one for each network they are serving. [net] is the network number specified in the command line when using hosts_to_named. This file should contain a PTR (pointer) record for every host in the zone. PTR records are needed because the DNS resolves the address of a name by tracing down the domain tree and contacting a server for each label of

Domain Data Files

the name. This name-based indexing does not allow easy translation of an IP address back into its hostname. Hence, PTR records point back to the canonical name of the host owning the IP address.

The `in-addr.arpa` domain was created to allow this inverse mapping. The `in-addr.arpa` domain is preceded by four labels corresponding to the four bytes (octets) of an Internet address. Each byte must be specified even if it is zero. For example, the Internet address 143.22.0.3 is located in the domain `3.0.22.143.in-addr.arpa`. Note that the four octets of the address are reversed. This address reversal is awkward to read but allows for the natural grouping of hosts in a network.

In data files, `@` refers to the current origin. The current origin is also appended to names that do not end with a dot. For example, 119 has `8.19.15.in-addr.arpa` appended to it.

```

;
; db.15.19.8
;
@           IN      SOA      rabbit.div.inc.com. root.moon.div.inc.com. (
                                1           ; Serial
                                10800      ; Refresh every 3 hours
                                3600       ; Retry every hour
                                604800     ; Expire after a week
                                86400 ) ; Minimum ttl of 1 day
                                IN      NS      rabbit.div.inc.com.
                                IN      NS      indigo.div.inc.com.
119         IN      PTR      rabbit.div.inc.com.
64          IN      PTR      cheetah.div.inc.com.
197        IN      PTR      indigo.div.inc.com.
```

Experimental Resource Records

Option Lines

This section describes option lines which can be used in domain data files.

\$INCLUDE

An **\$INCLUDE** line allows you to separate different types of data in the same zone by putting them in multiple files. For instance, host data and mail exchanger data can be put in separate files. **\$INCLUDE** starts in column 1, followed by a file name. For example:

```
$INCLUDE /usr/named/data/mail exchangers
```

\$ORIGIN

An **\$ORIGIN** line allows you to change the origin in a data file. It is useful for putting more than one domain in a data file. **\$ORIGIN** starts in column 1, followed by a domain origin. For example:

```
$ORIGIN div.inc.com
```

Experimental Resource Records

This section describes the experimental resource records available.

GID - Group ID

```
:[name] [ttl] [class] GID Integer  
user1 IN GID 25
```

GID is the integer representing the group ID of the user.

MB - Mailbox Domain Name

B-6 Standard Data Files and Experimental Resource Records

Experimental Resource Records

```
;[name] [ttl] [class] MB Host
John      IN      MB      kelly.int.inc.com.
```

MB, the Mailbox record, lists the host where a user wishes to receive mail. The name field is the user's login, and the host field designates the host where mail will be delivered. Mailbox names should be unique to the zone.

MG - Mail Group Member

```
:[mail group name] [ttl] [class] MG member name
admin              IN      MG      Milo
```

The Mail Group record, MG, lists members of a mail group. An example for setting up a mail list is as follows:

```
Bind      IN      MINFO      Bind-request  jjw.ucb.edu.
          IN      MG         tony.ucb.edu.
          IN      MG         korba.ucb.edu.
          IN      MG         norwood.div.inc.com.
          IN      MG         sutton.div.inc.com.
```

MINFO - Mailbox or Mail List Information

```
:[name] [ttl] [class] MINFO rmailbx emailbx
Bind      IN      MINFO      Bind-request  jjw.ucb.edu.
```

The mail information record, MINFO, specifies mailing list information. This record contains two parts, rmailbx and emailbx. rmailbx is a domain name specifying a mailbox responsible for the mailing list or mailbox. This address is used for requests to be added to or deleted from the mailing list. If this domain name is the root, the owner of the MINFO RR is responsible for itself. emailbx is a domain name specifying a mailbox which will receive error messages related to the mailing list or mailbox specified by the owner of the MINFO RR.

Experimental Resource Records

In the example above, any errors during a mailing to “Bind-request” will go to jjw.ucb.edu.

MR - Mail Rename Domain Name

```
      ; [name]  [ttl]  [class] MR      corresponding MB
      postmaster      IN      MR      john
```

Mail Rename, MR, lists aliases for a user. The name field lists the alias for the name in the fourth field, which should have a corresponding MB record.

UID - User ID

```
      ; [name]      [ttl]      [class] UID      User ID
      User1              IN      UID      35
```

User ID, UID, is an integer used to identify a particular user.

UINFO - User Information

```
      ; [name]      [ttl]      [class] UINFO      User Info.
      User1              IN      UINFO      administrator
```

User information, UINFO, provides information about a particular user.

Index

A

Action, 3-16

Address determination, with BOOTP, 7-4

ARPA

See ARPA Services

ARPA Daemons

See Daemons

ARPA Diagnostics

See Diagnostics

ARPA Servers

See Servers

ARPA Services, 1-1

Administering, 3-1

Allow access to, 2-20

Configuring, 2-7

Deny access to, 2-20

Initialization script, 3-2

Installing, 2-1

See also: Installing ARPA Services

Naming conventions, 4-15

Product overview, 1-1, 1-3

Software components, 1-1 - 1-3

Troubleshooting, 8-8

B

Berkeley Internet Name Domain

See BIND Name server

Berkeley Services, 1-2

BIND Name Server, 4-1, 4-8

/etc/hosts, 4-9

Boot file, 4-18, B-1

Boot file options, 4-38

Caching only name server, 4-16 - 4-17

Caching only name server, boot file, 4-30

Caching only name server, configuring,
4-30

Configuring, 4-25 - 4-31

Configuring, forwarding option, 4-39

Configuring, slave option, 4-39

Configuring, sort option, 4-39

Configuring, wildcards, 4-38

Data files, 4-18, 4-25, B-1

Data files, creating, 4-25

Data files, maintaining and updating, 4-34

Debug levels, 4-55

Debugging, 4-41, 4-50

See also: DNS (Domain Name System)

Domain, 4-7

Domain data files, B-1

Domain data files, updating, 4-35

Domain, choosing, A-2

Domain, creating a new domain, 4-15

Handling queries, 4-10

Host, adding, 4-35

Host, deleting, 4-35

How it works, 4-10

Key terms, 4-3 - 4-5

Local Domain, 4-32

Network files, maintaining and updating,
4-34

Network Information Service (NIS), 4-8

Primary name server, 4-16 - 4-17

Primary name server, boot file, 4-26

Primary name server, configuring, 4-25

- Remote name server, 4-16 - 4-17
- Remote name server, setting up, 4-30
- Resolver, 4-8, 4-11, 4-30
- Resource records, 4-19, B-6
- Root name server, 4-11, B-2
- Root name server, configuring, 4-37
- Secondary name server, 4-16 - 4-17
- Secondary name server, boot file, 4-29
- Secondary name server, configuring, 4-27
- Send signals to, 4-54
- Starting, 4-32 - 4-33
- Statistics, 4-56
- Subdomains, delegating, 4-36
- Troubleshooting, 4-40
- Troubleshooting, nslookup, 4-41
- Troubleshooting, syslogd, 4-41
- Verify configuration, 4-33
- BIND Name Service**
 - Advantages of using, 4-10
 - Configuration overview, 4-14
 - Key terms, 4-3 - 4-5
 - Public network contacts, A-1 - A-2
- BOOTP (Bootstrap Protocol)**
 - See* BOOTP server
- BOOTP server**
 - Adding a client, examples, 7-15
 - Address determination, 7-4
 - ARP (Address Resolution Protocol), cache entry, 7-4
 - Boot file, collecting client information, 7-12
 - Boot file, format, 7-12
 - bootpd, BOOTP server implementation, 7-4
 - bootpd, common problems, 7-25
 - bootpquery command, 7-11
 - Bootreply packet, process of, 7-4
 - Bootrequest packet, process of, 7-4
 - Client parameters, in boot file, 7-13
 - Client, adding in SAM, 7-6
 - Client, collecting information about, 7-12
 - Client, examples for adding, 7-15
 - Client, manually adding, 7-12
 - Configuring bootpd manually, overview, 7-10
 - Configuring bootpd manually, procedure for, 7-10
 - Configuring bootpd, with SAM, 7-5
 - Error logging, error log level, 7-37
 - Error logging, information log level, 7-34
 - Error logging, notice log level, 7-36
 - Error logging, overview, 7-34
 - /etc/bootptab file, 7-12
 - Hardware type, client information, 7-12
 - Host name, client information, 7-12
 - How BOOTP works, 7-4
 - Introduction, chapter overview, 7-1
 - IP address, client information, 7-12
 - Key terms, 7-3
 - Link level address, client information, 7-12
 - Nonsupported products, 7-1
 - Options, for bootpd configuration, 7-14
 - SAM configuration, overview, 7-5
 - SAM configuration, procedure for, 7-5
 - Subnet mask, client information, 7-12
 - Tags, boolean, 7-13
 - Tags, boot file format, 7-13
 - Tags, description of, 7-14
 - Troubleshooting, common bootpd problems, 7-25
 - Troubleshooting, helpful hints, 7-24
 - Troubleshooting, overview, 7-24
 - Verifying manual bootpd configuration, 7-11
- Bootstrap Protocol (BOOTP)**
 - See* BOOTP server
- C**
 - Caching Only Name Server**
 - See* BIND Name Server

Client, BOOTP

See BOOTP server

Configuration files

`$HOME/.netrc`, 2-34

`$HOME/.rhosts`, 2-30

`/etc/ftusers`, 2-24

`/etc/gated.conf`, 5-12

`/etc/hosts.equiv`, 2-9, 2-26

`/etc/inetd.conf`, 2-9, 2-16

`/usr/adm/inetd.sec`, 2-19

`/etc/hosts`, 2-8, 2-10

Configuring BOOTP servers

See BOOTP server

Configuring TFTP servers

See TFTP server

`Convert_rhosts`, 4-34

Core dump, 8-21

D

Daemons, 3-1

`inetd`, 3-4, 3-6 - 3-7

`rwhod`, 3-2

Data corruption, 8-3

Diagnostics

`Landiag`, 8-3

`Linkloop`, 8-3

`Netstat`, 8-3

`Ping`, 8-3

`Rlb`, 8-3

Tools summary, 8-3

DNS (Domain Name System)

Domain, 4-7

Domain name, 4-7

Illustration, 4-7

Domain

See BIND Name Server

Domain Name System

See DNS (Domain Name System)

Dynamic routing with `gated`, 5-1

E

EGP (External Gateway Protocol), 5-3, 5-6

Error logging, by `bootpd`, 7-34

Errors

Command errors, 8-1

Intermittent, 8-3

Link level, 8-3

Messages, 8-8

Programmatic errors, 8-1

`/etc/ftusers` file, 2-9, 2-24

Format, 2-25

Permissions, 2-26

`/etc/gated.conf` file, 5-4

`/etc/hosts` file, 2-8, 2-10

Edit, 2-14

Edit with SAM, 2-11

Format, 2-15

Permissions, 2-15

`/etc/hosts.equiv` file, 2-9, 2-26

Permissions, 2-30

`/etc/inetd.conf` file, 2-9, 2-16

Edit, 2-16, 2-18

Permissions, 2-19

`/etc/netbsdsrc` file, 3-2

`/etc/netgroup` file, 2-28, 2-32

`/etc/newconfig/aliases` file, 6-19

`/etc/newconfig/netbsdsrc` file, 5-13

`/etc/newconfig/sendmail.cf` file, 6-16

`/etc/resolv.conf` file, 4-8, 4-30

`/etc/savecore` utility, 8-21

`/etc/syslog.conf` file

Configure `syslogd`, 3-16

Default configuration, 3-17

Event logging, 8-3

External Gateway Protocol

See EGP

F

- Facility, 3-16
 - Daemon, 3-16
 - Mail, 3-16
 - Syslog, 3-16
- Fingerd server, 3-9
- Ftpd server, 3-10
 - Anonymous account, 3-11 - 3-12, 3-14
 - Logging, 3-11
 - Timeout, 3-10

G

- Gated, 5-1
 - /etc/gated.conf, 5-12
 - /etc/gated.conf file, 5-4
 - /etc/newconfig/netbsdsrc file, 5-13
 - Advantages of using, 5-2
 - announce option, 5-9
 - Avoiding routes, 5-8
 - Command line options, 5-17
 - Configuration files, samples, 5-15
 - Configuration options, 5-4
 - Configuring, 5-12
 - Configuring, sample files, 5-14
 - Customizing routes, 5-6
 - Default gateways, 5-6
 - donotlistenhost option, 5-11
 - EGP, 5-3, 5-6
 - Forcing routes, 5-8
 - HELLO, 5-3, 5-6
 - interfacemetric option, 5-8
 - listen option, 5-11
 - listenhost option, 5-11
 - Metrics, defined, 5-3
 - Metrics, setting, 5-8
 - noannounce option, 5-9
 - noripfrominterface option, 5-10
 - noripoutinterface option, 5-8
 - Passive ooption, 5-7

- RIP, 5-3, 5-5
- RIP, configuring, 5-5
- Routes, avoiding, 5-8
- Routes, customizing, 5-6
- Routes, forcing, 5-8
- Routes, static, 5-7
- Routing protocols, 5-3, 5-5
- sourceripgateways option, 5-8
- src.addr , 5-11
- Starting, 5-17
- Starting, automatic, 5-13
- Static routes, 5-7
- Traceflags, 5-18
- traceflags option, 5-11
- Tracing, 5-11, 5-20
- Troubleshooting, 5-20
- Troubleshooting, dump routing table, 5-21
- Troubleshooting, ripquery, 5-21
- Troubleshooting, syslog output, 5-20
- trustedripgateways option, 5-11

H

- Header flags, 6-63
- HELLO, 5-3, 5-6
- Holddown mode, 5-3
- \$HOME/.netrc file, 2-9, 2-34
 - Format, 2-34
 - Permissions, 2-35
- \$HOME/.rhosts file, 2-9, 2-30
 - Disable, 3-7
 - Permissions, 2-34
- Hostname command, 2-15
- Hostnames
 - Spelling conventions, 2-14
- hosts_to_named, 4-26

I

- Inetd daemon, 2-16, 3-4, 3-7, 3-18
 - /usr/adm/inetd.sec file, 2-9

- Kill server, 3-18
- Log connections, 3-18
- Reconfigure log file, 3-18
- Installing ARPA Services, 2-1
 - Check installation, 2-36
 - Configure hostname to address mapping, 2-5
 - Configure software, 2-7
 - Files created during installation, 2-3
 - Install the software, 2-2
 - Reboot after installation, 2-36
 - Update network map, 2-2
 - Update program, 2-3
- Internet File Transfer Protocol Server
 - See* Ftpd server
- Internetwork mail routing
 - See* Sendmail
- IP address determination,with BOOTP, 7-4

L

- Landiag diagnostic, 8-3
- Level, 3-17
- Link level errors, 8-3
- Linkloop diagnostic, 8-3
- Logging messages, 8-3
- Lookup service, 4-1

M

- metric option, gated, 5-8
- Metrics, Gated, 5-3
- Mflags, 6-60
- MIL-STD specifications, 1-7

N

- Name Server
 - See* BIND Name Server

- Netstat command, 8-3
- Network Information Control Center
 - See* NIC
- Network Information Service (NIS), 4-8
 - BIND Name Server, 4-2
- NFS Services configuration, 2-16, 2-28, 2-32
- NIC (Network Information Control Center), 2-10
- NIS
 - See* Network Information Service, 4-8
- Nslookup utility, 2-6, 4-41
 - Verify name server, 4-33

P

- Ping diagnostic, 8-3
- Primary Name Server
 - See* BIND Name Server
- Problems
 - Characterize, 8-2
 - Identify causes, 8-3
 - Network-wide problems, 8-3
 - Report to support contact, 8-20 - 8-21
 - Resolve, 8-1
 - See also:* Troubleshooting
- Pseudo terminal files, 3-7, 3-9

R

- Rebooting, 2-36
- Remote Name Server
 - See* BIND Name Server
 - See also:* BIND Name Server, Resolver
- Remote systems
 - View list of, 2-13
- Remshd server, 3-6
- Resolver
 - See* BIND Name Server
- Resource records, 4-19
 - See also:* BIND Name Service

- Standard format, 4-19
- Rexecd server, 3-8
- RFCs (Requests for Comment), 1-7
- RIP (Routing Information Protocol), 5-3
 - Holddown mode, 5-3
 - Infinity metric, 5-3
- Rlb diagnostic, 8-3
- Rlogind server, 3-7
- Root Name Server
 - See BIND Name Server
- Routing Information Protocol
 - See RIP
- Routing protocols
 - EGP, 5-3
 - Gated, 5-3
 - HELLO, 5-3
 - Metrics, 5-3
- RPC Services, 2-16
- Rwhod daemon, 3-2

S

- SAM, 3-12, 6-14, 6-27
 - /etc/hosts, BIND database, 2-11
 - /etc/hosts, configure , 2-10
 - /etc/hosts, edit, 2-11
 - /etc/hosts, gateways and routing, 2-11
 - /etc/hosts, NFS YP database, 2-11
 - Anonymous ftp account, create, 3-12
 - ARPA security, 2-20
 - hosts.equiv, edit, 2-27
 - inetd.sec, edit, 2-20
 - netlinkrc, edit, 2-11
 - Network mail, 6-14, 6-27
 - Open ARPA access, 2-27, 2-31, 3-12
 - .rhosts, edit, 2-31
 - Sendmail installation, 6-14
 - Sendmail restart, 6-27
 - Tips for using, 2-11
- SAM (System Administration Manager)
 - Overview for configuring TFTP and
- BOOTP servers, 7-5
 - Procedure for configuring bootpd and tftpd, 7-5
- Secondary Name Server
 - See BIND Name Server
- Security
 - Ftp, /etc/ftpusers file, 2-9, 2-24
 - Gated, 5-8
 - Inetd, /usr/adm/inetd.sec file, 2-8 - 2-9, 2-19 - 2-20
 - TFTP server, enforcing strict security, 7-8
- Selector, 3-16
 - Level, 3-17
- Sendmail, 6-1
 - Default processing of header lines, 6-61
 - Special processing of standard header lines, 6-62
 - :include: specifications, 6-41
 - Address resolution and aliasing, 6-98
 - Address rewriting, 6-80
 - Alias Database, 6-38, 6-43
 - Aliases, 6-19
 - Aliasing, 6-38
 - Aliasing loops, 6-43
 - Aliasing to command lines, 6-40
 - Aliasing to filename, 6-40
 - Class definition from program output, 6-54
 - Cluster, on a, 6-27
 - Collecting message from argument vector, 6-8
 - Collecting message from stdin, 6-8
 - Collecting message from the queue, 6-10
 - Conditionals, 6-56
 - Configuration consistencies, 6-97
 - Configuration file syntax, 6-51
 - Contacting the daemon, 6-100
 - Daemon, 6-24
 - Default routing, 6-16
 - Defining classes, 6-52
 - Defining headers, 6-60
 - Defining macros, 6-55

- Defining mailers, 6-65
- Defining precedence, 6-79
- Defining trusted users, 6-80
- Delivery agent argv field, 6-71
- Delivery agent eol field, 6-71
- Delivery agent flags field, 6-66
- Delivery agent path field, 6-66
- Delivery agent sender and recipient fields, 6-70
- Envelope, 6-9
- Examples to send, 6-21
- Features, 6-5
- File class definition, 6-53
- Freezing configuration file, 6-26
- Function, 6-4
- Header address rewriting, 6-84
- Header lines, default processing, 6-61
- How it works, 6-6
- Installation guidelines, 6-13
- Installing, 6-13
- Installing configuration file, 6-15
- Installing manually, 6-14
- Invoking, 6-6
- Invoking as a daemon, 6-20
- Key terms, 6-2
- Literal class definition, 6-53
- Load limit, 6-31
- Local mailing, 6-21
- Logging, 6-101
- Macro interpolation, 6-55
- Macros defined at run time, 6-57
- Macros, required, 6-56
- Mail error handling, 6-12, 6-49
- Mail queue, 6-28
- Mail queue file format, 6-31
- Mailer flags, 6-66
- Mailer name, 6-66
- Making sendmail executable, 6-18
- Message collection, 6-8
- Message delivery, 6-99
- Message routing, 6-6
- MX failures, 6-48
- MX records, 6-46
- MX records, use of, 6-46 - 6-47
- Operating, 6-24
- Option specifications, 6-72
- Overview, 6-4
- Permanent failures, 6-12, 6-49
- Printing the mail queue, 6-34
- Queue priorities, 6-30
- Recipient address rewriting, 6-82
- Remote mailing via SMTP, 6-23
- Remote mailing via UUCP, 6-22
- Rewriting rulesets, 6-85
- RFC 822, 6-19
- Routing messages, 6-11
- Routing to local destinations, 6-12
- Routing to remote systems, 6-11
- Rulesets, left side, 6-86
- Rulesets, right side, 6-88
- SMTP, Using, 6-9
- Special header lines, 6-64
- System log, 6-35
- System-log log levels, 6-36
- Temporary failures, 6-12, 6-49
- Testing address rewriting rulesets, 6-93
- Token metasymbols, 6-86, 6-88
- Troubleshooting, 6-97
- User-controlled aliasing, 6-42
- Writable alias databases, 6-41
- Servers
 - remshd, 3-6
 - rexecd, 3-8
 - See BOOTP server
 - fingerd, 3-9
 - ftpd, 3-10
 - rlogind, 3-7
 - telnetd, 3-9
 - See TFTP server
- Service Request (SR), 8-20 - 8-21
- Subdomains, 4-36

Syslogd, 3-16, 4-41
 Maintaining files, 3-17
System Administration Manager
 See SAM
System log files, 3-16 - 3-19
 Syslogd, 3-16

T

Telnetd server, 3-9
TFTP (Trivial File Transfer Protocol)
 See TFTP server
TFTP server
 Configuring tftpd manually, overview, 7-7
 Configuring tftpd manually, procedure for, 7-7
 Configuring tftpd, with SAM, 7-5
 File transfer options, 7-22
 File transfer, process of, 7-4
 Introduction, chapter overview, 7-1
 Key terms, 7-3
 Nonsupported products, 7-1
 Options, for tftp file transfers, 7-22
 SAM configuration, overview, 7-5
 SAM configuration, procedure for, 7-5
 Security, strictly enforcing, 7-8
 tftp command, options, 7-22
 tftpd, common problems, 7-31
 tftpd, TFTP server implementation, 7-4
 Troubleshooting, common tftpd problems, 7-31
 Troubleshooting, helpful hints, 7-24
 Troubleshooting, overview, 7-24
 Verifying manual tftpd configuration, 7-9
Tracing
 Network, 8-3
Trivial File Transfer Protocol (TFTP)
 See TFTP server
Troubleshooting, 8-1
 ARPA Services, 8-1, 8-8
 BIND Name Server, 4-40

Data communications, 8-1
Gated, 5-20 - 5-24
Network interface configuration, 8-1
Networks using repeaters or gateways, 8-5
- 8-6
 See also: Problems
 Sendmail, 6-97

U

Update program, 2-3
 /usr/adm/inetd.sec file, 2-19
 Edit, 2-19, 2-21
 Format, 2-22
 Permissions, 2-24
 /usr/lib/aliases, 6-14, 6-19 - 6-20, 6-38, 6-44
 /usr/lib/aliases.pag, 6-44
 /usr/lib/sendmail, 6-14
 /usr/lib/sendmail.cf, 6-9, 6-14, 6-16, 6-26
 /usr/lib/sendmail.fc, 6-26

Reader Comment Card

**HP 9000 Computers
Administering ARPA Services
B1014-90008 E1092**

We welcome your evaluation of this manual. Your comments and suggestions will help us improve our publications. Please tear this card out and mail it in. Use and attach additional pages if necessary.

Please circle the following Yes or No:

- | | | |
|--|-----|----|
| • Is this manual well organized? | Yes | No |
| • Is the information technically accurate? | Yes | No |
| • Are instructions complete? | Yes | No |
| • Are concepts and wording easy to understand? | Yes | No |
| • Are examples and pictures helpful? | Yes | No |
| • Are there enough examples and pictures? | Yes | No |

Comments: _____

Name: _____

Title: _____

Company: _____

Address: _____

City & State: _____

Zip: _____





NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

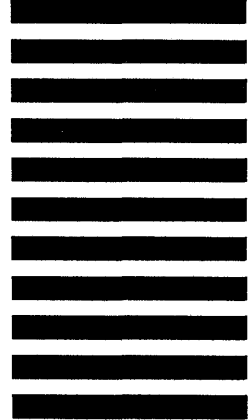
BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.

POSTAGE WILL BE PAID BY ADDRESSEE

**Hewlett-Packard Company
Information Networks Division
19420 Homestead Road
Cupertino, CA 95014**

ATTN: Network Usability Department



Fold Here

Tape

Please do not staple

Tape

Customer Order No.
None

Copyright © 1992
Hewlett-Packard Company
Printed in USA 10/92

Manufacturing No.
B1014-90008
Mfg. number is for HP internal use only



B1014-90008