HP 9000 Networking

Installing and Administering Internet Services

HP Part No. B2355-90110
Printed in U.S.A.
E0696

Edition 5

HEWLETT® PACKARD

Legal Notices

Hewlett-Packard Co.
19420 Homestead Road
Cupertino, CA 95014 USA

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

1

Product Overview

The HP 9000 Internet Services enable your HP 9000 computer to transfer files, log into remote hosts, execute commands remotely, and exchange mail with remote hosts on the network. The Internet Services product was previously called the ARPA Services.

A link product, such as LAN/9000 or X.25/9000, must be installed for the Internet Services 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 the Internet Services.

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

The Internet Services

The HP 9000 Internet Services product combines services developed by the University of California at Berkeley (UCB), Cornell University, Carnegie-Mellon University (CMU), and Hewlett-Packard.

ARPA Services include the set of services developed by UCB for the Advanced Research Projects Agency (ARPA): **`ftp`**, and **`telnet`**. 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: BIND, **`sendmail`**, **`finger`**, the **`rexec`** library, **`rcp`**, **`rlogin`**, **`remsh`**, **`ruptime`**, **`rwho`**, and **`rdist`**. Berkeley Services are used to communicate with HP-UX or UNIX systems.

The Internet Services product also contains several other services: BOOTP, **`tftp`**, **`rbootd`**, NTP, and DDFA.

For more information on the Internet Services, see *TCP/IP Network Administration* by Craig Hunt, published by O'Reilly and Associates.

For more information on DNS and BIND, see *DNS and BIND*, by Paul Albitz and Cricket Liu, published by O'Reilly and Associates, Inc.

For more information on **`sendmail`**, see *sendmail*, by Bryan Costales with Eric Allman and Neil Richert, published by O'Reilly and Associates, Inc.

Table 1 lists the Internet Services.

Table 1 The Internet Services

ftp	Copies files among hosts on the network that support Internet Services. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 1 ftp** or **man 1M ftpd**.
telnet	Allows you to log onto a remote host that supports Internet Services. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 1 telnet** or **man 1M telenetd**.
sendmail	Works with your network’s mailers (for example, **elm** and **mailx**) to perform internetwork mail routing among UNIX and non-UNIX hosts on the network. For more information, see “Installing and Administering sendmail” on page 159, or type **man 1M sendmail**.
BIND	Implements the Domain Name System (DNS). The Berkeley Internet Name Domain (BIND) Service, is a distributed database service that resolves host names and facilitates internetwork mail. For more information, see “Configuring and Administering the BIND Name Service” on page 95, or type **man 1M named**.
finger	Allows users to look up information about other users on the network. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 1 finger** or **man 1M fingerd**.
BOOTP	Allows some diskless systems, such as the HP 700/X terminal, to load network and configuration parameters from a server on the network. For more information, see “Configuring TFTP and BOOTP Servers” on page 213, or type **man 1M bootpd**.
tftp	Used with **bootp** to allow some diskless systems, such as the HP 700/X terminal, to transfer files containing bootstrap code, fonts, or other configuration information. For more information, see “Configuring TFTP and BOOTP Servers” on page 213, or type **man 1 tftp** or **man 1M tftpd**.
gated	Dynamically determines routing over internets from one node to another. For more information, see “Configuring gated” on page 289, or type **man 1M gated**.
mrouted	Implements the Distance-Vector Multicast Routing Protocol (DVMRP) for routing IP multicast datagrams. For more information, see “Configuring mrouted” on page 355, or type **man 1M mrouted**.
NTP	Maintains the local clock on an HP-UX workstation in agreement with Internet-standard time servers. For more information, see “Configuring NTP” on page 261, or type **man 1M xntpd**.
rexec	A library routine used to execute commands on a remote UNIX host on the network. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 3N rexec** or **man 1M rexecd**.

Table 1 The Internet Services

rcp	Allows you to transfer files between UNIX hosts on the network. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 1 rcp**.
rlogin	Allows you to log onto a remote UNIX host. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 1 rlogin** or **man 1M rlogind**.
remsh	Allows you to execute commands on a remote UNIX host. **remsh** is the same command as **rsh** in 4.3 BSD. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 1 remsh** or **man 1M remshd**.
ruptime	Lists information about specified UNIX nodes that are running the **rwhod** daemon. **ruptime** is not supported over X.25 networks or networks using the PPL (SLIP) product. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 1 ruptime** or **man 1M rwhod**.
rwho	Lists information about specified UNIX nodes that are running the **rwhod** daemon. **rwho** is not supported over X.25 networks or networks using the PPL (SLIP) product. For more information, see “Installing and Configuring Internet Services” on page 25, or type **man 1 rwho** or **man 1M rwhod**.
rdist	Distributes and maintains identical copies of files across multiple hosts. For more information, see “Using rdist” on page 371, or type **man 1 rdist**.
rbootd	RMP is an HP-proprietary boot and file transfer protocol used in early Series 700 workstations and in the Datacommunications and Terminal Controllers (DTC/9000). For more information, see “Configuring TFTP and BOOTP Servers” on page 213, or type **man 1M rbootd**.
whois	Lists information about specified people and organizations listed in the Network Information Center (NIC) database. A direct socket connection to the NIC is required. For more information, type **man 1 whois**.
DDFA	Allows access from HP-UX systems and user-written applications to HP DTCs. For more information, see the *DTC Device File Access Utilities* Manual.
Secure Internet Services	An optionally installable product that includes the following services: **ftp**, **rcp**, **remsh**, **rlogin** and **telnet**. The alternate versions of these services have been enhanced to incorporate Kerberos Version 5 Beta 4 authentication and authorization. For more information, see “Secure Internet Services” on page 63.

Military Standards and Request for Comment Documents

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

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

To obtain information about available RFCs, contact the following:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021
phone: (703) 802-8400

You can also obtain copies of RFCs by anonymous **ftp**, from **venera.isi.edu**. The RFCs are in the directory **in-notes** under the anonymous **ftp** directory. The RFC files are called **rfc###.txt**, where **###** is the number of the RFC.

Also, the following RFCs are located in the **/usr/share/doc** directory:

1034: “Domain Names—Concepts and Facilities”
1035: “Domain Names—Implementation and Specification”
1535: “A Security Problem and Proposed Correction
With Widely Deployed DNS Software”

2

Installing and Configuring Internet Services

This chapter describes how to install the Internet Services and configure them for your system. It contains the following sections:

- Updating Your Network Map
- Installing the Internet Services Software
- Configuring the Name Service Switch
- Configuring Internet Addresses
- Configuring the Internet Daemon, inetd
- Configuring rwhod, the Server for rwho and ruptime
- Configuring Logging for the Internet Services
- Configuring Anonymous ftp Access
- Restricting ftp Access with /etc/ftpusers
- Configuring Files to Bypass Security

Updating Your Network Map

Before you install the Internet Services, take the time to update your network map to indicate that the Internet Services are installed on your node. A network map provides 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.

See the *Installing and Administering LAN/9000 Software* manual for information about creating and maintaining a network map.

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

Installing the Internet 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 Internet Services software must all be the same version. You can check your HP-UX operating system version with the **`uname -r`** command.

Use the HP-UX Software Distributor (SD) to install the Internet Services file set. Issue the following command to start the SD **`swinstall`** utility:

```
/usr/sbin/swinstall
```

The Software Distributor is documented in *Managing HP-UX Software with SD-UX*.

Configuring the Name Service Switch

The Name Service Switch determines where your system will look for the information that is traditionally stored in the following files:

```
/etc/hosts
/etc/protocols
/etc/services
/etc/networks
/etc/netgroup
/etc/rpc
```

For all types of information except host information, you can configure your system to use NIS (one of the NFS Services), the local **/etc** file, or both, in any order. For host information, you can configure your system to use BIND (DNS), NIS, the **/etc/hosts** file, or any combination of the three, in any order.

The default Name Service Switch configuration is adequate for most installations, so you probably do not have to change it. The default configuration is explained in “Default Configuration” on page 32.

NOTE: Configuring the Name Service Switch is a separate task from configuring the name services themselves. You must also configure the name services before you can use them. The Name Service Switch just determines which name services are queried and in what order.

The ability to consult more than one name service for host information is often called **hostname fallback**. The Name Service Switch provides **client-side hostname fallback**, because it is incorporated into client-side programs (for example, **gethostbyname**), which request host information.

The Network Information Service (NIS), one of the NFS Services, allows you to configure a **server-side hostname fallback**. This feature causes the NIS server to query BIND when it fails to find requested host information in its database. The NIS server then returns the host information to the client through NIS. This server-side hostname fallback is intended for use with clients like PCs that do not have a feature like the Name Service Switch. Hewlett-Packard recommends that you use the Name Service Switch if possible, instead of the server-side hostname fallback provided by NIS. For more information on the NIS server-side hostname fallback, see *Installing and Administering NFS Services*.

You can use SAM to configure the Name Service Switch. Type **sam** at the HP-UX prompt.

Following are some suggestions for customizing your Name Service Switch configuration:

- If you want your system to consult the local **/etc/netgroup** file when it fails to find a netgroup in the NIS **netgroup** database, create or modify the **netgroup** line in the **/etc/nsswitch.conf** file as follows:

```
netgroup: nis [NOTFOUND=continue] files
```

- If you want your system to consult BIND (DNS) when it fails to find a host name in NIS, create or modify the **hosts** line in the **/etc/nsswitch.conf** file as follows:

```
hosts: nis [NOTFOUND=continue] dns files
```

  With this configuration, if NIS does not contain the requested information, and BIND is not configured, the **/etc/hosts** file is consulted.

- If you want your system to consult NIS if it fails to find a host name in BIND or if the BIND name servers are not responding, create or modify the **hosts** line in the **/etc/nsswitch.conf** file as follows:

```
hosts: dns [NOTFOUND=continue TRYAGAIN=continue] nis files
```

  With this configuration, if BIND does not return the requested information, and NIS is not running, the **/etc/hosts** file is consulted.

HP recommends that you maintain at least a minimal **/etc/hosts** file that includes important addresses like gateways, diskless boot servers and root servers, and your host's own IP address. HP also recommends that you include the word **files** in the **hosts** line to help ensure a successful system boot using the **/etc/hosts** file when BIND and NIS are not available.

CAUTION: Changing the default configuration can complicate troubleshooting. The default configuration is designed to preserve the authority of the name service you are using. It switches from BIND to NIS only if BIND is not enabled. It switches from NIS to the local **/etc** file only if NIS is not enabled. It is very difficult to diagnose problems when multiple name servers are configured and enabled for use.

Default Configuration

A default **nsswitch.conf** file is supplied in the **/usr/newconfig/etc** directory. It contains the following lines:

```
hosts:      dns  nis  files
protocols: nis  files
services:  nis  files
networks:  nis  files
netgroup:  nis  files
rpc:        nis  files
```

This is the default configuration. In other words, if you copy **/usr/newconfig/etc/nsswitch.conf** to **/etc/nsswitch.conf**, the Name Service Switch behaves the same way it would if no **/etc/nsswitch.conf** file existed.

Figure 1 illustrates the default behavior of the Name Service Switch for host information lookups.

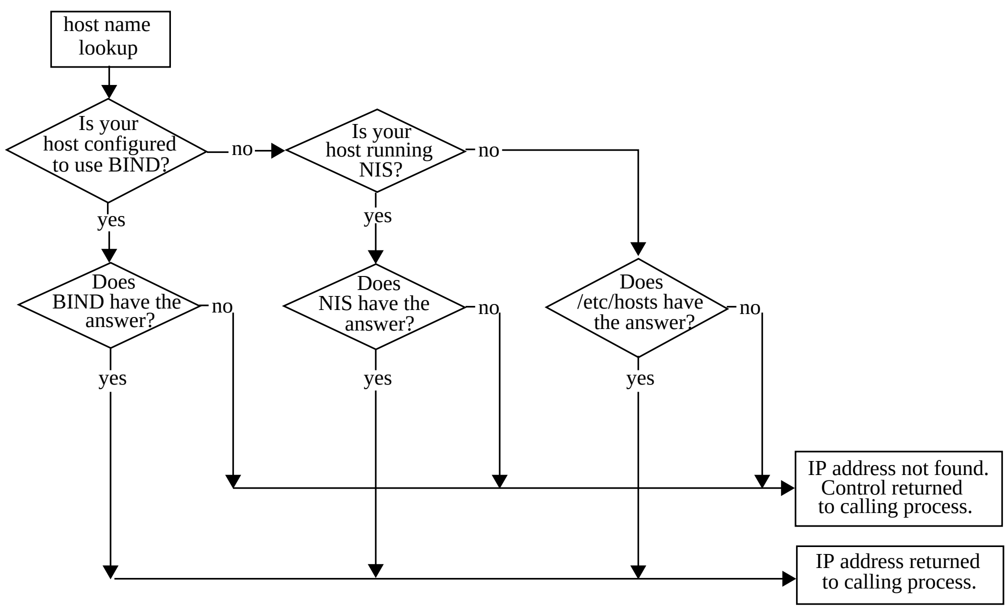


Figure 1 **Default Behavior of the Name Service Switch**

The `/etc/nsswitch.conf` File

The configuration file for the Name Service Switch is **`/etc/nsswitch.conf`**, which consists of lines with the following syntax:

```
info_type: source [status=action status=action...] source ...
```

Table 2 displays the possible values for each variable.

Table 2 Values for Variables in the /etc/nsswitch.conf File

info_type	A type of configuration information. Possible values are as follows:	
	hosts	Host names and IP addresses, as in **/etc/hosts**.
	protocols	Protocol names and numbers, as in **/etc/protocols**.
	services	Service names, port numbers, and protocols, as in **/etc/services**.
	networks	Network names and IP addresses, as in **/etc/networks**.
	netgroup	NFS netgroup names and members, as in **/etc/netgroup**.
	rpc	RPC program names and numbers, as in **/etc/rpc**.
source	A name service where information can be found. Possible values are as follows:	
	dns	Berkeley Internet Name Domain (BIND), the Berkeley implementation of the Domain Name System (DNS).
	nis	The Network Information Service (NIS), one of the NFS Services.
	files	The appropriate **/etc** file (**/etc/hosts**, **/etc/services**, etc.).
status	The result of querying the ***source***. Possible values are as follows:	
	SUCCESS	The query was successful, and the information was found.
	NOTFOUND	The ***source*** responded to the query, indicating that it did not have the requested information.
	UNAVAIL	The query failed, because the ***source*** is not configured on your local system, or because the server system is not running the name service.
	TRYAGAIN	The query failed or timed out because the server system is not responding.

Table 2 **Values for Variables in the /etc/nsswitch.conf File**

action	The action to be taken based on the ***status*** of the query. Possible values are as follows:	
	return	End the search and return control to the calling process, without querying the next ***source*** in the list.
	continue	Continue the search by querying the next ***source*** in the list.

If you specify any ***status*=*action*** pairs, the set of ***status*=*action*** pairs for each source must be enclosed in square brackets [ ].

If the **/etc/nsswitch.conf** file does not exist, or if no source is specified in it, the default search order is as follows:

1 DNS (for host information only)
2 NIS
3 local **/etc** file

The default ***status*=*action*** pairs are as follows:

```
SUCCESS=return
NOTFOUND=return
UNAVAIL=continue
TRYAGAIN=return
```

The default search order for host information is shown in Figure 1 on page 32.

For more information on the Name Service Switch, type **man 4 switch** at the HP-UX prompt.

To Check the Syntax of the `hosts` Line

To check the syntax of the **`hosts`** line in **`/etc/nsswitch.conf`** file, start **`nslookup`** with the **`swdebug`** option, as follows:

```
nslookup -swdebug
```

You will see the output of the parser as it reads the **`hosts`** line in your **`nsswitch.conf`** file. If your **`hosts`** line is syntactically correct, you will see the line **`__nsw_getconfig: PARSE SUCCESSFUL`**. If your **`hosts`** line contains a syntax error, you will see the line **`__nsw_getconfig: ERR-SYNTAX ERROR`**.

The following example checks the syntax of a **`hosts`** line that is missing a closing square bracket:

```
# cat /etc/nsswitch.conf
hosts: dns [notfound=continue] nis [notfound=continue files

# nslookup -swdebug
__nsw[/etc/nsswitch.conf] 1->hosts: dns [notfound=continue] nis [notf
ound=continue files
__nsw[/etc/nsswitch.conf]LS->L<hosts>L<:>L<dns>L<[>L<notfound>L<=>L<c
ontinue>L<]>L<nis>L<[>L<notfound>L<=>L<continue>L<files>^Missing =^
__nsw.error_recovery: ERR- Error Recovery Completed
__nsw_getconfig: ERR- SYNTAX ERROR
__nsw_getdefault: default hosts lookup policy
Default Name Server:  hpindbu.cup.hp.com
```

The parser indicates the error with carats (^). In this case, the parser reads the word **`files`** as another status following **`notfound=continue`**, because it has not encountered a closing square bracket. If the word **`files`** were a status, it must be followed by an equal sign, and it is not. So the parser displays the message **`^Missing =^`**.

NOTE: The parser checks only the position of the elements with respect to the delimiters **`:`**, **`[`**, and **`]`**. It does not check the spelling of all the elements. For example, if you type **`dsn`** instead of **`dns`**, you receive the **`PARSE SUCCESSFUL`** message. However, when you attempt a host name lookup, **`dsn`** is not a known name service, so DNS is not queried, and the lookup switches to the next configured source.

To Check the Current `hosts` Configuration

To check the Name Service Switch configuration that your system is currently using for host information, start **nslookup** and issue the **policy** command, as follows:

```
# nslookup
> policy
```

The output for the default configuration is as follows:

```
# Lookups = 3
dns [RRCR]     nis [RRCR]     files [RRRR]
```

The letters in square brackets stand for (**R**)eturn or (**C**)ontinue. They represent the values of the four status values, **SUCCESS**, **NOTFOUND**, **UNAVAIL**, and **TRYAGAIN**. In the example, the ***status*=*action*** pairs configured for **dns** and **nis** are

```
SUCCESS=return
NOTFOUND=return
UNAVAIL=continue
TRYAGAIN=return
```

For the following **hosts** line

```
hosts:  dns [NOTFOUND=continue]  files
```

the **policy** command displays the following:

```
# Lookups = 2
dns [RCCR]     files [RRRR]
```

To stop the **nslookup** program, type **exit**.

To Trace a Host Name Lookup

To trace a host name lookup, start **`nslookup`**, set the **`swtrace`** option, and perform a lookup, as follows:

```
# nslookup
> set swtrace
> hostname
```

For the **`nsswitch.conf`** file containing the **`hosts`** line

```
hosts: dns [NOTFOUND=continue] nis [NOTFOUND=continue] files
```

the following example tries all three name services before it finds an answer:

```
# nslookup
> set swtrace
> romney
Name Server: hpindbu.cup.hp.com
Address: 15.13.104.13

lookup source is DNS
Name Server: hpindbu.cup.hp.com
Address: 15.13.104.13

*** hpindbu.cup.hp.com can't find romney: Non-existent domain

Switching to next source in the policy
lookup source is NIS
Default NIS Server: hpntc43c
Address: 15.13.119.52
Aliases: hpntc43c.cup.hp.com, hpntc43c-119, 3c-119

*** No address information is available for "romney"

Switching to next source in the policy
lookup source is FILES
Using /etc/hosts on: hpntc2k

Name: romney
Address: 15.13.104.128
```

NOTE: If you do not set **`swtrace`**, **`nslookup`** displays only the first name service where it looks for a host, even if it finds the host in another name service.

Configuring Internet Addresses

This section tells you how to configure your host to find other hosts on the network, by host name or IP address. It contains the following sections:

- To Choose a Name Service
- To Edit the /etc/hosts File
- To Configure Routes
- To Change a Host’s IP Address

To Choose a Name Service

HP-UX provides three ways of translating host names to IP addresses or IP addresses to host names:

- The **/etc/hosts** file, a simple ASCII file that is searched sequentially.
- BIND (Berkeley Internet Name Domain), which is Berkeley's implementation of the Domain Name System (DNS).
- NIS (Network Information Service), one of the NFS Services. (NIS used to be called "Yellow Pages".)

By configuring the Name Service Switch, you can use these name services in any order you choose. See "Configuring the Name Service Switch" on page 29.

If you have a large network, or you need to connect to Internet hosts outside your local network, use BIND as your primary name service. When you use BIND, you administer a central database containing only the hosts on your local network, and you have access to the databases on all the other hosts on the Internet. See Chapter 4 for instructions on configuring BIND.

If you have a large network and little need for Internet connectivity, you can use NIS as your primary name service. The NIS hosts database is administered centrally on one of your hosts, but it must contain the names and IP addresses of all the other hosts in your network. For information on NIS, see *Installing and Administering NFS Services*.

If you have a small network and little need for Internet connectivity, you can use the **/etc/hosts** file as your primary name service. Each host in your network needs a copy of the **/etc/hosts** file containing the names and addresses of all the other hosts in your network. For information on the **/etc/hosts** file, see "To Edit the /etc/hosts File" on page 40.

If you choose to use BIND or NIS as your primary name service, you still need to configure a minimal **/etc/hosts** file so that your host can boot if BIND or NIS is not available.

To Edit the `/etc/hosts` File

You can use any text editor to edit the **/etc/hosts** file. If you are not running BIND or NIS, you can use SAM. SAM (System Administration Manager) is Hewlett-Packard's windows-based user interface for performing system administration tasks. To run SAM, type **sam** at the HP-UX prompt. SAM has an extensive online help facility.

1 If no **/etc/hosts** file exists on your host, copy **/usr/newconfig/etc/hosts** to **/etc/hosts**, or use **ftp** to copy the **/etc/hosts** file to your host from another host on your network. Type **man 1 ftp** for more information.

2 Make sure your **/etc/hosts** file contains the following line:

```
127.0.0.1        localhost        loopback
```

3 Add your own host's IP address, name, and aliases to the **/etc/hosts** file, as in the following example:

```
15.13.131.213    hpindlpk    romney
```

The first field is the IP address, the second is the official host name (as returned by the **hostname** command), and any remaining fields are aliases. Type **man 4 hosts** for more information.

4 If your host has more than one network interface installed, add a line to **/etc/hosts** for each interface. The **/etc/hosts** entries for your host will have the same official host name but different aliases and different IP addresses.

5 Add any other hosts to the **/etc/hosts** file that you need to reach. If you will use a BIND or NIS server on a different host, add that host to your **/etc/hosts** file.

If you have no default gateway configured, and you add a host that is not on your subnet, SAM will prompt you for the gateway. To stop the prompting, configure a default gateway.

6 If you are not using SAM, you must configure a gateway for each host that is not on your subnet. See "To Configure Routes" on page 41.

7 Make sure the **/etc/hosts** file is owned by user **root** and group **other**, and make sure the permissions are set to 0444 (**-r--r--r--**).

To Configure Routes

1 If you use only one gateway to reach all systems on other parts of the network, configure a default gateway.

You can use SAM to configure a default gateway, or if you are not using SAM, issue the following command:

```
/usr/sbin/route add default gateway_address 1
```

where ***gateway_address*** is the IP address of the gateway host.

Then, set the following environment variables in the **/etc/rc.config.d/netconf** file:

```
ROUTE_DESTINATION[0]="default"
ROUTE_GATEWAY[0]="gateway_address"
ROUTE_COUNT[0]="1"
```

If the default gateway is your own host, set the **ROUTE_COUNT** variable to 0. Otherwise, set it to 1.

2 If your host is a gateway, configure the destination networks that can be reached from its network interfaces. Issue the following command for each network interface on your host:

```
/usr/sbin/route add net destination IP_address
```

where ***destination*** is a network address reachable by your host, and ***IP_address*** is the address of the network interface.

Then, create a new set of routing variables in the **/etc/rc.config.d/netconf** file for each network interface. Whenever you create a new set of variables, increment the number in square brackets, as in the following example:

```
ROUTE_DESTINATION[1]="15.13.131.0"
ROUTE_GATEWAY[1]="15.13.131.213"
ROUTE_COUNT[1]="0"
```

3 If you will not be using **gated**, configure routes to all the networks you need to reach. Type the following command for each network you need to reach from your host:

```
/usr/sbin/route add net network_address gateway_address
```

Then, create a new set of routing variables in the **/etc/rc.config.d/netconf** file for each new route. Whenever you create a new set of variables, increment the number in square brackets.

```
ROUTE_DESTINATION[n]="network_address"
ROUTE_GATEWAY[n]="gateway_address"
ROUTE_COUNT[n]="1"
```

If **ROUTE_GATEWAY[*n*]** is your own host, set **ROUTE_COUNT[*n*]** to 0. Otherwise, set it to 1.

4 Type the following command to verify the routes you have configured:

```
/usr/bin/netstat -r
```

For more information on static routing, type **man 1M route** or **man 7 routing** at the HP-UX prompt.

If you have a large and complicated network, use **gated** for dynamic routing. See Chapter 9 for more information.

To Change a Host's IP Address

When you use SAM to change a host's IP address, SAM does *not* perform all these steps. For example, SAM does not update BIND or NIS databases.

1. Change the host's IP address in the **`/etc/hosts`** file. See "To Edit the /etc/hosts File" on page 40.
2. Change the **`IP_ADDRESS[`*`n`*`]`** variable in the **`/etc/rc.config.d/netconf`** file to the new IP address.
3. If the host is on a network that uses BIND, change the host's IP address in the data files of the authoritative name servers. See "Configuring and Administering the BIND Name Service" on page 95.

   If the host is on a network that uses NIS, change its IP address in the **`/etc/hosts`** file on the NIS master server, and issue the following commands to regenerate the **`hosts`** database and push it out to the NIS slave servers:

   ```
   cd var/yp
   /usr/ccs/bin/make hosts
   ```

4. If the host is moving to a different subnet, change the **`ROUTE_DESTINATION`**, **`ROUTE_GATEWAY`**, and **`BROADCAST_ADDRESS[`*`n`*`]`** variables in **`/etc/rc.config.d/netconf`**.

   If the host is moving to a nework that uses a different subnet mask, change the **`SUBNET_MASK[`*`n`*`]`** variable in **`/etc/rc.config.d/netconf`**.
5. If the host is moving to a different network, you may have to configure new routes for it. See "To Configure Routes" on page 41.
6. If the host is on a network that uses **`gated`**, change its IP address on all the **`gated`** routers. See "Configuring gated" on page 289.
7. If the host is a BOOTP client, change its IP address in the **`/etc/bootptab`** file on the BOOTP server. If the host is a BOOTP server, and a BOOTP relay agent is configured to relay boot requests to the host, change the host's IP address in the **`/etc/bootptab`** file on the BOOTP relay agent. See "Configuring TFTP and BOOTP Servers" on page 213.
8. If the host is an NTP server, change its IP address in the **`/etc/ntp.conf`** file on NTP clients. If the host is an NTP client and is moving to another network, you might have to configure a different NTP server in its **`/etc/ntp.conf`** file. See "Configuring NTP" on page 261.
9. Reboot the host.

Configuring the Internet Daemon, inetd

The internet daemon, **/usr/sbin/inetd**, is the master server for many of the Internet Services. The **inetd** daemon listens for connection requests for the services listed in its configuration file and starts up the appropriate server when it receives a request.

The **inetd** daemon is always started as part of the boot process, by the startup script **/sbin/init.d/inetd**.

The **/etc/inetd.conf** file is the **inetd** configuration file, which lists the services that may be started by **inetd**. In addition to the configuration file, you can configure an optional security file called **/var/adm/inetd.sec**, which restricts access to the services started by **inetd**.

This section gives instructions for completing the following tasks:

- To Edit the /etc/inetd.conf File
- To Edit the /var/adm/inetd.sec File

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

To Edit the /etc/inetd.conf File

1 Make sure the following lines exist in **/etc/inetd.conf**. If any of the lines starts with a sharp sign (#), remove the sharp sign to enable the service.

```
ftp    stream tcp nowait root /usr/lbin/ftpd    ftpd -l
telnet stream tcp nowait root /usr/lbin/telnetd telnetd
tftp   dgram  udp wait   root /usr/lbin/tftpd   tftpd
bootps dgram  udp wait   root /usr/lbin/bootpd  bootpd
finger stream tcp nowait bin  /usr/lbin/fingerd fingerd
login  stream tcp nowait root /usr/lbin/rlogind rlogind
shell  stream tcp nowait root /usr/lbin/remshd  remshd
exec   stream tcp nowait root /usr/lbin/rexecd  rexecd
```

To disable any of these services, comment out the line by typing a sharp sign (#) as the first character on the line.

2 If you made any changes to **/etc/inetd.conf**, type the following command to force **inetd** to read its configuration file:

```
/usr/sbin/inetd -c
```

3 Make sure **/etc/inetd.conf** is owned by user **root** and group **other**, and make sure its permissions are set to 0444 (**-r--r--r--**).

For more information, type **man 4 inetd.conf** or **man 1M inetd**.

To Edit the `/var/adm/inetd.sec` File

The **/var/adm/inetd.sec** file is a security file that **inetd** reads to determine which remote hosts are allowed access to the services on your host. The **inetd.sec** file is optional; you do not need it to run the Internet Services.

You can use either a text editor or SAM to edit the **inetd.sec** file. SAM (System Administration Manager) is Hewlett-Packard's windows-based user interface for performing system administration tasks. To run SAM, type **sam** at the HP-UX prompt. SAM has an extensive online help facility.

1. If the **/var/adm/inetd.sec** file does not exist on your host, copy **/usr/newconfig/var/adm/inetd.sec** to **/var/adm/inetd.sec**.
2. Create one line in **inetd.sec** for each service to which you want to restrict access. Do not create more than one line for any service.

   Each line in the **/var/adm/inetd.sec** file has the following syntax:

```
service_name {allow} host_specifier [host_specifier...]
             {deny}
```

   where ***service_name*** is the first field in an entry in the **/etc/inetd.conf** file, and **host_specifier** is a host name, IP address, IP address range, or the wildcard character (*).
3. Make sure the **/var/adm/inetd.sec** file is owned by user **root** and group **other**, and make sure its permissions are set to 0444 (**-r--r--r--**).

Following are some example lines from an **inetd.sec** file:

```
login allow 10.*
shell deny vandal hun
tftp deny *
```

The first example allows access to **rlogin** from any IP address beginning with 10. The second example denies access to **remsh** and **rcp** from hosts **vandal** and **hun**. The third example denies everyone access to **tftp**.

Only the services configured in **/etc/inetd.conf** can be configured in **/var/adm/inetd.sec**.

For more information, type **man 4 inetd.sec** or **man 1M inetd**.

Configuring rwhod, the Server for rwho and ruptime

The **rwhod** daemon checks the state of your host and generates status messages, which it broadcasts on the network every 180 seconds. It also listens for status messages broadcast by **rwhod** daemons on remote hosts, and it records these messages in a database of files in **/var/spool/rwho**. The files are named **whod.*hostname***, where ***hostname*** is the name of the remote host from which the status information came. The status messages are displayed when users issue the **rwho** or **ruptime** command.

1 In the **/etc/rc.config.d/netdaemons** file, set the **RWHOD** variable to 1.

2 Issue the following command to start the **rwhod** daemon:

```
/sbin/init.d/rwhod start
```

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.

Because UDP (User Datagram protocol) broadcasts do not go through gateways, **rwho** and **ruptime** do not report status for hosts that can be reached only through a gateway.

For more information on **rwhod**, see the following man pages: **rwhod**(1M), **rwho**(1), and **ruptime**(1).

Configuring Logging for the Internet Services

This section tells you how to complete the following tasks:

- To Configure syslogd
- To Maintain System Log Files
- To Configure inetd Connection Logging
- To Configure ftpd Logging

To Configure `syslogd`

The Internet daemons and servers log informational and error messages through **`syslog`**. You can monitor these messages by running **`syslogd`**. You can determine the type and extent of monitoring through **`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 Internet daemons and servers, except **`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 (*).

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           /var/adm/syslog/mail.log
*.info,mail.none     /var/adm/syslog/syslog.log
*.alert              /det/console
*.alert              root
*.emerg              *
```

With this configuration, all mail log messages at the **`debug`** level or higher are sent to **`/var/adm/syslog/mail.log`**. Log messages from any facility at the **`information`** level or higher (but no mail messages) are sent to **`/var/adm/syslog/syslog.log`**. Log messages from any facility at the **`alert`** level or higher are sent to the console and any terminal where the superuser 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, type **man 3C syslog** or **man 1M syslogd** at the HP-UX prompt.

To Maintain 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 your log files as in the following example:

```
cd /var/adm/syslog
mv mail.log mail.log.old
mv syslog.log sylog.log.old
```

2 Restart **syslogd** with the following commands:

```
cd /sbin/init.d
syslogd stop
syslogd start
```

When you reboot your system, each log file is moved to ***filename*.old** automatically, and new log files are started.

To Configure `inetd` Connection Logging

The **`inetd`** daemon can log connection requests through **`syslogd`**. It logs successful connections at the **`information`** level and unsuccessful connection attempts at the **`notice`** level. By default, **`inetd`** starts up with connection logging turned off.

If **`inetd`** is running with connection logging turned off, issue the following command to start it:

```
/usr/sbin/inetd -l
```

If **`inetd`** is running with connection logging turned on, the same command turns it off. For more information, type **`man 1M inetd`**.

To Configure `ftpd` Logging

To configure **`ftpd`** to log messages about logins, login failures, and anonymous **`ftp`** activity, follow these steps:

1. Add the **`-l`** or **`-v`** (verbose) option to the **`ftp`** line in the **`/etc/inetd.conf`** file, as in the following example:

   ```
   ftp stream tcp nowait root /usr/lbin/ftpd ftpd -l
   ```

   The **`-v`** option provides more detailed logging than the **`-l`** option, except for anonymous **`ftp`**. For anonymous **`ftp`**, the **`-l`** and **`-v`** options provide the same level of logging.

2. Issue the following command to force **`inetd`** to read its configuration file:

   ```
   /usr/sbin/inetd -c
   ```

For more information, type **`man 1M ftpd`** at the HP-UX prompt. Included in this man page is a complete list of error messages.

Configuring Anonymous ftp Access

Anonymous **ftp** allows a user without a login on your host to transfer files to and from a public directory. A user types the **ftp** command to connect to your host and types **anonymous** or **ftp** as a login name. The user can type any string of characters as a password. (By convention, the password is the host name of the user's host). The anonymous user is then given access only to user **ftp**'s home directory, usually called **/home/ftp**.

Configuring anonymous **ftp** access involves the following tasks, described in this section:

- To Add User ftp to /etc/passwd
- To Create the Anonymous ftp Directory

You can follow the instructions in this section, or you can use SAM to configure anonymous **ftp** access. SAM (System Administration Manager) is Hewlett-Packard's windows-based user interface for performing system administration tasks. To run SAM, type **sam** at the HP-UX prompt. SAM has an extensive online help facility.

To Add User ftp to /etc/passwd

Use a text editor to add a line for user **ftp** to the **/etc/passwd** file, as in the following example:

```
ftp:*:500:guest:anonymous ftp:/home/ftp:/usr/bin/false
```

The password field should be *, the group membership should be **guest**, and the login shell should be **/usr/bin/false**. In this example, user **ftp**'s user ID is 500, and the anonymous **ftp** directory is **/home/ftp**.

Type **man 4 passwd** at the HP-UX prompt for information on the **passwd** file.

To Create the Anonymous `ftp` Directory

1 Create the **`ftp`** home directory that you configured in the **`/etc/passwd`** file, as in the following example:

```
cd /home
mkdir ftp
```

2 Create the subdirectory **`/usr/bin`** under the **`ftp`** home directory:

```
cd /home/ftp
mkdir usr
cd usr
mkdir bin
```

3 Copy the **`ls`** and **`pwd`** commands from **`/usr/bin`** to **`~ftp/usr/bin`**, and set the permissions on the commands to 0111 (executable only):

```
cp /usr/bin/ls /home/ftp/usr/bin
cp /usr/bin/pwd /home/ftp/usr/bin
chmod 0111 /home/ftp/usr/bin/ls
chmod 0111 /home/ftp/usr/bin/pwd
```

4 Set the owner of the **`~ftp/usr/bin`** and **`~ftp/usr`** directories to **`root`**, and set the permissions to 0555 (not writeable):

```
chown root /home/ftp/usr/bin
chmod 0555 /home/ftp/usr/bin
chown root /home/ftp/usr
chmod 0555 /home/ftp/usr
```

5 Create the subdirectory **`etc`** under the **`ftp`** home directory:

```
cd /home/ftp
mkdir etc
```

6 Copy **`/etc/passwd`** and **`/etc/group`** to **`~ftp/etc`**. These files are required by the **`ls`** command, to display the owners of files and directories under **`~ftp`**.

```
cp /etc/passwd /home/ftp/etc
cp /etc/group /home/ftp/etc
```

7 Replace the password field in all entries in **/home/ftp/etc/passwd** with *, and delete the shell field from the end of each entry:

```
ftp:*:500:guest:anonymous ftp:/home/ftp:
acb:*:8996:20::/home/acb:
```

8 Replace the password field in all entries in **/home/ftp/etc/group** with *:

```
users:*:20:acb
guest:*:21:ftp
```

9 Set the owner of the files in **~ftp/etc** to **root**, and set the permissions to 0444 (read only):

```
chown root /home/ftp/etc/passwd
chmod 0444 /home/ftp/etc/passwd
chown root /home/ftp/etc/group
chmod 0444 /home/ftp/etc/group
```

10 Set the owner of **~ftp/etc** to **root**, and set the permissions to 0555 (not writeable):

```
chown root /home/ftp/etc
chmod 0555 /home/ftp/etc
```

11 Create a directory called **pub** and under **~ftp**. Set its owner to user **ftp** and its permissions to 0777 (writeable by all). Anonymous **ftp** users can put files in this directory to make them available to other anonymous **ftp** users.

```
mkdir /home/ftp/pub
chown ftp /home/ftp/pub
chmod 0777 /home/ftp/pub
```

12 Create a directory called **dist** and under **~ftp**. Set its owner to user **root** and its permissions to 0755 (writeable only by **root**). The superuser can put read-only files in this directory to make them available to anonymous **ftp** users.

```
mkdir /home/ftp/dist
chown root /home/ftp/dist
chmod 0755 /home/ftp/dist
```

13 Set the owner of user **`ftp`**'s home directory to **`root`** and the permissions to 0555 (not writeable).

```
chown root /home/ftp
chmod 0555 /home/ftp
```

An anonymous **`ftp`** directory has the structure shown in Figure 2:

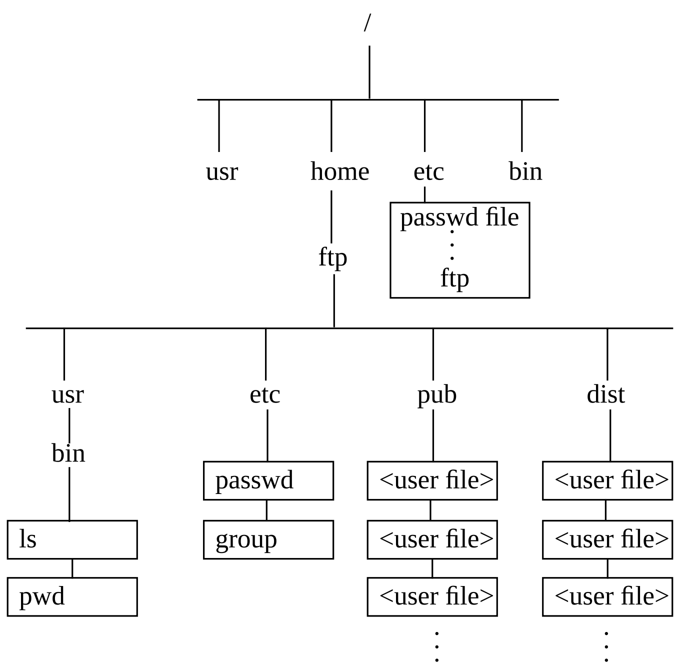


Figure 2 Directory Structure for Anonymous `ftp` Account

Restricting ftp Access with /etc/ftpusers

When a user attempts to log into your system using **ftp**, the **ftpd** daemon checks the **/etc/ftpusers** file. If the file exists, and the user's login name is listed in it, **ftpd** denies access to the user.

User accounts that specify a restricted login shell in **/etc/passwd** should be listed in **/etc/ftpusers**, because **ftpd** accesses local accounts without using their login shells. UUCP accounts should also be listed in **/etc/ftpusers**.

You can use either a text editor or SAM to create and edit the **/etc/ftpusers** file. SAM (System Administration Manager) is Hewlett-Packard's windows-based user interface for performing system administration tasks. To run SAM, type **sam** at the HP-UX prompt. SAM has an extensive online help facility.

Each line in **/etc/ftpusers** consists of a login name with no white space. Following is an example **/etc/ftpusers** file:

```
uucp
guest
nobody
```

For more information, type **man 4 ftpusers** at the HP-UX prompt.

Configuring Files to Bypass Security

The following files may be used to allow users access to your host without supplying a password:

- **`/etc/hosts.equiv`**, a file owned by user **`root`**. This file allows certain users to connect to your host with **`rcp`**, **`remsh`**, or **`rlogin`** without supplying a password.
- **`$HOME/.rhosts`**, a file that may be created by any user in his or her home directory. This file allows certain users to connect to your host with **`rcp`**, **`remsh`**, or **`rlogin`** without supplying a password.
- **`$HOME/.netrc`**, a file that may be created by any user in his or her home directory. This file allows certain users to connect to your host with **`ftp`** or **`rexec`** without supplying a password.

CAUTION: These files create a significant security risk.

The **`remshd`** and **`rlogind`** servers can be configured to ignore **`$HOME/.rhosts`** files. See "To Disable Use of $HOME/.rhosts" on page 60.

To Configure the `/etc/hosts.equiv` File

Each line in the **`/etc/hosts.equiv`** file has the following form:

`hostname` **`[`*`username`*`]`**

You can use either a text editor or SAM to configure the **`/etc/hosts.equiv`** file. To run SAM, type **`sam`** at the HP-UX prompt. SAM has an extensive online help facility.

If a user is logged into a host listed in your **`/etc/hosts.equiv`** file, and the user's login name is listed in your **`passwd`** database, the user may connect to your host with **`rcp`**, **`remsh`**, or **`rlogin`**, and the user will not be prompted for a password.

If a ***`username`*** is included in **`/etc/hosts.equiv`**, only the specified user on the associated host may connect to your host without supplying a password. However, the specified user may log in as any user on your system (except root) without supplying a password.

CAUTION: Hewlett-Packard recommends that you leave user names out of the **`/etc/hosts.equiv`** file, unless you intend to give a user the privilege of logging into all the accounts on the system without having to provide a password.

When a non-root user attempts to log into your host, the **`/etc/hosts.equiv`** file is checked before **`$HOME/.rhosts`**. If an entry is found in **`/etc/hosts.equiv`**, **`$HOME/.rhosts`** is not checked. When a user attempts to log into your host as root, the **`/etc/hosts.equiv`** file is not checked. Only the **`/.rhosts`** file is checked. See "To Configure the $HOME/.rhosts File" on page 59.

The **`/etc/hosts.equiv`** file may contain NFS netgroups. See *Installing and Administering NFS Services* for more information.

The **`/etc/hosts.equiv`** file should be owned by user **`root`**, with permissions set to 0444 (**`-r--r--r--`**).

CAUTION: The **`/etc/hosts.equiv`** file creates a significant security risk.

Type **`man 4 hosts.equiv`** for more information.

To Configure the **$HOME/.rhosts** File

Any user may create a **.rhosts** file in his or her home directory. Each line in the **.rhosts** file has the following form:

hostname **[*username*]**

To create a **.rhosts** file in any home directory other than the superuser's home directory, you must use a text editor. You can use SAM to configure the **/.rhosts** file (in the superuser's home directory). To run SAM, type **sam** at the HP-UX prompt. SAM has an extensive online help facility.

A remote user logged into a host specified in a local **$HOME/.rhosts** file can use **rcp**, **remsh**, or **rlogin** to log into that local user's account without supplying a password.

If your host has a **/.rhosts** file, the root user on any system listed in that file may use **rcp**, **remsh**, or **rlogin** to connect to your host without being prompted for a password.

The **remshd** and **rlogind** servers can be configured to ignore **$HOME/.rhosts** files. See "To Disable Use of $HOME/.rhosts" on page 60.

When a non-root user attempts to connect to your host, the **/etc/hosts.equiv** file is checked before **$HOME/.rhosts**. If an entry is found in **/etc/hosts.equiv**, **$HOME/.rhosts** is not checked. When a user attempts to connect to your host as root, the **/etc/hosts.equiv** file is not checked. Only the **/.rhosts** file is checked.

The **$HOME/.rhosts** file may contain NFS netgroups. See *Installing and Administering NFS Services* for more information.

Each **$HOME/.rhosts** file should be owned by the user of the home directory, with permissions set to 0600 (**-rw-------**). The user's home directory should be write-protected so that no other user can create a **.rhosts** file in it.

CAUTION: The **$HOME/.rhosts** file creates a significant security risk.

Type **man 4 hosts.equiv** for more information.

To Disable Use of `$HOME/.rhosts`

1 Add the **`-l`** option to the lines in **`/etc/inetd.conf`** that begin with **`login`** and **`shell`**, as in the following example:

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

2 Type the following command to force **`inetd`** to read its configuration file:

```
/usr/sbin/inetd -c
```

This procedure disables the use of **`$HOME/.rhosts`** files. It does *not* disable the use of the **`/etc/hosts.equiv`** file.

For more information, type **`man 1M rlogind`** or **`man 1M remshd`**.

To Configure the `$HOME/.netrc` File

Any user may create a **`.netrc`** file in his or her home directory. Each line in the **`.netrc`** file has the following form:

```
machine hostname login remote_login_name password password
```

Following is an example entry in a **`.netrc`** file:

```
machine broccoli login bill password try2Bhave
```

If user **`andrea`** has this entry in her **`.netrc`** file on host **`cabbage`**, she can use **`ftp`** or **`rexec`** to connect to user **`bill`**'s account on host **`broccoli`** without being prompted for a password.

Each **`$HOME/.netrc`** file should be owned by the user of the home directory, with permissions set to 0600 (**`-rw-------`**). The user's home directory should be write-protected so that no other user can create a **`.netrc`** file in it.

The fields in a **`.netrc`** entry may be separated by white space, line breaks, or commas. If you want to include a comma in a field, enclose the whole field in double quotes. For example, if you need to supply both account and user passwords for a login to an MPE/iX machine, enter both passwords in the password field, separated by a comma, and enclose the field in double quotes. Following is an example of a **`.netrc`** entry for an MPE/iX login with both account and user passwords:

```
machine corn login manager.sys password "usrpass,acctpass"
```

CAUTION: The **`$HOME/.netrc`** file creates a significant security risk. It contains unencrypted passwords.

For more information, type **`man 4 netrc`** at the HP-UX prompt.

3

Secure Internet Services

This chapter contains information about the optionally installable Secure Internet Services product, **`InternetSvcSec`**. This product provides alternative versions of the following Internet Services: **`ftp`**, **`rcp`**, **`remsh`**, **`rlogin`**, and **`telnet`**.

These alternative versions of the services incorporate Kerberos Version 5 Beta 4 authentication and authorization and are referred to as the Secure Internet Services.

This chapter includes the following sections:

- Overview of the Secure Internet Services
- Overview of the Secure Environment and the Kerberos V5 Protocol
- Configuration Requirements of the Secure Environment
- Installing and Enabling the Secure Internet Services
- Configuring the Secure Internet Services
- Verifying the Secure Internet Services
- Troubleshooting the Secure Internet Services
- Using the Secure Internet Services
- Sources for Additional Information

Overview of the Secure Internet Services

Network security concerns are becoming increasingly important to the computer system user. The purpose of the Secure Internet Services is to allow the user greater security when running these services.

When an Internet Services client connects to the server daemon, the server daemon requests authentication. The Secure Internet Services authenticate, or in other words validate, the identity of the client and server to each other in a secure way. Also, with the Secure Internet Services, users are authorized to access an account on a remote system by the transmission of encrypted tickets rather than by using the traditional password mechanism. The traditional password mechanism, used with non-secure Internet Services, sends the password in a readable form (unencrypted) over the network. This creates a security risk from intruders who may be listening over the network.

The Secure Internet Services are meant as replacements for their non-secure counterparts. The main benefit of running the Secure Internet Services is that user authorization no longer requires transmitting a password in a readable form over the network. Authorization is the process in which servers verify what access remote users should have on the local system.

The Secure Internet Services may only be used in conjunction with software products that provide a Kerberos V5 Network Authentication Services environment (for example, the HP DCE Security Server). The network authentication mechanism ensures that the local and remote hosts are mutually identified to each other in a secure and trusted manner and that the user is authorized to access the remote account.

For **ftp**/**ftpd**, **rlogin**/**rlogind**, and **telnet**/**telnetd**, the Kerberos V5 authentication involves sending encrypted tickets instead of a readable password over the network to verify and identify the user. Although **rcp**/**remshd**, and **remsh**/**remshd** (used with a command), do not prompt for a password, the secure versions of these services ensure that the user is authorized to access the remote account. (If **remsh** is used with no command specified, **rlogin/rlogind** is invoked.)

If any of the Secure Internet Services are installed in an environment where some of the remote systems on the network are running non-secure versions of the Internet Services, you can use a special command line option to bypass Kerberos authentication to access those remote systems. However, if a password is required to access the system, the password is sent in a readable form over the network.

CAUTION: None of the Secure Internet Services encrypts the session beyond what is necessary to authorize the user or authenticate the service. Thus, these services do not provide integrity-checking or encryption services on the data or on remote sessions.

Overview of the Secure Environment and the Kerberos V5 Protocol

This section gives an overview of the secure environment in which the Secure Internet Services operate, including a simplified overview of the Kerberos V5 authentication protocol and related Kerberos concepts.

Kerberos, originally developed by MIT, refers to an authentication protocol for open network computing environments. Kerberos V5 is the Kerberos version applicable to the Secure Internet Services. The Kerberos V5 protocol is specified in RFC 1510 : "The Kerberos Network Authentication Service (V5)".

In this chapter "non-HP Kerberos" refers to Kerberos implementations available directly from MIT, or to commercialized versions of Kerberos based on MIT source code.

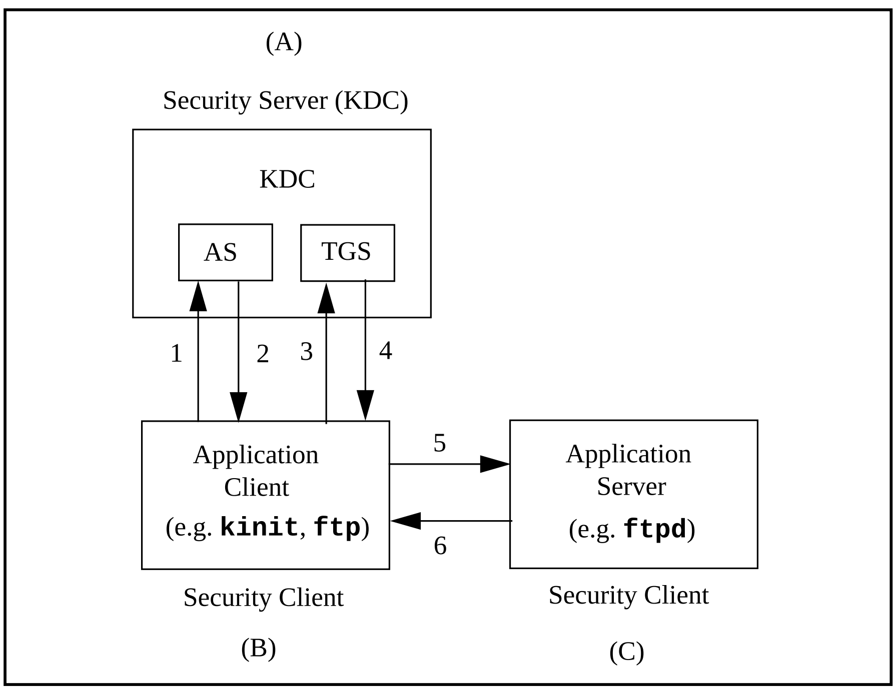


Figure 3 **The Secure Environment and the Kerberos V5 Protocol**

Figure 3 on page 67 shows the components of the secure environment in which the Secure Internet Services and the Kerberos V5 protocol operate. Each component and arrows 1-6 are explained below.

Components of the Secure Environment

As part of the Kerberos V5 protocol, security clients authenticate themselves (verify their identity) to a trusted host. This trusted host is called the security server (A in figure 1). It is highly recommended that the security server host machine be physically secure (e.g. located in a locked room).

The security server is also referred to as the Key Distribution Center (KDC). The KDC provides Kerberos authentication services by issuing encrypted tickets, which clients and servers share. Throughout the rest of this chapter the term KDC will be used to refer to a generic security server. HP's product that currently fulfills the role of the KDC is the HP DCE Security Server.

Security clients are hosts that run the Secure Internet Services clients and daemons (B and C in Figure 3 on page 67). Security clients communicate with the security server for authentication.There are two types of security clients: application clients and application servers.

Application clients (B in Figure 3 on page 67) are hosts that run one or more of the following Secure Internet Services clients: **ftp**, **rcp**, **remsh**, **rlogin**, and **telnet**. The Kerberos utilities **kinit**, **klist**, and **kdestroy** also run on the application client. (See “Related Kerberos Terms and Concepts” on page 70.) In some examples, application clients may be referred to as local hosts.

Application servers (C in Figure 3 on page 67) are hosts that run one or more of the following Secure Internet Services daemons: **ftpd**, **remshd**, **rlogind**, and **telnetd**. In some examples, application servers may be referred to as remote hosts.

A specific security client can be, and usually is, both an application client and application server. However, it makes sense to distinguish between these roles when describing the Kerberos V5 protocol.

A Simplified Description of the Kerberos V5 Protocol

The following steps refer to the arrows indicated in Figure 3 on page 67.

1. Users must first obtain credentials for themselves from a portion of the KDC called the Authentication Server **(AS)**. The AS is the portion of the KDC that verifies the authenticity of a principal. Users must issue the **`kinit`** command which then calls the AS. HP DCE users would generally use the **`dce_login`** command rather than the **`kinit`** command.

2. Once the AS finds an entry for the user principal, it issues encrypted credentials back to the client. The client will need these credentials to successfully run the Secure Internet Services. The credentials consist of a ticket, called the ticket granting ticket **(TGT)**, and a temporary encryption key, often called the session key. The session key is a temporary encryption key used by the server to authenticate the client. It is typically valid for a login session. It is encrypted in the server’s key. The user must obtain a TGT before running the Secure Internet Services.

   All the user has to do up to this point is issue the **`kinit`**, or **`dce_login`** command. The TGT and session key are automatically kept for the user in a temporary credentials cache file. The user does not need to explicitly do anything with them. However, at the end of the session, or when the credentials are no longer needed, it is recommended that the user destroy the credentials using a Kerberos utility called **`kdestroy`**.

   When users invoke one of the Secure Internet Services they enter the usual command along with any desired command options.

   From a user's perspective, the interface to the Secure Internet Services is virtually identical to that of the non-secure versions of the services. Aside from new kerberos-related security options, the only difference is that the user is not prompted for a password. If the Kerberos V5 authentication and authorization succeed, the command succeeds and the details are transparent to the user.

   Although it is not visible to the user, more is going on.

3. When a user invokes a Secure Internet Service, the client contacts the ticket granting service (TGS) portion of the KDC. The client passes along to the TGS the TGT, the name of the application server (remote host), and an authenticator. The authenticator is a record containing information that can be shown to have been recently generated using the session key known only by the client and the server.The encrypted authenticator is generated from the session key that was sent with the credentials from the AS.

1 The TGS generates new credentials that both the server and client use to authenticate each other. The TGS sends back to the client a new session key, called the sub-session key, that is encrypted in the old session key. The TGS also sends back to the client a ticket, called a service ticket. The service ticket contains a copy of the sub-session key and is encrypted in the target server's secret key.

The secret key is an encryption key shared by a principal and the KDC. These encrypted keys are stored in the KDC's principal database. A secret key has a relatively long lifetime as compared to the relatively short lifetime of a session key.

The same TGT can be used to obtain multiple service tickets.

2 The client then sends to the application server the service ticket and a new authenticator encrypted using the sub-session key. The application server decrypts the service ticket with its own secret key and extracts the sub-session key. This sub-session key is now a shared secret between the client and the application server.

3 At the client's request, the application server can also return to the client credentials encrypted in the sub-session key. This implies a mutual authentication between the client and the application server. This optional Kerberos V5 mutual authentication step is performed in each of the Secure Internet Services.

To summarize,

- The user obtains a TGT from the AS portion of the KDC when it first issues the **`kinit`**, or **`dce_login`**, command to the KDC.
- When the user invokes a Secure Internet Service the client requests a service ticket from the TGS portion of the KDC. It obtains this service ticket by presenting the TGT and other credentials to the TGS portion of the KDC.
- The client sends the service ticket and other credentials received from the TGS to the application server. This authenticates the client to the application server. This authentication replaces the non-secure authentication method of sending a password, in a readable form, to the application server.

Related Kerberos Terms and Concepts

Kerberos Utilities

The following utilities must exist on all security clients (HP provides these utilities on HP clients):

- **`kinit`**: This command obtains and caches a TGT for the user. For more information, refer to the **`kinit`**(1) man page.
- **`klist`**: This command displays the list of tickets in the user's credentials cache file. For more information, refer to the **`klist`**(1) man page.
- **`kdestroy`**: This command destroys the user's accumulated credentials. For more information, refer to the **`kdestroy`**(1) man page.

Realms/Cells

A realm defined an administrative boundary. It has a unique name. It consists of the KDC and all the security clients (application servers and application clients) registered to that KDC.

When using the HP DCE Security Server as a KDC, the term “cell” is used. A cell is roughly equivalent to a realm.

By convention, Kerberos uses uppercase realm names, which appear as suffixes in principal names (**`david@MYREALM.COM`**).

An HP DCE cell name must be lowercase. It appears as a prefix and has a leading "/.../" in a principal name (**`/.../my_kdc_cell.com/david`**).

Cross-Realm Authentication

Cross-realm authentication occurs when a client from one realm wishes to access a server from a different realm. Since each KDC administers tickets for a specific realm, cross-realm operation requires using inter-realm keys with the KDC. Cross-realm authentication is also referred to as inter-realm authentication.

Currently it is not possible to set up heterogeneous cross-realm authentication between a DCE KDC and a Kerberos V5 KDC. Cross-realm authentication is available between realms hosted by KDCs of the same type. In other words, for cross-realm configurations with the Secure Internet Services, all the KDCs must be HP DCE Security Servers, or all the KDCs must be Kerberos V5 KDCs.

Principals

Principals are uniquely named network entities, including users and services. Principals have names that include realm (or cell) information and an associated key. All principals that participate in Kerberos V5 authentication and authorization are required to be included in the KDC's database. The KDC database does not distinguish between types of principal names. However, it is useful to describe two kinds of principal names: user principal names and service principal names.

User Principal Names

A user principal name is associated with a specific user of the Secure Internet Services. User principal names consist of a user ID and a realm (or cell) name. All users must have one or more user principal names in the KDC's database. An example of a Kerberos user principal name is, **susan@MYREALM.COM**. An example of an HP DCE user principal name is, **/.../my_kdc_cell/susan**.

Service Principal Names

A service principal name is a principal name that authorizes a client to use a particular service, including the specific application server machine the service will access and the realm name.

For **rcp**, **remsh**, **rlogin**, and **telnet**, the service principal name is **host**. (The actual word is **host**. It is not meant to be replaced by a host name.)

For **ftp**, the service principal name is **ftp** (as a first choice) or **host** (as an acceptable second choice).

Following is an example of a Kerberos service principal name for **telnetd**:

host/abc.com@REALM_A.COM.

In this example, the system is **abc.com**, and the realm is **REALM_A.COM**.

Following is an example of an HP DCE service principal name for **telnetd**:

/.../cell_a.com/host/abc.com.

This example uses **cell_a.com** instead of **REALM_A.COM**.

Authorization

Authorization is the process in which users verify that they may access a remote account on a specified server. Authorization depends on successful user principal validation through the Kerberos V5 authentication protocol described earlier in this section.

For authorization to succeed, a mapping must exist on the application server machine authorizing the user principal to operate as the login user. The term “login user” refers to the user whose account is being accessed on the remote host. This is not necessarily the same user who originally issued the **kinit**, or **dce_login** command.

Assume **david** has already issued the **kinit** command. In this example, **david** enters the following:

```
$ ftp hostA
$ Connected to hostA
$ Name:(hostA:david): susan
```

In this example, **susan** is the login user.

Both of the following requirements must be met for authorization to succeed:

1. The login user must have an entry in the **/etc/passwd** file on the application server (remote host).
2. One of the following three conditions must be met:
   - A **$HOME/.k5login** file must exist in the login user's home directory on the application server and contain an entry for the authenticated user principal. This file must be owned by the login user and only the login user can have write permission.
   - An authorization name database file called **/krb5/aname** must exist on the application server and contain a mapping of the user principal to the login user.
   - The user name in the user principal must be the same as the login user name, and the client and server systems must be in the same realm.

Forwarded/Forwardable Tickets

When a user obtains service ticket credentials, they are for a remote system. However, the user may want to use a secure service to access a remote system and then run a secure service from that remote system to a second remote system. This would require possession of a valid TGT for the first remote system. However, running **kinit** on the first remote system to obtain a TGT would cause the user's password to be transmitted in a readable form over the network.

To avoid this problem, Kerberos provides the option to create TGTs with special attributes allowing them to be forwarded to remote systems within the realm.

The Secure Internet Services clients which offer TGT forwarding options (**-f**, **-F**) are **remsh**, **rlogin**, and **telnet**. However, before these options can be recognized, two prerequisite flags must be enabled.

First, the KDC's forwardable ticket option must be enabled. For Kerberos V5 KDCs, use the **kadmin** command. For the HP DCE Security Server, use the **dcecp** command to set the **forwardabletkt** account attribute.

Second, **kinit** must be invoked with the forwardable flag set (**-f**). If the **-f** option is selected when **kinit** is run, the TGT for the local system can be forwarded to the remote system. Then clients do not need to re-authenticate themselves from the remote system to the KDC.

HP DCE Clients must use **kinit -f** to enable forwarding as the **dce_login** utility does not have options for ticket attributes.

Provided these two flags are enabled, the forwarding options of **rlogin**, **remsh**, and **telnet** can take effect. For the **remsh**, **rlogin**, or **telnet** client that invokes the **-f** option, the forwarding of the TGT only happens to one remote system (one free hop). For the **remsh**, **rlogin**, or **telnet** client that invokes the **-F** option, it is possible to keep forwarding the TGT (potentially *n* free hops).

Multiple free hops are possible because using the **-F** option leaves the forwardable attribute enabled in the forwarded TGT ticket, whereas using the **-f** option does not. Thus, the client can forward the TGT to an unlimited number of remote systems if the **-F** option is used every time. Once the **-f** option is used, the forwarding chain stops at the next node.

API (Application Program Interface)

The Secure Internet Services versions of **rcp/remshd**, **remsh/remshd**, **rlogin/rlogind**, and **telnet/telnetd** use the Kerberos V5 Beta 4 API.

The Secure Internet Services versions of **ftp/ftpd** use the GSS-API (Generic Security Service Application Program Interface) Version 1. The GSS-API separates application logic from a given security mechanism.

For more information on GSS-API Version 1, refer to RFCs 1508 and 1509.

Secure Environment Configurations

Configurations consist of KDCs and client nodes. The figures below illustrate possible KDC/client configurations. The following paragraphs describe the nodes in more detail and also discuss interoperability among the nodes.

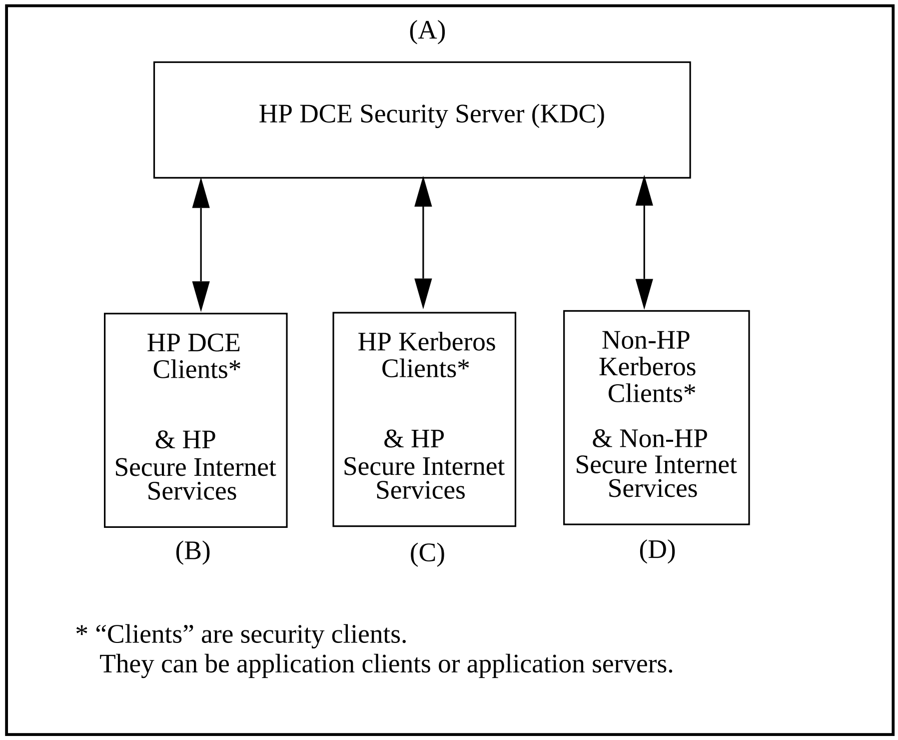


Figure 4 **Client Interoperability with HP DCE Security Servers**

Figure 4 illustrates which security clients can interoperate in configurations using HP DCE Security Servers. Though not shown here, there may be multiple HP DCE Security Servers in the configuration.

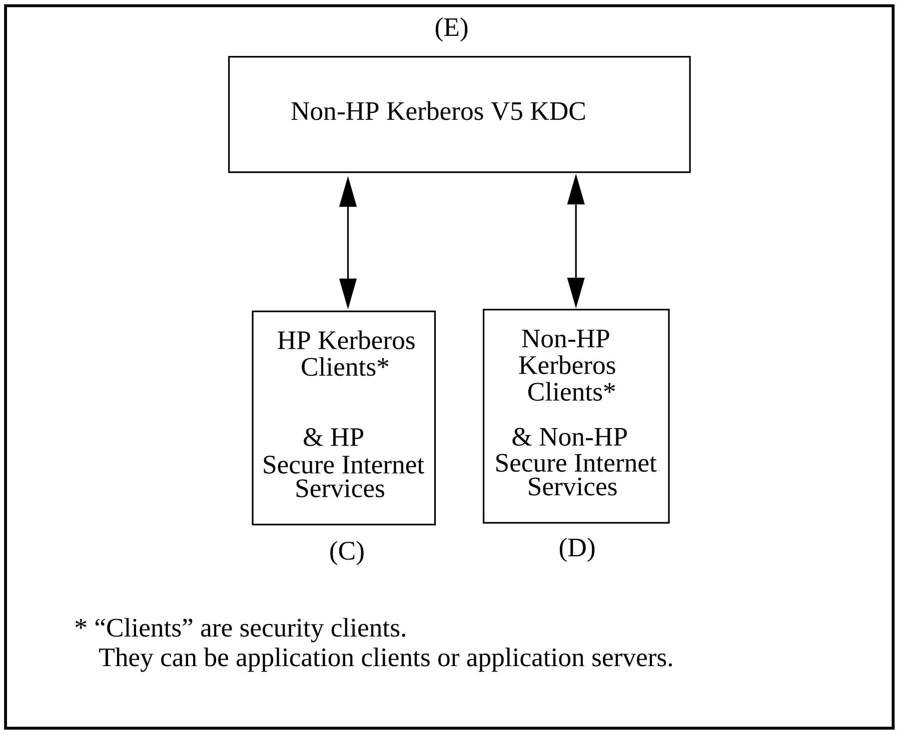


Figure 5 **Client Interoperability with Non-HP Kerberos V5 KDCs**

Figure 5 illustrates which security clients can interoperate in configurations using non-HP Kerberos V5 KDCs. Though not shown here, there may be multiple non-HP Kerberos V5 KDCs in the configuration.

Types of KDC Nodes

- The HP DCE Security Server can be configured to run with security clients using the Secure Internet Services and fulfill the role of the KDC. An HP DCE Security Server node runs the HP DCE security daemon **secd**. This node can be configured as the only member of a single-node DCE cell, or as a member of a multi-node cell with HP DCE clients.

  For more information on how to configure an HP DCE Security Server, see *Planning and Configuring HP DCE*.

  The HP DCE Security Server is shown as node A in Figure 4 on page 75.

- The Non-HP Kerberos V5 KDC can be configured to run with security clients using the Secure Internet Services. A non-HP Kerberos V5 KDC is any non-HP KDC that implements the Kerberos V5 protocol (described in RFC 1510).

  For more information, refer to your KDC provider's documentation.

  The Non-HP Kerberos V5 KDC is shown as node E in Figure 5 on page 76.

Types of Security Client Nodes Using the Secure Internet Services

- The HP DCE Client is a node configured into a DCE cell using the **dce_config** utility. The HP DCE file set **DCE-Core.DCE-CORE-RUN**, which is automatically installed, must be configured on this client. The HP Secure Internet Services product, **InternetSvcSec**, must be installed, enabled and configured on this client.

  The Kerberos utilities **kinit**, **klist**, and **kdestroy** are supplied by HP on this client. However, this client generally obtains credentials using the **dce_login** command, rather than the Kerberos **kinit** command. This client can use **dcecp** and other administrative tools for Kerberos-related management tasks.

  For more information, see *Using HP DCE 9000 Security with Kerberos Applications*, available in postscript and ASCII form in the directory **/opt/dce/newconfig/RelNotes/** in the files **krbWhitePaper.ps** and **krbWhitePaper.text**.

  The HP DCE Client is shown as node B in Figure 4 on page 75.

- The HP Kerberos Client is a node with the same client software as the HP DCE client. This node, however, is *not* configured into a DCE cell. The HP DCE file set **DCE-Core.DCE-CORE-RUN**, which is automatically installed, must be configured on this client. The HP Secure Internet Services product, **InternetSvcSec**, must be installed, enabled and configured on this client.

  The Kerberos utilities **kinit**, **klist**, and **kdestroy** are supplied by HP. The HP Kerberos client treats the HP DCE Security Server as an ordinary Kerberos KDC. Credentials are obtained with the Kerberos command **kinit**, not the HP DCE command **dce_login**. The HP Kerberos client cannot use HP DCE administration tools for Kerberos-related management tasks. The creation and update of Kerberos-related files must be done manually.

  For more information, see *Using HP DCE 9000 Security with Kerberos Applications*, available in postscript and ASCII form in the directory **/opt/dce/newconfig/RelNotes/** in the files **krbWhitePaper.ps** and **krbWhitePaper.text**.

The HP Kerberos Client is shown as node C in Figure 4 on page 75 and Figure 5 on page 76.

Allowable Non-HP Security Client Nodes

The Non-HP Kerberos Client is a node running non-HP security client software. This includes non-HP versions of the Kerberos utilities **kinit**, **klist** and **kdestroy**, and non-HP secure versions of internet services.

Generally, configurations that contain non-HP security clients will interoperate securely with configurations that include the HP Secure Internet Services, provided the following is true:

- The Kerberos utilities **kinit**, **klist** and **kdestroy** are based on Kerberos V5 Beta 4.
- Secure versions of **rcp**/**remshd**, **remsh**/**remshd**, **rlogin**/**rlogind** and **telnet**/**telnetd** are implemented with Kerberos V5 Beta 4 API.
- Secure versions of **ftp**/**ftpd** are implemented according to the FTP security extension standard and use the GSS-API Version 1 based on the Kerberos V5 Beta 4 API.

For information on the non-HP Kerberos client, refer to your provider's documentation.

The Non-HP Kerberos Client is shown as node D in Figure 4 on page 75 and Figure 5 on page 76.

Interoperability

Within a given realm, all KDCs must be of the same type. In other words, for configurations that include the Secure Internet Services, KDCs must be either all HP DCE Security Servers or all non-HP Kerberos V5 KDCs (implementing RFC 1510). Multiple KDCs of the same type may exist. In these cases there is effectively one “master” KDC. The additional KDCs contain duplicate, read-only, database information from the master. This can be helpful for load balancing purposes.

Currently it is not possible to set up heterogeneous cross-realm authentication between a DCE KDC and a Kerberos V5 KDC. Thus, even in cross-realm configurations, all KDCs must be of the same type. In other words, they must be either all HP DCE Security Servers or all non-HP Kerberos V5 KDCs (implementing RFC 1510).

For more specific interoperability information with non-HP Kerberos clients (node D in Figure 4 on page 75 and Figure 5 on page 76) contact your HP support representative.

Configuration Requirements of the Secure Environment

The main purpose of this chapter is to provide information required specifically for the Secure Internet Services. However, since the successful usage of the Secure Internet Services requires a correctly configured secure environment, this section discusses some general requirements of the secure environment.

For specific configuration information, refer to your KDC (security server) provider's and your security client provider's documentation.

For configurations that include any HP nodes (HP DCE Security Server, HP DCE client, HP Kerberos Client), see *Using HP DCE 9000 Security with Kerberos Applications*, available in postscript and ASCII form in the directory **/opt/dce/newconfig/RelNotes/** in the files **krbWhitePaper.ps** and **krbWhitePaper.text**.

Requirements on the KDC

1. The KDC (Security Server) software should be running.
2. User accounts should be created, as necessary.
3. User and service (**host** and optionally **ftp**) principals should exist in the KDC database.

Requirements on the Security Clients

1. The following port must exist in the **/etc/services** file or in the NIS services map.

   ```
   kerberos5   88/udp   kdc
   ```

2. The security client software should be installed.

   The Kerberos commands **kinit**, **klist** and **kdestroy** should all exist.

   For HP DCE and HP Kerberos clients the HP DCE file set, **DCE-Core.DCE-CORE-RUN**, must be configured.

3. A configuration file called **/krb5/krb.conf** must exist.

This file specifies the default realm or cell name and also maps realm or cell names to KDCs. Suggested ownership and permissions for this file are **root**, **sys**, **-r--r--r--**.

For HP DCE Clients this file is automatically created when the client is configured into the HP DCE cell. Additional entries can be added manually.

4 A realms file called **/krb5/krb.realms** must exist.

This file is used to associate host names to realm or cell names. Suggested ownership and permissions for this file are **root**, **sys**, **-r--r--r--**.

5 A keytab file called **/krb5/v5srvtab** must exist.

This file must be owned by **root** and only **root** can have read and write permissions.

This keytab file must contain the service principal names and their associated secret keys. The application server uses the key found in its keytab file to decrypt the service ticket sent to it by the application client.

HP Kerberos Security Clients

For HP Kerberos security clients even though the service principal's secret key is required to be in a file on the security client, it must first be created on the KDC. On an HP DCE Security Server use the **dcecp** command. On a non-HP Kerberos V5 KDC use the appropriate command.

The keytab then needs to be *securely* copied to the target client node. This can be somewhat difficult if you have no secure means to copy the file over the network. A removable media (for example, a floppy disk) may be necessary to ensure proper security.

HP DCE Security Clients

For HP DCE security clients the keytab file can be created and edited on the client itself, using **dcecp** keytab commands. This is very useful in that the problem of securely copying the keytab file information from the KDC is no longer an issue, since the file is created on the client.

6 A **$HOME/.k5login** file should exist in each login user's home directory.

This file must be owned by the login user, and only the login user can have write permission.

This file lists the user principals and their associated realm or cell names that have access permission to the login user's account. The user principals are for the user that originally performed the **kinit** or **dce_login** command. The term “login user” refers to the user whose account is being accessed on the remote host. This is not necessarily the same user who originally issued the **kinit** or **dce_login** command.

Assume **amy** has already issued the **kinit** command. In this example, **amy** enters the following:

```
$ rlogin hostA -l robert
```

In this example, **robert** is the login user, and **amy** must have an entry in Robert’s **$HOME/.k5login** file on the application server (**hostA**).

Alternatively, the client can use an authorization name database file called **/krb5/aname**. An entry in this file will authorize a user principal name to the specified login name. A tool for the administration of an **aname** file is not provided by DCE.

For the Secure Internet Services, login is allowed even without entries in the login user’s **$HOME/.k5login** file or the **aname** database, provided that the login user’s name matches the user principal user’s name, and that the Kerberos realm of the client matches the default realm of the application server.

7 The login user must have an entry in the **/etc/passwd** file on the application server.

Installing and Enabling the Secure Internet Services

A properly configured KDC must be running for the Secure Internet Services to work.

System Requirements for the Secure Internet Services

Hardware Requirements	HP 9000 S700 or S800
Software Requirements	HP-UX 10.20
Disk Space	This product requires approximately 3.6 Mbytes of additional disk space. This is due to the static linkage to the Kerberos libraries, which provide the actual authentication functions.
Memory	No additional memory is required.
Prerequisite Software for all HP security clients (HP DCE and HP Kerberos)	HP DCE file set (Rev 1.4.1 or later) **`DCE-Core.DCE-CORE-RUN`** Internet Services file set **`InternetSrvcs.INETSVCS-RUN`**

NOTE: The Internet Services file set is still required. The Secure Internet Services product only replaces some of the Internet Services files.

Installing and Enabling the Secure Internet Services Product

1 Log in as **`root`** on the system where you want to install and enable the product.

2 Invoke **`swinstall`**. The default view of the software is in the form of bundles. Change the software view to products and select the **`InternetSvcSec`** product for installation. For more information on the swinstall utility, see *Managing HP-UX Software with SD-UX.*

The product contains the run-time file set **INETSVCS-SEC** as well as file sets for the man pages. The **INETSVCS-SEC** file set contains the secure versions of the services (**kftp**/**kftpd**, **krcp**, **kremsh**/**kremshd**, **krlogin**/**krlogind**, and **ktelnet**/**ktelnetd**). In addition to the client and daemon man pages for the services there is a new man page called **sis**(5) which contains information common to all the Secure Internet Services, including warning and error messages.

Within **INETSVCS-SEC** is a required startup script called **inetsvcs_sec**. This script must be run to enable the product. (See step 5.)

NOTE: If a user wants to activate the HP DCE Integrated Login Utilities package *and* install this product, the HP DCE Integrated Login must be activated before this product is installed. Similarly, if a user wants to deactivate the HP DCE Integrated Login Utilities *and* install this product, deactivation must take place before the installation of this product. The order is important because the HP DCE Integrated Login Utilities package offers **ftp** in addition to its other services. When activated, it overwrites the existing **ftp** with its own version of **ftp**.

The Secure Internet Services **ftp** service can be used to replace the **ftp** service provided by HP DCE Integrated Login Utilities package. The secure version will ensure that a password is not sent over the network in a readable form. However, users will not be allowed access to remote DFS (Distributed File Service) cells as they are with the HP DCE Integrated Login Utilities **ftp** service.

3. Review the **swinstall** log files for warnings or errors.

   Any logged errors will be accompanied by information describing the appropriate action for resolving the installation problem.

4. Verify the installation of the new executable.

   The clients **kftp**, **krcp**, **kremsh**, **krlogin**, and **ktelnet** should be present in **/usr/bin**.

   The daemons **kftpd**, **kremshd**, **krlogind**, and **ktelnetd** should be present in **/usr/lbin**.

   The following client man pages should be present in **/usr/share/man/man1.Z**:

   kftp(1), **krcp**(1), **kremsh**(1), **krlogin**(1), and **telnet**(1).

   The following daemon man pages should be present in **/usr/share/man/man1m.Z**:

kftpd(1M), **kremshd**(1M), **krlogind**(1M), and **telnetd**(1M)

The **sis**(5) man page should be present in **/usr/share/man/man5.Z**.

The enable script **inetsvcs_sec** should be present in **/usr/sbin**.

5 To enable the product, invoke the following command:

```
/usr/sbin/inetsvcs_sec enable
```

When the product is enabled the non-secure executables are stored in files of the same name, but with the extension **.noauth**. The original service names are then symbolically linked to their respective secure versions. The original man pages are moved to files with the same name, but with the extension **.safe**. The secure versions of the man pages are then copied over the original versions of the man pages (i.e. **ftp**(1) is moved to **ftp.safe**, **kftp**(1) is moved to **ftp**(1)).

To verify that the product has been successfully enabled check that the **.noauth** files, **.safe** files, and linkages exist as described.

Disabling and Removing the Secure Internet Services Product

1 Log in as **root** on the system where you want to disable and remove the product.

2 To disable the product without removing the files, invoke the following command:

```
/usr/sbin/inetsvcs_sec disable
```

Verify that the prior executables and man pages were restored.

3 To remove the product invoke **swremove** and remove the **InternetSvcSec** product.

Configuring the Secure Internet Services

Provided that the general secure environment configuration requirements have been met, the following are the tasks required specific to configuring the Secure Internet Services.

Requirements on the KDC

You do not need to perform any specific tasks on the KDC for the configuration of the Secure Internet Services.

Requirements on the Security Clients

The following are required on security clients:

1. Log in as **root** on the security client system.
2. Make sure the following ports exist in the **/etc/services** file or in the NIS services map.

```
klogin        543/tcp
kshell        544/tcp   krcmd kcmd
```

   The secure versions of **telnet/telnetd** and **ftp/ftpd** applications run on the same ports as the non-secure versions. The **telnet** service uses port 23 and the **ftp** service uses port 21.

   If you are using NIS, then these entries should be made in the NIS services database.

3. Make sure the **/etc/inetd.conf** file has the following lines:

```
klogin  stream tcp nowait root  /usr/lbin/rlogind   rlogind -K
kshell  stream tcp nowait root  /usr/lbin/remshd    remshd -K
ftp     stream tcp nowait root  /usr/lbin/ftpd      ftpd
telnet  stream tcp nowait root  /usr/lbin/telnetd   telnetd
```

   You may choose to set different options from the default options listed above. For example, to enforce Kerberos V5 authentication on **ftp** and **telnet**, add the **-A** option after **ftpd** and **telnetd**. To prevent non-secure access from **rcp**, **remsh**, and **rlogin**, comment the following two lines out of the **/etc/inetd.conf file**:

```
#shell  stream tcp nowait root  /usr/lbin/remshd   remshd
#login  stream tcp nowait root  /usr/lbin/rlogind  rlogind
```

CAUTION: If the **`shell`** line is commented out, the **`rdist`** command will no longer work.

4. If you modified the **`/etc/inetd.conf`** file, run the **`inetd -c`** command to force **`inetd`** to reread its configuration file.
5. Repeat steps 1-4 for all security client systems.

Verifying the Secure Internet Services

Secure Environment Checklist

The following is a quick checklist to verify that the secure environment is properly configured.

1. On the KDC, issue a **ps -ef** command and verify that the necessary security server executables are running. Look for **secd** on an HP DCE Security Server or **krb5kdc** on a non-HP Kerberos V5 KDC.

2. Use an appropriate tool to verify that the desired principals exist in the KDC database. This can usually be done remotely. For the HP DCE Security Server, use **dcecp**.

3. Issue a **what**(1) command for the appropriate Secure Internet Services client and daemon. Verify that the string includes “Secure Internet Svcs”.

4. Ensure that the following entries exist in the **/etc/services** file or in the NIS services map.

```
kerberos5  88/udp   kdc
klogin    543/tcp
kshell    544/tcp   krcmd  kcmd
```

5. Ensure that the following entries exist in **/etc/inetd.conf**:

```
klogin  stream tcp nowait root  /usr/lbin/rlogind   rlogind -K
kshell  stream tcp nowait root  /usr/lbin/remshd    remshd -K
ftp     stream tcp nowait root  /usr/lbin/ftpd      ftpd
telnet  stream tcp nowait root  /usr/lbin/telnetd   telnetd
```

   Different options may be set from the default options shown above. If you modified the **/etc/inetd.conf** file, you must run the **inetd -c** command to force **inetd** to reread its configuration file.

6. To ensure that the client configurations are correct, invoke the validation application, **krbval**. The **krbval** tool checks for proper configuration of security clients. It can be used to “ping” a particular realm's KDC. It can also check the keys in the keytab file for agreement with the KDC. By acting as a client/daemon service itself, it can further assist in verifying the correctness of the configuration.

For more information refer to the **krbval**(1M) man page.The **krbval** tool is also described in *Using HP DCE 9000 Security with Kerberos Applications,* available in postscript and ASCII form in the directory **/opt/dce/newconfig/RelNotes/** in the files **krbWhitePaper.ps** and **krbWhitePaper.text**.

Verifying Usage of Secure Internet Services

You may first want to read the section "Using the Secure Internet Services" on page 91 before continuing with this section.

1. Obtain a TGT (ticket granting ticket) from the KDC. On an HP DCE security client, use the **dce_login** command. On an HP Kerberos Client or a non-HP Kerberos Client, use the **kinit** command.
2. Invoke the desired Secure Internet Service in the same manner as in a non-secure environment.

   If the secure versions of **ftp**, **rlogin**, and **telnet** work successfully, the only observable difference from execution on a non-secure system will be that, if a password was required on the non-secure version, then the password prompt will not be displayed on the secure version.

   If the secure versions of **remsh** (used with a command) and **rcp** work successfully, there are no observable differences from execution on a non-secure system.
3. Before logging off the local system, invoke the command **kdestroy**. This will remove the credentials cache file.

Troubleshooting the Secure Internet Services

The Verification Checklist

Go through the checklist, described in the section “Verifying the Secure Internet Services” on page 88.

- Verify that the secure environment is correct.
- Verify that the correct versions of the Secure Internet Services (clients and daemons) are installed.
- Use the **krbval** validation tool.

Security-related Error Messages

All of the Secure Internet Services obtain security-specific error messages from the Kerberos API. Secure **ftp/ftpd** uses the GSS-API, but as that API depends on the Kerberos API, its error messages will be consistent with the other services.

There are several security-related messages specific to the Secure Internet Services that are generated outside of the Kerberos API. For a list of these error messages, refer to the DIAGNOSTICS section of the Secure Internet Services man page, **sis**(5).

In general, the Secure Internet Services client will write error messages to standard error and the Secure Internet Services daemon will write error messages to syslog (typically **/var/adm/syslog/syslog.log**).

Common Problems

The most common problem likely to occur when using the Secure Internet Services will be the failure to obtain a TGT, which is required for using the Secure Internet Services. Use the **klist** command to determine if a ticket has been granted, and if none has been, run **kinit** or **dce_login**.

Other common problems will most likely relate to an incorrect configuration.

Using the Secure Internet Services

Overview of the User's Session

- Users must issue a **kinit** (or for HP DCE clients a **dce_login**) command so that they get a TGT from the KDC (for example, **kinit amy@realm1.com**). The TGT credentials received from the **kinit** (or **dce_login**) will typically be valid for a default lifetime. The **kinit**(1) man page describes TGT lifetime and renewable options.

  For more information, refer to the **kinit**(1) and **dce_login**(1) man pages.

- Once users have obtained a TGT, they can use the Secure Internet Services throughout the time period that their TGT is valid. The lifetime of a TGT is configurable and is typically eight hours.

  There are no visible differences between using the secure and non-secure versions of the Internet Services, except that users are not prompted for a password, and there are some new Kerberos-related options available.The new Kerberos-related options relate to various Kerberos concepts, including authentication, authorization and TGT forwarding. For information on these and other Kerberos concepts refer to the “Overview of the Secure Environment and the Kerberos V5 Protocol” section on page 67 of this chapter.

  The **klist** command is one of the Kerberos utilities users may want to use during their secure session. This command will display their accumulated credentials. For more information, refer to the **klist**(1) man page.

- When users are finished for the day (or secure session), they should issue the **kdestroy** command to remove the credentials they have accumulated during their session. These credentials are not automatically removed when they exit a shell or log out of their session. Thus, it is highly recommended that they issue this command so that any credentials they accumulated are not susceptible to misuse from intruders. For more information refer to the **kdestroy**(1) man page.

Bypassing and Enforcing Kerberos Authentication

Depending on how certain options are used with these services, the Secure Internet Services clients will still be able to access non-secure remote hosts, and the daemons will still be able to accept requests from non-secure clients.

To access a non-secure remote system on the network, users can use the **-P** option when issuing the client command to bypass Kerberos authentication. However, if accessing the host requires a password, then the password will be sent in a readable form over the network.

To prevent remote users from gaining access in a non-secure manner, administrators can enforce Kerberos authentication. For **ftpd** and **telnetd**, to prevent access from non-secure clients these daemons should be invoked with the **-A** option. For **remshd** and **rlogind**, to prevent access from non-secure clients the entries for **shell** and **login** in the **/etc/inetd.conf** file should be commented out. If these steps have been taken, the client cannot use the **-P** option to bypass authentication.

CAUTION: If the **shell** line is commented out, the **rdist** command will no longer work.

Other Comments on Using the Secure Internet Services

- There is no change to the way in which the secure version of **ftp** handles anonymous users. However, in secure environments, it serves no purpose to authenticate or authorize an anonymous user. An anonymous user does not have a password to protect, and any data accessible through an **ftp** account has been made publicly available. Therefore, it does not make sense to add an anonymous user to the KDC's database. To access a secure system anonymously, use the **-P** option **ftp** provides. This approach requires that **ftpd** was not invoked with the **-A** option on the remote host.
- The secure version of **rlogin** accesses **rlogind** through the new port specified by the /**etc**/**services** entry **klogin** when operating as a secure client. If you invoke **rlogin** with the **-P** option, or if you run a non-secure version of **rlogin**, then **rlogin** will behave as a non-secure client and access **rlogind** through the **login** port.

  The secure version of **remsh** accesses **remshd** through the new port specified by the /**etc**/**services** entry **kshell** when operating as a secure client. If you invoke **remsh** with the **-P** option, or if you run a non-secure version of **remsh**, then **remsh** will behave as a non-secure client and access **remshd** through the **shell** port.

The secure version of **`rcp`** accesses **`remshd`** through the new port specified by the /**`etc`**/**`services`** entry **`kshell`** when operating as a secure client. If you invoke **`rcp`** with the **`-P`** option, or if you run a non-secure version of **`rcp`**, then **`rcp`** will behave as a non-secure client and access **`remshd`** through the **`shell`** port.

Sources for Additional Information

Additional HP Documentation

- *Using HP DCE 9000 Security with Kerberos Applications*

  Available in postscript and ASCII form in the directory **/opt/dce/newconfig/RelNotes/** in the files **krbWhitePaper.ps** and **krbWhitePaper.text**. This document is highly recommended reading for customers with any HP KDC or security client nodes in their configuration (not just HP DCE). Especially important is the detailed configuration information it contains.

Relevant Man Pages

See the following man pages for more information: **ftp**(1), **ftpd**(1M), **kdestroy**(1), **kinit**(1), **klist**(1), **krbval**(1M), **rcp**(1), **remsh**(1), **remshd**(1M), **rlogin**(1), **rlogind**(1M), **sis**(5), **telnet**(1), and **telnetd**(1M).

Related RFCs

- 1510 : “The Kerberos Network Authentication Service (V5)”
- 1508 : “Generic Security Service Application Program Interface”
- 1509 : “Generic Security Service API : C-bindings”
- Working Specification: “FTP Security Extensions” (Internet Draft 8)

4

Configuring and Administering the BIND Name Service

The Berkeley Internet Name Domain (BIND) is a distributed network information lookup service. It allows you to retrieve host names and internet addresses for any node on the network. It also provides mail routing capability by supplying a list of hosts that will accept mail for other hosts. This chapter includes the following sections:

- Overview of the BIND Name Service
- Creating and Registering a New Domain
- Configuring the Name Service Switch
- Choosing Name Servers for Your Domain
- Configuring a Primary Master Name Server
- Configuring a Secondary Master Name Server
- Configuring a Caching-Only Name Server
- Starting the Name Server Daemon
- Configuring the Resolver to Query a Remote Name Server
- Updating Network-Related Files
- Delegating a Subdomain
- Configuring a Root Name Server
- Configuring BIND in sam
- Troubleshooting the BIND Name Server

For more information on DNS and BIND, see *DNS and BIND*, by Paul Albitz and Cricket Liu, published by O'Reilly and Associates, Inc.

RFCs 1034 and 1035, located in the **`/usr/share/doc`** directory, explain the DNS database format and domain name structure.

Overview of the BIND Name Service

The Berkeley Internet Name Domain (BIND) is the Berkeley implementation of DNS (Domain Name System). It is a database, distributed across the Internet, which maps host names to internet addresses, maps internet addresses to host names, and facilitates internet mail routing. This section describes the components of BIND and how they work. It contains the following sections:

- Benefits of Using BIND
- The DNS Name Space
- How BIND Works
- How BIND Resolves Host Names

Benefits of Using BIND

This section explains the advantages of BIND over the other name services available on HP-UX (NIS and the **/etc/hosts** file).

- **You store information for only the hosts in your local domain.** You configure the hosts in your own domain, and you configure the addresses of name servers in other domains. Your name server can contact these other name servers when it fails to resolve a host name from its local database.

  If you use the **/etc/hosts** file or the NIS **hosts** database for host name resolution, you must explicitly configure every host you might need to contact.

- **You can store all host information on one host.** You configure one machine as a name server, and all other machines query the name server. Information must be kept up to date on only one host instead of many.

  If you use the **/etc/hosts** file for host name resolution, you must keep an up-to-date copy of it on every host in your domain. If you use NIS, you must make sure that your NIS slave servers receive regular updates from the master server.

- **You can contact almost any host on the Internet.** Because BIND spans network boundaries, you can locate almost any host on the network by starting at the root server and working down.

  An NIS server can serve only the hosts on its local LAN. NIS clients send out broadcasts to locate and bind to NIS servers, and broadcasts do not cross network boundaries. Each NIS server must be able to answer all the host name queries from the hosts on its local LAN.

Many people use BIND for host information and NIS for other configuration information, like the **passwd** and **group** databases. NIS has the advantage that it can easily manage many different types of information that would otherwise have to be maintained separately on each host. However, NIS does not easily span networks, so the hosts in an NIS domain do not have access to information from other domains.

The DNS Name Space

The DNS **name space** is a hierarchical organization of all the hosts on the internet. It is a tree structure, like the structure of UNIX directories. The root of the hierarchy is represented by a dot (**.**). Underneath the root, top-level internet domains include **com** (commercial businesses), **edu** (educational institutions), **gov** (government agencies), **mil** (military and defence), **net** (network-related organizations), and **org** (other organizations). Under each top-level domain are subdomains. For example, the **edu** domain has subdomains like **purdue**, **ukans**, and **berkeley**. In turn, each subdomain contains other subdomains. For example, the **purdue** subdomain could contain **econ**, **cs**, and **biol** subdomains.

At the deepest level of the hierarchy, the "leaves" of the name space are hosts. A **fully qualified host name** begins with the host's canonical name and continues with a list of the subdomains in the path from the host to the root of the name space. For example, the fully qualified host name of host **arthur** in the **cs** domain at Purdue University would be **arthur.cs.purdue.edu.**

Figure 6 shows the hierarchical structure of the DNS name space.

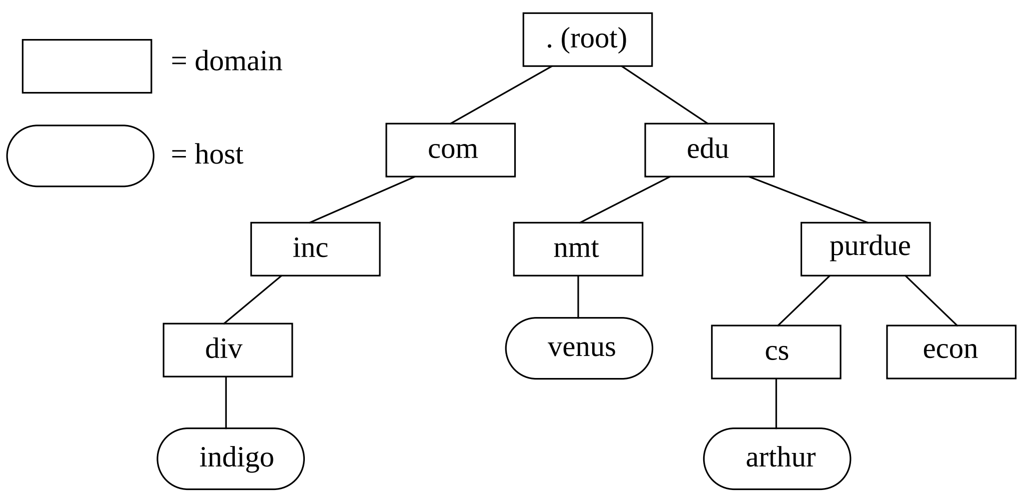


Figure 6 **Structure of the DNS Name Space**

How BIND Works

When a user who is logged into host **venus** in the **nmt.edu** domain types the following command,

```
telnet indigo.div.inc.com
```

the following events occur:

1. The **telnet** process calls **gethostbyname** to get the internet address of **indigo.div.inc.com**.
2. The **gethostbyname** routine invokes the BIND resolver, a set of routines for querying name servers.
3. The resolver constructs a query and sends it to a name server. If the local host is not running a name server, it should have a file called **/etc/resolv.conf**, which contains one or more internet addresses for name servers that serve the local domain. If the local host does not have an **/etc/resolv.conf** file, the resolver sends the query to the local name server.
4. The name server daemon, **named**, receives the query from the resolver. Since the name server has information about only the hosts in its local domain (**nmt.edu**), it cannot answer the query with the information in its local database.
5. The local name server queries a root name server to find the address of **indigo.div.inc.com**. A root name server serves the root domain. It typically stores information about hosts and name servers one and two levels below the root.
6. If the root name server cannot resolve the host name, it returns the address of a name server for the **inc.com** domain.
7. The local name server queries the server for the **inc.com** domain to find the address of **indigo.div.inc.com**.
8. The name server for the **inc.com** domain may not have information for the **div.inc.com** domain. If it does not, it returns the address of a name server for the **div.inc.com** domain.
9. The local name server queries the server for the **div.inc.com** domain to find the address of **indigo.div.inc.com**.
10. The server for the **div.inc.com** domain returns the address of **indigo.div.inc.com** to the local name server.

11 The local name server passes host **indigo**'s address to the resolver, which passes it to **gethostbyname**, which returns it to the **telnet** process.

The local name server in the **nmt.edu** domain caches the addresses of remote name servers, so the next time a local user needs the address of a host in the **inc.com** domain, the local name server sends its query directly to the name server for **inc.com** instead of querying the root name server.

How BIND Resolves Host Names

Because complete domain names can be cumbersome to type, BIND allows you to type host names that are not fully qualified (that is, that do not contain every label from the host to the root and end with a dot). This section describes how the name server resolves host names.

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, the **hostname** command, and network-related configuration files. Names that contain all of the name components and end in a dot are used with commands like **nslookup**, **ping**, and **telnet**, to facilitate the lookup process.

- If the input host name ends with a dot, BIND looks it up as is, without appending any domains to it.
- If the input host name contains at least the number of dots specified by the **ndots** option in the **/etc/resolv.conf** file, BIND looks it up as is, before appending any domains to it. (The default value of **ndots** is 1, so if the input host name contains at least one dot, it will be looked up as is before any domains are appended to it.)
- If the input host name consists of a single component (contains no dots), and you have set up a host aliases file, BIND looks in your aliases file to translate the alias to a fully qualified host name.

  You can create a host aliases file for frequently typed host names, like the following example file:

```
john      zircon.chem.purdue.edu
melody    fermata.music.purdue.edu
```

  The alias (the first field on each line) must be all one word, with no dots.

  To use the file, set the **HOSTALIASES** environment variable to the name of the file, as in the following example:

```
export HOSTALIASES=/home/andrea/myaliases
```

- If the input host name does not end with a dot, BIND looks it up with domain names appended to it. The domain names that BIND appends to it can be configured in four places:

  1 The **`LOCALDOMAIN`** environment variable.

  2 The **`hostname`** command.

  3 The **`search`** option in the **`/etc/resolv.conf`** file.

  4 The **`domain`** option in the **`/etc/resolv.conf`** file.

  If a user has set the **`LOCALDOMAIN`** variable, as in the following example,

```
export LOCALDOMAIN="nmt.edu div.inc.com inc.com"
```

  the **`LOCALDOMAIN`** variable overrides the **`hostname`** and any **`search`** or **`domain`** option in **`/etc/resolv.conf`**, for BIND requests made within the context of the user's shell environment. The input host name is looked up in each of the domains in the variable, in the order they are listed.

  If the local **`hostname`** is set to a fully qualified domain name, and the **`search`** and **`domain`** options are not specified in **`/etc/resolv.conf`**, the input host name is looked up in the domain configured in the fully qualified **`hostname`**.

  The **`search`** option specifies a list of domains to search. Following is an example of a **`search`** option in **`/etc/resolv.conf`**:

```
search div.inc.com inc.com
```

  You can set the **`search`** option to any list of domains, but the first domain in the list must be the domain of the local host. BIND looks up host names in each domain, in the order they are listed. BIND uses the **`search`** option only if the **`LOCALDOMAIN`** variable is not set.

  The **`domain`** option specifies the local domain. If you use the **`domain`** option, BIND will search only the specified domain to resolve host names. BIND uses the **`domain`** option for host name lookups only if the **`LOCALDOMAIN`** variable is not set and the **`search`** option is not specified. (Do not use the **`domain`** and **`search`** options together in the same **`/etc/resolv.conf`** file. If you do, the one that appears last in the file will be used, and any previous ones will be ignored.)

For more information on how BIND resolves host names, type **`man 5 hostname`** or **`man 4 resolver`** at the HP-UX prompt.

Creating and Registering a New Domain

Follow the steps in this section if you need to set up a new domain. Skip this section if you are interested only in adding hosts to an existing domain.

1 Ask the appropriate person or organization for a range of internet addresses to be assigned to the hosts in your domain.

- If your organization already has a domain on a public network, ask the person in charge of the domain to set up a subdomain for you.
- If your organization does not yet have a domain on a public network, and you want to set one up, ask for a domain registration form from Government Systems, Inc. at the following address:

  Government Systems, Inc.
  ATTN: Network Information Center
  14200 Park Meadow Drive
  Chantilly, VA 22021

  email: **`hostmaster@nic.ddn.mil`** phone: (703) 802-8400

  If your organization belongs to several networks, register your domain with only one of them.
- If your organization is not connected to a network, you may set up domains without registering them. However, we suggest that you follow Internet naming conventions in case you later decide to join a public network.

2 Come up with a name for your domain.

- Use only letters (A-Z), digits (0-9), and hyphens (-). No distinction is made between uppercase and lowercase letters.
- Avoid labels longer than 12 characters. (A label is a single component of a fully qualified name, like **`indigo`** or **`com`**.)
- If a host connects to more than one network, it should have the same name on each network.
- Do not use **`nic`** or other well known acronyms as leftmost (most specific) labels in a name. Contact Government Systems, Inc. for a list of top-level and second-level domain names already in use.

3 After you have registered your domain, you can create subdomains without registering them with the public network.

Configuring the Name Service Switch

The Name Service Switch determines where your system will look for host information when it needs to resolve a host name to an IP address. You can configure your system to use BIND, NIS (one of the NFS Services), the **`/etc/hosts`** file, or any combination of the three, in any order.

The default Name Service Switch configuration is adequate for most installations, so you probably do not have to change it. Figure 7 illustrates the default behavior of the Name Service Switch.

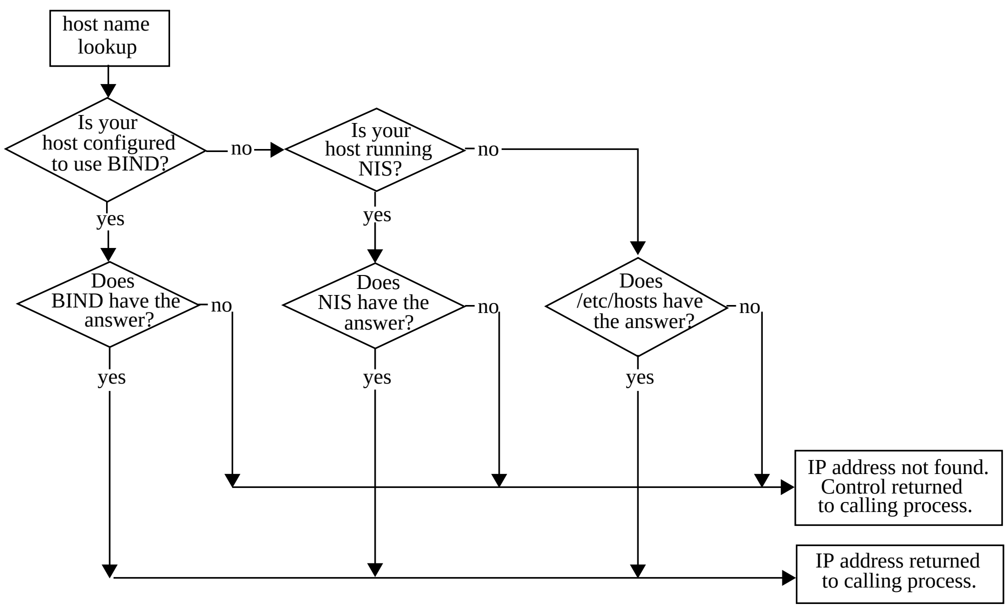


Figure 7 **Default Behavior of the Name Service Switch**

Following are some suggestions for customizing your Name Service Switch configuration:

- If you want your system to consult BIND when it fails to find a host name in NIS, create an **/etc/nsswitch.conf** file that contains only the following line:

```
hosts: nis [NOTFOUND=continue] dns files
```

With this configuration, if NIS does not contain the requested information, and BIND is not configured, the **/etc/hosts** file is consulted.

- If you want your system to consult NIS if it fails to find a host name in BIND or if the BIND name servers are not responding, create an **/etc/nsswitch.conf** file that contains only the following line:

```
hosts: dns [NOTFOUND=continue TRYAGAIN=continue] nis files
```

With this configuration, if BIND does not return the requested information, and NIS is not running, the **/etc/hosts** file is consulted.

HP recommends that you maintain at least a minimal **/etc/hosts** file that includes important addresses like gateways, diskless boot servers and root servers, and your host's own IP address. HP also recommends that you include the word **files** in your **/etc/nsswitch.conf** file to help ensure a successful system boot using the **/etc/hosts** file when BIND and NIS are not available.

CAUTION: Changing the default configuration can complicate troubleshooting. The default configuration is designed to preserve the authority of the name service you are using. It switches from BIND to NIS only if BIND is not enabled. It switches from NIS to the **/etc/hosts** file only if NIS is not enabled. It is very difficult to diagnose problems when multiple name servers are configured and enabled for use.

For more information on the Name Service Switch, see "Configuring the Name Service Switch" on page 29.

Choosing Name Servers for Your Domain

You can configure your host as any of three types of BIND name servers:

Primary Master Server 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.

Secondary Master Server A secondary is also the authority for its domain and contains that domain's data, but it gets its data over the network from another master server.

Caching-Only Server A caching-only server is not authoritative for any domain. It gets its data from an authoritative server and places it in its cache.

If you do not want to run a name server at all on your host, you can configure the resolver to query a name server on another host. By default, the resolver is configured to query the name server on the local host.

To Choose the Type of Name Server to Run

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.
- 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. Running a caching-only server gives you better performance than querying a name server on a remote system, especially if the remote system is on the other side of a gateway or router..
- PCs, workstations that do not want to maintain a server, and other small networked systems should be configured to query a name server on another host. Cluster nodes should query the name server on the cluster server.
- If your network is isolated from the Internet, and your host will be the only BIND nameserver in your organization, you need to configure a root name server. See “Configuring a Root Name Server” on page 136.

To Choose Which Servers Will Be Master Servers

Follow these guidelines when selecting a master server:

- You must have at least two master servers per domain: a primary master and one or more secondary masters for redundancy. One host may be master for multiple domains: primary for some, secondary for others.
- Choose hosts that are as independent as possible for redundancy. For example, choose hosts that use different power sources or cables.
- Choose hosts that have the most reliable Internet connectivity, with the best gateway connections.
- 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 domain.

  A **zone** is the portion of the name space for which a name server has the complete set of authoritative data files.

Configuring a Primary Master Name Server

This section explains how to configure a primary master server in your domain. It also describes the name server data files in the primary master server configuration. It contains the following sections:

- To Create the Data Files for a Primary Master Server
- To Set the Default Domain Name
- The Primary Master Server's Boot File
- The Primary Master Server's Cache File
- The db.127.0.0 File
- The Primary Master Server's db.domain Files
- The Primary Master Server's db.net Files
- To Add a Host to the Domain Data Files
- To Delete a Host from the Domain Data Files

To Create the Data Files for a Primary Master Server

1 Make sure the **/etc/hosts** file is up to date on the host that will be the primary master server.

2 On the host that will be the primary master, create the **/etc/named.data** directory, where the name server data files will reside, and make it the current directory:

```
mkdir /etc/named.data
cd /etc/named.data
```

3 Issue the following command to generate the name server data files from the **/etc/hosts** file:

```
/usr/sbin/hosts_to_named -d domainname -n network_number
```

Following is an example:

```
/usr/sbin/hosts_to_named -d div.inc.com -n 15.19.8
```

4 Move the **named.boot** file to the **/etc** directory:

```
mv /etc/named.data/named.boot /etc/named.boot
```

5 Copy the file **/usr/examples/bind/db.cache.arpa** to the **/etc/named.data** directory. This file is a list of root name servers. You can also use anonymous **ftp** to get the current list of root name servers from **rs.internic.net**. Instructions are included in the **/usr/examples/bind/db.cache.arpa** file.

6 Use the list of root name servers from the **/usr/examples/bind/db.cache.arpa** file or from **rs.internic.net** to update the **/etc/named.data/db.cache** file. The **hosts_to_named** program creates this file but does not add any data to it. The format of the **db.cache** file is described in “The Primary Master Server’s Cache File” on page 113.

If your network is isolated from the Internet, contact the BIND administrator responsible for your domain to get the names and addresses of the root nameservers.

The **hosts_to_named** program creates the following data files in the directory from which it is run. These files are described in the next few sections.

named.boot
db.cache (initially empty)
db.127.0.0
db.*domain* (one file for each domain specified with the **-d** option)
db.*net* (one file for each network number specified with the **-n** option)

Naming these files **db.*name*** is a Hewlett-Packard convention.

You can also create these files manually using a text editor. If you choose to create them manually, you must convert all host names to fully qualified domain names (names containing all labels from the host to the root, terminated with a dot; for example, **indigo.div.inc.com.**)

The **hosts_to_named** program completely rewrites the **db.*domain*** and **db.*net*** files. All manual modifications to these files will be lost the next time you run **hosts_to_named**, except changes to **SOA** records.

For more information, type **man 1M hosts_to_named** or **man 1M named** at the HP-UX prompt.

To Set the Default Domain Name

If you will be using an **/etc/resolv.conf** file on your host, configure the default domain name with the **search** or **domain** keyword. See "Configuring the Resolver to Query a Remote Name Server" on page 130. If you will not be using an **/etc/resolv.conf** file, follow these steps:

1 Set the default domain name with the **hostname** command, by appending the domain name to the host name, as in the following example:

```
/usr/bin/hostname indigo.div.inc.com
```

Do not put a trailing dot at the end of the domain name.

2 Set the **HOSTNAME** variable in the **/etc/rc.config.d/netconf** file to the same value, as in the following example:

```
HOSTNAME=indigo.div.inc.com
```

The Primary Master Server's Boot File

The boot file, **/etc/named.boot**, tells the primary master server the location of all the data files it needs. The primary name server loads its database from these data files. The **hosts_to_named** program creates the **named.boot** file.

Following is 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:

```
;
; 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
```

Every name server must have data for the **0.0.127.IN-ADDR.ARPA** domain. 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.

; Lines beginning with semicolon (;) are comments.

directory Indicates the directory where data files are located.

primary Designates a primary server for the domain in the second field. The third field is the name of the file containing the data for that domain.

cache Indicates the location of the cache file, which contains the addresses of network root name servers.

The Primary Master Server's Cache File

The cache file, **/etc/named.data/db.cache**, lists the servers for the root domain. Every name server must have a cache file. When a name server cannot resolve a host name query from its local database or its local cache, it queries a root server.

The **hosts_to_named** program creates the **db.cache** file, but it leaves it empty. To add data to this file, copy it from the file **/usr/examples/bind/db.cache.arpa**. You can also use anonymous **ftp** to get the list of root name servers from **nic.ddn.mil**. Instructions are included in the file **/usr/examples/bind/db.cache.arpa**.

Following is an example **db.cache** file for a primary master server:

```
;
; This file holds the information on root name servers needed
; to initialize cache of Internet domain name servers
;
;       last update:    May 11, 1994
;       related version of root zone:   940516
;
; name              ttl         class   type    data
;
.                   99999999    IN      NS      NS.INTERNIC.NET.
NS.INTERNIC.NET.    99999999            A       198.41.0.4
.                   99999999            NS      NS1.ISI.EDU.
NS1.ISI.EDU.        99999999            A       128.9.0.107
.                   99999999            NS      C.NYSER.NET.
C.NYSER.NET.        99999999            A       192.33.4.12
.                   99999999            NS      TERP.UMD.EDU.
TERP.UMD.EDU.       99999999            A       128.8.10.90
.                   99999999            NS      NS.NASA.GOV.
NS.NASA.GOV.        99999999            A       128.102.16.10
                    99999999            A       192.52.195.10
.                   99999999            NS      NS.NIC.DDN.MIL.
NS.NIC.DDN.MIL.     99999999            A       192.112.36.4
.                   99999999            NS      AOS.ARL.ARMY.MIL.
AOS.ARL.ARMY.MIL.   99999999            A       128.63.4.82
                    99999999            A       192.5.25.82
.                   99999999            NS      NIC.NORDU.NET.
NIC.NORDU.NET.      99999999            A       192.36.148.17
```

;	Lines beginning with semicolon (;) are comments.
name	In **NS** records, the name of the domain served by the name server listed in the **data** column. A period (.) in the **name** column represents the root domain (the root of the DNS name space hierarchy). In **A** records, the **name** column contains the name of the name server whose address appears in the **data** column.
ttl	The optional time-to-live (**ttl**) indicates how long, in seconds, a server may cache the data it receives in response to a query.
class	The optional **class** field specifies the protocol group. **IN**, for internet addresses, is the most common class. If left blank, the class defaults to the last class specified. So, all the entries in this example **db.cache** file are of class **IN**.
type	Type **NS** records list name servers. The first field in an **NS** record is the domain for which the name server has authority. The last field in an **NS** record is the fully qualified name of the name server. Type **A** records list addresses. The first field in an **A** record is the name of the name server. The last field in an **A** record is the internet address of the name server.
data	The **data** field for an **NS** record gives the fully qualified name of a name server. The **data** field for an **A** record gives an internet address.

The db.127.0.0 File

Each name server must have an **/etc/named.data/db.127.0.0** file. 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**. The **hosts_to_named** program creates this file.

```
;name   class   type    data
@       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.
```

name The name of the subdomain. In data files, **@** represents the current origin. The current origin is the domain configured in this file, according to the boot file. The boot file says that the **0.0.127.in-addr.arpa** domain is configured in the **db.127.0.0** file. Therefore, every instance of **@** in the **db.127.0.0** file represents **0.0.127.in-addr.arpa**.

The current origin is also appended to names that do not end with a dot. For example, the **1** in the **PTR** line would be interpreted as **1.0.0.127.in-addr.arpa**.

class The optional **class** field specifies the protocol group. **IN**, for internet addresses, is the most common class.

type

The **SOA** (start-of-authority) record designates the start of a domain, and indicates that this server is authoritative for the data in the domain.

The **NS** record designates a name server for the current origin (**0.0.127.in-addr.arpa**).

PTR records are usually used to associate an address in the **in-addr.arpa** domain with the canonical name of a host. The **PTR** record in the example **db.127.0.0** file associates the name **localhost** with the address **1.0.0.127.in-addr.arpa**. (The current origin is appended to the 1 in the **name** field, because it does not end with a dot.)

data

The **SOA** data includes the name of the host this data file was created on, the mailing address of the person responsible for the name server, and the following values:

Serial	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 (in seconds) how often 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	The minimum number of seconds for the time to live field on other resource records for this domain.

The **NS** data is the fully qualified name of the name server.

The **PTR** data is the loopback address of **localhost**, in the **in-addr.arpa** domain.

The Primary Master Server's `db.`*`domain`* Files

A primary server has one **`/etc/named.data/db.`*`domain`*** file for each domain for which it is authoritative. ***`domain`*** is the first part of the domain specified with the **`-d`** option in the **`hosts_to_named`** command. This file should contain an **`A`** (address) record for every host in the zone.

The example file shown below, **`db.div`**, contains the following types of records:

`SOA`
: Start of Address record. The **`SOA`** record designates the start of a domain, and indicates that this server is authoritative for the data in the domain.

  In data files, `@` represents the current origin. The current origin is the domain configured in this file, according to the boot file. The boot file says that the **`div.inc.com`** domain is configured in the **`db.div`** file. Therefore, every instance of `@` in the **`db.div`** file represents **`div.inc.com`**.

  The **`SOA`** record indicates the name of the host this data file was created on, the mailing address of the person responsible for the name server, and the following values:

  `Serial`
  : 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 (in seconds) how often 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`
  : The minimum number of seconds for the time to live field on other resource records for this domain.

NS — Name Server records. The **NS** records give the names of the name servers and the domains for which they have authority. The domain for the name servers in the example is the current origin (**div.inc.com**), because **@** was the last domain specified.

A — Address records. The **A** records give the internet addresses for all the hosts in the domain.

The current origin is appended to names that do not end with a dot. For example, **localhost** in the first **A** record is interpreted as **localhost.div.inc.com**.

HINFO — Host Information records. The **HINFO** records indicate the hardware and operating system of the host.

CNAME — Canonical Name record. The **CNAME** record 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. All other resource records should use the canonical name instead of the alias.

WKS — Well Known Service records. The **WKS** record lists the services supported by a host. The list of services comes from the host's **/etc/services** file. There should be only one **WKS** record per protocol per address.

MX — Mail Exchanger records. **MX** records specify a weighted list of hosts to try when mailing to a destination on the Internet. The **MX** data indicates an alternate host or list of hosts that accept mail for the target host if the target host is down or inaccessible. The preference field 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 example below, mail for **rabbit** should go first to **rabbit.div.inc.com**. If **rabbit** is down, its mail should be sent to **indigo.div.inc.com**.

See Chapter 5 for information on **sendmail** and how it uses the name server's **MX** records for mail routing.

```
;
; 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
            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
```

The Primary Master Server's db.*net* Files

A primary server has one **db.*net*** file for each network it serves. ***net*** is the network number specified with the **-n** option in the **hosts_to_named** command. This file should contain a **PTR** (pointer) record for every host in the zone. A **PTR** record allows BIND to translate an IP address back into its host name. BIND resolves the address of a name by tracing down the domain tree and contacting a server for each label of the name.

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 address 143.22.0.3 has the domain name **3.0.22.143.in-addr.arpa**. Note that the four octets of the address are reversed.

```
;
;  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.
```

This example file, **`db.15.19.8`**, contains the following records:

`SOA` Start of Address record. The **`SOA`** record designates the start of a domain, and indicates that this server is authoritative for the data in the domain.

In data files, `@` represents the current origin. The current origin is the domain configured in this file, according to the boot file. The boot file says that the **`8.19.15.in-addr.arpa`** domain is configured in the **`db.15.19.8`** file. Therefore, every instance of `@` in the **`db.15.19.8`** file represents **`8.19.15.in-addr.arpa`**.

The **`SOA`** record indicates the name of the host this data file was created on, the mailing address of the person responsible for the name server, and the following values:

`Serial` 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 (in seconds) how often 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` The minimum number of seconds for the time to live field on other resource records for this domain.

`NS` Name Server records. The **`NS`** records give the names of the name servers and the domains for which they have authority. The domain for the name servers in the example is the current origin (**`8.19.15.in-addr.arpa`**), because `@` was the last domain specified.

PTR Pointer records. **PTR** records are usually used to associate an address in the **in-addr.arpa** domain with the canonical name of a host. The first **PTR** record in the example file associates the name **rabbit.div.inc.com** with the address **119.8.19.15.in-addr.arpa**. (The current origin is appended to the **119** in the first field, because it does not end with a dot.)

To Add a Host to the Domain Data Files

1 Add the host to **/etc/hosts** and run **hosts_to_named** again.

or

Add the host manually, as follows:

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.

c Add the host to the **/etc/hosts** file. If the host is not listed in **/etc/hosts**, someone might run **hosts_to_named**, which overwrites your **db.*domain*** and **db.*net*** files, and the host will be lost.

Examples of these records are shown in “The Primary Master Server’s db.domain Files” on page 117 and “The Primary Master Server’s db.net Files” on page 120.

2 After modifying the domain data files, issue the following command to restart the name server and force it to reload its databases:

```
/usr/sbin/sig_named restart
```

To Delete a Host from the Domain Data Files

1 Delete the host from **/etc/hosts** and run **hosts_to_named** again.

or

Delete the host manually, as follows:

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.

2 After modifying the domain data files, issue the following command to restart the name server and force it to reload its databases:

```
/usr/sbin/sig_named restart
```

Configuring a Secondary Master Name Server

A secondary master server can operate in either of two ways:

1. It can store the authoritative data in backup files on its disk. When this type of secondary server reboots, it reads its data from the backup files and does not have to rely on loading data from a primary server. After it is booted, the secondary server will check with the primary server to verify that its data is up to date.
2. It can store the authoritative data in memory only. When this type of secondary server boots, it always loads its data from a primary master server.

This section explains how to configure a secondary master server in your domain. It contains the following sections:

- To Create the Secondary Master Server's Data Files Using hosts_to_named
- To Create the Secondary Master Server's Data Files Manually
- To Set the Default Domain Name

To Create the Secondary Master Server's Data Files Using `hosts_to_named`

1 If you want your secondary server to store its data in backup files on its disk, run **`hosts_to_named`** on the primary server as follows:

```
/usr/sbin/hosts_to_named -z primary_server's_IP_address
```

If you want your secondary server to always load its data from the primary server, run **`hosts_to_named`** on the primary server as follows:

```
/usr/sbin/hosts_to_named -Z primary_server's_IP_address
```

2 If you ran **`hosts_to_named`** with the **`-z`** option, copy the file **`boot.sec.save`** from the current directory on the primary server to the **`/etc`** directory on the secondary server.

If you ran **`hosts_to_named`** with the **`-Z`** option, copy the file **`boot.sec`** fromt the current directory on the primary server to the **`/etc`** directory on the secondary server.

3 On the secondary server, rename **`/etc/boot.sec.save`** or **`/etc/boot.sec`** to **`/etc/named.boot`**.

4 Copy the files **`/etc/named.data/db.cache`** and **`/etc/named.data/db.127.0.0`** from the primary server to the secondary server.

The format of the data files copied from the primary master server are described in "Configuring a Primary Master Name Server" on page 109.

An example boot file for a secondary master server is shown in "To Create the Secondary Master Server's Data Files Manually" on page 126.

For more information on **`hosts_to_named`**, type **`man 1M hosts_to_named`** at the HP-UX prompt.

To Create the Secondary Master Server's Data Files Manually

1 Copy the files **`/etc/named.boot`**, **`/etc/named.data/db.cache`**, and **`/etc/named.data/db.127.0.0`** from the primary server to the secondary server.

2 On the secondary server, use a text editor to make the following changes to **`/etc/named.boot`**:

   a In every **`primary`** line except the one containing **`db.127.0.0`**, replace the word **`primary`** with the word **`secondary`**.

   b In every **`secondary`** line, add the internet address of the primary server after the domain name.

   c If you do not want your secondary server to store backup files on disk, delete the last field of every **`secondary`** line (the field that specifies the file name).

Following is an example boot file from a secondary master server:

```
;
; type       domain                        server        backup file
;                                          address

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
```

This file specifies a file name in the fourth field for each domain. The secondary server will use this file as a backup file. It will read the authoritative data from the backup file when it reboots, and later it will contact the primary master server to verify the data.

The format of the data files copied from the primary master server are described in "Configuring a Primary Master Name Server" on page 109.

To Set the Default Domain Name

If you will be using an **/etc/resolv.conf** file on your host, configure the default domain name with the **search** or **domain** keyword. See "Configuring the Resolver to Query a Remote Name Server" on page 130. If you will not be using an **/etc/resolv.conf** file, follow these steps:

1 Set the default domain name with the **hostname** command, by appending the domain name to the host name, as in the following example:

```
/usr/bin/hostname indigo.div.inc.com
```

Do not put a trailing dot at the end of the domain name.

2 Set the **HOSTNAME** variable in the **/etc/rc.config.d/netconf** file to the same value, as in the following example:

```
HOSTNAME=indigo.div.inc.com
```

Configuring a Caching-Only Name Server

The boot file of a caching-only name server has no primary or secondary lines, except the primary line for the **0.0.127.in-addr.arpa** domain (the loopback interface). 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.

Follow these steps to create a caching-only server:

1. Copy the files **/etc/named.data/db.127.0.0** and **/etc/named.data/db.cache** from the primary server to the caching-only server.

2. If you ran **hosts_to_named** to create the primary master server, **hosts_to_named** created a file called **boot.cacheonly** in the directory from which it was run. Copy this file to the caching-only server, and rename it **/etc/named.boot**.

   If you created the primary master server manually, without running **hosts_to_named**, create a boot file for the caching-only server called **/etc/named.boot**. It should look like the following example:

```
;
; 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
```

3 If you will be using an **/etc/resolv.conf** file on your host, configure the default domain name with the **search** or **domain** keyword. See "Configuring the Resolver to Query a Remote Name Server" on page 130.You can also configure remote nameservers in **/etc/resolv.conf**. If you will not be using an **/etc/resolv.conf** file, follow these steps:

Set the default domain name with the **hostname** command, as in the following example,

```
/usr/bin/hostname indigo.div.inc.com
```

and set the **HOSTNAME** variable in the **/etc/rc.config.d/netconf** file to the same value, as in the following example:

```
HOSTNAME=indigo.div.inc.com
```

Do not put a trailing dot at the end of the domain name.

Configuring the Resolver to Query a Remote Name Server

Follow these steps if you want your host to query a name server on a remote host:

1 Create a file on your host called **/etc/resolv.conf**. The **/etc/resolv.conf** file has three configuration options:

- **domain** followed by the default domain name. The **domain** entry is needed only when the local system's host name (as returned by the **hostname** command) is not a domain name, and the **search** option is not configured.
- **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 resolver will append these domains, one at a time, to a host name that does not end in a dot, when it constructs queries to send to a name server. The **domain** and **search** keywords are mutually exclusive.

  If you do not specify the **search** option, the default search list will contain only the local domain.
- **nameserver** followed by the internet address (in dot notation) of a name server that the resolver should query. You can configure up to three **nameserver** entries.

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

```
search cs.Berkeley.Edu Berkeley.Edu
nameserver 132.22.0.4
nameserver 132.22.0.12
```

2 If you did not specify the local domain with the **search** or **domain** option, set the default domain name with the **hostname** command, as in the following example,

```
/usr/bin/hostname indigo.div.inc.com
```

and set the **HOSTNAME** variable in the **/etc/rc.config.d/netconf** file to the same value, as in the following example:

```
HOSTNAME=indigo.div.inc.com
```

Do not put a trailing dot at the end of the domain name.

NOTE: If you want to run both BIND and HP VUE, you *must* have an **/etc/resolv.conf** file on your system, or HP VUE will not start.

If a user sets the **LOCALDOMAIN** environment variable, any BIND requests made within the context of the user's shell environment will use the search list specified in the **LOCALDOMAIN** variable. The **LOCALDOMAIN** variable overrides the **domain** and **search** options in **/etc/resolv.conf**.

On HP-UX releases before 10.0, by default, if the resolver could not find the requested host by appending the local domain, it would append the parent of the local domain and the grandparent of the local domain. It would not append just the top-level domain (like **com** or **edu**). For example, if BIND could not find host name **aardvark** in the local domain **zoo.bio.nmt.edu**, it would look for **aardvark.bio.nmt.edu** and **aardvark.nmt.edu** but not **aardvark.edu**.

On HP-UX release 10.0 and later releases, by default, if you do not specify a **search** list in **/etc/resolv.conf**, the resolver will append only the local domain to the input host name.

If you want BIND to behave as it did in releases before 10.0, configure a **search** list in the **/etc/resolv.conf** file. The following **search** list causes BIND to search the **zoo.bio.nmt.edu** domain as it did by default in releases before 10.0:

```
search zoo.bio.nmt.edu bio.nmt.edu nmt.edu
```

CAUTION: In order to reduce situations that may cause connections to unintended destinations, you should carefully select which domains you put in the **search** list in the **/etc/resolv.conf** file. Hewlett-Packard recommends that the possible domains for the **search** list be limited to those domains administered within your trusted organization. For more information on the security implications of search lists, please read RFC 1535, located in the **/usr/share/doc** directory.

Type **man 4 resolver** or **man 5 hostname** the HP-UX prompt for more details, or see "How BIND Resolves Host Names" on page 102.

Starting the Name Server Daemon

The name server daemon, **/usr/sbin/named**, must be running on every primary, secondary, and caching-only name server. If you have configured your system to query a remote name server (that is, if you have created an **/etc/resolv.conf** file that directs BIND queries to a name server on another host), you do not have to run the **named** daemon on your host.

Before you start the name server daemon, make sure **syslogd** is running. **syslogd** logs informational and error messages. For information on configuring **syslogd**, see Chapter 2 in this manual.

Follow these steps to start the name server daemon:

1 In the **/etc/rc.config.d/namesvrs** file, set the **NAMED** environment variable to 1, as follows:

```
NAMED=1
```

2 Issue the following command to determine whether **named** is already running:

```
ps -ef | grep named
```

3 If **named** is not running, issue the following command to start it:

```
/sbin/init.d/named start
```

For more information, type **man 1M named** at the HP-UX prompt.

Verifying the Name Server

1. If you are running **syslogd**, check the **/var/adm/syslog/syslog.log** file for error messages. If error messages are recorded, see "Troubleshooting the BIND Name Server" on page 139.

2. Start **nslookup**(1) with the following command:

```
/usr/bin/nslookup
```

3. At the **>** prompt, issue the **server** command to force **nslookup** to use the server you want to test:

```
> server BIND_server_hostname
```

4. At the **>** prompt, type the name of a host for the name server to look up, as in the following example

```
> charlie
```

   You should see output similar to the following:

```
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
```

5. Look up several host names and IP addresses of hosts in the name server's domain.

6. At the **>** prompt, type the following commands to verify that your name server can query root name servers:

```
> set type=ns
> .
```

   nslookup should display a list of the root name servers in your **db.cache** file. If it does not, see "Troubleshooting the BIND Name Server" on page 139.

7. Type **exit** to exit from **nslookup**.

Updating Network-Related Files

After you configure your system to use BIND, the following network-related configuration files require fully-qualified domain names for all hosts outside your local domain:

/etc/hosts.equiv
$HOME/.rhosts
/var/adm/inetd.sec
$HOME/.netrc

To Update /etc/hosts.equiv and $HOME/.rhosts

Flat or string-type host names that are not hosts in the local domain must be converted to fully qualified domain names in the **/etc/hosts.equiv** file and in all **$HOME/.rhosts** files.

The shell script **convert_rhosts**, found in **/usr/examples/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.

To Update /var/adm/inetd.sec and $HOME/.netrc

Flat or string-type host names that are not hosts in the local domain must be converted to fully qualified domain names in the **/var/adm/inetd.sec** file and in all **$HOME/.netrc** files. No automated utility exists for performing this task, so you must do it manually.

To Update /etc/hosts

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

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 ensuring that up-to-date data is available from multiple, redundant servers.

Follow these steps to add a subdomain:

1. Set up the name servers for the subdomain.
2. Edit the existing zone file, **db.*domain*** on the name server for the parent domain, as follows:
   a. Add an **NS** resource record for each server of the new domain.
   b. Add **A** records to specify the internet addresses of the name servers listed in the **NS** records.

   Following are some lines from the example file **db.nmt**. Hosts **venus.nmt.edu** and **moon.nmt.edu** are name servers for the **nmt.edu** domain. Each of these hosts has two connections to the network, so each requires two **A** records: one for each of its internet addresses.

```
nmt.edu.          86400   IN    NS    venus.nmt.edu.
                  86400   IN    NS    moon.nmt.edu.
venus.nmt.edu.    86400   IN    A     123.4.5.678
                  86400   IN    A     45.6.7.890
moon.nmt.edu.     86400   IN    A     123.9.0.12
                  86400   IN    A     67.8.9.10
```

3. After modifying the domain data files, issue the following command to restart the name server for the parent domain and force it to reload its databases:

```
/usr/sbin/sig_named restart
```

Configuring a Root Name Server

If you are connected to the Internet, use the root servers already available. (For a list of root servers, use anonymous **ftp** to get the file **/domain/named.ca** from **nic.ddn.mil**.) 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, which indicates that the server is primary for the root domain:

```
primary    .    db.root
```

The **db.root** file typically contains only **NS** and **A** resource records for the authoritative name space tree. You can use the **hosts_to_named** command with the **-r** option to create the **db.root** file. Type **man hosts_to_named** for more information.

The **db.cache** file on the other name servers in the domain should contain an entry for this root server.

A domain may have more than one root name server.

Following is an example of the root zone file, **db.root**. In the example **db.root** file, hosts **rabbit.div.inc.com**, **denny.dept.inc.com**, and **sally.doc.inc.com** are authoritative name servers for the root domain. Hosts **eduardo.inc.com** and **labs.inc.com** are authoritative for the **inc.com** subdomain.

```
@               IN     SOA     rabbit.div.inc.com.
                               root.moon.div.inc.com. (
                                    3          ; Serial
                                    10800      ; Refresh after 3 hours
                                    3600       ; Retry after 1 hour
                                    604800     ; Expire after 1 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
```

Configuring BIND in sam

On the local system, you can configure a primary server, a secondary server, a caching-only server, and resolver; start, restart, or stop the server; specify a parent domain, update the DNS database files; and configure NS resource records.

More information on configuring BIND in **sam** can be found by running **sam** and referring to the help screens. You can get to the DNS section by selecting “Networking and Communications” and “DNS (BIND).”

Troubleshooting the BIND Name Server

This section tells you how to identify and correct problems with the BIND name server. It contains the following sections:

- Troubleshooting Tools and Techniques
- Problem Symptoms
- Name Server Problems
- Understanding Name Server Debugging Output
- Name Server Statistics

NOTE: After you configure 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. See “Updating Network-Related Files” on page 134.

If you want to run both BIND and HP VUE, you *must* have an **`/etc/resolv.conf`** file on your system, or HP VUE will not start. See “Configuring the Resolver to Query a Remote Name Server” on page 130.

After you configure the BIND name service, **`sendmail`** will use the name server's **`MX`** (mail exchanger) records for mail routing. See Chapter 5 for information on **`sendmail`**.

Troubleshooting Tools and Techniques

This section describes the available tools for troubleshooting of the BIND name server.

The `ping` command

Use the **`ping`** command to test whether a specific host name can be looked up. You can also use it to check network connectivity to the name server.

```
$ /usr/sbin/ping hostname
```

If host name lookups are failing, use **`ping`** with an IP address to test network connectivity.

```
$ /usr/sbin/ping IP_address
```

The `nslookup` command

The nslookup command sends queries to name servers and displays the contents of their responses. It also tells you whether BIND, NIS, or the **`/etc/hosts`** file is being used for name lookups, and which server your host is using. You can pass a host name to **`nslookup`**, and it will return the corresponding IP address, or you can pass an IP address to **`nslookup`**, and it will return the corresponding host name. For more information, type **`man 1 nslookup`** at the HP-UX prompt.

```
$ /usr/bin/nslookup

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

The `syslogd` Utility

Informational and error messages relating to **`named`** are logged using **`syslogd`**. By default, **`syslogd`** logs messages to the file **`/var/adm/syslog/syslog.log`**, but the destination of these messages is configurable. See Chapter 2 for information on **`syslogd`**.

Name Server Debugging

The debugging output from the name server goes to the file **/var/tmp/named.run**. To turn on **named** debugging, issue the following command:

```
/usr/sbin/sig_named debug level
```

where ***level*** is one of the following debugging levels:

1 This is the most useful debug level. It logs information about transactions being processed. It logs the IP address of the sender, the name looked up, and the IP addresses of other servers queried.

2 The level lists the IP addresses about to be queried and their current round trip time calculations. A secondary server displays information about each zone it is maintaining when it contacts a primary master to see if a zone is up to date.

3 This level gives detailed information about internal operation, 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 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 This level displays the initial query packet and the response packets from other remote servers.

5 This level gives more internal operation information, most of it not helpful.

10 This level shows the packet sent to other servers during name lookup. It also shows the packet the local server sent back to the querying process.

At certain debugging levels, the actual packets are displayed. See RFC 1035 for the format of DNS packets. This RFC is in **/usr/share/doc**.

To turn off named debugging, issue the following command:

```
/usr/sbin/sig_named debug 0
```

See “Understanding Name Server Debugging Output” on page 151. For more information, type **man 1M sig_named** or **man 1M named** at the HP-UX prompt.

Dumping the Name Server Database

The name server dumps its current database and cache to the file **/var/tmp/named_dump.db** when you issue the following command:

```
/usr/sbin/sig_named dump
```

For more information, type **man 1M sig_named** or **man 1M named** at the HP-UX prompt.

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” on page 145.

1 After configuring the primary server for the first time, names in the local domain cannot be found. Check the following:

- 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 the following:

- 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 the following:

- Problem 4, Syntax Errors in **`/etc/resolv.conf`**
- Problem 8, Local Domain Not Set
- Problem 9, **`/etc/nsswitch.conf`** Not Configured Correctly

4 A remote name lookup now fails that has completed successfully before. Check the following:

- Problem 5, Network Connectivity
- Problem 2, Syntax Errors
- Problem 4, Syntax Errors in **`/etc/resolv.conf`**
- Problem 10, **`/etc/hosts`** or NIS Contains Incorrect Data

5 A local name lookup now fails that has completed successfully before. Check the following:

- 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
- Problem 10, **/etc/hosts** or NIS Contains Incorrect Data

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 the following:

- 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.

1 Incorrect parameters supplied to **`hosts_to_named`**.

Check the domain data files to be sure they contain 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.

a **`syslogd`**

Syntax error messages are logged indicating the file name and line number.

b Name server debugging output

Start the name server at debug level 1. Check for syntax error messages in **`/var/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 host names to internet addresses, **`db.`*****`domain`***.

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 **`syslogd`** to log the following message: **`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 the following:

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

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 **/var/tmp/named.run** contains the following: **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 **/var/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. Errors in **/etc/resolv.conf** are silently ignored by the resolver code.

a **nslookup**

This indicates that NIS or the **/etc/hosts** file is being used for lookups.

b **ping *IP_address*** or **ping *hostname***

Only names in the NIS database or **/etc/hosts** file can be looked up. **ping** the remote server's address to verify connectivity.

c Name server debugging output

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.

5 Network connectivity problems may cause certain lookups to fail. See the *Installing and Administering LAN/9000 Software* manual for information on troubleshooting network connectivity.

a Name server debugging output

Turn on debug level 1. **ping** the host name. Check the name server debugging output in **/var/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 **ping**ing 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**

nslookup times out while trying to look up the name.

c **ping *hostname***

A message is returned saying that the host is unknown.

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 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.

```
$ 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.

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

a **nslookup**

Use **nslookup** to query the parent server for delegation information. Execute the following:

```
$ nslookup
> server parent_server_name or IP_address
> set type=NS
> subdomain_name
```

This should show you the **NS** and **A** 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`
	hershey is the default name server for this host.

`>  server eduardo.doc.inc.com.`
`Default Name Server:  eduardo.doc.inc.com`
`Address:  15.19.11.2`
	Set the default name server to be this subdomain's parent server, **eduardo**.

`>  set type=ns`
`>  div.inc.com`
`Name Server:  eduardo.doc.inc.com`
`Address:  15.19.11.2`
	Set query type to **ns** (nameserver). Look up the **div.inc.com** domain.

`Non-authoritative answer:`
`div.inc.com     nameserver = walleye.div.inc.com`
`div.inc.com     nameserver = friday.div.inc.com`
	Name server records for **div.inc.com**, the delegated subdomain.

`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`
	Address records for 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 **A** records for the subdomain. If no **NS** or **A** records exist, the parent server for the subdomain or the root servers are not reachable. If **NS** and **A** records exist, check their correctness. Then try **ping**ing 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.

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**.

a **nslookup**

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

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

9 The **/etc/nsswitch.conf** file, if it exists, is not configured correctly. If you want to query BIND before querying NIS or the **/etc/hosts** file, make sure **dns** is listed first on the **hosts** line. See “Configuring the Name Service Switch” on page 29.

10 The **/etc/hosts** file or NIS contains incorrect data. The name service switch (**/etc/nsswitch.conf**) allows host name lookups in **/etc/hosts** or NIS, and one of those databases contains incorrect data. For information on configuring the **/etc/hosts** file, see “To Edit the /etc/hosts File” on page 40. For information on NIS, see *Installing and Administering NFS Services*.

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

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

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

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
```

- In the first group of four lines, a query is received for **john.dept.inc.com**. The query is forwarded to a root server, **ns.inc.ddn.mil** at address 192.67.67.53
- In the second group of four lines, **ns.nic.ddn.mil** responded with **NS** records for **inc.com**.
- In the third group of four lines, the **inc.com** server responded with **NS** records for **dept.inc.com**.
- In the fourth group of four lines, 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. Debug levels are explained in “Name Server Debugging” on page 141.

`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.

```
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

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

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

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

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

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
```

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.)

Name Server Statistics

The name server keeps track of various statistics. You can print these statistics to the file **/var/tmp/named.stats** by issuing the following command:

```
/usr/sbin/sig_named stats
```

Statistics are appended to the file. The statistics look similar to this:

```
1273431         time since boot (secs)
29802           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
10569           MX querys
424             AXFR querys
179263          ANY querys
```

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 is 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 is 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 is 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.

iqueries is 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 is the number of response packets that the name server receives from queries to other name servers.

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 is the number of responses to queries that contain some information.

FAIL answers is 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 is 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 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 **/var/adm/sendmail/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 is 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 **/var/adm/sendmail/sendmail.cf**.

5

Installing and Administering sendmail

This chapter describes **sendmail**, the Internet Services mail routing facility. **sendmail** relays incoming and outgoing mail to the appropriate programs for delivery and further routing. **sendmail** allows you to send mail to and receive mail from other hosts on a local area network or through a gateway.

This chapter contains the following sections:

- Deciding Whether to Install sendmail
- Installing sendmail
- Creating sendmail Aliases
- How sendmail Works
- Modifying the Default sendmail Configuration File
- Migrating the sendmail Configuration File
- Security
- Troubleshooting sendmail

For more information on **sendmail**, see *sendmail*, by Bryan Costales with Eric Allman and Neil Richert, published by O'Reilly and Associates, Inc.

You cannot use SAM to install, configure, or enable **sendmail**.

NOTE: **sendmail** for HP-UX 10.20 is an HP implementation of version 8.7 of publicly-available **sendmail** software. HP provides support for the features documented in this chapter and in the **sendmail** man page.

Deciding Whether to Install sendmail

You must install **`sendmail`** in order to do the following things:

- Deliver mail to other machines using the SMTP protocol over a LAN or WAN.
- Route X.400 mail using the X.400/9000 delivery agent.
- Route OpenMail or X.400 mail using the OpenMail product.

If you do not install **`sendmail`**, only local and UUCP mail will work. HP-supported user agents (programs that send messages to **`sendmail`**) and delivery agents (programs that **`sendmail`** uses to route messages) are listed in the section "How sendmail Works" on page 179.

NOTE: If you are running an earlier version of **`sendmail`** on your HP-UX system, you cannot use the same **`/etc/mail/sendmail.cf`** configuration file with the HP-UX 10.20 version of **`sendmail`**. See the section "Migrating the sendmail Configuration File" on page 194 for information on how to migrate your pre-HP-UX 10.20 configuration file.

Installing sendmail

When you install **sendmail**, the installation script creates and modifies files on the system that are needed for **sendmail** operation. The **sendmail** configuration file supplied with HP-UX 10.20 will work without modifications for most installations. Therefore, the only steps you must do are: set up **sendmail** servers to run with NFS, configure and start **sendmail** clients, and verify that **sendmail** is running properly.

This section contains information about the following tasks:

- Installing sendmail on a Standalone System
- Installing sendmail on a Mail Server
- Installing sendmail on a Mail Client
- Verifying Your sendmail Installation

NOTE: HP recommends that you use **sendmail** with the BIND nameserver. The BIND nameserver should have an **MX** record for every host in the domain(s) that it serves. For more information on how **sendmail** uses **MX** records, see “MX Records” on page 184.

Installing sendmail on a Standalone System

When **sendmail** is installed, it is automatically configured to send and receive mail for users on the local system only. The standalone system processes all outbound mail and establishes connections to the message destination host or to Mail Exchanger (**MX**) hosts (see “MX Records” on page 184 for more information). The **sendmail** daemon is then started when you reboot the system, so you do not need to make any changes to any system files.

The **sendmail** installation script makes the following configuration changes:

- Sets the **SENDMAIL_SERVER** variable in the /**etc/rc.config.d/mailservs** file to **1**. This ensures that the

sendmail daemon is started whenever you reboot your system or run the **sendmail** startup script.

- Creates **/etc/mail/sendmail.cf** and **/etc/mail/aliases** files with default configurations. These files are created with **root** as the owner, **other** as the group, and permissions set to 0444.

NOTE: If an **/etc/mail/sendmail.cf** file already exists, the existing file is saved to **/etc/mail/#sendmail**. If an **/etc/mail/aliases** file already exists, then the **sendmail** installation script does not create it.

- Creates the file **/etc/mail/sendmail.cw** that contains the hostname and the fully-qualified hostname for the system. For example, the system **dog** in the **domain cup.hp.com** has the following entries in the file:

```
dog
dog.cup.hp.com
```

- Finally, the installation script issues the following command to run the **sendmail** startup script:

```
/sbin/init.d/sendmail start
```

The **sendmail** startup script generates the aliases database from the **/etc/mail/aliases** source file. The generated database is located in the file **/etc/mail/aliases.db**.

The **sendmail** startup script then starts the **sendmail** daemon by issuing the following command:

```
/usr/sbin/sendmail  -bd  -q30m
```

The **-q30m** option tells **sendmail** to process the mail queue every 30 minutes.

For more information about **sendmail** command line options, type **man 1M sendmail** at the HP-UX prompt.

Installing sendmail on a Mail Server

This section describes how to configure a system to allow users on other (client) systems to use **`sendmail`**. The mail server receives mail for local users and for the users on client systems. Users on client systems then NFS mount the mail directory from the server and read mail over an NFS link. For more information on how **`sendmail`** clients and servers work, see "Default Client-Server Operation" on page 187.

The **`sendmail`** installation script performs the configuration changes that are described in "Installing sendmail on a Standalone System" on page 162. To set the system up as an NFS server and allow the **`sendmail`** clients to read and write to the **`/var/mail`** directory, do the following:

1 Make sure all mail users have accounts on the mail server and that their user IDs and group IDs on the mail server are the same as on the client machines. (This step is not necessary if you are using NIS and your mail server is in the same NIS domain as the clients.)

2 In the **`/etc/rc.config.d/nfsconf`** file, use a text editor to set the **`NFS_SERVER`** variable to 1.

3 Use a text editor to add the following line to the **`/etc/exports`** file:

```
/var/mail  -access=client,client...
```

where each mail client is listed in the access list. If the **`/etc/exports`** file does not exist, you will have to create it.

4 Issue the following command to run the NFS startup script:

```
/sbin/init.d/nfs.server  start
```

For more information on NFS, see *Installing and Administering NFS Services*.

Installing sendmail on a Mail Client

sendmail clients do not receive mail on their local system; instead, users on the client systems obtain their mail on the mail server. User mail directories reside on the server, and users read their mail over an NFS link. By default, a **sendmail** client forwards to the server any local mail (a user address destined for the client system) and sends non-local mail directly to the destination system or **MX** host. Outgoing mail appears to originate from the server, so replies are sent to the server. For more information on how **sendmail** clients and servers work, see "Default Client-Server Operation" on page 187. **sendmail** clients can be diskless systems.

To configure a **sendmail** client system to access a **sendmail** server:

1. In the **/etc/rc.config.d/mailservs** file, use a text editor to set the **SENDMAIL_SERVER** variable to **0**. This ensures that the **sendmail** daemon will *not* be started when you reboot your system or run the **sendmail** startup script.
2. In the **/etc/rc.config.d/mailservs** file, use a text editor to set the **SENDMAIL_SERVER_NAME** variable to the host name or IP address of the mail server you will use (the machine that will run the **sendmail** daemon).
3. In the **/etc/rc.config.d/nfsconf** file, use a text editor to set the **NFS_CLIENT** variable to **1**.
4. Use a text editor to add the following line to the **/etc/fstab** file:

   ```
   servername:/var/mail  /var/mail  nfs  0  0
   ```

   where ***servername*** is the name configured in the **SENDMAIL_SERVER_NAME** variable in **/etc/rc.config.d/mailservs**. If the **/etc/fstab** file does not exist, you will have to create it.
5. Issue the following command to run the **sendmail** startup script:

   ```
   /sbin/init.d/sendmail start
   ```

6. Issue the following command to run the NFS startup script:

   ```
   /sbin/init.d/nfs.client start
   ```

The **sendmail** startup script assumes that this system will use the host specified by the **SENDMAIL_SERVER_NAME** variable as the mail hub. The script also assumes that mail sent from this system should appear to be from the host specified by the **SENDMAIL_SERVER_NAME** variable (this feature may previously have been known as “site hiding”). The script therefore modifies the macros **DM** (for “masquerade”) and **DH** (for “mail hub”) in the system’s **/etc/mail/sendmail.cf** file to use the host specified by the **SENDMAIL_SERVER_NAME** variable. Note that if the **DM** and **DH** macros have previously been defined, the startup script does not modify them.

As mentioned earlier, the client system now forwards local mail to the mail server and forwards other mail directly to remote systems. To configure the client system to relay all mail to the mail server for delivery, see “Modifying the Default sendmail Configuration File” on page 191.

The NFS startup script NFS-mounts the **/var/mail** directory from the mail server to your system. For more information on NFS, see *Installing and Administering NFS Services*.

Verifying Your sendmail Installation

You can verify that **`sendmail`** has been installed properly and is working properly by doing the following:

- Mailing to a Local User
- Mailing to a Remote User with UUCP Addressing (if you are using it).
- Mailing to a Remote User with the SMTP Transport (if you are using it).

Mailing to a Local User

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 user **`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 **`/var/adm/syslog/mail.log`** file should have been logged for the local message transaction. See “Configuring and Reading the sendmail Log” on page 204 for more information.

Mailing to a Remote User with UUCP Addressing

For this test, mail a message to a remote user with the UUCP transport by using a ***host*!*user*** address, where ***host*** is a system to which your local host has a direct UUCP connection. (The **uuname** command lists the UUCP names of known systems. Type **man 1 uuname** at the HP-UX prompt for more information.)

To verify both inbound and outbound UUCP connections, mail the message in a loop, using the syntax ***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 **/var/adm/syslog/mail.log** file should have been logged for the UUCP mail transaction. See “Configuring and Reading the sendmail Log” on page 204 for more information.

NOTE: In this example, if you mail to yourself, and if the remote system is running **sendmail**, be sure the configuration file on the remote system has set the **m** option (for a pre-version 6 configuration file) or the **MeToo** option (for a version 6 configuration file). The remote system’s configuration file should contain a line beginning with **Om** or **O MeToo**. 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.

Mailing to a Remote User with the SMTP Transport

For this test, mail a message to a remote user with the SMTP transport using 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, mail the message in a loop, using the syntax ***user%my_host@remote_host***. For example, if you try,

```
date | mailx -s "Round Robin SMTP" joe%node2@node1
```

you should receive a message similar to the following:

```
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
```

An entry in your **/var/adm/syslog/mail.log** file should have been logged for the SMTP mail transaction. See “Configuring and Reading the sendmail Log” on page 204 for more information.

NOTE: In this example, if you mail to yourself, and if the remote system is running **sendmail**, be sure the configuration file on the remote system has set the **m** option (for a pre-version 6 configuration file) or the **MeToo** option (for a version 6 configuration file). The remote system’s configuration file should contain a line beginning with **Om** or **O MeToo**. 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.

Creating sendmail Aliases

The **`sendmail`** aliases database stores mailing lists and mail aliases. You create the aliases database by adding aliases to the file **`/etc/mail/aliases`** and then running the **`newaliases`** script to generate the database from the file. The generated database is stored in the file **`/etc/mail/aliases.db`**. The **`sendmail`** startup script also generates the aliases database when you reboot your system.

Each user on your system can create a list of alternate mailing addresses in a **`.forward`** file in his or her home directory. The **`.forward`** file allows the user to forward his or her own mail to files or to other mailing addresses.

This section tells you how to perform the following tasks:

- Adding sendmail Aliases to the Alias Database
- Verifying Your sendmail Aliases
- Managing sendmail Aliases with NIS (Network Information Service)
- Rewriting the “From” Line on Outgoing Mail
- Forwarding Your Own Mail with a .forward File

Adding sendmail Aliases to the Alias Database

1. If the file **`/etc/mail/aliases`** does not exist on your system, copy it from **`/usr/newconfig/etc/mail/aliases`** to **`/etc/mail/aliases`**.
2. Use a text editor to add lines to the file. Each line has the following form:

   ```
   alias: mailing_list
   ```

   where ***alias*** is a local address, local user name, or local alias, and ***mailing_list*** is a comma-separated list of local user names or aliases, remote addresses, file names, commands, or included files. Table 3 lists the types of things you can include in a mailing list and the syntax for each one.
3. Issue the following command to regenerate the aliases database from the **`/etc/mail/aliases`** file:

   ```
   /usr/sbin/newaliases
   ```

   This command creates the aliases database, which is located in the file **`/etc/mail/aliases.db`**.

Table 3 **Things That May Be Included in a Mailing List**

user_name	A local user name will be looked up in the aliases database unless you put a backslash (\) before it. To prevent **`sendmail`** from performing unnecessary alias lookups, put backslashes before local user names. Example:
	`local_users: \amy, \carrie, \sandy, \anne, \david,`
	`             \tony`
	`remote_users: mike, denise`
	`mike: mike@chem.tech.edu`
	`denise: bigvax!amlabs!denise`

remote_address	The remote address syntax that **`sendmail`** understands is configured in the **`sendmail`** configuration file and usually includes RFC 822 style addressing (***user@domain***) and UUCP style addressing (***host!user***). Example:
	`chess_club: mike@chem.tech.edu, marie@buffalo,`
	`            bigvax!amlabs!denise`

Table 3 Things That May Be Included in a Mailing List

<table>
<tr><td>filename</td><td>An absolute pathname on the local machine. sendmail appends a message to the file if the following conditions are true:
1 The file exists, is not executable, and is writable by all.
2 The directory where the file resides is readable and searchable by all. Example:
<pre>public: /tmp/publicfile
terminal: /dev/tty</pre>Mail addressed to public is appended to /tmp/publicfile. Mail addressed to terminal appears on the sender’s terminal.</td></tr>
<tr><td>"| command"</td><td>sendmail pipes the message as standard input to the specified command. The double quotes are required to protect the command line from being interpreted by sendmail. Commands must be listed as full pathnames.

If stdout and stderr are not redirected, they are not printed to the terminal, and they disappear. However, if a command returns a non-zero exit status, its output to stderr becomes part of the sendmail error transcript.

The command is executed by the prog mailer defined in the configuration file. In the configuration file supplied with HP-UX, the prog mailer is configured as “sh -c”. Example:
<pre>prog: "| /usr/bin/cat | /usr/bin/sed
       's/Z/z/g' > /tmp/outputfile"</pre>Mail addressed to prog is saved in /tmp/outputfile with all capital Z’s changed to lowercase z’s.</td></tr>
<tr><td>:include:filename</td><td>Any mail addressed to the alias is sent to all the recipients listed in the included file. The file must be a full pathname. Non-root users can create :include files for maintaining their own mailing lists. An :include file can contain anything that can be specified in the right side of an alias definition. Example alias dfinition:
<pre>dogbreeders: :include:/users/andrea/dogbreeders</pre>Example :include file:
<pre>#file included in dogbreeders alias definition:
terriers@akc.ny.com, coonhounders@ukc.sc.com</pre></td></tr>
</table>

An alias can be continued across multiple lines in the aliases file. Lines beginning with blanks or tabs are continuation lines.

The aliases file can contain comment lines, which begin with #. Blank lines in the aliases file are ignored.

NOTE: You cannot address messages directly to file names, command lines, or **`:include`** files. **`sendmail`** will deliver messages to these only if they appear in the right side of an alias definition.

Configuring Owners for Mailing Lists

Because the sender of a message often does not control the mailing list to which the message is addressed, **`sendmail`** allows you to configure an owner for a mailing list. If **`sendmail`** encounters an error while attempting to deliver a message to the members of a mailing list, it looks for an alias of the form **`owner-`*`mailing_list`*** and sends the error message to the owner. For example, if mike were responsible for maintaining the **`chess_club`** mailing list, he could be configured as the owner:

```
chess_club:  mike@chem.tech.edu, marie@buffalo,
             bigvax!amlabs!denise, margaret@hp.com

owner-chess_club:  mike@chem.tech.edu
```

Any errors **`sendmail`** encountered while trying to deliver mail to the members of the **`chess_club`** mailing list would be reported to mike.

Avoiding Alias Loops

You should avoid creating aliasing loops. Loops can occur either locally or remotely. Following is an example of a local alias loop:

```
#Example of a local aliasing loop
first : second
second : first
```

When regenerating the alias database, **newaliases** does not notice a loop like the one shown in the previous example. However, after the alias database is generated, 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**.

In the previous 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.

Following is an example of a remote aliasing loop:

```
# Example alias entry on host sage
dave : dave@basil

# Example alias entry on host basil
dave : dave@sage
```

Mail sent to **dave** at either host **sage** or host **basil** 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.

Creating a Postmaster Alias

RFC 822 requires that a “postmaster” alias be defined on every host. The postmaster is the person in charge of handling problems with the mail system on that host. The default aliases file supplied with HP-UX defines the postmaster to be root. You can change this alias to the appropriate user for your system.

Verifying Your sendmail Aliases

After you have created a **sendmail** alias and regenerated the aliases database, issue the following command to verify that your alias is valid:

```
/usr/sbin/sendmail  -bv  -v  alias, alias, . . .
```

The **-bv** option causes **sendmail** to verify the aliases without collecting or sending any messages. Any errors in the specified aliases will be logged to standard output.

Users can use the HP **expand_alias** utility to expand an alias or mailing list as far as is possible. For more information on the **expand_alias** utility, type **man 1M expand_alias** at the HP-UX prompt.

Managing sendmail Aliases with NIS (Network Information Service)

The **sendmail** aliases database can be managed through the Network Information Service (NIS), which is one of the NFS Services. NIS allows you to maintain an aliases database on one server system. All other systems request alias information from the server. In order to use NIS, you must set up an NIS domain and configure the machines in your network as NIS servers and clients. For information on NIS, see *Installing and Administering NFS Services*.

When you configure NIS in your network, it manages your **sendmail** aliases by default, so you do not have to make any changes to your NIS configuration. The **ypinit**(1M) script creates NIS “maps” from system configuration files like **/etc/hosts** and **/etc/passwd**. By default, it also creates the **mail.aliases** and **mail.byaddr** maps from the **/etc/mail/aliases** file.

Before you run the **ypinit** script, make sure the **/etc/mail/aliases** file on the NIS master server contains all the **sendmail** aliases you want to make globally available through NIS.

The **sendmail** program attempts to resolve aliases first in the local **sendmail** aliases database. If it fails to find an alias in the local database, and if NIS is running, **sendmail** consults NIS to resolve the alias. On HP-UX, no plus sign is required in the local aliases file to force **sendmail** to search the NIS aliases database.

Modifying Your NIS Aliases Database

Follow these steps to make changes to your NIS **aliases** database:

1. On the NIS master server, make your changes to the **/etc/mail/aliases** file.
2. Issue the following command on the NIS master server to regenerate the **aliases** database and push it out to the NIS slave servers:

```
cd /var/yp
/usr/ccs/bin/make aliases
```

To look up hostnames, **sendmail** first tries to use DNS, then the **/etc/hosts** file. To look up aliases, sendmail tries to the **/etc/mail/aliases** file. To specify a different lookup sequence or to specify a different lookup source (such as NIS), do the following:

1 Edit the **/etc/mail/service.switch** file to specify the lookup sequence. For example, to specify NIS for both hostname and aliases lookup enter the following:

```
hosts    NIS
aliases  NIS
```

For information on the syntax of this file, type **man 1M service.switch** at the HP-UX prompt.

2 Open the **/etc/mail/sendmail.cf** file with a text editor and uncomment the following line:

```
O ServiceSwitchFile=/etc/mail/service.switch
```

Rewriting the "From" Line on Outgoing Mail

HP provides a method that allows the "From" line on mail to be rewritten. This can be useful where a user's login name does not clearly identify the user to intended mail recipients. For example, mail sent by "bkelley (mailname)" can be changed to read from "Bob_Kelley (maildrop)".

To rewrite "From" lines on outgoing mail:

1 Create the file **/etc/mail/userdb** that contains two entries for each mail user who will have outgoing mail on the system. The entries should be in the following format:

```
bkelley:mailname        Bob_Kelley
Bob_Kelley:maildrop     bkelley
```

2 Build the **/etc/mail/userdb.db** file with the **makemap** routine:

```
makemap btree /etc/mail/userdb.db < /etc/mail/userdb
```

3 Uncomment the following line in the **`/etc/mail/sendmail.cf`** file:

```
UserDatabaseSpec=/etc/mail/userdb.db
```

Forwarding Your Own Mail with a .forward File

You can redirect your own mail by creating a **`.forward`** file in your home directory. If a **`.forward`** file exists in your home directory and is owned by you, **`sendmail`** will redirect mail addressed to you to the addresses in the **`.forward`** file.

A **`.forward`** file can contain anything that can appear on the right side of an alias definition, including programs and files. (See Table 3.) Following is an example of a **`.forward`** file owned by user alice on host **`chicago`**:

```
alice@miami, alice@toronto, \alice,  mycrew
```

Mail sent to **`alice@chicago`** will be delivered to alice's accounts on hosts **`miami`** and **`toronto`** as well as to her account on local host **`chicago`**. It will also be delivered to all the recipients of the mailing list **`mycrew`**, which must be defined in the local aliases database or in an **`:include`** file on host **`chicago`**.

The aliases database is read before a **`.forward`** file. The **`.forward`** file is read only if the user's name is not defined as an alias or if an alias expands to the user's name.

How sendmail Works

`sendmail` acts as a post office to which all messages can be submitted for routing. **`sendmail`** can interpret both Internet-style addressing (that is, ***`user@domain`***) and UUCP-style addressing (that is, ***`host!user`***). How addresses are interpreted is controlled by the **`sendmail`** configuration file. **`sendmail`** can rewrite message addresses to conform to standards on many common target networks.

This section discusses the following topics:

- Message Structure
- How sendmail Collects Messages
- How sendmail Routes Messages
- Default Client-Server Operation
- How sendmail Handles Errors

Message Structure

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.

How sendmail Collects Messages

sendmail can receive messages from any of the following:

- A user agent that calls **sendmail** to route a piece of mail. User agents that are supported by HP for use with HP-UX 10.20 **sendmail** include **elm**, **mail**, **mailx**, and **rmail**.
- A **sendmail** daemon or other mail program that calls **sendmail** to route a piece of mail received from the network or the mail queue.
- A user that calls **sendmail** directly from the command line.

How sendmail Routes Messages

To route the message, **sendmail** does the following:

1. Rewrites the recipient and sender addresses given to it to conform to the standards of the target network.
2. If necessary, adds lines to the message header so that the recipient is able to reply.
3. Passes the mail to one of several specialized delivery agents for delivery.

Figure 8 on page 181 outlines the flow of messages through **sendmail**.

Once **sendmail** collects a message, 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, **/etc/mail/sendmail.cf**.

A separate delivery agent is invoked for each host to which messages are being routed. Some delivery agents can accept multiple users in a given invocation. Others must be invoked separately for each recipient. Delivery agents that are supported by HP for use with HP-UX 10.20 **sendmail** include SMTP, UUCP, X.400, and OpenMail.

To invoke a delivery agent, **sendmail** constructs a command line according to a template in the configuration file.

If the delivery agent is specified as IPC, **sendmail** does not invoke 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.

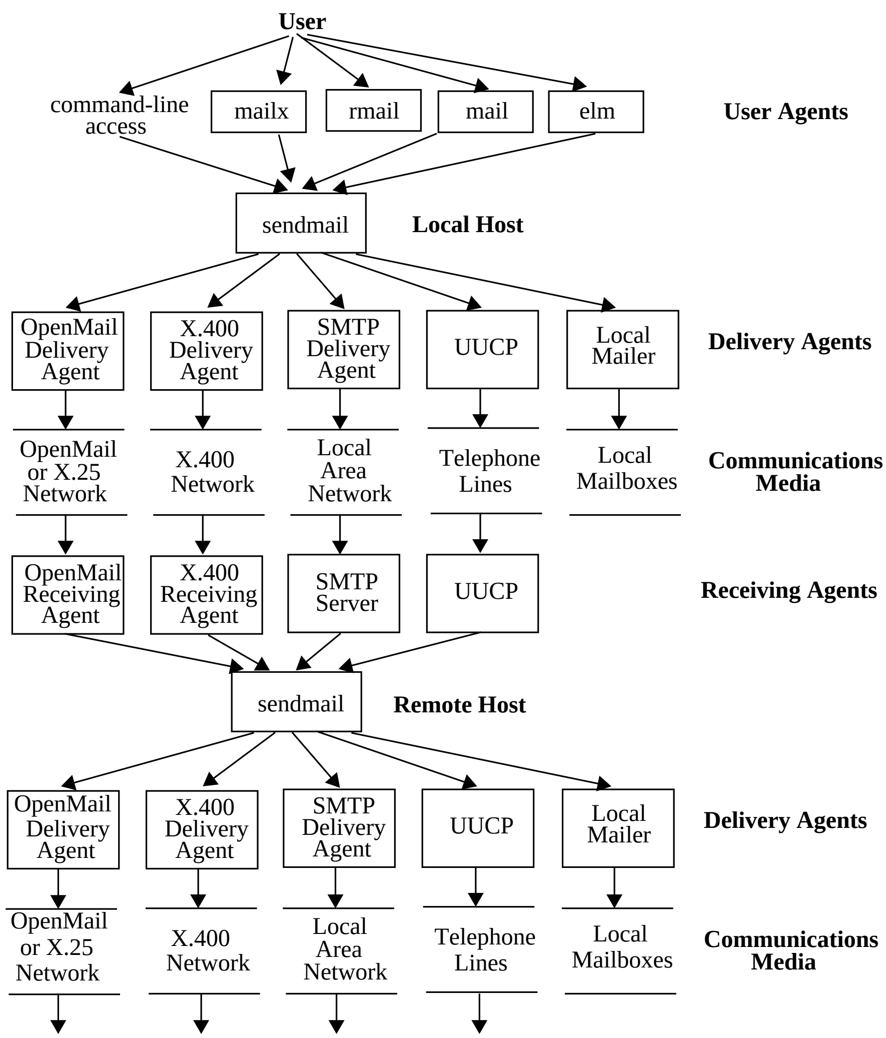


Figure 8 **Flow of Mail Through sendmail**

If an address resolves to the local mailer, **`sendmail`** looks up the address in its alias database and expands it appropriately if it is found.

The aliasing facility or a user's **`.forward`** file may be used to route mail to programs and to files. (**`sendmail`** does not mail directly to programs or files.) Mail to programs is normally piped to the **`prog`** mailer (**`/usr/bin/sh -c`**), which executes the command specified in the alias or **`.forward`** file definition. (You can restrict the programs that can be run through the **`aliases`** or **`.forward`** files. See "Security" on page 196 for more information.) Mail to a file is directly appended to the file by **`sendmail`** if certain conditions of ownership and permission are met.

After all alias expansion is complete, mail that is addressed to a local user name is routed to the local mailer (**`/usr/bin/rmail`**), which deposits the message in the user's mailbox.

The Default Routing Configuration

The installed configuration file, if unmodified, routes mail depending on the syntax of the recipient addresses as described in the following sections.

Local Addresses The following forms are recognized as local addresses and are 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, addresses of the following forms are recognized as UUCP addresses:

```
host!user
host!host!user
user@host.uucp
```

If your host has a direct UUCP connection to the next host in the path, the mail is delivered to that host through 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 following forms, where ***host*** is not the local host name, are routed by SMTP over TCP/IP:

```
user@host
user@host.domain
<@host,@host2,@host3:user@host4>
user@[remote_host's_internet_address]
```

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.

Mixed Addresses The supplied configuration file interprets address operators with the following precedence:

@, !, %

This means that recipient addresses using mixtures of these operators are resolved as shown Table 4.

Table 4 How sendmail Resolves Addresses with Mixed Operators

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**

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. For information on creating **MX** records, see “Configuring and Administering the BIND Name Service” on page 95.

MX records are used only 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 successfully connects. 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.

If **sendmail** tries all the **MX** hosts in the list and fails, the message is returned to the sender with an error message. If you want **sendmail** to try to connect to the host to which the message is addressed, uncomment the following line in the **/etc/mail/sendmail.cf** file:

```
TryNullMXList
```

sendmail then tries to connect to the host to which the message is addressed, if any of the following conditions occur:

1. The nameserver returns no **MX** records.
2. The nameserver is not running.
3. The local host is the highest preference mail exchanger in the list.

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.

MX records are used for two main purposes:

1. To arrange that one host "back up" another by receiving mail for it when it is down.
2. To arrange that mail addressed to remote networks be relayed through the appropriate gateways.

In the following example, the nameserver serving the domain **paf.edu** has the following **MX** records configured to provide backup for host **bling**:

```
;name    ttl     class    MX   preference    mail exchanger

bling            IN       MX   0             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 (that is, the lowest number) in the list.

The following example relays messages through a gateway:

```
;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.

MX Failures Several possible failures are associated with **MX** configuration:

- The nameserver query for **MX** records fails.

  The query fails because no **MX** records exist for the target host or because the nameserver is not running. You can set the **TryNullMXList** option in the **/etc/mail/sendmail.cf** file if you want **sendmail** to always try to connect to the host to which the message is addressed (see "MX Records" on page 184).

  If the query fails temporarily (that is, **h_errno** is set to **TRY_AGAIN**) the message will be queued. 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 (that is, 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 queued.

- 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.

Default Client-Server Operation

This section describes the operation of **`sendmail`** servers and clients. This section assumes that **`sendmail`** is installed as described earlier in this chapter.

Figure 9 shows a **`sendmail`** server called **`mailserv`** and a **`sendmail`** client called **`mailclient`** in the **`company.com`** domain. On **`mailclient`**, the **`SENDMAIL_SERVER_NAME`** in the **`/etc/rc.config.d/mailservs`** file is set to **`mailserv.company.com`**. **`user1`** is a user on **`mailclient`**.

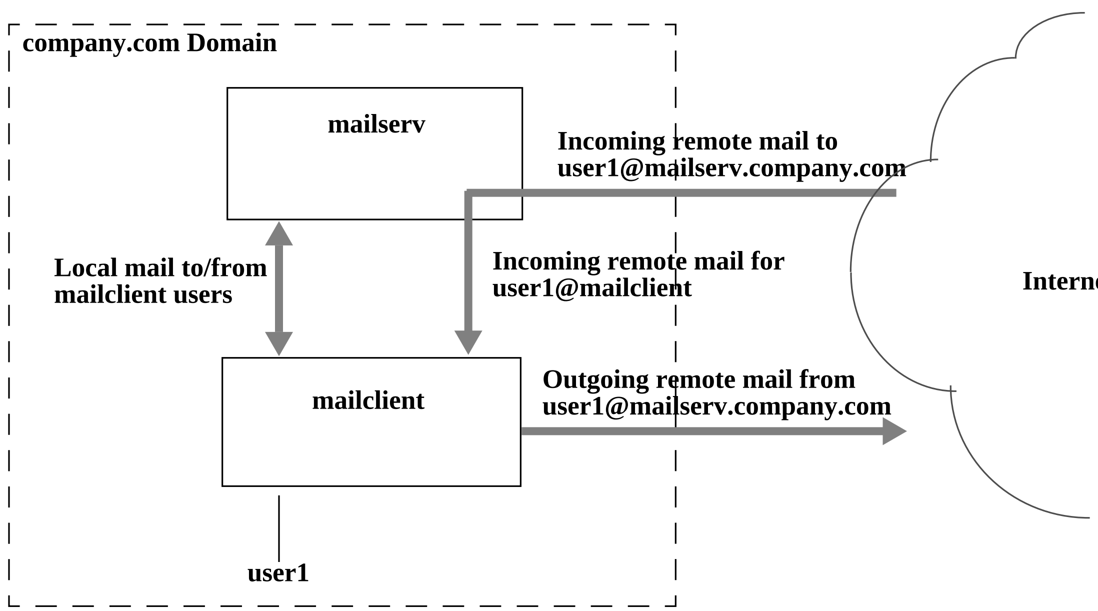


Figure 9 **sendmail Client-Server Operation**

Outgoing mail from **`user1`** can be "local" mail that is intended for any user on **`mailclient`**. Local mail is forwarded to **`mailserv`**; this is specified by the setting of the **`DH`** macro entry in the **`/etc/mail/sendmail.cf`** file on **`mailclient`**. (The **`sendmail`** installation script sets the **`DH`** macro value to the host specified by **`SENDMAIL_SERVER_NAME`**.) Outgoing mail that is not local is sent by **`mailclient`** to the remote host using **`MX`** records. Note that because the **`DM`** macro entry in the **`/etc/mail/sendmail.cf`** file on **`mailclient`** is set to **`mailserv.company.com`**, mail from **`user1`** appears to be from **`user1@mailserv.company.com`**.

Since mail sent to remote hosts from **user1** is sent from **user1@mailserv.company.com**, replies to **user1**'s messages are returned to **mailserv**. On **mailserv**, when **sendmail** receives mail for **user1**, it looks up **user1** in the aliases database and redirects mail for **user1** to **user1@mailclient**.

You can modify **sendmail** server and client operations. Most modifications involve changing or re-creating the **/etc/mail/sendmail.cf** file on the server or client systems. For example, you can define the **DM** macro on a mail server system. You can also modify the **/etc/mail/sendmail.cf** file so that the clients relay all outbound mail to the server; this is described in "Modifying the Default sendmail Configuration File" on page 191.

How sendmail Handles Errors

By default **sendmail** immediately reports to standard output any errors that occur during the routing or delivery of a message. **sendmail** distinguishes between "temporary failures" and "permanent 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. A delivery failure of the local mailer because the file system is full is also a permanent failure.

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.

How sendmail Handles "Permanent" Failures

Permanent failures include the following:

- Temporary failures that have remained in the mail queue for the queue timeout period (set with the **Timeout.queuereturn** option in the **/etc/mail/sendmail.cf** file), which is normally five days.
- Local recipient user unknown.
- The recipient address cannot be resolved by the configuration file.
- 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.

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.

If **sendmail** tries all **MX** hosts in its preference list and fails to deliver a message, the message is returned to the sender with an error message. For more information, see "MX Records" on page 184.

If delivery failed on an alias, and an owner is configured for that alias in the aliases database, **sendmail** returns the message and transcript to the alias owner.

If there is an **Errors-To:** header line in the message header, **sendmail** returns the message and transcript to the address on the **Errors-To:** line instead of to the sender.

If the Postmaster Copy option (option **P**) is set to a valid address, **sendmail** sends a copy of the transcript and failed message (with the message body deleted) to the Postmaster Copy address.

If the attempt to return the failed message itself fails, **sendmail** returns the message and transcript to the alias **postmaster** on the local system. The **postmaster** alias in the default alias file (**/usr/newconfig/etc/mail/aliases**) resolves to root.

If **sendmail** is unable to return the message to any of the addresses described above, as a last resort it appends the error transcript and returned message to the file **/var/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.

How sendmail Handles "Temporary" Failures

Messages that fail temporarily are saved in the mail queue and retried later. By default, the mail queue is stored in the directory **/var/spool/mqueue**. **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, as in the following example:

```
/usr/sbin/sendmail -bd -q30m
```

In this example, every 30 minutes, **sendmail** processes any messages currently in the queue.

When processing the queue, **sendmail** first creates and sorts a list of the messages in the queue. **sendmail** reads the queue control file for each message to collect the pre-processed envelope information, the header lines, and the name of the data file containing the message body. **sendmail** then processes the message just as it did when it was originally collected.

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 **Timeout.queuereturn** option in the **/etc/mail/sendmail.cf** file and, by default, is five days.

Modifying the Default sendmail Configuration File

The **sendmail** configuration file that is supplied with HP-UX will work correctly for most **sendmail** configurations, so you probably do not need to modify it. However, certain modifications to the file are supported. This section describes examples of modifications that you may want to make. The configuration file itself also contains instructions for making the supported modifications.

This section contains the following subsections:

- The sendmail Configuration File
- Restarting sendmail
- Example Modifications

CAUTION: Hewlett-Packard supports the default configuration file and all the modifications described in it. If you make any changes other than the ones described in the default configuration file, Hewlett-Packard cannot support your configuration.

The sendmail Configuration File

The default configuration file is located in **/usr/newconfig/etc/mail/sendmail.cf** and is installed in **/etc/mail/sendmail.cf**. It is recommended that you leave the copy in **/usr/newconfig** unmodified, in case you need to reinstall the default configuration.

The **sendmail** configuration file performs the following functions:

- 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.”

Restarting sendmail

- Issue the following commands, on a standalone system or on the mail server, to restart **sendmail**:

```
/sbin/init.d/sendmail stop
/sbin/init.d/sendmail start
```

You must restart **sendmail** if changes are made to any of the following:

- The **sendmail** configuration file, **/etc/mail/sendmail.cf**.
- The UUCP configuration, as reflected in the output of the **uuname** command.

Example Modifications

This section describes some modifications to the **/etc/sendmail.cf** file that you may find useful.

Configuring a sendmail Client to Relay All Mail to a Server

As mentioned previously, the default behavior for a **sendmail** client is to forward only local mail to the **sendmail** server for delivery; the client system sends non-local mail directly to remote systems. If you want the client to relay all mail (local and non-local) to the **sendmail** server, you will need to build a new **sendmail.cf** file using the **m4** macros. (**m4** macros are a set of library routines used to create customized **sendmail** configuration files.) Follow these steps to create a new **sendmail.cf** file:

1 Change directory to the **/usr/newconfig/etc/mail/cf/cf** directory:

```
cd /usr/newconfig/etc/mail/cf/cf
```

2 Copy the file **examples/clientproto.mc** to this directory:

```
cp examples/clientproto.mc .
```

3 Edit the following statements in the **`clientproto.mc`** file:

```
OSTYPE(hpux10)
...

FEATURE(mailhost.$m)
```

For ***`mailhost`***, enter the hostname of the **`sendmail`** server. Example statements are shown below:

```
OSTYPE(hpux10)
...

FEATURE(dog.$m)
```

4 Use the **`m4`** macros to build a new **`/etc/mail/sendmail.cf`** configuration file. (If you made changes to the existing **`sendmail`** configuration file that you want to retrofit to the new configuration file, you should first save the file to another name.)

```
m4 ../m4/cf.m4 clientproto.mc > /etc/mail/sendmail.cf
```

Forwarding Non-Domain Mail to a Gateway

Mail that is being sent to a domain other than the sender's domain can be forwarded to a mail gateway. To have non-domain mail forwarded to a mail gateway:

- Edit the DS line in the /**`etc/mail/sendmail.cf`** file to specify the hostname of the mail gateway:

```
DSmailgw.cup.hp.com
```

Migrating the sendmail Configuration File

sendmail for HP-UX 10.20 includes many new features as well as changes in operations from earlier versions of **sendmail**. See the *HP-UX 10.20 Release Notes* (part number 5964-5283) for more information on the new and changed features of **sendmail**.

In particular, the functions and format of the **sendmail** configuration file **/etc/mail/sendmail.cf** have changed; you cannot use an earlier version of the **sendmail** configuration file with HP-UX 10.20. This section discusses the methods of migrating an earlier version of the **sendmail** configuration file to the HP-UX 10.20 version.

The **/etc/mail/sendmail.cf** file that is installed with the HP-UX 10.20 **sendmail** software contains a default configuration for use with HP-UX 10.20 systems. The script for installing Internet services on HP-UX 10.20 moves an existing **/etc/mail/sendmail.cf** file to **/etc/mail/#sendmail.cf**, so you will still have the original file for your reference.

NOTE: If there is an existing version 6 /**etc/mail/sendmail.cf** file, then that file is *not* overwritten by the HP-UX 10.20 script. If the existing **/etc/mail/sendmail.cf** file is not version 6, then the file is copied to **/etc/mail/#sendmail.cf** and a version 6 **/etc/mail/sendmail.cf** file is written. In either case, a sample version 6 configuration file can be found in **/usr/newconfig/etc/mail/sendmail.cf**.

There are three methods of migrating your configuration file:

- Make any modifications that you need to **/etc/mail/sendmail.cf** file that is installed with HP-UX 10.20. This method is recommended where you have minimal site-specific changes to the **sendmail.cf** file.
- Create a new **sendmail.cf** file using the **m4** macros. The **m4** macros are a set of library routines that you can use to create customized **sendmail.cf** files. Generally you use the **m4** macros only for highly-customized **sendmail** applications (for example, setting up mail systems for an Internet service provider).

If you must use the **m4** macros to create your **sendmail** configuration file, read the information in the following files first:

- **/usr/newconfig/etc/mail/cf/README** contains information about creating a configuration file from the **m4** macros.
- **/usr/newconfig/etc/mail/cf/README.hpux10** contains information about HP macros and macro changes that are recommended for HP systems.

- Use the **convert_awk** utility to convert the old configuration file into a format required by HP-UX 10.20 **sendmail**. Note that while the resulting file is usable by HP-UX 10.20 **sendmail**, it does not allow you to use the format and options available with the HP-UX 10.20 **sendmail.cf** file. For this reason, you should only use **convert_awk** if your existing **sendmail** configuration file contains many site-specific rulesets that are not easily redefined in the HP-UX 10.20 **sendmail.cf** format.

Security

This version of **sendmail** supports the IDENT protocol, as defined in RFC 1413. Specifically, **sendmail** includes the following programs:

- The **identd** daemon logs the names of users who initiate **sendmail** connections. To execute **identd** from **inetd**, uncomment the following line in the **/etc/inetd.conf** file:

```
ident stream tcp wait bin /usr/bin/identd identd -w -t120
```

  For more information on **identd**, type **man 1M identd** at the HP-UX prompt.

- The **idlookup** utility identifies the user of an incoming connection by contacting **identd** on the remote system. For more information on **idlookup**, type **man 1M idlookup** at the HP-UX prompt.
- The **owners** utility identifies the user of an outgoing connection by contacting **identd** on the local system. For more information on **owners**, type **man 1M owners** at the HP-UX prompt.

sendmail on HP-UX 10.20 allows the **aliases** file or a user's **.forward** file to specify programs to be run. These programs are by default invoked through **/usr/bin/sh -c**. The **sendmail** restricted shell (**smrsh**) program allows you to restrict the programs that can be run through the **aliases** file or through a **.forward** file; only programs that are linked to the **/var/adm/sm.bin** directory can be invoked.

To use the **smrsh** program:

1. In the **/etc/mail/sendmail.cf** file, comment out the following lines (by inserting a pound sign (#) before each line):

```
#Mprog, P=/usr/bin/sh, F=lsDFMoeu, S=10/30, R=20/40, D=$z:/,
#       T=X-Unix,
#       A=sh -c $u
```

2 In the **/etc/mail/sendmail.cf** file, uncomment the following lines (by deleting the pound sign (#) before each line):

```
Mprog, P=/usr/bin/smrsh, F=lsDFMoeu, S=10/30, R=20/40, D=$z:/,
       T=X-Unix,
       A=smrsh -c $u
```

3 Create the directory **/var/adm/sm.bin/** with root:bin ownership and 755 permissions. Place the binaries of the programs that you want to allow into this directory. Typically, programs such as **vacation**, **rmail**, and **AutoReply** are replaced in this directory. (You can also specify hard links to the binaries.) You should not place shells such as **ksh**, **sh**, **csh**, and **perl** in this directory as they have too many security issues.

Troubleshooting sendmail

This section describes the following techniques for troubleshooting **`sendmail`**:

- Keeping the Aliases Database Up to Date
- Verifying Address Resolution and Aliasing
- Verifying Message Delivery
- Contacting the sendmail Daemon to Verify Connectivity
- Setting Your Domain Name
- Attempting to Start Multiple sendmail Daemons
- Configuring and Reading the sendmail Log
- Printing and Reading the Mail Queue

Almost all **`sendmail`** troubleshooting must be done as superuser.

Keeping the Aliases Database Up to Date

The aliases database must be rebuilt if changes have been made to the aliases text file.

You must restart **sendmail** after you change the configuration file or the aliases database.

Issue the following commands, on a standalone system or on the mail server, to rebuild the aliases database and restart **sendmail**:

```
/sbin/init.d/sendmail stop
/sbin/init.d/sendmail start
```

Updating Your NIS Aliases Database

If you are using NIS to manage your aliases database, follow these steps to make changes to the database:

1. On the NIS master server, make your changes to the **/etc/mail/aliases** file.
2. Issue the following command on the NIS master server to regenerate the **aliases** database and push it out to the NIS slave servers:

```
/usr/ccs/bin/make aliases
```

Verifying 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, issue the following command:

```
/usr/sbin/sendmail  -bv  -v  -oL10  address [address...]
```

The **-bv** (verify mode) option causes **sendmail** to verify addresses without collecting or sending a message.

The **-v** (verbose) flag causes **sendmail** to report alias expansion and duplicate suppression.

The **-oL10** (log level) option sets the log level to 10. At log level 10 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, because **MX** records are not collected until delivery is actually attempted.

If the address is not being resolved as you expect, you may have to modify one or more of the following:

- 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/sbin/sendmail  -bt
```

Verifying Message Delivery

You can observe **sendmail**'s interaction with the delivery agents by delivering the message in verbose mode, as in the following example:

```
/usr/sbin/sendmail -v  myname@cup.hp.com
```

sendmail responds with the following information:

```
myname@cup.hp.com... aliased to myname@mymachine.cup.hp.com
```

sendmail is now ready for you to type a message. After the message, type a period (.) on a line by itself, as in the following example:

```
This is only a test.
.
```

sendmail responds with the following information:

```
myname@cup.hp.com... Connecting to local host (local)...
myname@cup.hp.com... Executing "/bin/rmail -d myname"
myname@cup.hp.com... Sent
```

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 from the **sendmail** command line and expect message headers and message 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.

Contacting the sendmail Daemon to Verify Connectivity

It is possible to talk to the **sendmail** daemon and other SMTP servers directly with the following 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."

Once you establish a connection to the **sendmail** daemon, you can use the SMTP **VRFY** command 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
```

Not all SMTP servers support the **VRFY** and **EXPN** commands.

Setting Your Domain Name

If **sendmail** cannot resolve your domain name, you may see the following warning message in your **syslog** file:

```
WARNING: local host name name is not qualified; fix $j in
config file
```

This message can occur if you are using NIS and the first entry in the **/etc/hosts** file is not a fully-qualified host name. To resolve this problem, do one of the following:

- In the **/etc/mail/sendmail.cf** file, uncomment the following line by deleting the pound sign (#) at the beginning of the line:

```
Dj$w.Foo.COM
```

Change "Foo.COM" to the name of your domain (for example, "HP.COM").

- Modify the **/etc/hosts** file, making sure that the fully-qualified name of the system is listed first. For example, the entry in the file should be "**255.255.255.255 dog.cup.hp.com dog**" and not "**255.255.255.255 dog dog.cup.hp.com**."

Attempting to Start Multiple sendmail Daemons

If you attempt to start **sendmail** when there is already a **sendmail** daemon running, the following message may be logged to both syslog file and to the console:

```
NO QUEUE: SYSERR (root) opendaemonsocket: cannot bind:
Address already in use
```

This message means that a **sendmail** daemon is already running. You can use either **/sbin/init.d/sendmail stop** or **killsm** to stop the running daemon.

Configuring and Reading the sendmail Log

sendmail logs its mail messages through the **syslogd** logging facility.

The **syslogd** configuration should write mail logging to the file **/var/adm/syslog/mail.log**. You can do this by adding the following line in **/etc/syslog.conf**:

```
mail.debug  /var/adm/syslog/mail.log
```

You can use the HP **mtail** utility to look at a specified number of the last lines of the log file:

```
mtail 15
```

By default, **mtail** displays the last 20 lines of the log file. For more information on the **mtail** utility, type **man 1M mtail** at the HP-UX prompt.

For more information about configuring **syslogd**, see "Installing and Configuring Internet Services" on page 25.

Setting Log Levels

You can set the log level with the **-oL** option on the **sendmail** command line or on the **OL** line in the **sendmail** 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 11 and lower are considered useful. Log levels above 11 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 5.

Table 5 sendmail Logging Levels

0	No logging.
1	Major problems only.
2	Message collections and failed deliveries.
3	Successful deliveries.
4	Messages being queued (due to a host being down, and so on).
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.
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 only of interest when debugging.
16	Verbose information regarding the queue.

Understanding syslog Entries

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.

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 message succeeded, failed, or was queued), the mailer, and the host used are logged.

Queued messages and **SYSERR**s are also logged.

Storing Off Old sendmail Log Files

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. Following is an example of a **crontab** entry for a script called **newsyslog**:

```
0 0 * * * /var/adm/syslog/newsyslog
```

The following example shows what the script **/var/adm/syslog/newsyslog** might contain. The script assumes that **syslog** is configured to direct mail logging to **/var/adm/syslog/mail.log**.

```
#!/usr/bin/sh
#
#  NEWSYSLOG:  save only the last week's sendmail logging
#
cd /var/adm/syslog
mv mail.log.6  mail.log.7
mv mail.log.5  mail.log.6
mv mail.log.4  mail.log.5
mv mail.log.3  mail.log.4
mv mail.log.2  mail.log.3
mv mail.log.1  mail.log.2
cp mail.log    mail.log.1
kill -1 `cat /var/run/syslog.pid`
```

Printing and Reading the Mail Queue

The current contents of the mail queue can be printed with the following command:

```
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
                                       sys6!sysloc!njm
                                       vls@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.

Two other messages in the queue are also routed for delivery the next time the mail queue is processed.

If **`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 Files in the Mail Queue

The files that **`sendmail`** creates in the mail queue all have names of the form ***zz*T*AAnnnnn***, where ***zz*** is the type of the queue file and **T*AA*** is an identifier used to distinguish separate queue entries that happen to have the same process ID. **`sendmail`** starts with **`TAA`** and loops through **`TAB`**, **`TAC`**, and so on, 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.

A file whose name begins with **`df`** is a data file. The message body, excluding the header, is kept in this file.

A file whose name begins with **`qf`** is a queue-control file, which contains the information necessary to process the job.

A file whose name begins with **`xf`** is a 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.

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. Lines in queue-control files are described in Table 6.

Table 6 Lines in Queue-Control Files

Initial Letter	Content of Line
B	The message body type (either **7bit** or **8bitmime**).
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 that **sendmail** should run as when it is delivering a message into a file or a program's **stdin**.
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. 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 **mailq** command and is generally used to store status information (that is, the reason the message was queued). It can contain any text.
R	A recipient address. Normally this has already been completely aliased, but it 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.

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 job creation 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
```

6

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 client 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 manually from the shell prompt. Examples are provided to help you configure the servers. (You can also use SAM, the online configuration interface, to configure BOOTP and TFTP 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.

NOTE: As of Release 10.02, Dynamic Host Configuration Protocol (DHCP) is available for advanced IP address allocation and management of TCP/IP LAN computing environments. DHCP is a superset of BOOTP and can be used with the SAM graphical interface. See the DHCP chapter for more information.

Chapter Overview

The topics covered in this chapter include the following:

- How BOOTP Works
- Booting RMP Clients
- Configuring the TFTP Server
- Configuring the BOOTP Server
- Adding Client or Relay Information
- Command Options for Using TFTP
- Troubleshooting BOOTP and TFTP Servers

How BOOTP Works

The Bootstrap Protocol (BOOTP) allows a client system to discover its own IP address, the address of a bootpserver, 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`**.

Address Determination and Bootfile Selection

The first phase involves a **bootrequest** packet that is broadcast by the BOOTP client. A BOOTP server that receives the bootrequest can send a **bootreply** to the client if it finds the client's boot information in its database. Or, it can relay the bootrequest to other BOOTP servers if it finds relay information for the client in its database.

1 The BOOTP client formulates a bootrequest that it will broadcast. Before sending the bootrequest, the client does the following:

- It sets the **`hops`** field of the bootrequest packet to 0. Each time a BOOTP server relays the client's bootrequest, the **`hops`** field is incremented by 1. If the **`hops`** value exceeds the maximum hop value configured for this client on a BOOTP server, the bootrequest is dropped. The **`hops`** value limits the number of times a bootrequest can be relayed.
- It sets the **`secs`** field of the bootrequest packet to 0 for a first-time request. If the client does not receive a reply to this request, it sets the value of this field to the number of seconds since the first request was sent. If the value of the **`secs`** field is less than the threshold value configured for this client on a BOOTP server, the bootrequest is dropped. The threshold value ensures that enough time is allowed for a bootreply to be received by the client before a subsequent bootrequest for the same client is relayed.
- It sets the **`giaddr`** (gateway IP address) field to 0. If a BOOTP server finds that this field is 0, it fills it with its own IP address.

2 The client broadcasts the bootrequest packet on its first LAN interface (**`lan0`**). The bootrequest also contains the client's hardware address, and, if known, its IP address.

3 The BOOTP server checks to see if boot information for the client is in its database. If boot information for the client is available in the server's database, the server answers the bootrequest with a bootreply packet.

4 If the BOOTP server does not find boot information for the client in its database, it checks to see if there is relay information for the client. If there is no relay information for the client in the database, the bootrequest is dropped. If there is relay information available and the relay function is enabled for the client, the server checks the following:

- Does the **`hops`** value in the bootrequest packet exceed the maximum configured for the client? If it does, the request is dropped. If not, the **`hops`** field in the bootrequest packet is incremented.
- Is the **`secs`** value in the bootrequest packet less than the threshold configured on the server for the client? If it is, the request is dropped.

If the request has not been dropped during the above checks, the server then relays the bootrequest to the BOOTP server(s) that have been configured for the client. If the **`giaddr`** field of the bootrequest packet is 0, the server puts its IP address in the field.

Steps 3 and 4 are repeated until either the bootrequest is received by a BOOTP server that finds boot information about the client in its database, or the request is dropped.

When a server finds client information about a particular client in its database, the server answers the bootrequest with a bootreply packet. The client's IP address is placed into a field in the bootreply. The bootreply may also contain a file name of a boot file, which the client should load with TFTP. Other information that can be included in the bootreply are the client's subnet mask, the addresses of nameservers, and the addresses of gateways.

If the bootrequest has been relayed to one or more BOOTP servers, the bootreply is sent to the IP address in the **`giaddr`** field. This should be the IP address of the BOOTP server that initially relayed the bootrequest. That BOOTP server then sends the bootreply to the client.

Figure 10 shows an example of a bootrequest that is relayed from server A to server B to server C. Server C finds the client's boot information in its database, and sends the bootreply back to server A. Server A then sends the bootreply to the client.

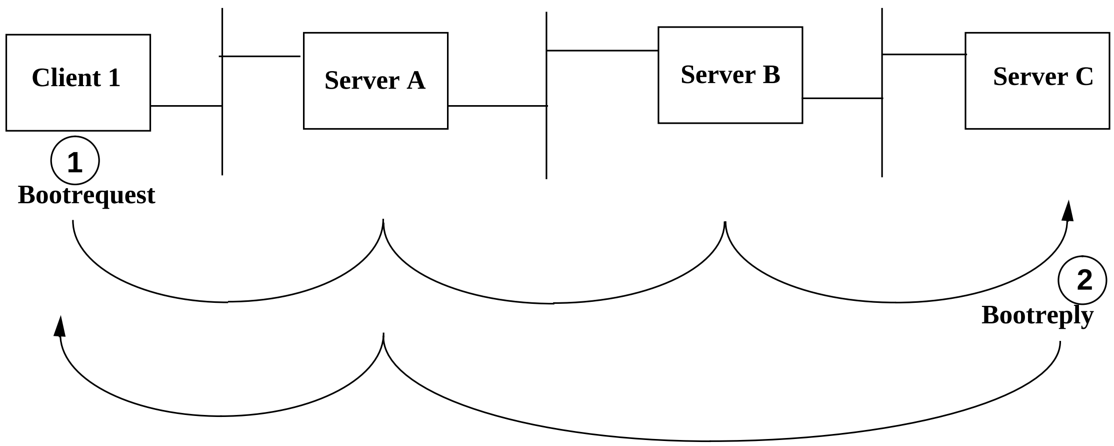


Figure 10 **Bootrequest Relay Example**

NOTE: BOOTP clients can be booted over a gateway; however, the BOOTP server with the relay information for the client must be on the same side of the gateway as the client.

File Transfer

The second phase, file transfer by the BOOTP client using TFTP, is optional. Some BOOTP clients use BOOTP only for IP address resolution and do not use TFTP. If the boot file is transferred, it must be publicly available.

Booting RMP Clients

Remote Maintenance Protocol (RMP) is an HP-proprietary boot and file transfer protocol used in early Series 700 workstations and in the Datacommunications and Terminal Controllers (DTC/9000). The **`rbootd`** daemon allows BOOTP servers to serve clients that use RMP. **`rbootd`** must be run on a BOOTP server on the same subnet as the RMP client. That is, both **`rbootd`** and **`bootpd`** must run on the same system.

The **`rbootd`** daemon translates RMP bootrequests into a BOOTP bootrequest using the client's hardware address. **`rbootd`** then forwards the bootrequest to **`bootpd`**. **`bootpd`** can send a bootreply back to **`rbootd`** if it finds the client's boot information in its database. Or, it can relay the bootrequest to other BOOTP servers if it has relay information for the client in its database. **`rbootd`** translates the BOOTP bootreply back to RMP and sends it to the client.

Figure 11 shows an example of an RMP bootrequest that is sent to **`rbootd`**, which then forwards a BOOTP bootrequest for the client to **`bootpd`**. **`bootpd`** finds the client's boot information in its database and sends a BOOTP bootreply back to **`rbootd`**. **`rbootd`** then sends an RMP bootreply to the client.

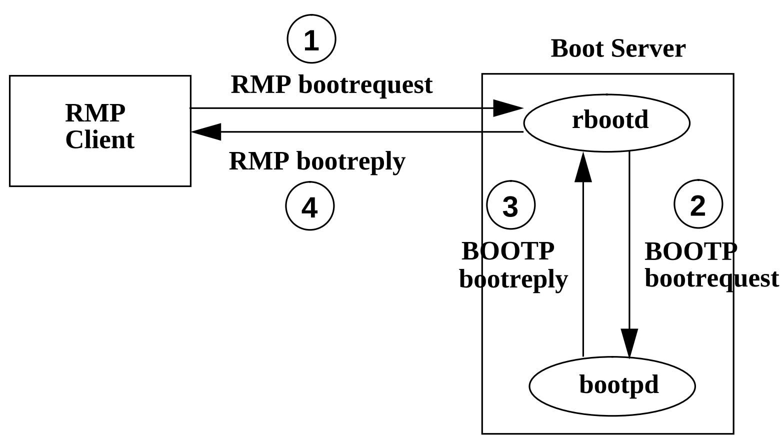


Figure 11 **BOOTP Server for RMP Client**

As mentioned previously, the BOOTP bootrequest can be relayed to other BOOTP servers. A BOOTP bootreply is sent back to the original **bootpd** daemon, which then sends the bootreply back to the **rbootd** daemon on its local system. **rbootd** uses either NFS or TFTP to transfer boot files from the remote server to its local system. (TFTP is the default file transfer method.) **rbootd** then transfers bootable images to the client in the form of RMP packets.

If TFTP is used to transfer boot files from a remote server, the boot files must be accessible via TFTP. For more information, see “Configuring the TFTP Server” on page 221. There must also be temporary file space available in **/var/rbootd/C0809*** on the **rbootd** server. Generally, at least 6 to 8 Mbytes of space should be allowed for each BOOTP client. The temporary files are removed automatically after a certain period of inactivity; by default, this time period is 10 minutes. You can specify a different time period by using the **-t** option when starting **rbootd**.

If NFS is used to transfer boot files from a remote server, use the NFS **mount** command to mount the path of the boot files on the **rbootd** server system. The path that is specified with the **mount** command must be defined with the **bf** tag for the client configuration in the **/etc/bootptab** file. (See “Adding Client or Relay Information” on page 227.) Note that a directory or file must be exported with the **exportfs** command before it can be NFS-mounted.

To start the **rbootd** daemon:

1. Set the environment variable **START_RBOOTD** to **1** in the file **/etc/rc.config.d/netdaemons**. This causes **rbootd** to start automatically whenever the system is booted.
2. Run the **rbootd** startup script with the following command:

```
/sbin/init.d/rbootd start
```

Configuring the TFTP Server

To manually configure the TFTP server, **tftpd**, you need to modify the **tftpd** entry in the **/etc/inetd.conf** file or create an entry for the user **tftp** in the **/etc/passwd** file. If you use SAM to configure your system as a BOOTP server, your system is automatically configured as a TFTP server. 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. For new **tftpd** installations, you can do this in one of two ways:

- Add the user **tftp** to **/etc/passwd**. For example,

```
tftp:*:510:10:TFTP:/home/tftpdir:/usr/bin/false
```

  HP *recommends* that you use this method. If there is no **/etc/passwd** entry for the user **tftp**, **tftpd** has root access to any files or directories you specify in the entry for **tftp** in the **/etc/inetd.conf** file. If an **/etc/passwd** entry exists for the user **tftp**, **tftpd** cannot read or write files unless they are readable or writable by the user **tftp**.

  If you create an **/etc/passwd** entry for the user **tftp**, **tftpd** first looks for a file relative to the home directory of the user **tftp**. If the file is not found there, then **tftpd** looks for the file relative to the path(s) specified with the **tftpd** command. If you want to give remote systems permission to retrieve a file through TFTP, the file must be readable by the user **tftp**. If you want to give remote systems permission to transmit a file to your system through TFTP, the file must be writeable by the user **tftp**. For example, to create a home directory for the user **tftp**, make the directory owner the user **tftp**, and ensure the directory gives the user **tftp** read, write and execute permissions. For example:

```
$ mkdir /home/tftpdir$
$ chown tftp /home/tftpdir
$ chgrp guest /home/tftpdir
$ chmod 700 /home/tftpdir
```

- Specify the files available to clients in the **tftpd** command line in **/etc/inetd.conf**:

```
tftpd dgram udp wait root /usr/lbin/tftpd tftpd [path...]
```

[*path*...] is a list of the files or directories that you want to make available to TFTP clients. File or directory names are separated by spaces. Each file or directory is assumed to be relative to **/**.

Reconfigure **/usr/sbin/inetd**:

```
/usr/sbin/inetd  -c
```

If you have both an **/etc/passwd** entry for the user **tftp** and files specified in the **tftpd** command line, **tftpd** first looks for a file relative to the user **tftp**'s home directory. If the file is not found, then **tftpd** looks for the file relative to the path specified in the **tftpd** command. If two files with the same name are in both locations, **tftpd** accesses the one under **tftp**'s home directory.

Verify Your tftpd Installation

To verify your **`tftpd`** installation, create a file and use the **`tftp`** program to perform a file transfer:

1 Create a file that is readable by the user **`tftp`**. The file should be in the user **`tftp`**'s home directory or in a directory specified with the **`tftpd`** command. For example,

```
$ echo "Hello, this is a test." > /export/testfile
$ chown tftp /export/testfile
$ chmod 400 /export/testfile
```

Make sure that an **`/etc/passwd`** entry exists for the user **`tftp`**.

2 Using a TFTP client, try to retrieve the file:

```
$ tftp localhost
tftp> get /export/testfile
Received 24 bytes in 0.6 seconds
tftp> quit
```

You can specify either the IP address or name of the remote host. In order to get a file from a directory specified as an argument to the **`tftpd`** command, you must specify the full path name. If this step fails, see "Troubleshooting BOOTP and TFTP Servers" on page 238.

3 Compare the ASCII files to verify data transfer:

```
$ diff testfile /export/testfile
$
```

4 Remove the test file once you have verified the installation.

Configuring the BOOTP Server

To manually configure the BOOTP server daemon, **`bootpd`**, you need to add entries to the files **`/etc/services`** and **`/etc/inetd.conf`**. When you use SAM to do the configuration, 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 Make sure that the BOOTP server and client protocols are added to **`/etc/services`**:

```
bootps  67/udp # Bootstrap protocol server
bootpc  68/udp # Bootstrap protocol client
```

2 Uncomment the following entry to **`/etc/inetd.conf`**:

```
bootps dgram udp wait root /usr/lbin/bootpd bootpd
```

3 Reconfigure **`/usr/sbin/inetd`**:

```
/usr/sbin/inetd  -c
```

You are now ready to add client or relay information to the configuration file **`/etc/bootptab`**. This step is discussed in the section “Adding Client or Relay Information” on page 227. If you wish to verify your **`bootpd`** installation, continue to the next section.

NOTE: SAM does not add relay information to the configuration file. You must manually configure relay information on a BOOTP server.

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. (Type **man 1M bootpquery** for more information.) For example, if you configured **bootpd** on a system named **myhost**, enter:

```
/usr/sbin/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 following 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**, **bootpd** logs informative messages to the daemon facility. (Type **man 1M syslogd** for more information.) In the default configuration, where **syslogd** sends daemon information messages to **/var/adm/syslog/syslog.log**, 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 configuration file **/etc/bootptab** and that it correctly rejected the test bootrequest that you sent with **bootpquery**.

Having verified that **bootpd** is configured to start from **inetd**, you should add to the configuration file any BOOTP clients that the system is to serve, or any BOOTP clients that are to be relayed to another server. The next section, “Adding Client or Relay Information” on page 227, describes how to add client information or client relay information and how to verify that the BOOTP server will respond to the client.

Adding Client or Relay Information

To allow a client to boot from your local system or to allow a bootrequest to be relayed to the appropriate boot server, you must add information about the client in your **/etc/bootptab** file. **bootpd** uses the **/etc/bootptab** file as the database for two types of entries:

- Client entries that contain information that allows the clients to boot from your system.
- Relay entries that contain information to relay the bootrequest to one or more BOOTP servers.

Collecting Client Information

To make an entry for the client in the **/etc/bootptab** file, you need to collect the following information about the client:

- Host name—the name of the client's system.
- Hardware type—the type of network interface.
- Link level address—the client's hardware address.
- IP address—the client's assigned internet address.
- Subnet mask—the mask (IP address) that identifies the network where the client resides.
- Gateway address—the gateway from the client’s local subnet.
- Boot file—the name of the file that the client will retrieve using **tftp**.

Collecting Relay Information

To make a relay entry for the client in the **`/etc/bootptab`** file, you need to collect the following information about the client:

- Host name—the name of the client's system.
- Hardware type—the type of network interface (IEEE 802.3 or Ethernet).
- Link level address—the client's hardware address.
- Subnet mask—the mask that is used to identify the network address where the client resides.
- Gateway address—the address of the gateway that connects the client's local subnet to the BOOTP server's subnet.
- Boot server(s) for client—the boot servers to which the local system will relay the client's bootrequest.
- Threshold value—the number of seconds since the client sent its first request.
- Maximum hops—the maximum number of hops that the client's bootrequest can be forwarded.

Understanding Boot File Configurations

A configuration entry is a single line with the following format:

```
hostname:tag=value:tag=value:...tag=value
```

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 colon separates each ***`tag=value`*** parameter definition. **`bootpd`** uses these tags and values to recognize a client's bootrequest, supply parameters in the bootreply to the client, or relay the bootrequest.

For example, parameters for the BOOTP client **`xterm01`** are represented with the following entry in **`/etc/bootptab`**:

```
xterm01:  ht=ether:  ha=080009030166:  ip=15.19.8.2:\
sm=255.255.248.0:  gw=15.19.8.1:  bf=/xterm01
```

This entry tells **bootpd** the following information about **xterm01**:

- Hardware type is an Ethernet network interface.
- Hardware address is 080009030166.
- IP address is 15.19.8.2.
- Subnet mask is 255.255.248.0.
- The address of the gateway is 15.19.8.1.
- The file **/xterm01** should be retrieved with TFTP.

You may enter tags in any order, with the following exceptions:

- The client's hostname must be the first field of an entry.
- The **ht** (hardware type) tag, if specified, must precede the **ha** (hardware address) and **hm** (hardware mask) tags.
- If the **gw** (gateway IP address) tag is specified, the **sm** (subnet mask) tag must also be specified.

Other points to know when adding an entry in **/etc/bootptab** include the following:

- 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 sharp sign (#) are ignored.

Parameter Tags and Descriptions

Table 7 lists the tags most commonly used to define the client parameters. For more information on these and the other tags available, type **`man 1M bootpd`**.

Table 7 Tags for Defining Client Options in bootptab

`ba`	Forces **`bootpd`** to broadcast the bootreply to the client's network. This tag should be used only when troubleshooting with the **`bootpquery`** program.
`bf`	Boot file name that the client downloads with TFTP.
`bs`	Boot file size in 512-byte blocks. If this tag is specified with no equal sign or value, the server automatically calculates the boot file size at each request.
`ds`	IP address(es) of the BIND name server(s).
`gw`	IP address(es) of the gateway(s) for the client's subnet.
`ha`	Client's hardware address.
`hd`	Directory to which the boot file is appended (see **`bf`** tag). The directory specified must end with /. The default is /.
`hn`	Send the host name in the bootreply. This tag is strictly Boolean; it does not need an equals sign or an assigned value.
`ht`	Client's hardware type. May be assigned the value **`ieee`** or **`ether`**. If used, this tag must precede the **`ha`** tag.
`ip`	BOOTP Client's IP address. This tag takes only one IP address. This tag distinguishes a boot entry from a relay entry.
`sm`	The subnet mask for the client's network.
`tc`	Specifies previously-listed entry that contains tag values that are shared by several client entries.
`vm`	The format of the vendor extensions on the bootrequest and bootreply. Possible values are **`auto`** (the bootreply uses the format used in the bootrequest), **`rfc1048`** (the most commonly used format, described in RFC 1048), and **`cmu`** (another format used by some BOOTP clients). If you do not specify the **`vm`** tag, the bootreply will use the format sent by the client in the bootrequest.

Table 8 lists the tags most commonly used to define the relay parameters. For more information on these and the other tags available, type **man 1M bootpd**.

Table 8 **Tags for Defining Relay Options in bootptab**

bp	List of boot servers to which the client’s bootrequests will be forwarded. The list can contain individual IP addresses, hostnames, or network broadcast addresses.
ha	Client's hardware address.
hm	Mask for the link level address. This value is ANDed with the **ha** value to determine a match for a group relay entry. If this tag is specified, the **ha** and **ht** tags must also be specified.
hp	Maximum number of hops for the entry. Default is 4.
ht	Client's hardware type. See the **bootp** man page for supported hardware types and the corresponding values. If used, this tag must precede the **ha** tag.
tc	Specifies previously-listed entry that contains tag values that are shared by several client entries.

A relay entry can contain relay parameters for an individual system or for a group of systems. If a BOOTP client does not have an individual entry in the BOOTP server’s **/etc/bootptab** file, the group relay entries are searched. The first group relay entry that matches the BOOTP client is used.

Examples of Adding BOOTP Clients

This section shows examples of adding entries to the **`/etc/bootptab`** file. The first example shows how to configure a BOOTP server for an HP 700/X terminal. The second example shows how to configure a BOOTP server to relay a client’s bootrequest to another server.

Example 1: Adding an HP 700/X Terminal as a Client

Figure 12 shows the network configuration for this example.

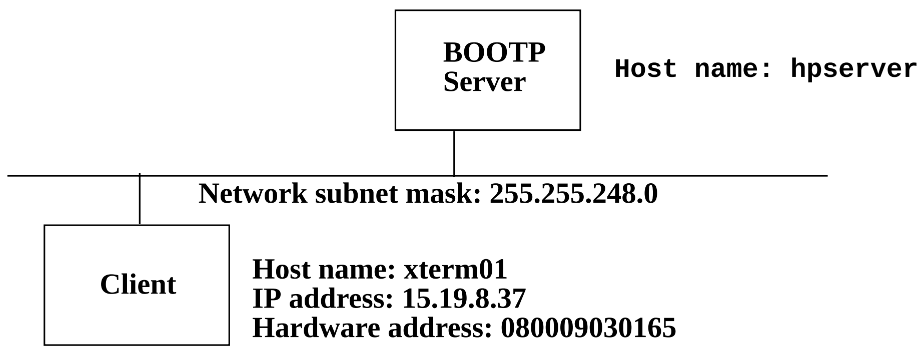


Figure 12 **Example Configuration: HP 700/X Terminal as Client**

The following information is added to the **`/etc/bootptab`** file on the BOOTP server (**`hpserver`**):

```
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
```

To verify the new **/etc/bootptab** entry, do the following on the BOOTP server:

1. Add the **ba** (broadcast address) tag to the entry so that the bootreply is not sent directly to **xterm01**. This allows the **bootpquery** diagnostic tool to intercept any bootreply packets for **xterm01**.

```
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
```

2. Run the **bootpquery** tool to see how **bootpd** on your local system responds to a request from **xterm01**. For the example configuration, the following would be entered (as superuser):

```
/usr/sbin/bootpquery 080009030165 -s hpserver
```

The following output is displayed:

```
Received BOOTREPLY from hpserver.hp.com (15.19.8.119)

    Hardware Address:     08:00:09:03:01:65
    Hardware Type:        ethernet
    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:            term01.hp.com
```

This shows that the BOOTP server responded with information that corresponds to the entry in the **/etc/bootptab** file.

3. Remove the **ba** tag entry from the **/etc/bootptab** file.

Example 2: Adding a Relay Entry

Figure 13 shows the network configuration for this example. In this example, the network contains HP workstations and other vendors' systems. Server B is the BOOTP server that contains boot information for the HP workstations. When server A receives a bootrequest, it relays requests from HP workstations to server B. Bootrequests for other vendors' systems are relayed to server C. In this example, Server A (the BOOTP relay agent) is also the gateway between the client's network and the server's network.

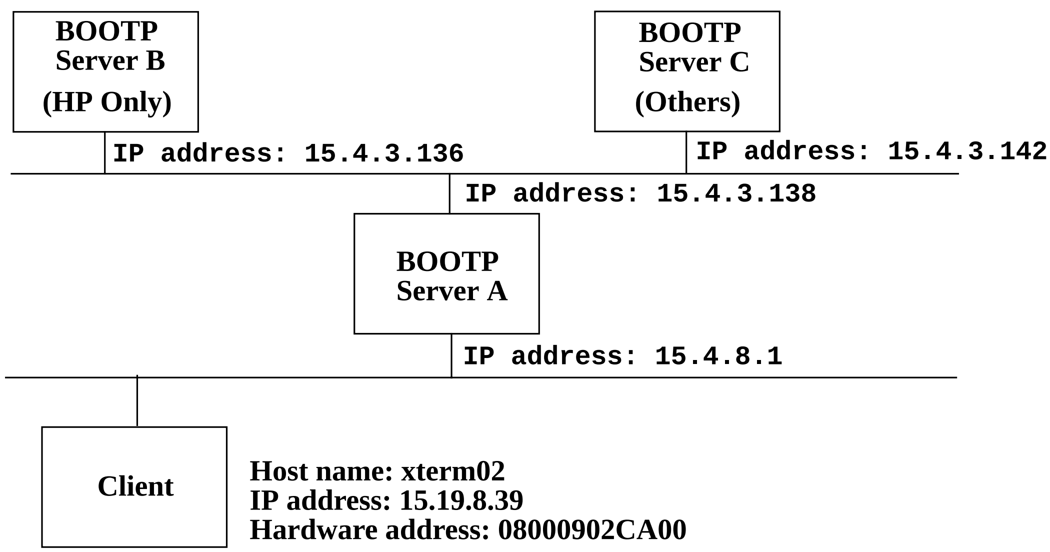


Figure 13 **Example Configuration: Relay Entry**

The following information is added to the **/etc/bootptab** file on BOOTP server A:

```
defaults: ht=ether
all_hp:\
   tc=defaults:\
   ha=080009000000:\
   hm=FFFFFF000000:\
   bp=15.4.3.136
others:\
   tc=defaults:\
   ha=000000000000:\
   hm=000000000000:\
   bp=15.4.3.142
```

The **all_hp** entry causes bootrequests from HP workstations (machines with hardware addresses that begin with 080009) to be relayed to IP address 15.4.3.136 (server B). Bootrequests from other hardware addresses (presumed to be non-HP machines) are relayed to IP address 15.4.3.142 (server C).

The following information is added to the **/etc/bootptab** file on BOOTP server B:

```
xterm02:  hn: ht=ether:  ha=08000902CA00: \
    ip=15.19.8.39:  sm=255.255.248.0: \
    gw=15.19.8.1:  ds=15.19.8.119:  bf=/xterminal:
```

The gateway address (**gw=15.19.8.1**) is passed back to the client in the bootreply and allows the client to send a TFTP request to the BOOTP server to get its boot file.

To verify the new **/etc/bootptab** entry, do the following:

1. Add the **ba** (broadcast address) tag to the **xterm02** entry on the BOOTP server that contains the client's boot entry (server B) so that the bootreply is not sent directly to **xterm02**. This allows the **bootpquery** diagnostic tool to intercept any bootreply packets for **xterm02**.

```
xterm02:  ht=ether:  ha=08000902CA00: \
ip=15.19.8.39:  sm=255.255.248.0:\
gw=15.19.8.1: ds=15.19.8.119:  bf=/xterminal ba
```

2. If you can boot the client in standalone mode, run the **bootpquery** tool on the client to see how **bootpd** on the server responds to a request from **xterm02**. For the example configuration, the following would be entered (as superuser):

```
/usr/sbin/bootpquery 08000902CA00
```

   You can also run **bootpquery** from another machine that is up and running on the same subnet as the client.

   Output like the following is displayed:

```
Received BOOTREPLY from hpserver.hp.com (15.4.3.136)

    Hardware Address:     08:00:09:02:CA:00
    Hardware Type:        ethernet
    IP Address:           15.19.8.39
    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:            xterm02.hp.com
```

   This shows that the BOOTP server responded with information that corresponds to the client entry in the **/etc/bootptab** file. You can also conclude that the bootrequest was correctly relayed to the BOOTP server that contains the client's boot information.

3. Remove the **ba** tag entry from the **/etc/bootptab** file.

Command Options for Using TFTP

Internet Services includes a TFTP client implementation, **/usr/bin/tftp**. You can use this client to verify that your TFTP server is working correctly. For example, to retrieve the file **bootf** from the TFTP server **duncan**, enter the following:

```
/usr/bin/tftp duncan
```

At the **tftp** prompt, enter:

```
get bootf
```

Table 9 describes the most common **tftp** commands you can use when transferring files. For information on the other **tftp** options, type **man 1 tftp**.

Table 9 **tftp File Transfer Options**

ascii	Sets the TFTP file transfer type to ASCII. This is the default type.
binary	Sets the TFTP file transfer type to binary.
get remote_file [*local_file*]	Copy ***remote_file*** to ***local_file***. If ***local_file*** is unspecified, **tftpd** uses the specified ***remote_file*** name as the ***local_file*** name. If ***local_file*** is specified as "**-**", the remote file is copied to standard output.
put local_file [*remote_file*]	Copy ***local_file*** to ***remote_file***. If ***remote_file*** is unspecified, **tftpd** assigns the ***local_file*** name to the ***remote_file*** 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.

Helpful Configuration Changes

To make troubleshooting easier, configure your system as follows:

- Ensure **syslogd** is configured to log daemon information messages to the file **/var/adm/syslog/syslog.log**. To check this configuration, make sure **/etc/syslog.conf** includes one of the following lines:

  ```
  *.info /var/adm/syslog/syslog.log
  ```

  or

  ```
  daemon.info /var/adm/syslog/syslog.log
  ```

- 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. Follow these steps to set the debug log level:

  1. Add the **-d 2** option to the **bootpd** line in **/etc/inetd.conf**:

```
bootps dgram udp wait root /usr/lbin/bootpd bootpd -d 2
```

  2. Reconfigure **inetd** with the following command:

     ```
     /usr/sbin/inetd -c
     ```

  3. Kill any **bootpd** daemon that is still running on your system. For example,

     ```
     $ /usr/bin/ps -e | /usr/bin/grep bootpd
     429 ? 0:00 bootpd
     $ /usr/bin/kill 429
     ```

Common bootpd Problems

If you experience a problem with **`bootpd`**, read through this section for possible remedies. The problems listed in this section are ordered by symptom.

To view the information that **`bootpd`** places in the bootreply, enable a broadcast bootreply by adding the **`ba`** tap to the client’s **`/etc/bootptab`** entry. Use the **`bootpquery`** command to emulate the client’s bootrequest:

```
bootpquery client_link_address -s servername
```

`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.

Symptom: The server’s system log file **`/var/adm/syslog/syslog.log`** does not contain any log messages from **`/usr/lbin/bootpd`** showing that the server started. A **`ps -ef`** listing does not show a running **`/usr/lbin/bootpd`**.

Cause: The server may not be started or it may not be receiving the client’s bootrequest.

Action:

- ❒ Make sure 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 **`/var/adm/syslog/syslog.log`** to ensure **`inetd`** is configured to start **`bootpd`**.
- ❒ 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 BOOTP servers are configured to relay bootrequest broadcasts.

Symptom: The system log **/var/adm/syslog/syslog.log** contains one of the following messages:

```
hardware address not found: hardware_address
IP address not found: ip_address
```

Cause: **bootpd** does not have an entry in **/etc/bootptab** for this client's hardware address or IP address.

Action:

- ❒ 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 **/usr/lbin/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.

Symptom: The system log **/var/adm/syslog/syslog.log** contains a message that looks like this:

```
requested file not found: filename
```

Cause: The client specified ***filename*** as the boot file in its bootrequest, but **bootpd** could not find the file in the **tftp** directory.

Action:

- ❒ Make sure 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 **/home/tftpdir** or in the directory specified with the **tftpd** command. If it is not, place the file in the directory and reboot the BOOTP client. If the requested file exists in the directory, be sure it is readable by the user **tftp**. (See “Common tftpd Problems” on page 243.)

Symptom: The system log **/var/adm/syslog/syslog.log** contains the following message:

```
cannot route reply to client's_IP_address
```

Cause: The IP address you have specified for the client is one which the server's system cannot reach directly.

Action:

- ❐ 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. If the client resides on another network, ensure that intervening servers are configured to relay the bootrequests.
- ❐ Ensure the IP address you have chosen for the client is a valid IP address for the server's network.

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
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
```

Cause: Any of these error messages means there are errors in the configuration file entry for the client.

Action: See “Error Logging” on page 246 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 **/usr/lbin/bootpd** for the client whose entry you have corrected. Check the system log **/var/adm/syslog/syslog.log** to see if the server replies. At debug level 2 (see “Helpful Configuration Changes” on page 238),

bootpd logs the following sequence of messages when it responds to a bootrequest:

```
request from hardware address link_address
found ip_address hostname
vendor magic field is magic_cookie
sending RFC1048-style reply
```

Symptom: The client does not receive configuration information for the tags that pertain to RFC 1048 vendor information:

```
bs = boot_file_size
ds = domain_nameserver_addresses
gw = gateway_addresses
hn = hostname
lg = log_server_addresses
sm = subnet_mask
to = time_offset
Tnnn = generic_information
```

Cause: Too many RFC-1048 options have been specified for the client's configuration entry in **/etc/bootptab**. The BOOTP protocol allows only 64 bytes of "vendor extension" 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-byte 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.

Action: 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.

Common tftpd Problems

If you experience a problem with **`tftpd`**, read through this section for possible remedies. The problems listed in this section are ordered by symptom.

Symptom: File transfer “timed out.” **`inetd`** connection logging (enabled with the **`inetd -l`** command) does not show any connection to the TFTP server.

Cause: The TFTP server, **`tftpd`**, did not start.

Action:

- ❒ 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” on page 224, verify that 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. Refer to *Installing and Administering LAN/9000 Software* or the BOOTP client manual (for example, HP 700/X documentation).

Symptom: File transfer "timed out." The system log contains one of the following messages:

```
User tftp unknown
system_call: error
```

Cause: The TFTP server, **tftpd**, exited prematurely.

Action: If you suspect that there is a problem on the network, you can increase the per-packet retransmission and the total retransmission timeouts used by **tftpd**. These timeouts are specified (in seconds) with the **-R** or **-T** options. See the **tftpd** man page for more information.

The **User tftp unknown** message can also mean that the password database entry for the user **tftp** is either missing or incorrect. Verify that 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*.

Symptom: File transfer fails with **File Not Found**, **No Such File or Directory**, or **TFTP Error Code 1** message.

Cause: The file the client is attempting to read from or write to the server does not exist within the home directory of the user **tftp** or in the path specified with the **tftpd** command.

Action: Ensure the full path name that the client is requesting from the server exists within the **tftp** directory or in a path specified with the **tftpd** command. For example, if the **tftp** directory is **/home/tftpdir** and the TFTP client is requesting the file **/usr/lib/X11/700X/C2300A**, the file must exist as **/home/tftpdir/usr/lib/X11/700X/C2300A**.

If no entry exists for the user **tftp** in the **/etc/passwd** file, you must specify at least one file or directory with the **tftpd** command. Make sure that you specify the full path name when attempting to get a file from a directory specified with the **tftpd** command.

Symptom: File transfer fails with **Access Violation**, **Permission Denied**, or **TFTP Error Code 2** message.

Cause: **tftpd** does not have permission to read the file.

Action: 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 that the file the client is reading has read permissions for the user **tftp**. For example, if the client was attempting to read the file named **xterm**, **xterm** should be mode 0400 and owned by the user **tftp**:

```
$ ll /home/tftpdir/xterm
-r-------- 1 tftp guest 438 May 10 1989 xterm
```

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 be mode 0600 and owned by **tftp**:

```
$ ll /home/tftpdir/fonts
-rw------- 1 tftp guest 0 May 10 1989 fonts
```

Error Logging

This section explains the error messages that **bootpd** logs through **syslogd**. The three levels of error logging documented in this section are as follows:

- Information Log Level
- Notice Log Level
- Error Log Level

The **bootpd** debug level must be set for 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.

- **exiting after *time* minutes of inactivity**

  If **bootpd** hasn't received a bootrequest within ***time*** minutes (the timeout set with the **-t** option), it issues this message and exits.

- **reading *configuration_file***

  reading new *configuration_file*

  bootpd is reading or rereading configuration information from the indicated ***configuration_file***.

- **read *number* entries from *configuration_file***

  Shows that **bootpd** successfully read ***number*** configuration entries, including table continuation entries, from the indicated ***configuration_file***.

- **request from hardware address hardware_address**

  bootpd received a bootrequest from a client with the indicated ***hardware_address***. This message is logged at debug level 1.

- **request from IP addr *ip_address***

  bootpd received a bootrequest from a client with the indicated ***ip_address***. This message is logged at debug level 1.

- **`found`** ***`ip_address hostname`***

  `bootpd` located information for the specified client in its configuration database. This message is logged at debug level 1.

- **`broadcasting reply on`** ***`ip_address`***

  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.

- **`vendor magic field is`** ***`magic_cookie`***

  `sending CMU-style reply`

  `sending RFC1048-style reply`

  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.

- **`bootptab mtime is`** ***`time`***

  `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.

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.

- **hardware address not found: *hardware_address***

  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.

- **IP address not found: *ip_address***

  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.

- **requested file not found: *filename***

  The client requested the boot file ***filename***, 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.

- **cannot route reply to *ip_address***

  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.

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.

- **bad bootp server address for host *hostname***

  A value specified for the **bp** tag is invalid. Values can be individual IP addresses separated by a space, and/or one or more network broadcast addresses.

- **bad hardware mask value for host *hostname***

  The value for the hardware address mask tag **hm** was incorrectly formatted in the configuration file entry for ***hostname***. Correct the configuration entry and try to reboot the BOOTP client. The subnet mask must be specified in hex.

- **bad hardware type for host *hostname***

  The value specified for the **ht** tag is an unsupported hardware type. See the **bootpd** man page for a list of supported hardware types.

- **bad hostname: *hostname***

  The name given in the **hostname** field was not a valid host name. Correct the host name and try to reboot the BOOTP client. A valid host name consists a letter followed by any number of letters, digits, periods, or hyphens.

- **bad IP address for host *hostname***

  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.

  Correct the configuration entry and try to reboot the BOOTP client. IP addresses must be specified in standard Internet “dot” notation. They can use decimal, octal, or hexadecimal numbers. (Octal numbers begin with 0, and hexadecimal numbers begin with 0x or 0X.) If more than one IP address is listed, separate addresses with white space.

- **bad reply broadcast address for host hostname**

  The address given for the **ba** tag was invalid or incorrectly formatted. Correct the configuration entry and try to reboot the BOOTP client. Type **man 1M bootpd** for more information.

- **`bad subnet mask for host`** ***`hostname`***

  The value for the subnet mask tag **`sm`** was incorrectly formatted in the configuration file entry for ***`hostname`***. Correct the configuration entry and try to reboot the BOOTP client. The subnet mask must be specified as a single IP address.

- **`bad time offset for host`** ***`hostname`***

  The value for the **`to=`** tag was not a valid number. Correct the configuration entry and try to reboot the BOOTP client. The **`to=`** value may be either a signed decimal integer or the keyword **`auto`**, which uses the server's timezone offset.

- **`bad vendor magic cookie for host`** ***`hostname`***

  The vendor magic cookie, specified with the **`vm`** tag, was incorrectly formatted. Correct the configuration entry and try to reboot the BOOTP client. The **`vm`** tag can be one of the following values: **`auto`**, **`rfc1048`**, or **`cmu`**.

- **`can't find tc=`*****`label`***

  `bootpd` could not find a table continuation configuration entry with the host field ***`label`***. Correct the configuration entry and try to reboot the BOOTP client. Type **`man 1M bootpd`** for more information.

- **`duplicate hardware address:`** ***`hardware_address`***

  More than one configuration entry was specified for the client with the indicated ***`hardware_address`***. Ensure that only one configuration entry exists for the hardware address in **`/etc/bootptab`**. Then, try to reboot the BOOTP client.

- **`missing ha values for host`** ***`hostname`***

  The hardware address must be specified in hex and must be preceded by the **`ht`** tag. If the **`hm`** tag is specified, the **`ha`** and **`ht`** tags must also be specified.

- **`syntax error in entry for host`** ***`hostname`***

  The configuration entry for the indicated host ***`hostname`*** is incorrectly formatted. Correct the configuration entry and try to reboot the BOOTP client. Type **`man 1M bootpd`** for the correct syntax of the BOOTP configuration file.

- **`unknown symbol in entry for host`** ***`hostname`***

  The configuration entry contains an unknown tag or invalid character. Correct the configuration entry and try to reboot the BOOTP client. Type **`man 1M bootpd`** for the correct syntax of the BOOTP configuration file.

7

DHCP

DHCP (Dynamic Host Configuration Protocol) provides advanced IP address allocation and management for TCP/IP LAN computing environments. By automating IP address allocation, the server provides a level of automation not provided with the BOOTP client-server bootstrap protocol.

DHCP also supports more configuration options than BOOTP. DHCP clients can include TCP/IP network printers, X terminals and Microsoft Windows machines. In addition to supporting new DHCP clients, DHCP supports new and existing BOOTP clients.

NOTE: DHCP allows clients to run on stations that boot from their own disks. A DHCP client will automatically be assigned an IP address from a DHCP server. For detailed information on DHCP clients, see the latest edition of the *Installing HP-UX* manual.

Configuration Overview

The DHCP server is configured and administered through SAM (or by editing the files /etc/bootptab and /etc/dhcptab, described later). DHCP consists of three branches of configuration, each of which involves a different method of clients receiving booting information:

- DHCP Device Groups
- Fixed-Address Devices
- Devices Booting From Remote Servers

DHCP Device Groups

DHCP allows you to configure groups of devices, specifying a unique IP address range for each group configured. Each device in a specific group is automatically assigned an available IP address from its group upon requesting booting information.

By creating various groups of devices you can compose each group with a device type specific to that group. For example, you may want one group to contain nothing except printers, and you may want another group to contain a certain type of terminal.

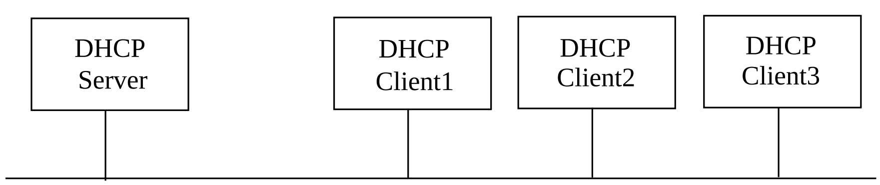


Figure 14 Devices Can be Configured as Part of a DHCP Group

In the drawing, assume that a particular group has been configured so that Client1, Client2 and Client3 all belong to this group. This means that each device in this group will have the same group name and will be given an IP address that is within the group’s IP address range. The IP addresses within the group's range make up what is known as a pool of addresses. When Client1, Client2 or Client3 perform a boot request, they will automatically be assigned an IP address not already in use from this pool.

DHCP allows you to exclude certain addresses within a group if you wish to make them unavailable for assignment. You also have the capability to define many values for the devices of a group including address lease times, DNS servers, NIS servers and many other optional parameters.

For detailed configuration information, refer to the on-line help that is provided with the DHCP graphical user interface.

Fixed-Address Devices

In addition to having addresses assigned from groups, DHCP allows IP addresses to be individually configured for devices. For administrative or security reasons, you may want certain devices to have fixed addresses.

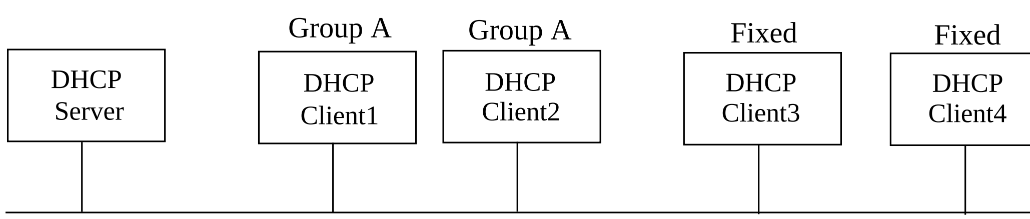


Figure 15 **DHCP Devices Can Have Fixed IP Addresses**

Through SAM, you must configure each fixed-address device with information about the device, including its own IP address. In the drawing, assume that you have configured a DHCP group to include Client1 and Client2, meaning that each will receive an IP address from a pool of available addresses at boot request. However, suppose that you have configured Client3 and Client4 to have fixed IP addresses. Client3 and Client4, therefore, will be assigned the addresses you configured for them upon boot request. Client3 and Client4 will always be assigned these same addresses unless you change the configuration.

DHCP also allows you to define many optional parameter values for clients with fixed addresses.

For more detailed configuration information, refer to the on-line help that is provided with the DHCP graphical user interface.

Devices Booting From Remote Servers

The third method of DHCP clients receiving IP addresses is through the use of what is called a BOOTP Relay Agent. A BOOTP Relay Agent is a machine on the local network which forwards boot requests from a DHCP or BOOTP client to a configured DHCP or BOOTP server.

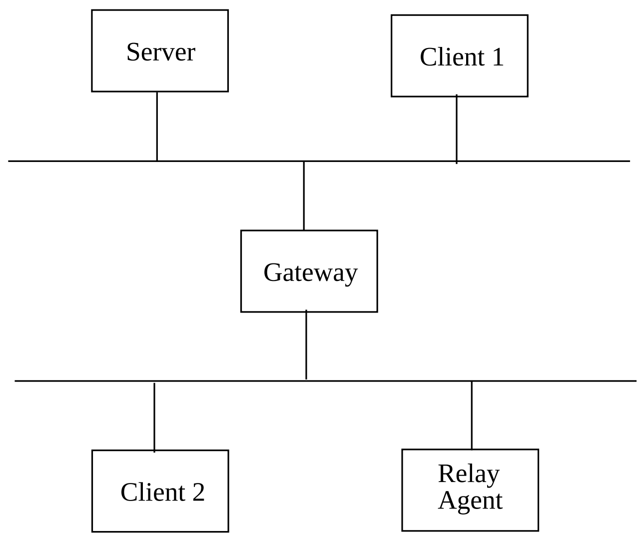


Figure 16 **Relay Agent Scenario**

In the drawing, suppose that Client2 broadcasts a boot request. The server containing the booting information belongs to a remote network. Therefore, the broadcast message is received by the local machine known as the relay agent. The relay agent sends the message across the gateway to the remote server, which in turns sends the boot information for Client2 back to the relay agent. The relay agent then broadcasts a message which is received by Client2. The message contains booting information for Client2.

As for the gateway, the gateway could be configured to also serve as a relay agent if the gateway is “DHCP-smart.” However, if the gateway does not have knowledge of DHCP, then a relay agent must be used, as shown in the drawing.

Client1 in the drawing does not need to use a relay agent because Client1 is on the same network as the server.

For more detailed configuration information, refer to the on-line help that is provided with the DHCP graphical user interface.

Configuration Files

Two configuration files, bootptab and dhcptab, are used for your DHCP configuration. These files are written to when you perform configuration through SAM. You can also manually edit these files if desired, although most of your work will probably be performed using SAM.

The bootptab file contains configuration information for old BOOTP clients as well as DHCP clients with fixed IP addresses. The bootptab file also contains configuration for relay agents.

The dhcptab file contains configuration information for DHCP groups, where clients are assigned IP addresses from a pool of currently unused addresses.

Options

Two options within the bootptab and dhcptab files you may want to be aware of are the t and v options.

The t option allows you to set specific values for optional DHCP parameters. Most optional parameters are configurable through SAM and are shown and described (through on-line help) on the corresponding SAM screen, but some additional optional parameters can be specified only by editing the option of the appropriate configuration file.

The v option is for DHCP clients and is used to configure vendor-specific options.

Migration

Because of the easy IP address allocation made possible by DHCP, you may want to convert old BOOTP clients to become DHCP clients. From the server perspective, this can be accomplished by using the allow-bootp-clients option of the bootpd(1M) command. You can refer to this manpage for detailed information. In short, you can make an old BOOTP client part of a DHCP group that has been defined. Bootpd is the internet boot protocol server daemon that implements DHCP, BOOTP and DHCP/BOOTP relay agents.

DHCP is backwards compatible with BOOTP, so no changes are required of existing users of BOOTP.

Tools

The command-line tool known as dhcptools(1M) is available to provide access to DHCP-related options for the bootpd server. The options provide control for dumping internal data structures, generating a host's file, previewing client address assignment, reclaiming unused addresses, tracing packets, and validating configuration files.

Refer to the dhcptools(1M) manpage for detailed information about the various options. The -v option should be used after you have completed configuration to verify that no detectable errors exist in either the bootptab or dhcptab configuration files.

If you are experiencing trouble with communication between the server and client at a protocol level, and you have verified that no errors exist in the configuration files, you may want to use the -t option of the dhcptools command. This option performs packet tracing. You may want to use this option in conjunction with the -d option of the bootpd(1M) command. Refer to the bootpd(1M) manpage for details. Bootpd is the internet boot protocol server daemon that implements DHCP, BOOTP and DHCP/BOOTP relay agents.

Getting Started Information

Before startup, you must set the broadcast address for the LAN0 interface name to 255.255.255.255. You can do this either manually or through SAM.

To manually adjust the address, do the following commands:

1. ifconfig lan0 broadcast 255.255.255.255

2. /etc/rc.config.d/netconf

Then edit the BROADCAST_ADDRESS variable for lan0 to 255.255.255.255.

To change the address through SAM, do the following:

1. Choose the Networking and Communications area.

2. Choose the Network Interface Cards area.

3. Go to Advanced Options and set the broadcast address to 255.255.255.255.

If there is more than one LAN interface, each must have a broadcast address of 255.255.255.255.

8

Configuring NTP

This chapter contains information about how to configure and use **xntpd**. **xntpd** is a daemon that maintains the local clock on an HP-UX workstation in agreement with Internet-standard time servers. **xntpd** is an implementation of the Network Time Protocol (NTP) version 3 standard, as defined in RFC 1305.

This chapter includes the following sections:

- Overview
- Configuration
- Starting and Stopping **xntpd**
- Modifying and Querying **xntpd**

You can use SAM to configure **xntpd**.

NOTE: **xntpd** is an HP implementation of version 3.2 of a publicly-available NTP daemon. HP provides support for the features documented in this chapter and in the **xnptd** man page. Other features of the publicly-available daemon may work, however they are not supported by HP.

Overview

Many internetwork services and applications depend upon the system clocks of the networked systems being synchronized with each other. For example, network management systems need to be able to determine the order in which events in an internetwork occur. Without clock synchronization, the timestamps of networked or distributed file systems (such as NFS, AFS, or DFS) may not reflect the actual time at which a file was created. Software distribution programs like **`rdist`** depend upon reliable timestamps to update software. NTP is a way to help synchronize time in an internetwork so that these types of services or applications operate properly.

An NTP **synchronization subnet** is a network of timekeeping systems, called **time servers**. These time servers are a subset of the systems on a network or an internetwork. Each time server synchronizes to Universal Coordinated Time (also known by the acronym UTC). Each server measures the time difference between its local system clock and the system clocks of its neighbors in the subnet.

NTP Time Server Hierarchy

Time servers are organized into levels, or **strata**. Stratum-1 servers are directly connected to an external time source. The external time source can be a device such as a radio clock, which decodes UTC timecodes that are broadcast from radio services in the United States, Canada, and in some European countries.

NTP assumes that servers that are not stratum-1 servers have several possible sources to which they can synchronize their time. NTP then chooses a server to synchronize to, based on factors such as which server is at the lowest-numbered stratum, and which is the closest in terms of network delay. If the chosen synchronization source becomes unavailable (due to failure of the server or in the network path), NTP automatically selects a different source from the available servers.

The stratum level of your local NTP server is always one more than the stratum level of the time server to which your server is synchronizing. Thus, if your server is synchronizing to a stratum-1 server, your server is a stratum-2 server. If your server is synchronizing to a stratum-2 server, then it is a stratum-3 server. Because the synchronization source can change due to server or network path availability, the stratum level of your server can also change. The maximum stratum level that a server can assume is 15.

Figure 17 depicts the organization of time servers into strata. The arrows show the direction of time synchronization.

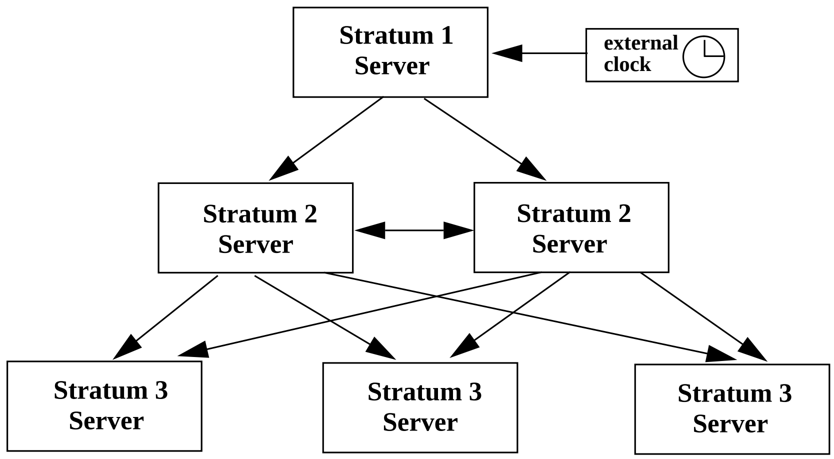


Figure 17 **Hierarchy of NTP Time Servers**

A time server processes a client's request for time and immediately returns a message to the client. From the returned message, the client determines the server's time, compares it to its local time, and then adjusts its local clock. Instead of adjusting the local clock all at once (which could cause the clock to be set backward), **xntpd** adjusts the clock to its new value at small, constant increments over a period of time.

Note that a client of a time server can be another time server. Or a client can be a workstation that is simply synchronizing its local system clock to a time server but is not providing time to any other system.

NOTE: **/usr/sbin/ntpdate** is a program that can be used to set the date and time on a system by polling specified NTP servers. If precise timing is not absolutely essential on your local host, **ntpdate** is an alternative to running **xntpd** that consumes less memory than running the daemon. The program must be run as root; typically, the **ntpdate** command is included in the startup script or a **cron** script. See the **ntpdate** man page for more information.

Time Server Roles

An NTP time server can assume different roles in its relationships with other time servers in the synchronization subnet. A time server can assume one or more of the following roles:

- **Server** — The local host provides time to clients when requested. This role can be assumed by time servers at various strata. The server role is illustrated in Figure 18.

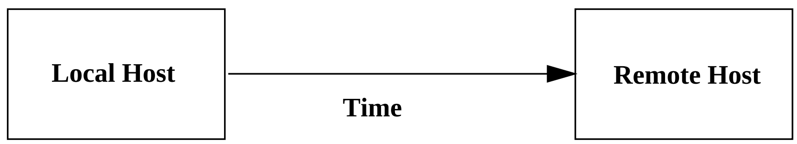


Figure 18 **Local Host as Server**

- **Peer** — The local host obtains time from a specified server and provides time to that server, if requested. This role is most appropriate for stratum-1 and stratum-2 servers or for time servers that are interconnected via multiple network paths. The peer role is illustrated in Figure 19.

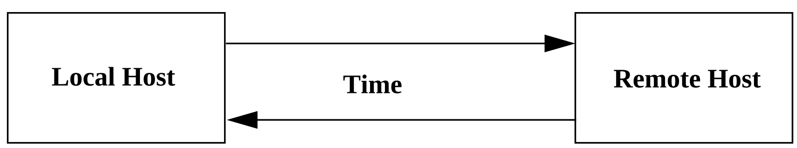


Figure 19 **Local Host as Peer**

- **Client** — The local host obtains time from a specified server, but does not provide time to that server. This role is appropriate for time servers that obtain time from a server of a lower-numbered stratum (for example, a stratum-1 server). The local host may, in turn, provide synchronization to its clients or peers. The client role is illustrated in Figure 20.

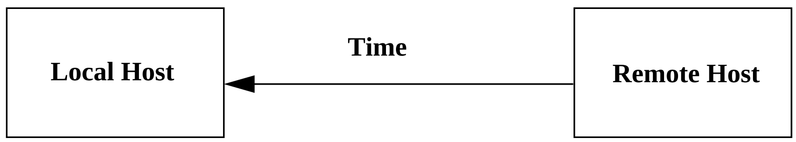


Figure 20 **Local Host as Client**

- **Broadcaster** — The local host provides time to the specified remote host, or more typically, the broadcast address on a LAN. This role is most appropriate for an NTP time server that provides time to workstation clients on a LAN. The broadcaster role is illustrated in Figure 21.

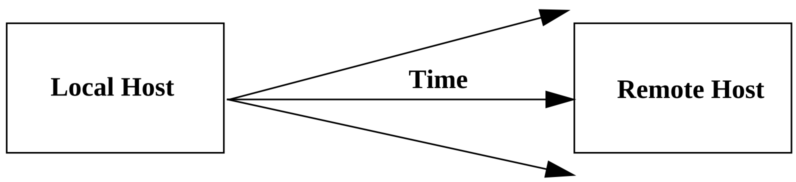


Figure 21 **Local Host as Broadcaster**

- **Broadcast Client** — The local host listens for and synchronizes to broadcast time. This role is most appropriate for time server clients on a LAN. The broadcast client role is illustrated in Figure 22.

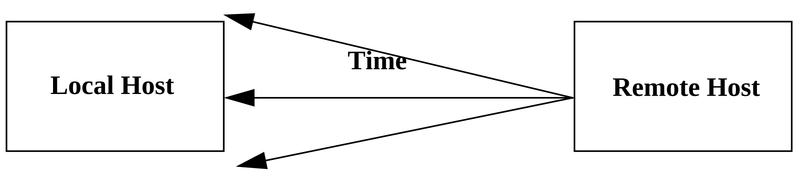


Figure 22 **Local Host as Broadcast Client**

Figure 23 illustrates relationships between time servers in a synchronization subnet.

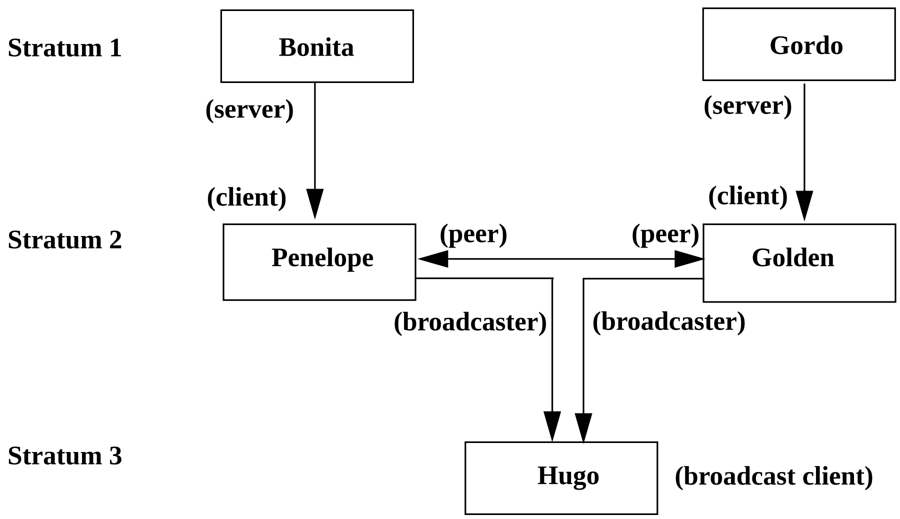


Figure 23 Example of Relationships Between Time Servers

In Figure 23, Gordo and Bonita are stratum-1 servers. Because they are stratum-1 servers, they receive time from external clocks or use their local system clocks as time sources for NTP. Gordo and Bonita may have peer relationships with other stratum-1 time servers (preferably in other administrative domains). Penelope is a client of Bonita and Golden is a client of Gordo. Penelope and Golden are peers to each other. They are also broadcasters. Hugo is a broadcast client to both Penelope and Golden.

“Configuration” on page 268 contains examples of how these roles are configured for **`xntpd`**.

Configuration

This section covers the following topics:

- Overview of the **xntpd** configuration file and the steps needed to configure **xntpd**.
- Guidelines for configuring a synchronization subnet.
- Descriptions of how to configure various characteristics of **xntpd** in the **/etc/ntp.conf** file.

Configuration Overview

When **xntpd** starts, it reads a configuration file to find out its operating characteristics. The configuration file is called **/etc/ntp.conf**. This file is owned by root and is writable only by root (it is readable by anyone). Modifying the configuration file is usually the responsibility of the system administrator.

To configure **xntpd**:

1. Edit the **xntpd** configuration file **/etc/ntp.conf**. You can also use SAM to configure **xntpd**.

   Determine how you want to configure **xntpd** by reading the rest of this chapter and the **xntpd** man page. Then add the appropriate statements in **/etc/ntp.conf**.

2. Set the environment variable **XNTPD** to **1** in the file **/etc/rc.config.d/netdaemons**. This causes **xntpd** to start automatically whenever the system is booted.

3. Set the appropriate value for your local time zone in the file **/etc/TIMEZONE**.

4. Run the **xntpd** startup script with the following command:

   ```
   /sbin/init.d/xntpd start
   ```

5. Run the **ntpq** program with the **-p** switch to verify that **xntpd** is forming the correct relationships with other NTP hosts. (See “Querying xntpd” on page 282.)

Guidelines for Configuration

The following are guidelines that you should consider when planning your configuration:

- Every NTP hierarchy must have at least one stratum-1 server. You may configure your administrative domain to have outside sources of synchronization which ultimately link to stratum-1 server(s), or you may implement your own hierarchy of NTP time servers with one or more stratum-1 servers. For example, an NFS-Diskless cluster may be configured as its own NTP hierarchy. In this topology, the NFS-Diskless server is configured as a stratum-1 NTP server, and may use its own system clock as the time server.
- Configure at least three time servers in your administrative domain. It is important to provide multiple, redundant sources of synchronization, as NTP is specifically designed to select an optimal source of synchronization from several potential candidates. Each time server should be a peer with each of the other time servers. In Figure 24, each of these servers are depicted as a “Stratum 2 Server” within the administrative domain.
- For each time server, select 1-3 *outside* sources of synchronization. This assures a relative degree of reliability in obtaining time, especially if you can select sources that do not share common paths. The sources should operate at a stratum level that is one less than the local time servers. In Figure 24, there are two stratum-1 sources shown for each server in the administrative domain.

NOTE: An enterprise may implement its own hierarchy of NTP time servers, including stratum-1 servers. If your administrative domain is part of an enterprise-wide internet, you should check for available NTP resources in your enterprise. If your administrative domain does *not* have access to lower-stratum time servers, there are NTP servers on the Internet that are willing to provide public time synchronization. (Many stratum-1 and stratum-2 servers can only be used by permission of the administrator of the system; you should always check with the administrator before using an NTP server on the Internet.) The list of servers is available by anonymous **`ftp`** in the file **`pub/ntp/doc/clock.txt`** on Internet host **`louie.udel.edu`** (Internet address 128.175.1.3).

- The outside sources of synchronization should each be in different administrative domains, and should be accessed from different gateways and access paths. Avoid loops and common points of failure. Do not synchronize multiple time servers in an administrative domain to the same outside source, if possible. See Figure 24.

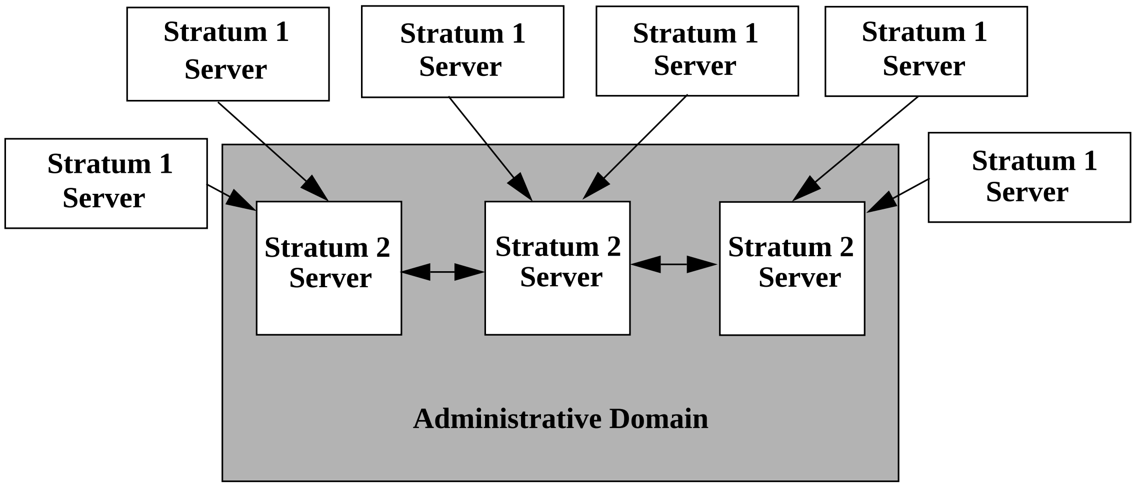


Figure 24 **Example Configuration for an Administrative Domain**

- For enterprise networks that contain hundreds or thousands of file servers and workstations, the local time servers should obtain service from stratum-1 servers. See the previously-mentioned **`clock.txt`** file for stratum-1 sources if your enterprise does not have its own NTP time server hierarchy.
- Single, isolated workstations should not obtain time from a stratum-1 server. Workstations located in sparsely-populated domains without a local synchronization structure should request synchronization from servers that are stratum-2 or higher.
- When defining a relationship between a server of a higher-numbered stratum and a server of a lower-numbered stratum, configure the relationship in the server of the higher-numbered stratum. For example, if a stratum-3 server is a client of a stratum-2 server, configure the relationship in the stratum-3 server. This simplifies configuration maintenance, since there is likely to be more configuration change in systems of higher-numbered stratums, such as workstations.
- Use NTP broadcasting where possible and practical in order to reduce NTP traffic on subnets.

Configuration File

This section describes the statements that can be defined in the /**etc/ntp.conf** configuration file. Configuration file statements are described in the following subsections:

- Configuring relationships with other time servers.
- Configuring a driftfile.
- Configuring authentication.
- Configuring external clocks.
- Restricting incoming packets.

Configuring Relationships with Other Time Servers

The roles of a time server are its relationships to other servers in the synchronization subnet. In the configuration file, a role is defined with one of four statements.

peer *host|IP_address* specifies that the named host is to provide time that the local host may synchronize to, and the local host is willing to provide time to which the named host may be synchronized.

server *host|IP_address* specifies that the named host is to provide time that the local host might synchronize to, but the local host does not provide time to which the named host may be synchronized. (The local host is a client of the named host.) In addition, server statements are used to configure external clocks (radio clocks or local system clocks) for stratum-1 servers. Refer to “Configuring External Clocks” for more information.

broadcast *host|broadcast_address* specifies that the **xntpd** daemon in the local host transmits broadcast NTP messages to a named address, usually the broadcast address on your local network. (The local host is a broadcaster.)

NOTE: Every node in an NTP hierarchy must have either a **server** statement or a **broadcastclient yes** statement in its configuration file. Every node must have an upper-level server. A stratum-1 server must also have a server statement in its configuration file, which specifies a radio clock or internal system clock as a time source.

With the **`peer`**, **`server`**, or **`broadcast`** statement, you can also specify one or more of the following options:

`key` ***`number`*** specifies that the NTP packets sent to the named host are encrypted using the key that is associated with ***`number`***. The authentication feature of **`xntpd`** must be enabled. See "Configuring Authentication" on page 275.

`version 1` must be specified if **`xntpd`** will be requesting time from a host that is running **`ntpd`**, a daemon that is based on version 1 of the NTP protocol. **`version 2`** must be specified if **`xntpd`** will be requesting time from a host that is running an **`xntpd`** implementation that is based on version 2 of the NTP protocol. If either of these options is *not* specified, **`xntpd`** sends out version 3 NTP packets when polling the host; if the host is a version 1 or 2 implementation, the packets will be discarded.

`prefer` specifies that the named host should be the primary source for synchronization when it is one of several valid sources. This option is most useful for a time server on a high-speed LAN that is equipped with an external time source, such as a radio clock. As mentioned in "Guidelines for Configuration" on page 269, synchronization may be provided by outside sources. However, the local time server should be the preferred synchronization source.

The other role that you can define in the configuration file is that of a broadcast client. The statement **`broadcastclient yes`** indicates that the local host should listen for and attempt to synchronize to broadcast NTP packets. The optional statement **`broadcastdelay`** ***`seconds`*** specifies the default round trip delay to the broadcaster.

Note that if the local host is to assume the role of a server in providing time to clients, there is no configuration of this role on the local system. Instead, the configuration file on the client system would contain a **`server`** statement with the name or IP address of the host.

Also note that if authentication is enabled on the local host, the roles you configure are subject to the authentication process. For example, the local host can be configured as a peer or a client of a stratum-1 server, but if the remote server does not meet the criteria for an authenticated synchronization source, it will never be used as a time source by the local host. See "Configuring Authentication" on page 275.

NOTE: **`xntpd`** is an HP implementation of version 3.2 of a publicly-available NTP daemon. HP does not guarantee that **`xntpd`** is fully compatible with version 1 or version 2 implementations of the daemon.

Configuring External Clocks

You can configure **`xntpd`** to support an external clock. Clocks are normally configured with **`server`** statements in the configuration file. Clock addresses can be used anywhere else in the configuration file that a normal IP address is used — for example, in **`restrict`** statements.

Clocks are referenced by an address of the format **`127.127.`*`t`*`.`*`u`***, where ***`t`*** specifies the clock type, and ***`u`*** is a unit number, which is dependent on the clock type for interpretation (this allows multiple instances of the same clock type on the same host).

`xntpd` supports two kinds of clocks:

- Netclock/2 WWVB Synchronized Clock. A system with this type of clock attached and configured is, by definition, a stratum-1 time server. The address used to configure the clock is **`127.127.4.`*`u`***, where ***`u`*** is value between 1 and 4. You must create a device file **`/dev/wwvb%`*`u`***.
- Local synchronization clock, also known as a "pseudo" clock. A system with this type of clock configured uses the local system clock as a time source. The address used to configure this clock is **`127.127.1.`*`u`***, where ***`u`*** is a value between 0 and 15 and specifies the stratum level at which the clock runs. The local host, when synchronized to the clock, operates at one higher stratum level than the clock. This type of clock can be used in an isolated synchronization subnet where there is no access to a stratum-1 time server.

See the **`xntpd`** man page for more information on configuring external clocks.

Figure 23, shown earlier in this chapter, depicts an example of servers in a synchronization subnet and their relationships to each other. Figure 25 on page 274 shows the peer, server, and broadcast statements that are configured for each of the servers. The system that will assume the server role is configured on its client systems. For example, if Penelope is to be a client of Bonita, you configure the name or address of Bonita on Penelope. You do not need to configure Penelope as a client on Bonita.

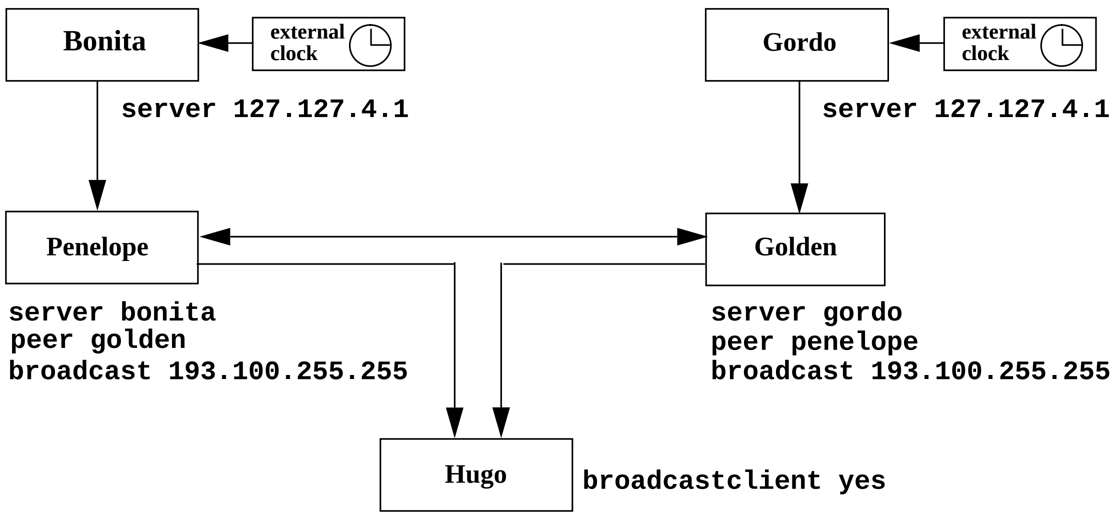


Figure 25 **Example Configurations**

Configuring a Driftfile

`xntpd` computes the error in the frequency of the clock in the local host. It usually takes **`xntpd`** a day or so after it is started to compute a good estimate of the frequency error. The current value of the frequency error may be stored in a **driftfile**. The driftfile allows a restarted **`xntpd`** to reinitialize itself to the estimate stored in the driftfile, saving about a day's worth of time in recomputing a good frequency estimate. You specify the path and name of the driftfile.

NOTE: **`xntpd`** should be operated on a continuous basis. If it is necessary to stop **`xntpd`**, the interval when it is *not* running should be kept to a minimum.

To specify the driftfile, define the keyword **`driftfile`**, followed by the name of the file in which the frequency error value is to be stored. The recommended location for the driftfile is **`/etc/ntp.drift`**. The following is an example of a driftfile statement:

```
driftfile /etc/ntp.drift
```

Configuring Authentication

Authentication is a mechanism that helps protect against unauthorized access to time servers. Authentication is enabled on a system by system basis. Once enabled on a system, authentication applies to *all* NTP relationships configured on the system. When authentication is enabled on a host, only those time servers that send messages encrypted with a configured **key** are considered as candidates to which the host would be synchronized.

In authenticated mode, each NTP packet transmitted by a host has appended to it a **key number** and an **encrypted checksum** of the packet contents. The key number is specified in the **`peer`**, **`server`**, or **`broadcast`** statement for the remote host. You specify either the Data Encryption Standard (DES) or the Message Digest (MD5) algorithm to be used for the encryption of NTP packets.

Upon receipt of an encrypted NTP packet, the receiving host recomputes the checksum and compares it with the one included in the packet. Both the sending and receiving systems must use the same encryption key, defined by the key number.

When authentication is enabled on a host, the following time servers will *not* be considered by the host for synchronization:

- Time servers that send unauthenticated NTP packets.
- Time servers that send authenticated packets that the host is unable to decrypt.
- Time servers that send authenticated packets encrypted using a non-trusted key.

An **authentication key file** is specified on the host. The key file contains a list of keys and their corresponding key numbers. Each key-key number combination is further defined by a key format, which determines the encryption method being used. See the **`xntpd`** man page for more information about the content of the authentication key file. A sample key file is provided in **`/usr/newconfig/etc/ntp.keys`**. The recommended location for the key file is **`/etc/ntp.keys`**. The key file should be secured to allow only the system administrator to have read and write access (mode 600).

While the key file can contain many keys, you can declare a subset of these keys as **trusted keys**. Trusted keys are used to determine if a time server is "trusted" as a potential synchronization candidate. Only time servers that use a specified trusted key for encryption, and whose authenticity is verified by successful decryption, are considered synchronization candidates.

Figure 26 illustrates how authentication works.

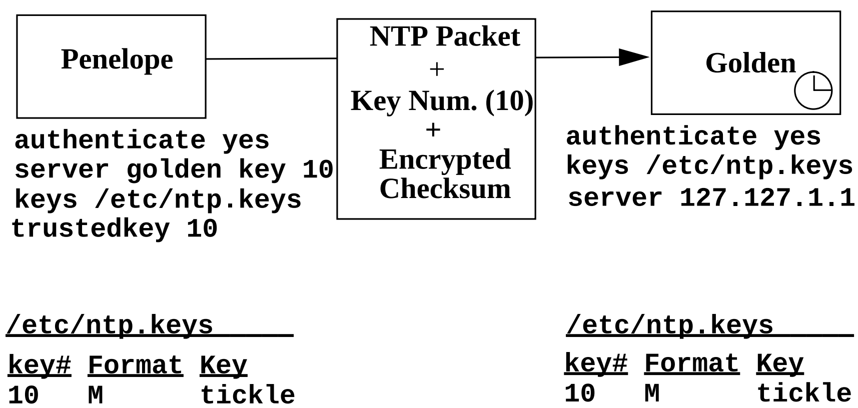


Figure 26 **Authentication Example**

In the example in Figure 26, authentication is enabled for both Penelope and Golden. An NTP time request from Penelope to Golden will include authentication fields--the key ID 10, and a checksum encrypted with the key corresponding to the key ID 10, "tickle." When Golden receives this request, it recomputes the checksum using the packet's key ID field (10) to look up the key for ID 10 in its key file ("tickle") and compares it to the authentication field in the request.

Golden will send back time information with the key ID 10 and a checksum encrypted using "tickle."

In addition, Penelope will only accept time synchronization that have used the key ID 10 and the corresponding encryption key "tickle."

To enable authentication on the local host, include the following statement in the **/etc/ntp.conf** configuration file:

```
authenticate yes
```

If the above statement is not specified, no authentication is used. When authentication is enabled, the following keywords and parameters may also be specified:

`authdelay` ***`seconds`*** indicates the amount of time (in seconds) needed to encrypt an NTP authentication field on the local host. The ***`seconds`*** value is used to correct transmit timestamps for authenticated outgoing packets. The value depends upon the CPU speed of the local host.

CAUTION: The startup script automatically calculates the proper value for **`authdelay`** for the local system and writes it into the configuration file **`/etc/ntp.conf`**. Do *not* modify this value.

`keys` ***`filename`*** specifies the file that contains the encryption keys used by **`xntpd`**. See the **`xntpd`** man page for the format of the file.

`trustedkey` ***`key#`*** **`[`*****`key#2`*****`]...`** specifies the encryption key ID(s) that are trusted as synchronization sources.

Restricting Incoming NTP Packets

`xntpd` provides a mechanism for restricting access to the local daemon from certain source addresses. In the **`/etc/ntp.conf`** file, you can define a **restriction list** that contains the addresses or addresses-and-masks of sources that may send NTP packets to the local host. For each address or address-mask specified in the restriction list, you can define zero or more flags to restrict time service or queries to the local host.

The source address of each incoming NTP packet is then compared to the restriction list. If a source address matches an entry in the restriction list, the restriction defined by the corresponding flag is applied to the incoming packet. If an address-mask is specified in the restriction list, the source address of each incoming NTP packet is ANDed with the mask, and then compared with the associated address for a match.

The restriction list should not be considered an alternative to authentication. It is most useful for keeping unwanted or broken remote time servers from affecting your local host. An entry in the restriction list has the following format:

```
restrict address [mask mask] [ntpport] [flag] [flag2]...
```

The keyword **`ntpport`** causes the restriction list entry to be matched only if the source port in the packet is the NTP UDP port 123.

Table 10 shows the flags that can be specified for **`xntpd`**:

Table 10 **Restrict Option Flags**

Flag	Effect
`ignore`	Ignore all packets.
`noquery`	Ignore **`ntpq`** queries.
`nomodify`	Ignore **`ntpq`** packets that attempt to modify the state of the server.
`noserve`	Ignore requests for time, but permit **`ntpq`** queries.
`nopeer`	Provide time service, but do not form peer association.
`notrust`	Do not use the host as a synchronization source.

A restriction list entry with no flags set leaves matching hosts unrestricted. A source address of an incoming packet may match several entries in the restriction list. The entry that matches the source address most specifically is the entry that is applied. For example, consider the following restriction list entries:

```
restrict 193.100.0.0 mask 255.255.0.0 ignore
restrict 193.100.10.8
```

The first entry causes packets from source addresses on net 193.100 to be ignored. However, packets from host 193.100.10.8 are unrestricted, as specified by the second entry. The two restriction list entries effectively cause all packets from net 193.100 to be ignored, with the exception of packets from host 193.100.10.8.

The following are examples of restriction list entries for a local host with the address 193.100.100.7. These entries assume that **`ntpq`** requests to the local host can be made only from the local host or the host with address 193.8.10.1, while the local host only synchronizes to a time source on net 193.100.

```
#default entry - matches *all* source addresses
restrict default notrust nomodify

#trust for time, but do not allow ntpq requests
restrict 193.100.0.0 mask 255.255.0.0 nomodify noquery

#ignore time requests, but allow ntpq requests
restrict 193.8.10.1 noserve

#local host address is unrestricted
restrict 193.100.100.7
```

Starting xntpd

To start **xntpd**:

1. Set the environment variable **XNTPD** to **1** in the file **/etc/rc.config.d/netdaemons**. This causes **xntpd** to start automatically whenever the system is booted.
2. Issue the following command to run the **xntpd** startup script:

```
/sbin/init.d/xntpd start
```

Command line arguments for starting **xntpd** may be specified with the **XNTPD_ARGS** environment variable in the file **/etc/rc.config.d/netdaemons**. See the **xntpd** man page for more information about command line arguments.

Stopping xntpd

NOTE: **xntpd** should be operated on a continuous basis. If it is necessary to stop **xntpd**, the interval when it is *not* running should be kept to a minimum.

If you modify the configuration file or the **XNTPD_ARGS** environment variable in the file **/etc/rc.config.d/netdaemons** while **xntpd** is running, you have to stop and restart the daemon in order for the configuration changes to take effect.

To stop **xntpd**, issue the following command:

```
/sbin/init.d/xntpd stop
```

Querying xntpd

ntpq is a program used to query systems that are running **xntpd** about the current state of the server. It can also be used to obtain a list of a server's peers. **ntpq** sends requests to and receives responses from NTP time servers using a special form of NTP messages called **mode-6 control messages**. The program can be run either interactively or from a command line. See the **ntpq** man page for details about using this program.

ntpq is most useful for querying remote NTP implementations to assess their timekeeping accuracy and to expose problems in configuration or operation.

NOTE: When you specify time-related configuration options in **/etc/ntp.conf**, you specify the values in seconds. **ntpq**, however, displays time values in milliseconds, as specified by the RFC 1305 NTP standard.

Use **ntpq** to verify the following:

- **xntpd** can form associations with other NTP hosts.
- Synchronization is taking place correctly.

After **xntpd** starts, run the **ntpq** program with the **-p** option:

```
/usr/sbin/ntpq -p
```

The **p** option prints a list of NTP hosts known to the server, along with a summary of their states. After a while, a display like the following appears:

```
  remote           refid          st when poll reach   delay   offset    disp
=============================================================================
+node1         node3             2  131  256  373    9.89   16.28     23.25
*server1       .WWVB.            1  137  256  377  280.62   21.74     20.23
-node2         node4             2   49  128  376  294.14    5.94     17.47
+server2       .WWVB.            1  173  256  377  279.95   20.56     16.40
```

The **remote** column shows hosts specified in the local host's configuration file plus other hosts that are configured to be peers with the local host. The host address preceded with a '*' indicates the current synchronization source. An '-' indicates a host that was not considered for synchronization, while a '+' indicates a host that was considered for synchronization.

The **refid** column shows the current source of synchronization for the remote host. '.WWVB.' indicates that the host uses a radio clock that receives time signals from the U.S. government radio station WWVB.

The **st** column shows the stratum level of the remote host.

The **when** column shows the number of seconds since the remote host was last heard from.

The **poll** column shows the polling interval to the remote host, as determined by **xntpd**. You can define the minimum polling interval with the **minpoll** option in the **peer**, **server**, or **broadcast** definitions in the **/etc/ntp.conf** file. See the **xntpd** man page for more information on setting this option.

The **reach** column shows the status of the reachability register in octal format. See the RFC for information on how to interpret this value.

The **delay**, **offset**, and **dispersion** columns show the value, in milliseconds, computed for the remote host.

Troubleshooting ntp

If **ntp** is not operating properly, use this section to identify and correct the problem.

To Find Out if xntpd is Running

Issue the following command to find out if **xntpd** is running:

```
/usr/bin/ps -ef | /usr/bin/grep xntpd
```

This command reports the process identification (PID), current time, and the command invoked (**xntpd**). An example output is shown below:

```
daemon   4484      1  0   Feb 18      0:00 xntpd
user     3691   2396  2   15:08:45    0:00 grep xntp
```

Ensure **syslogd** is configured to log daemon information messages to the file **/var/adm/syslog/syslog.log**. To check this configuration, make sure **/etc/syslog.conf** includes one of the following lines:

```
*.info        /var/adm/syslog/syslog.log
```

or

```
daemon.info   /var/adm/syslog/syslog.log
```

If **xntpd** is not running, check the **syslog** file for related messages.

NTP Associations

Each NTP daemon must form an association with a time source: a higher-level (lower stratum) server or, for Stratum-1 servers, an external clock. NTP daemons may form additional associations with peer servers. To list the NTP associations the local NTP daemon has established, use the command:

```
/usr/sbin/ntpq -p
```

Note that in the output an asterisk (*****) must appear next to the node name to indicate that an association has been formed.

In the example below, the local NTP daemon has established an association with the NTP daemon on node **good.cup.hp**, but not with the node **bad**:

```
*good.cup.hp      LOCAL(1)      2   29   64  377    5.43   -0.16    16.40
 bad              0.0.0.0       -   31   64    0
```

If the local node cannot form an association with its higher-level server or its peer, log in to the higher-level server or peer and issue the command:

```
/usr/sbin/ntpq -p
```

Verify that the higher-level server/peer has itself established an association with a time source.

Query with Debug Option

If you cannot form an association with a server or peer, stop the local **xntpd** and send a time request to the server/peer with the **ntpdate** command and the debug (**-d**) option:

```
/sbin/init.d/xntpd stop
/usr/sbin/ntpdate -d server
```

The debug (**-d**) option prints information about the requests sent to the remote **xntpd** and the information returned by the remote **xntpd**. Note that **ntpdate** will fail if **xntpd** is already running on the local system.

Note also that **ntpdate** does not use authentication, so it should only be executable by **root**.

You can also use **`ntpdate`** on systems where exact time synchronization is not necessary. You could run **`ntpdate`** periodically from **`cron`** every hour or two to synchronize the local clock to another system's clock. Refer to the **`ntpdate`**(1M) man page for more information.

Common Problems

This section covers typical problems with **`ntp`** operation.

Problem 1: No suitable server for synchronization found.

Every NTP time hierarchy must have at least one stratum-1 server, with an external time source configured, either an attached radio clock (Netclock/2 WWVB Synchronized Clock) or the local system clock. If there is no stratum-1 server in the hierarchy, no associations will be formed. To verify that the local **`xntpd`** is able to form an association, issue the command:

```
/usr/sbin/ntpdate server
```

The ***`server`*** is the name of a trusted server, such as a peer or higher-level (lower stratum) server. If the local **`xntpd`** is unable to form any associations, this command will return the message "No suitable server for synchronization found." Check the sections below for possible causes.

Time Difference Greater than 1000 seconds When evaluating incoming time updates, clients and peers reject time from servers/peers if the time difference is 1000 seconds or greater. On a non-broadcast client or peer, the **`xntpd`** daemon will eventually die if it cannot find a suitable server after six consecutive polls, or five polling cycles (approximately 320 seconds if using the default polling interval).

Because of this behavior, you may have to issue the following command to synchronize the local system time with another NTP server before starting **`xntpd`**:

```
/usr/sbin/ntpdate server
```

For HP-UX NFS Diskless Clusters, the **`/sbin/init.d/xntpd`** script on the diskless clients will execute **`xntpdate`** to synchronize time with the diskless cluster server before starting **`xntpd`**.

You can also explicitly specify a trusted time server in **/etc/rc.config.d/netdaemons** and /**sbin/init.d/xntpd** will execute **xntpdate**, querying the specified time server.

Startup Delay When **xntpd** first starts, it takes five poll cycles (320 seconds using the default polling interval) to form an association with a higher-level server or peer. During this time window, **xntpd** will not respond to time requests from other NTP systems, since it does not have a suitable time source. This window exists even if **xntpd** is using an external clock, which can be either an attached radio clock (Netclock/2 WWVB Synchronized Clock) or the local system clock (**server 127.127.*n.n***).

For external clocks, **xntpd** will not form a complete association until it has sent five successful polls to itself using the local loopback address.

Problem 2: Version 1 and 2 NTP Servers Do Not Respond

NTP version 3 packets (HP-UX 10.0 NTP is version 3) are ignored by NTP version 1 and version 2 systems. The solution is to indicate the version 1 and 2 systems in the configuration entries on the version 3 systems. This will tell the version 3 system to use the older message formats when communicating with these systems.

The following configuration file entries tell **xntpd** to use NTP version 2 message formats when communicating with **some_ver2.sys** and NTP version 1 when communicating with **some_ver1.sys**.

```
server some_ver2.sys version 2
server some_ver1.sys version 1
```

Reporting Problems

Provide the following information when reporting NTP problems:

- **/etc/ntp.conf** (or an alternate configuration file, if used)
- **/etc/rc.config.d**
- NTP drift file (if configured)
- NTP statistics file (if configured)
- **/var/adm/syslog/syslog.log** (**xntpd**/NTP entries)
- output from **/usr/sbin/ntpq -p**
- output from **ntpdate -d server** (stop the local **xntpd** first)

9

Configuring gated

gated, (pronounced “gate D”), is a routing daemon that handles multiple routing protocols. The **gated** daemon can be configured to perform all or any combination of the supported protocols.

This chapter contains information about how to configure and use version 3.0 of **gated**. It includes the following sections:

- Overviews of **gated** functions and of the steps needed to configure **gated**.
- Configuring the RIP Protocol. **gated** version 3.0 supports version 2 of the RIP protocol.
- Configuring the OSPF Protocol. Support for this protocol is new with **gated** version 3.0.
- Customizing routes.
- Specifying trace options.
- Specifying route preferences.
- Starting **gated**.
- Troubleshooting **gated**.
- Migrating from **gated** version 2.1 to version 3.0.

For information on configuring the HELLO and EGP protocols for **gated**, type **man 4 gated.conf** at the HP-UX prompt.

You cannot use SAM to configure **gated**.

Overview

A **router** is a device that has multiple network interfaces and transfers Internet Protocol (IP) packets from one network or subnet to another within an internetwork. (In many IP-related documents, this device is also referred to as a "gateway." The term "router" is used in this chapter.) The **gated** daemon updates routing tables in internetwork routers. Developed at Cornell University, **gated** handles the RIP, EGP, HELLO, BGP, and OSPF routing protocols, or any combination of these protocols.

Routing protocols are designed to find a path between network nodes. If multiple paths exist for a given protocol, the shorter paths are usually chosen. Each protocol has a cost or a metric that it applies to each path. In most cases, the lower the cost or metric for a given path, the more likely a protocol will choose it.

When started, **gated** reads the kernel routing table on the local machine. **gated** maintains a complete routing table in the user space, and keeps the kernel routing table (in the kernel space) synchronized with this table.

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 link failures in paths.

Advantages

Using **gated** offers these advantages:

- Dynamic routing eliminates the need to reset routes manually. When network failures occur, routes are automatically re-routed.
- Dynamic routing makes it easier to add and administer nodes.
- Dynamic routing lowers the cost of operating complex internet systems.
- **gated** translates among several protocols, passing information within or between IP routing domains or autonomous systems. "**Autonomous system**" is used here to refer to a group of connected nodes and routers in the same administrative domain that are exchanging routing information via a common routing protocol.
- **gated** gives the system administrator flexibility in setting up and controlling

network routing. For example, **gated** can listen to network traffic at specified routers, 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.

gated should be run on routers (gateways) so its routing information can be sent to other routers. **gated** supports many routing protocols that allow routers to build and maintain dynamic routing tables. However, **gated** also supports RIP (Routing Information Protocol), which can run on end systems (systems with only one network interface) as well as routers.

gated is useful in topologies with multiple routers and multiple paths between parts of the network. **gated** allows the routers to exchange routing information and dynamically change routing information to reflect topology changes and maintain optimal routing paths.

Alternatively, you may configure IP routes manually with the **route** (1M) command. For end systems in subnets with only one router (gateway) to the rest of the internet, manually configuring a default route is usually more efficient than running **gated**. Type **man 1M route** at the HP-UX prompt.

When connected to wide-area networks, **gated** can be used to inject local routing information into the wide-area network's routing table.

Protocols

For routing purposes, networks and gateways are logically grouped into autonomous systems. An autonomous system (AS) is a set of networks and gateways that is administered by a single entity. Companies and organizations that wish to connect to the Internet and form an AS must obtain a unique AS number from the Internet Assigned Numbers Authority.

An interior gateway protocol is used to distribute routing information within the autonomous system. An exterior gateway protocol is used to distribute general routing information about an autonomous system to other autonomous systems.

Dividing networks into autonomous systems keeps route changes inside the autonomous system from affecting other autonomous systems. When routes change within an autonomous system, the new information need not be propagated outside the autonomous system if it is irrelevant to gateways outside the autonomous system.

gated supports the following interior gateway protocols, as defined in IETF RFCs:

- RIP (Routing Information Protocol) is a common routing protocol used within an autonomous system. 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. There are currently two versions of RIP implementations: version 1, as defined in RFC 1058, and version 2, as defined in RFC 1388. **gated** supports all version 1 features and most of the features of version 2. The following version 2 features are not supported: route aggregation, RIP management information base (MIB), route tag, and authentication.
- OSPF (Open Shortest Path First), like RIP, is a routing protocol that allows routing information to be distributed between routers in an autonomous system. Each router on the network transmits a packet that describes its local links to all other routers. The distributed database is then built from the collected descriptions. If a link fails, updated information floods the network, allowing all routers to recalculate their routing tables at the same time. OSPF is more suitable than RIP for routing in complex networks with many routers. **gated** 3.0 supports most of the features of OSPF version 2, as described in RFC 1247. The following version 2 feature is not supported: IP type of service (TOS) routing. Equal cost multipath routes are limited to one hop per destination, because the HP-UX kernel supports only one gateway per route.
- HELLO was designed to work with routers called "Fuzzballs." Most installations use RIP or OSPF instead of HELLO.

NOTE: Do not mix RIP and OSPF protocols within a single network, as the routing information may conflict.

Table 11 compares the advantages and disadvantages of the RIP and OSPF protocols.

Table 11 Comparison of RIP and OSPF Protocols

RIP	OSPF
Advantage: RIP is easy to configure.	*Disadvantage:* OSPF is complicated to configure and requires network design and planning.
Advantage: An end system (a system with only one network interface) can run RIP in passive mode to listen for routing information without supplying any.	*Disadvantage:* OSPF does not have a passive mode.
Disadvantage: RIP may be slow to adjust for link failures.	*Advantage:* OSPF is quick to adjust for link failures.
Disadvantage: RIP generates more protocol traffic than OSPF, because it propagates routing information by periodically transmitting the entire routing table to neighbor routers.	*Advantage:* OSPF generates less protocol traffic than RIP, because each router transmits only information about its links instead the whole routing table, and because OSPF allows you to divide an autonomous system into areas, each with a designated router that exchanges inter-area routing information with other routers. Intra-area routing information is isolated to a single area.
Disadvantage: RIP is not well suited to large networks, because RIP packet size increases as the number of networks increases.	*Advantage:* OSPF works well in large networks.

gated supports the following exterior gateway protocols:

- 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 routing information between autonomous systems.
- BGP (Border Gateway Protocol) is intended as a replacement for EGP. BGP uses path attributes to select routes. One of the attributes that BGP can pass is the sequence of autonomous systems that must be traversed to reach a destination. **`gated`** supports BGP versions 2 and 3, as described in RFCs 1163 and 1267.

NOTE: BGP is not currently supported on HP-UX systems.

Configuration Overview

When **gated** starts, it reads a configuration file to find out how each protocol should be used to manage routing. By default, it uses the configuration file called **/etc/gated.conf**. Creating the configuration file is usually the responsibility of the system administrator.

The configuration file may include up to eight sections (called **classes**) of configuration **statements**. Statements can be further defined with optional **clauses**. The eight classes of configuration statements are:

- Directives are statements that are immediately acted upon by the **gated** parser.
- Trace statements control **gated** tracing options.
- Options statements define global **gated** options.
- Interface statements define router interface options.
- Definition statements identify the autonomous system that the router belongs to, the router ID, and “martian” addresses (any addresses for which routing information should be ignored).
- Protocol statements enable or disable **gated** protocols and set protocol options.
- Static statements define static routes or default routers that are installed in the kernel routing table.
- Control statements define routes that are imported to the router from other routing protocols and routes that the router exports to other routing protocols.

Type **man 4 gated.conf** at the HP-UX prompt for a description of each configuration class and to determine which statements belong to which class.

NOTE: If you are currently using the RIP protocol with version 2.0 of **gated**, you need to convert the current **/etc/gated.conf** configuration file to the format used by **gated** 3.0. See “Migrating from gated 2.0 to 3.0” on page 353.

To configure **gated**:

1. Create the **gated** configuration file **/etc/gated.conf**.

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

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

2. Determine how you want to configure each routing protocol by reading the rest of this chapter and the **gated.conf**(4) man page. Then add the appropriate statements for each protocol in **/etc/gated.conf**.

   The next section, “Configuring the OSPF Protocol” on page 306, describes statements in the configuration file that affect OSPF routing. RIP configuration is described in “Configuring the RIP Protocol” on page 298. For more detailed descriptions of the configuration statements, as well as for descriptions of the HELLO and EGP protocol statements, type **man 4 gated.conf** at the HP-UX prompt.

3. Add statements as needed for any additional configuration information. See “Customizing Routes” on page 335, “Specifying Tracing Options” on page 337, and “Specifying Route Preference” on page 339 for other configuration options.

   In particular, you may want to prevent **gated** from deleting interfaces from the routing table if **gated** receives no routing protocol information from that interface. One way to do this is to insert passive interface definitions in the **interfaces** statements. For example:

   ```
   interfaces {
      interface 15.1.1.1 16.1.1.1 passive ;
   } ;
               :
               :
   <protocol statements follow>
   ```

4. If you normally use default routes, you must configure a static default route in the **gated** configuration file. If the default route is a gateway node, add the following entry to /**etc/inetd.conf** (enter the gateway node’s IP address for ***gateway_IP_Address***):

```
static {
    default gateway gateway_IP_Address retain ;
} ;
```

The default route may be a local interface, such as in topologies where there is a Proxy ARP server on the local network. If the default route is a local interface, add the following entry to **/etc/inetd.conf**:

```
static {
    default interface local_IP_Address retain ;
} ;
```

The ***local_IP_Address*** is the local system's IP address of the interface that acts as the default route. If a Proxy ARP server is used, this is the local address of the interface attached to the same network as the Proxy ARP server.

For more information, refer to the section "Customizing Routes" on page 335 and the section covering "Common Problems" on page 348 in the "Troubleshooting gated" section.

5. To check for syntax errors in the configuration file, run **gated** with the **-c** or **-C** option. (**gated** exits after parsing the configuration file.)
6. Set the environment variable **GATED** to **1** in the file **/etc/rc.config.d/netconf**. This causes **gated** to start automatically whenever the system is booted.
7. To start **gated**, reboot your system, or run the **gated** startup script with the following command:

```
/sbin/init.d/gated start
```

Examples of **gated** configuration files are included in the sections "Configuring the OSPF Protocol" on page 306 and "Configuring the RIP Protocol" on page 298. They are also included in the **/usr/newconfig/gated/conf** directory.

NOTE: It is best to use IP addresses in dot notation (for example, **a.b.c.d**) when you specify an address for a configuration option such as a router, host, or interface. Host names that have multiple IP addresses associated with them are considered an error.

Configuring the RIP Protocol

RIP uses hopcount to determine the shortest path to a destination. Hopcount is the number of routers a packet must pass through to reach its destination. If a path is directly connected, it has the lowest hopcount of 1. If the path passes through a single router, the hopcount increases to 2. Hopcount can increase to a maximum value of 16, which is RIP's "infinity metric," an indication that a network or node cannot be reached.

If **gated** encounters an unreachable node, 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.

Hosts with only one LAN interface may use the RIP protocol with **gated** to passively listen to routing information when there is more than one router on the LAN. If there is only one router on the LAN (leaving only one path off the local LAN), you may prefer to configure a static route to that router in **/etc/rc.config.d/net** instead of running **gated**.

In certain cases 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.

RIP Protocol Statement

The syntax for the RIP protocol statement is:

```
rip yes| no| on| off [ {
    broadcast | nobroadcast ;
    nocheckzero ;
    preference preference ;
    defaultmetric metric ;
    interface interface_list [noripin] [noripout]
      [metricin metric] [metricout metric]
     [version 1] | [version 2 [multicast | broadcast]] ;
    [interface ...]
    trustedgateways router_list ;
    sourcegateways router_list ;
    traceoptions traceoptions ;
} ] ;
```

Curly braces ({}) are part of the syntax for the RIP protocol statement. Square brackets ([]) are not part of the syntax; they are used here to indicate optional parameters.

yes (or **on**) tells **gated** to enable the RIP protocol at this node and process RIP packets coming in from other nodes. **no** (or **off**) tells **gated** to disable the RIP protocol at this node. 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 on** is assumed.

broadcast specifies that RIP packets are always generated. If the RIP protocols enabled and there is more than one interface specified, **broadcast** is assumed. Specifying **broadcast** with only one interface is useful only when propagating static routes or routes learned from other protocols.

nobroadcast specifies that RIP packets are sent only to routers listed in the **sourcegateways** clause. If the RIP protocol is enabled, but there is only one interface specified, **nobroadcast** is assumed.

nocheckzero specifies that the RIP protocol should not check to see if the reserved fields in the RIP packets are zero. In RIP version 1 (as described in RFC 1058), certain reserved fields should be zeroed out; however, this may vary in RIP implementations.

preference determines the order of routes from other protocols to the same destination in the routing table. **gated** allows one route to a destination per protocol for each autonomous system. In the case of multiple routes, the route used is determined by the value of **preference**.

Default: 100
Range: 0 (most preferred) - 255 (least preferred)

defaultmetric is the default metric used when propagating routes learned from other protocols.

Default: 16
Range: 1 - 16

interface is specified as one of the following (in order of precedence): an IP address (for example, 193.2.1.36), a domain or interface name (for example, **lan0** or **lan1**), a wildcard name (for example, **lan***), or **all** (which refers to all interfaces). Multiple **interface** statements may be specified with different clauses. If a clause is specified more than once, the instance with the most specific interface reference is used.

noripin specifies that **gated** does not process any RIP information received through the specified interface.

noripout specifies that **gated** does not send any RIP information through the specified interface.

metricin specifies the incoming metric for all routes propagated to this node through the specified interface.

Default: kernel interface metric plus 1 (the default RIP hop count)

metricout specifies the outgoing metric for all routes propagated by this node through the specified interface.

Default: 0

`version 1` specifies that RIP version 1 (as defined in RFC 1058) packets are sent; RIP version 2 packets (defined in RFC 1388) are *only* sent in response to a version 2 poll packet. **`version 2`** specifies that RIP version 2 packets are sent to the RIP multicast address or to the broadcast addresses. You can specify how the packets are sent with the **`multicast`** or **`broadcast`** clauses. If you do not specify a version, version 1 is assumed.

`trustedgateways` specifies a list of routers that provide valid RIP routing information; routing packets from other routers are ignored.

Default: all routers on the attached network(s).

`sourcegateways` specifies routers to which RIP routing packets may be sent. If the **`nobroadcast`** clause is specified, routing updates are sent only to routers listed in the **`sourcegateways`** clause.

`traceoptions` enables tracing for the RIP protocol. See “Specifying Tracing Options” on page 337.

Controlling RIP Traffic

This section describes configuration options for RIP routing information sent out by **gated** from the node. Use these options to hide all or part of your network from other networks or to limit network traffic.

Two options for limiting RIP routing information exported by **gated** are in the RIP protocol definition in the **/etc/gated.conf** file:

- The **noripout** clause in the interface definition tells **gated** not to send any RIP information through the listed interfaces.
- The **sourcegateways** clause tells **gated** to send RIP information directly to the specified routers.

See “RIP Protocol Statement” on page 299 for more information about these clauses.

Two options for limiting RIP routing information imported by **gated** are in the RIP protocol definition in the **/etc/gated.conf** file:

- The **noripin** clause in the interface definition tells **gated** not to process RIP information received through the listed interfaces.
- The **trustedgateways** clause tells **gated** to listen to RIP information received only from the specified routers.

See “RIP Protocol Statement” on page 299 for more information about these clauses.

You can also use the **gated** **import** and **export** statements to restrict and control the route information propagated from one routing protocol to another. See “Importing and Exporting Routes” on page 341.

Sample RIP Configurations

Figure 27 and accompanying text describe examples of how **gated** might be configured for the RIP protocol in each node within a networked system.

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 routers D and E can both provide a path to a network, but the path through router D is shorter, nodes B, C, and E will use router D when routing packets to that network. If D goes down, E becomes the new router to that network for nodes B, C, and E.

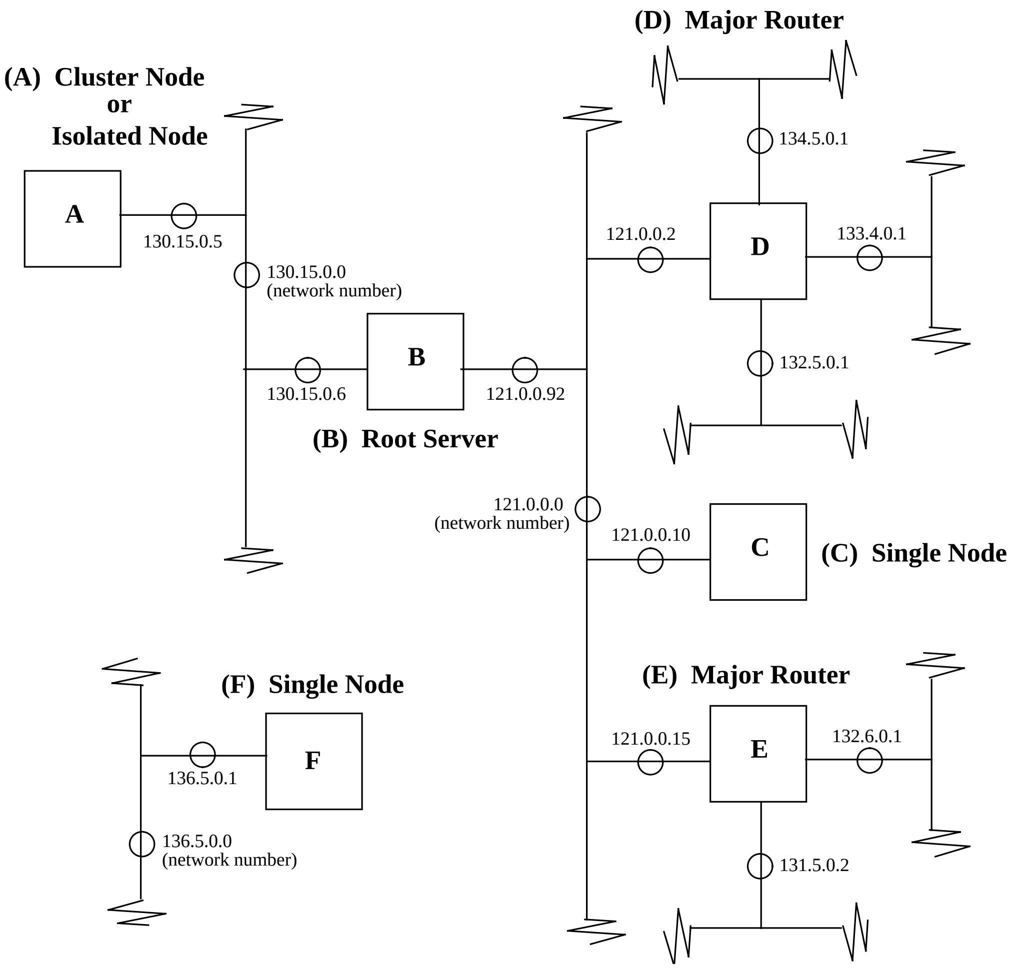


Figure 27 **Sample RIP Network**

A: Cluster Node (or Isolated Node)

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

```
ROUTE_DESTINATION[0]="default"
ROUTE_GATEWAY[0]="130.15.0.6"
ROUTE_COUNT[0]="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:

```
interfaces {
    interface 130.15.0.6 121.0.0.92 passive ;
};
rip yes {
    interface 130.15.0.6 noripout ;
    interface 121.0.0.92 version 2 ;
};
static {
    default gateway 121.0.0.2 preference 255 ;
};
```

In this case, setting **rip** to **yes** is like setting **rip** to **broadcast**. 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:

```
export proto rip interface 121.0.0.92 {
    proto direct {
        130.15.0.0 restrict ;
    };
};
```

To further isolate the LAN from the 121.0.0.0 LAN, do not specify any static routes that specify that you can reach the LAN through B. See “Importing and Exporting Routes” on page 341.

Always specify the **passive** option with the interface's IP address. It tells **gated** to maintain routes even if no other nodes on the 121.0.0.0 network are using RIP. Without this clause, **gated** may change the preference of the route to the interface if routing information is not received for the interface. The static default route adds the specified default to the kernel routing table. Setting the **preference** to **255** allows this route to be replaced whenever another default route is learned from one of the protocols.

C: End System on a LAN with RIP Routers

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

```
rip yes {
    interface 121.0.0.10 version 2;
};
static {
    default interface 121.0.0.10 preference 255 ;
};
```

In this case, setting **rip** to **yes** is equivalent to setting **rip** to **nobroadcast** because the C node has only one interface. With one interface, C can listen to RIP traffic on the network but does not broadcast. Routers must be broadcasting RIP packets on this network for C to learn about them and update its routing table.

D: Major Router

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

```
rip yes {
    interface all version 2 multicast ;
};
```

This runs RIP on all attached networks.

E: Major Router

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

```
rip yes {
    interface all version 2 ;
};
```

Configuring the OSPF Protocol

OSPF is a link-state routing protocol designed to distribute routing information between routers in a single autonomous system (AS). Each OSPF router transmits a packet with a description of its local links to all other OSPF routers. The distributed database is built from the collected descriptions. Using the database information, each router constructs its own routing table of shortest paths from itself to each destination in the AS.

OSPF allows routers, networks, and subnetworks within an AS to be organized into subsets called areas. An area is a grouping of logically contiguous networks and hosts. Instead of maintaining a topological database of the entire AS, routers in an area maintain the topology only for the area in which they reside. Therefore, all routers that belong to an area must be consistent in their configuration of the area. The topology of an area is hidden from systems that are not part of the area. The creation of separate areas can help minimize overall routing traffic in the AS. Figure 28 shows an example of three separate areas defined for an AS.

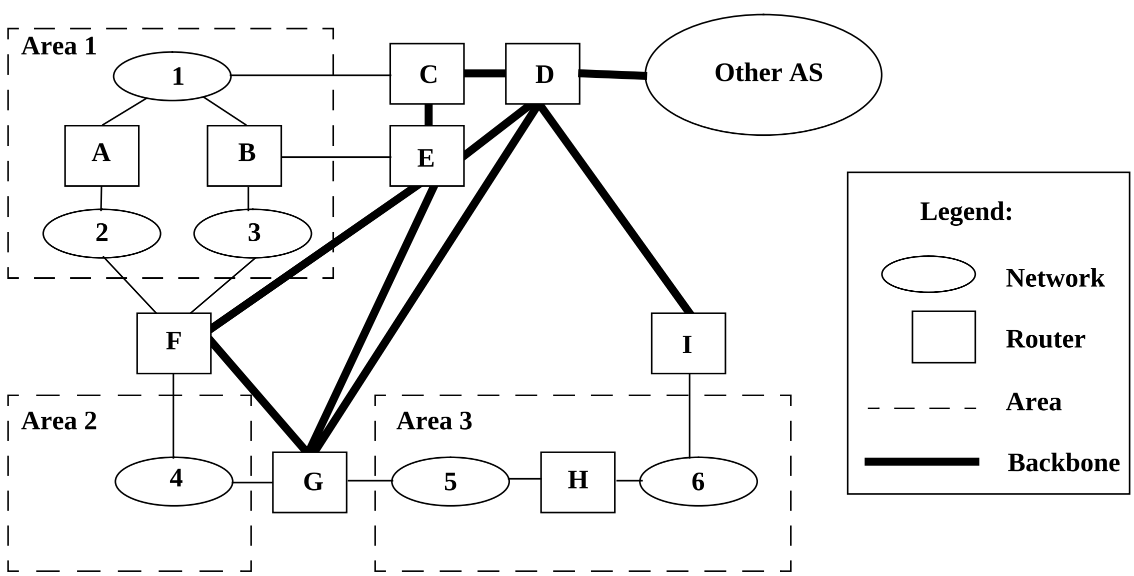


Figure 28 **Areas Defined in an Autonomous System**

Routers that have all their directly-connected networks in the same area are called **internal routers**. In Figure 28, routers A, B, and H are internal routers.

Routers that are connected to multiple areas are called **area border routers**. In Figure 28, routers F and G are area border routers.

Routers that connect one AS to another are called **AS boundary routers**. In Figure 28, router D is an AS boundary router.

Neighbor routers are routers that interface to a common network. OSPF uses its own Hello protocol to determine which routers are neighbors. In Figure 28, routers A, B, and C are a set of neighbor routers that interface to network 1, while routers A and F are another set of neighbor routers that interface to network 2.

Multi-access networks (networks that can be accessed through two or more neighbor routers) must have one of the routers identified as a **Designated Router**. The Designated Router initiates OSPF protocol functions on behalf of the network. In Figure 28, network 1 can be accessed through neighbor routers A, B, or C; one of these routers is elected to become the Designated Router for network 1.

The set of routers that exchange OSPF protocol packets between areas in an autonomous system is called the **backbone**. In Figure 28, routers C, D, E, F, G, and I form an AS backbone that allows protocol packets to travel between the three areas.

OSPF routers exchange various types of **link state advertisements** to build their topological databases. Most link state advertisements are flooded (sent to every router) throughout the attached area. An exception is the link state advertisement sent out by AS boundary routers that describe routes to destinations outside the AS; these advertisements are flooded throughout the AS. Table 12 shows the various types of link state advertisements used by the OSPF protocol.

Table 12 Types of Link State Advertisements

Type	Content	Originated By	Flooded Throughout
Router Link	Router’s links to area	Internal and area border routers	Area
Network Link	List of routers attached to network	Designated Router	Area
Summary link	Routes to destination outside area but within AS	Area border router	Area
AS external link	Routes to destinations outside AS	AS boundary router	AS

AS boundary routers exchange routing information with routers in other autonomous systems. An AS boundary router may be an area border router or an internal router. It can be a backbone router, but it is not *required* that an AS boundary router be a backbone router. An AS boundary router learns about routes outside of its attached AS through exchanges with other routing protocols (such as EGP) or through configuration information. Each AS boundary router calculates paths to destinations outside of its attached AS. It then advertises these paths to all routers in its AS.

There are two levels of routing in the AS:

- **Intra-area routing**, where the source and destination of a packet both reside in the same area. Routing is handled by internal routers.
- **Inter-area routing**, where the source and destination of a packet reside in different areas. Packets travel an intra-area route from the source to an area border router, then travel an inter-area route on a backbone path between areas, then finally travel another intra-area route to the destination.

Planning Your OSPF Configuration

The following is a suggested sequence of steps in planning for OSPF routing in your autonomous system.

1. If your AS will be exchanging routing information with other autonomous systems, you need to obtain a unique AS number from the Internet Assigned Numbers Authority.
2. Partition the AS into areas. Any inter-connected networks can be partitioned into lists of address ranges, with each address range represented as an address-mask pair. The area border routers will summarize the area contents for each address range and distribute the summaries to the backbone. For more information on specifying address ranges, see “Networks” on page 312.
3. Identify the internal routers for each area. An internal router configuration will contain only one area definition.
4. Identify the area border routers and the areas to which they interface. The configuration for each area border router will contain multiple area definitions.
5. For each router, determine the types of interface to each area. Router interfaces can be multicast, non-broadcast multi-access (NBMA), or point-to-point. For more information on router interfaces, see “Interfaces” on page 314.
6. For multi-access networks, identify a Designated Router. For NBMA networks, several routers can be Designated Router candidates. Designated Routers are specified in the interface definitions (see “Interfaces” on page 314).
7. Decide if you want to assign a cost to each interface. For more information about costs, see “Cost” on page 327.
8. Designate stub areas. AS external link advertisements are propagated to every router in every area in an AS, except for routers in configured stub areas. For more information, see “Stub Areas” on page 321.
9. Identify backbone routers. The router configuration will contain a backbone definition and a virtual link definition, if necessary. For more information, see “Defining Backbones” on page 323.
10. Determine if routing packets will be authenticated for each area. For more information, see “Authentication” on page 325.
11. Identify AS boundary routers. For more information, see “AS External Routes (AS Boundary Routers Only)” on page 329.

Enabling OSPF

The default router identifier used by OSPF is the address of the first interface on the router encountered by **gated**. To set the router identifier to a specific address, specify the **routerid *interface*** statement in the Definition class of the **/etc/gated.conf** file.

NOTE: The OSPF protocol should be enabled only for routers. Once the OSPF protocol is enabled for a system, the system is treated as a router by other routers, and not a host.

The OSPF protocol is enabled for a node with the **ospf** statement in the Protocol class of the **/etc/gated.conf** file. The clause **yes** (or **on**) tells **gated** to enable the OSPF protocol at this node and process all OSPF packets coming in from other nodes. If you do not specify an OSPF line in your configuration file, **ospf no** is assumed. The clause **no** (or **off**) tells **gated** to disable the OSPF protocol at this node.

The following is an example of the statement to enable OSPF:

```
ospf yes { ... }
```

Other statements that are defined for the OSPF protocol configuration are explained in the following sections.

Defining Areas

Every OSPF router is associated with one or more areas. The **area** statement identifies an OSPF area. The value is in the form of a dotted quad, or a number between 1 and 4294967295. To define an area, you also need to specify the following:

- The address(es) of the network(s) that make up the area.
- The router interface(s) used to communicate with the area.

Note that the configuration of an area border router contains multiple area definitions; a different router interface is defined for each area. Figure 29 shows an example of an area border router that is connected to area 0.0.0.1 through interface 193.2.1.33 and to area 0.0.0.2 through interface 193.2.1.17.

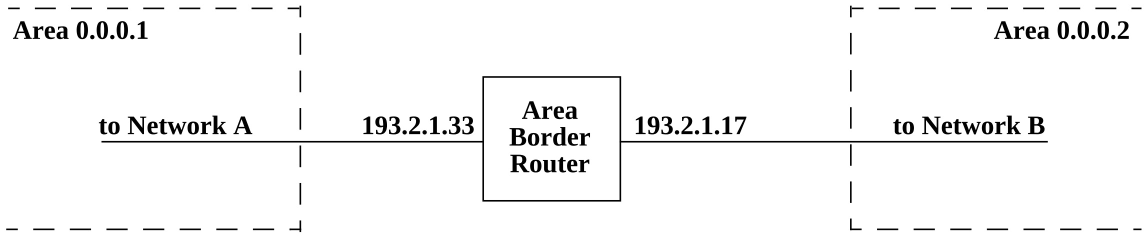


Figure 29 **Area Border Router Configuration Example**

The following is an example of the area definitions in the router's **`/etc/gated.conf`** file:

```
ospf yes {
      area 0.0.0.1 {
        interface 193.2.1.33 {
              ...
        } ;
      } ;
      area 0.0.0.2 {
        interface 193.2.1.17 {
              ...
        } ;

      } ;
   } ;
```

There are various other characteristics that you can define for the area and for the interface(s). The following sections describe the configuration statements that you use in defining an area.

Networks

The **networks** statement defines the address ranges that make up an OSPF area. This definition applies only to area border routers, where multiple areas are specified, and is only required if you need to compress a number of subnets using a network mask.

Inside the **networks** statement, each IP address range is specified by a network address followed by a hexadecimal bit mask. For example, the following address range begins with the network address 193.2.1.16 and includes the first 15 addresses in that network (193.2.1.17 through 193.2.1.31):

```
193.2.1.16 mask 0xfffffff0
```

Many separate networks can be specified in an address range. Area border routers advertise a single route for each address range.

Figure 30 shows an example of a router that is connected to area 0.0.0.1 through interface 193.2.1.33. The attached network consists of addresses 193.2.1.33 through 193.2.1.47. The other network in the area consists of addresses 193.2.1.17 through 193.2.1.31.

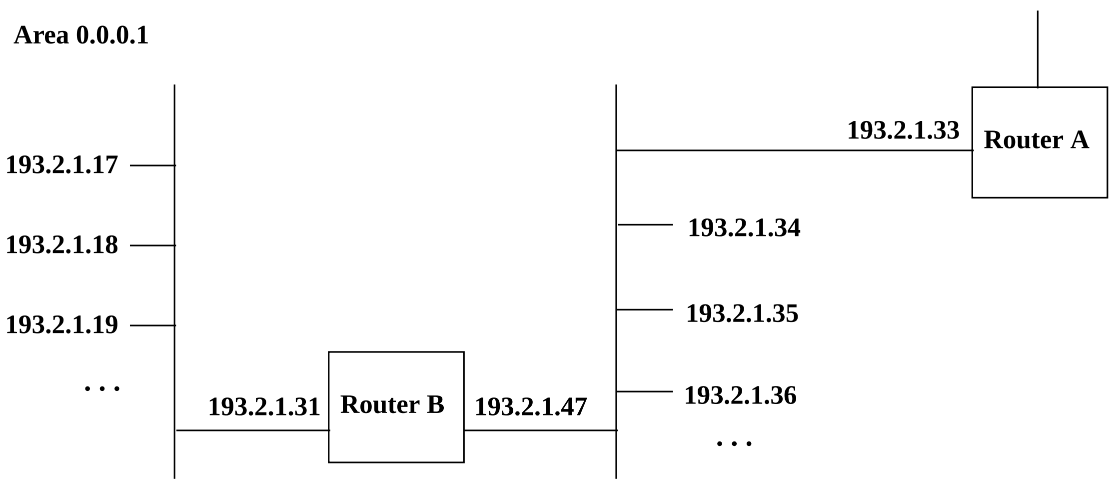


Figure 30 Network Configuration Example

The following is an example of the network definition in Router A's **/etc/gated.conf** file:

```
ospf yes
     area 0.0.0.1
       networks {
          193.2.1.16 mask 0xfffffff0 ;
          193.2.1.32 mask 0xfffffff0 ;
          } ;
          interface 193.2.1.33 {
            ...
          } ;
     } ;
 ...
```

Interfaces

The **interface** statement in the OSPF Protocol definition specifies which interface to use when communicating with the specified network(s). The interface may be specified with an address (for example, **193.2.1.36**), a domain or interface name (for example, **lan0** or **lan1**), a wildcard name (for example, **lan***), or **all**. (The order of precedence is address, name, wildcard name, **all**.) Multiple interface statements may be specified with different clauses. If a clause is specified more than once, the instance with the most specific interface reference is used.

The **cost** clause can optionally be specified to define a cost of sending a packet on the interface. This cost is advertised as the link cost for this interface. See “Cost” on page 327 for more information about setting interface costs.

You can also **enable** or **disable** the interface definition. If **disable** is not explicitly specified, an interface definition is assumed to be enabled.

OSPF supports three types of network interfaces:

- A multicast (or “broadcast”) network is a network that supports two or more attached routers and allows a single message to be addressed to a set of network nodes at the same time. An example of a multicast network is an Ethernet LAN.
- A non-broadcast multi-access (NBMA) network is a network that supports multiple attached routers, but does not support broadcasting of messages. An example of an NBMA network is an X.25 PDN.
- A point-to-point network is a network that joins a single pair of routers. An example of a point-to-point network is a 56Kb serial line.

The definition for each type of interface is described separately in the following sections.

Multicast Interfaces On multicast networks, an OSPF router dynamically detects its neighbor routers through the OSPF Hello message. The following statements are defined for a multicast type interface:

`retransmitinterval` is the number of seconds between retransmission of link states, database description, and link state request packets. This value should exceed the expected round-trip delay between any two routers in the network. A sample value for a LAN is 5 seconds.

> **Default:** None (you must specify a value)
> **Range:** Integer between 0 - 65535

`transitdelay` is the number of seconds it takes to transmit a Link State Update Packet over this interface. This value must take into account the transmission and propagation delays for the interface. It must be greater than 0. A sample value for a LAN is 1 second.

> **Default:** None (you must specify a value)
> **Range:** Integer between 0 - 65535

`priority` should be configured only for interfaces to multi-access networks. This value specifies the priority of the router to become the Designated Router. When two routers attached to a network both attempt to become the Designated Router, the one with the highest router priority value takes precedence.

> **Default:** None (you must specify a value for multi-access networks)
> **Range:** 8-bit unsigned integer between 0-255. 0 means that the router is ineligible to become a designated router on the attached network.

`hellointerval` specifies the number of seconds between transmission of OSPF Hello packets. Smaller intervals ensure that changes in network topology are detected faster, however routing traffic can increase. A sample value for an X.25 network is 30 seconds. A sample value for a LAN is 10 seconds.

> **Default:** None (you must specify a value)
> **Range:** Integer between 0 - 255

NOTE: The **`hellointerval`** value must be the same for all OSPF routers.

routerdeadinterval specifies the number of seconds that hello packets are not received from a router before it is considered “down” or “inactive” by its neighbors. This value should be some multiple of the **hellointerval** value.

> **Default:** None (you must specify a value)
> **Range:** 0 - 65535

NOTE: The **routerdeadinterval** value must be the same for all OSPF routers.

authkey is the password used to validate protocol packets received on the router interface. The value is one of the following: 1 to 8 decimal digits separated by periods, a 1-byte to 8-byte hexadecimal string preceded by **0x**, or a string of 1 to 8 characters in double quotes.

> **Default:** None
> **Range:** Up to 8 bytes

NOTE: To set an **authkey** value, the **authtype** clause must be set to **1** or **simple** for the area. See “Authentication” on page 325 for more information about using OSPF authentication.

Figure 31 shows an example of a router that is connected to a multicast network through interface 193.2.1.35.

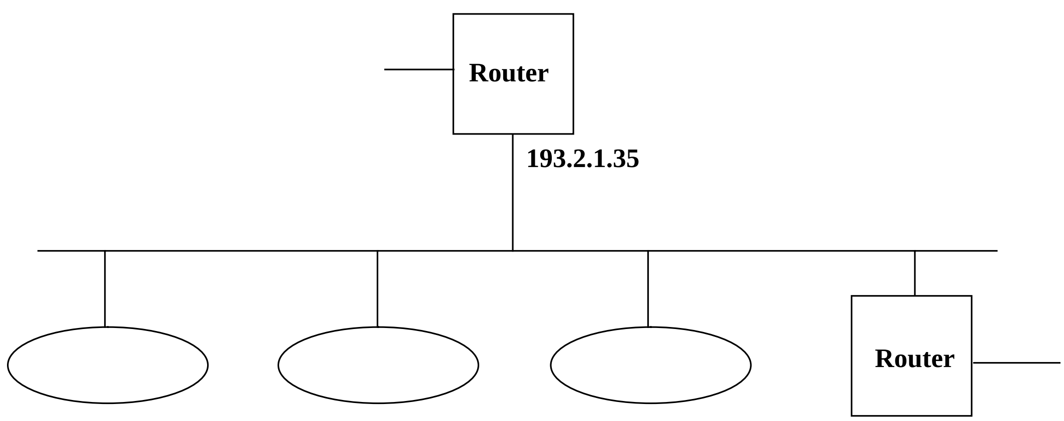


Figure 31 **Multicast Router Interface Example**

The following is an example of the multicast interface definition in the router's **/etc/gated.conf** file:

```
interface 193.2.1.35 cost 5 {
      enable ;
      priority 15 ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
    } ;
```

Non-Broadcast Multi-Access (NBMA) Interface On NBMA networks, certain configuration information, including the routers that are attached to the network, must be supplied in order for OSPF's Hello protocol to communicate with neighbor routers. An NBMA interface definition applies to both X.25 network interfaces as well as for systems that do not support IP multicast. An NBMA type interface is defined with the same statements as for a multicast type interface, with the following additions:

- The clause **nonbroadcast** must be specified in the **interface** statement.
- **pollinterval** specifies a rate at which hellos are sent when a neighboring router becomes inactive. (A router is considered inactive when hellos have not been received from the router for the amount of time specified by the **routerdeadinterval** definition.) The value of **pollinterval** should be larger than the value of **hellointerval**. A sample value for an X.25 network is 2 minutes.

  Default: None (you must specify a value)
  Range: 0 -255
- **routers** specifies the list of routers that are attached to the non-broadcast network. Routers are defined by their IP interface addresses. Routers that are eligible to become Designated Routers must be defined as **eligible**.

Figure 32 shows an example of a router (A) that is connected to an NBMA network through interface 193.2.1.35. Two other routers are also attached to the network: router B is connected through interface 193.2.1.33 and C is connected through interface 193.2.1.46. B and C are eligible to be Designated Routers.

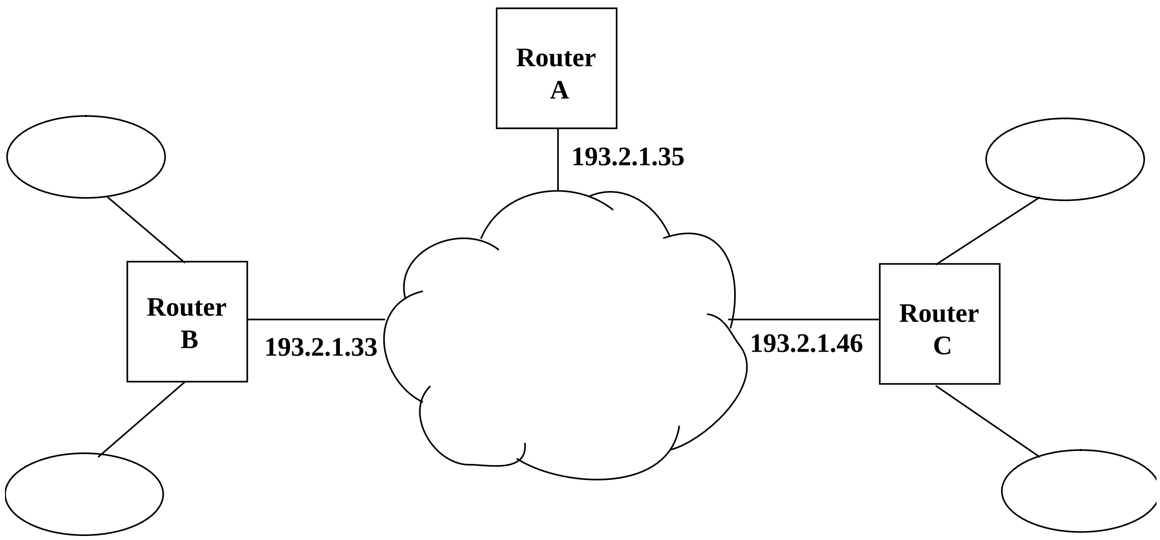


Figure 32 **Non-Broadcast Router Interface Example**

The following is an example of the non-broadcast interface definition in router A's **/etc/gated.conf** file:

```
interface 193.2.1.35 nonbroadcast cost 5 {
      routers {
        193.2.1.33 eligible ;
        193.2.1.46 eligible ;
      } ;
      priority 15 ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
      pollinterval 20 ;
} ;
```

Point-to-Point Interfaces On point-to-point networks, an OSPF router dynamically detects its neighbor router by sending OSPF Hello packets. The following statements are defined for a point-to-point interface:

`retransmitinterval` is the number of seconds between retransmission of link states, database description, and link state request packets. This value should exceed the expected round-trip delay between any two routers in the network. A sample value for a LAN is 5 seconds.

Default: None (you must specify a value)
Range: 0 - 65535

`hellointerval` specifies the number of seconds between transmission of OSPF Hello packets. Smaller intervals ensure that changes in network topology are detected faster, however routing traffic can increase. A sample value for an X.25 network is 30 seconds. A sample value for a LAN is 10 seconds.

Default: None (you must specify a value)
Range: 0 - 255

NOTE: The **`hellointerval`** value must be the same for all OSPF routers.

`routerdeadinterval` specifies the number of seconds that hello packets are not received from a router before it is considered “down” or “inactive” by its neighbors. This value should be some multiple of the **`hellointerval`** value.

Default: None (you must specify a value)
Range: 0 - 65535

NOTE: The **`routerdeadinterval`** value must be the same for all OSPF routers.

A point-to-point interface can be defined with or without a **`nonbroadcast`** clause. If the **`nonbroadcast`** clause is specified, then the **`pollinterval`** statement must be defined:

`pollinterval` specifies a rate at which hellos are sent when a neighboring router becomes inactive. (A router is considered inactive when hellos have not been received from the router for the amount of time specified by the

routerdeadinterval definition.) The value of **pollinterval** should be larger than the value of **hellointerval**. A sample value for an X.25 network is 2 minutes.

> **Default:** None (you must specify a value)
> **Range:** 0 -255

If the device at the other end of the point-to-point network is not an OSPF router, you can prevent Hello packets from being sent to it. This is done using the **stubhosts** statement. **stubhosts** specifies the IP address or domain name of the non-OSPF host. The cost of sending a packet to the host must also be specified. (In most cases, the host has only a single connection to the network so the cost configured has no effect on routing.)

Figure 33 shows an example of a router (A) that is connected to a non-broadcast, point-to-point network through interface 193.2.1.1.

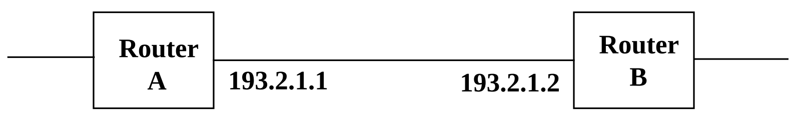


Figure 33 **Point-to-Point Router Interface Example**

The following is an example of the interface definition in router A's **/etc/gated.conf** file:

```
interface 193.2.1.1 nonbroadcast cost 5 {
      hellointerval 30 ;
      routerdeadinterval 30 ;
      retransmitinterval 30 ;
      pollinterval 30 ;
} ;
```

Note that if the router (A) were connected to a multicast, point-to-point network, the **nonbroadcast** clause and the **pollinterval** statement must be omitted.

Stub Areas

By default, AS external link advertisements (routes to destinations outside the AS) are propagated to every router in every area in the AS. Certain OSPF areas can be configured as stub areas. AS external link advertisements are not flooded through stub areas. This reduces the size of the topology database that must be maintained by internal routers in the stub area and reduces the protocol traffic through the area. For example, if all inter-area traffic for an area must go through a single router, then it is not necessary for all routers in the area to receive inter-area routing information.

An area border router advertises in the stub area a default route as the summary of all the IP destinations that are reachable outside the AS. Summary link advertisements (routes to destinations outside the area but within the AS) continue to be sent into the stub area.

The **`stub`** statement specifies that the area is a stub area. A **`cost`** clause can optionally be defined that specifies the cost associated with the default route to be advertised in the stub area.

Figure 34 shows an example of an area border router that is connected to area 0.0.0.2 through interface 193.2.1.20. Since all traffic in and out of area 0.0.0.2 must pass through router A, it is not necessary for the area's internal routers, such as router B, to receive inter-area routing information.

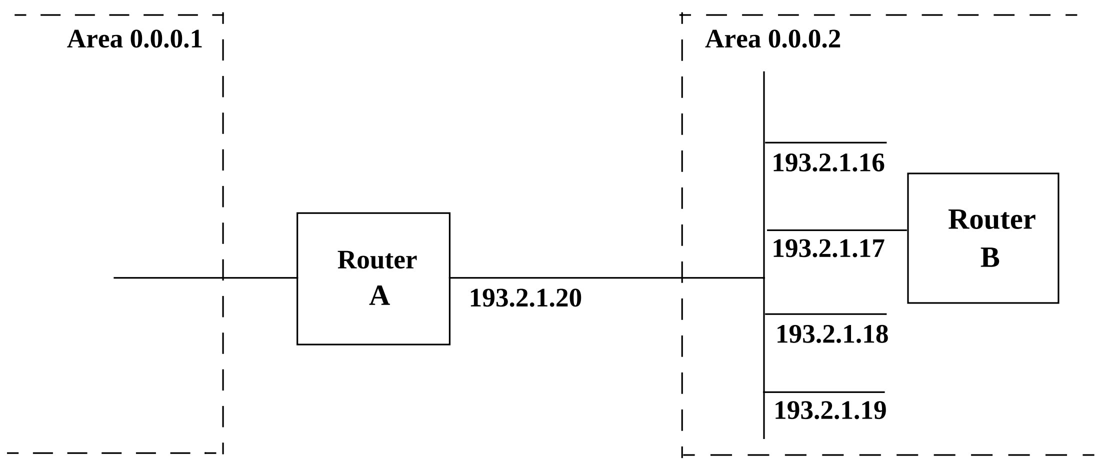


Figure 34 **Area Border Router Configuration Example**

The following is an example of the stub area definition in the router's **/etc/gated.conf** file:

```
OSPF yes {
  area 0.0.0.2 {
    stub cost 5 ;
    networks {
      193.2.1.16 mask 0xfffffff0 :
    } ;
    interface 193.2.1.20 nonbroadcast cost 5 {
      enable ;
      routers {
        193.2.1.17 eligible ;
       } ;
      priority 5 ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
      pollinterval 20 ;
    } ;
  } ;
} ;
```

Defining Backbones

The OSPF backbone distributes routing information between areas. Backbones are defined with the same statements and clauses as areas. The **stub** statement may not be defined for a backbone. The **backbone** statement is used to define a router as a backbone router. If an OSPF internal or area boarder router is also a backbone router, the **backbone** statement must follow the **area** statement(s) in the **/etc/gated.conf** file. Whenever an area border router (a router connected to multiple areas) is configured, backbone information must be provided.

Figure 35 shows an example of two area border routers that form part of a backbone. Router A has interfaces to both area 0.0.0.1 and area 0.0.0.2, while router B has interfaces to areas 0.0.0.3 and 0.0.0.4. Router A is connected to router B through interface 15.13.115.156.

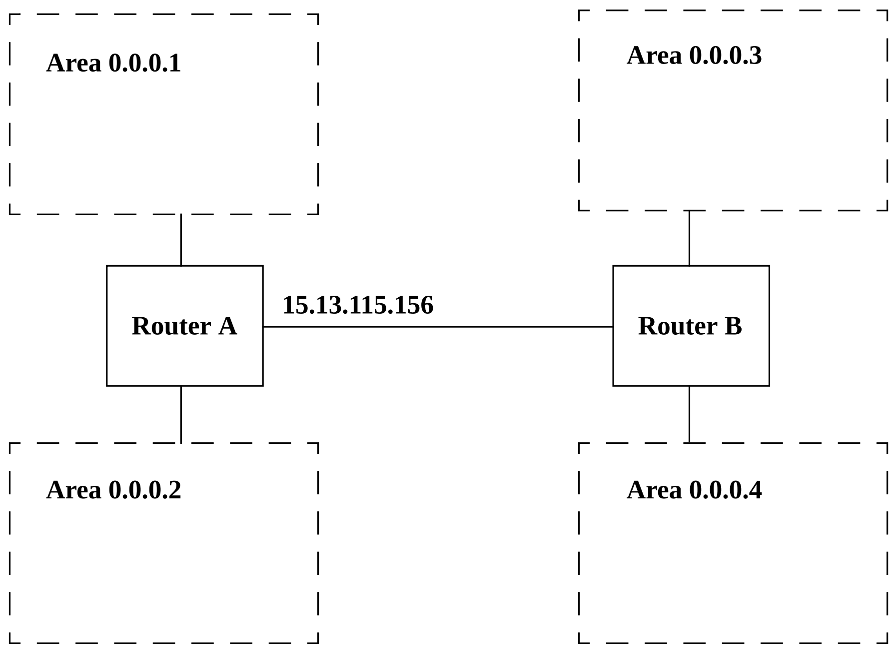


Figure 35 **Backbone Configuration Example**

The following is an example of the backbone router definition for router A's **/etc/gated.conf** file:

```
backbone {
      interface 15.13.115.156 {
        enable ;
        transitdelay 20 ;
        priority 20 ;
        hellointerval 30 ;
        routerdeadinterval 120 ;
        retransmitinterval 60 ;
        } ;
      } ;
```

If the router is directly attached via a point-to-point interface to a host that is not running OSPF, you can prevent OSPF Hello packets from being sent to the host. This is done by specifying the **subhost** statement with the host's address. A cost can optionally be defined.

NOTE: Backbones must be directly-connected or "contiguous". In some **gated** implementations, a "virtual link" can be configured to join non-contiguous backbone routers. Virtual links are not supported on HP-UX systems.

Authentication

The OSPF protocol allows packets containing routing information to be authenticated. The authentication method used is configured on a per-area basis; different authentication methods may be used in different areas.

gated supports a simple password authentication method. You can also choose to have no authentication. The **authtype** statement is used to define the authentication method used for the area. **0** or **none** specifies that routing exchanges in the area are not authenticated. **1** or **simple** specifies that network passwords of up to 64 bits (8 characters) are used to authenticate packets received from routers in the area.

In the simple password authentication method, all routers that interface to a given network use the same password. The password is defined by the **authkey** statement in the router's interface definition. If a router is not configured with the same password as other routers in the network, the router's packets are discarded by other network routers. Note that the password is configured on a per-interface basis. If a router has interfaces to more than one network, different passwords may be configured. This is illustrated in Figure 36.

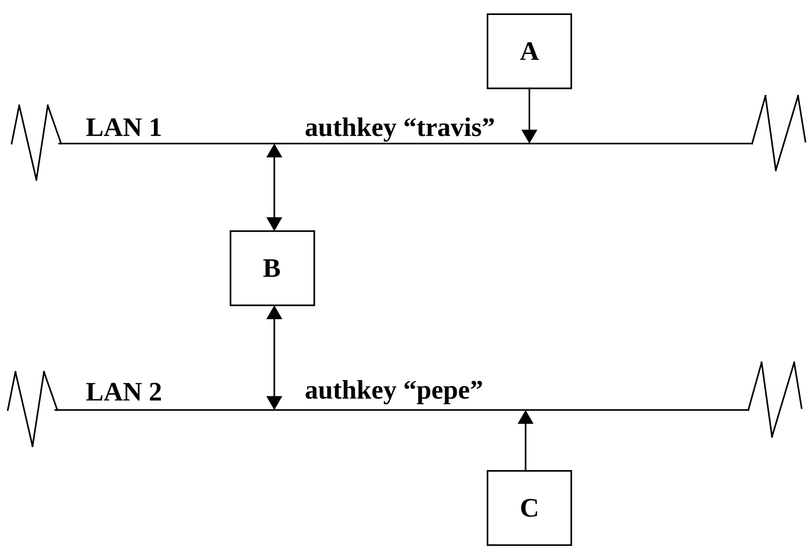


Figure 36 **Simple Password Authentication**

The following example shows an **authtype** statement that enables a simple password authentication for the routers in the area and an **authkey** statement in the interface definition that defines a password ("travis") to validate protocol packets received by the router:

```
area 0.0.0.1 {
    authtype 1 ;
    networks {
      193.2.1.16 mask 0xfffffff0 ;
      193.2.1.32 mask 0xfffffff0 ;
      } ;
    interface 193.2.1.35 nonbroadcast cost 5 {
      routers {
        193.2.1.33 eligible ;
        193.2.1.46 eligible ;
      } ;
      priority 15 ;
      enable ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
      pollinterval 20 ;
      authkey "travis" ;
    } ;
} ;
```

Cost

The outbound side of each router interface is associated with a configurable cost. Lower cost interfaces are more likely to be used in forwarding data traffic. Cost values are assigned at the discretion of the network or system administrator. While the value is arbitrary, it should be a function of throughput or capacity of the interface: the higher the value, the lower the throughput or capacity. Thus, the interfaces with the highest throughput or capacity should be assigned lower cost values than other interfaces. Interfaces from networks to routers have a cost of 0.

Figure 37 shows an example network where costs have been specified for each interface.

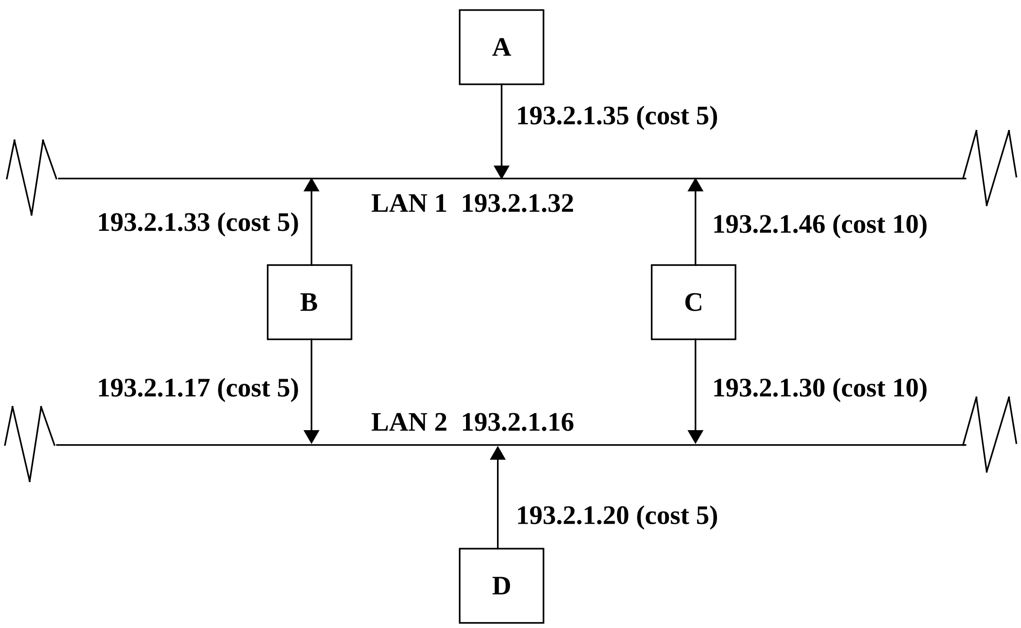


Figure 37 Cost Configuration Example

In Figure 37, there are two possible packet routes between nodes A and D: one route goes through node B and the other route goes through node C. The cost of each route is calculated as follows:

Node A to node B and node B to node D: 5+5 = 10

Node A to node C and node C to node D: 5+10 = 15

The lowest cost OSPF path between nodes A and D is therefore through node B. However, if there were a link failure between node B and LAN 2, packets would be rerouted through node C.

There are other places in the **`/etc/gated.conf`** file where cost can optionally be defined:

- In a **`defaults`** statement in the OSPF protocol configuration, which applies only to AS boundary routers. This cost definition applies to routes to destinations outside the AS. These routes may have been derived from other routing protocols, such as EGP. For more information, see “AS External Routes (AS Boundary Routers Only)” on page 329.
- In the **`export`** statement in the Control class in the **`/etc/gated.conf`** file, which applies only to AS boundary routers. This cost definition applies to routes that are exported from the AS boundary router to routers in other autonomous systems.
- In the stub area definition of the OSPF protocol configuration. This cost definition specifies the cost of the default summary link that is advertised into the area.
- In the **`stubhosts`** definition of the OSPF protocol configuration. This cost definition specifies the cost of a point-to-point interface between the router and a non-OSPF host.
- In the **`subhosts`** definition of the OSPF protocol configuration. This cost definition specifies the cost of a point-to-point interface between the backbone router and a non-OSPF host.

AS External Routes (AS Boundary Routers Only)

AS external (ASE) routes are paths to destinations that are outside the AS. Most ASE routes are routes to specific destinations. ASE routes are learned by AS boundary routers through another routing protocol, such as EGP, or through configured routes. **gated** supports the use of route information from other autonomous systems that use other routing protocols, such as EGP. AS external link advertisements are sent by AS boundary routers and are flooded throughout the AS (with the exception of configured stub areas). A single AS external link advertisement is sent for each external route that the AS boundary router has learned about.

Externally-defined routing information is kept separately from the OSPF routing information. In addition, the externally-defined routing information can be tagged, where the source of the information is identified and stored along with the route information.

Statements in the Control class of the **/etc/gated.conf** file control the importing of routes from routing protocols to a **gated** forwarding table and the exporting of routes from the **gated** forwarding table. See “Importing and Exporting Routes” on page 341.

The **defaults** statements in the OSPF protocol configuration are specified for AS boundary routers only. These statements specify how external routing information is handled by the OSPF protocol. The following can be defined in the **defaults** statements:

- **preference** specifies the preference value given to the ASE routes imported from other autonomous systems. The **preference** value determines the order of routes to the same destination in the routing table. **gated** allows one route to a destination per protocol for each autonomous system. In the case of multiple routes, the route used is determined by the lowest **preference** value. (See “Specifying Route Preference” on page 339.) If a preference value is not specified, ASE routes are imported with a preference of 150.

  Default: 150
  Range: 0 (most preferred) - 255 (least preferred)

- **cost** specifies the cost associated with an OSPF route that is exported to other AS boundary routers.

  Default: 0
  Range: 0 - 65535

- **`tag`** specifies an OSPF tag placed on all routes exported by **`gated`** into OSPF. Each external route can be tagged by the AS boundary router to identify the source of the routing information. The **`tag`** value can be an unsigned 31-bit number. Or, you can specify **`tag`** as ***`as_tag`***, where ***`as_tag`*** is an unsigned 12-bit number that is automatically assigned.

- **`type`** determines how ASE routes imported into OSPF are treated. Type 1 routes should be routes from internal gateway protocols with external metrics that are directly comparable to OSPF metrics. When OSPF is selecting a route, OSPF will use a type 1 route's external metric and add the OSPF internal cost to the AS border router. Type 2 routes should be routes from external gateway protocols with metrics that are not comparable to OSPF metrics. When OSPF is selecting a route, OSPF will ignore a type 2 route's metric and use only the OSPF internal cost to the AS border router.

  Default: 1

- **`exportlimit`** specifies the rate that ASE routes are imported into the **`gated`** routing table for each **`exportinterval`** (see below).

  Default: 100 (ASE routes)
  Range: 0 - 65535

- **`exportinterval`** specifies the interval, in seconds, between ASE exports into OSPF.

  Default: 1 (second)
  Range: 0 - 2147483647

Sample OSPF Configuration

Figure 38 shows an example of two areas. Area 1 is a non-stub area, while area 2 is configured as a stub area. Node B is an area border router between the two areas.

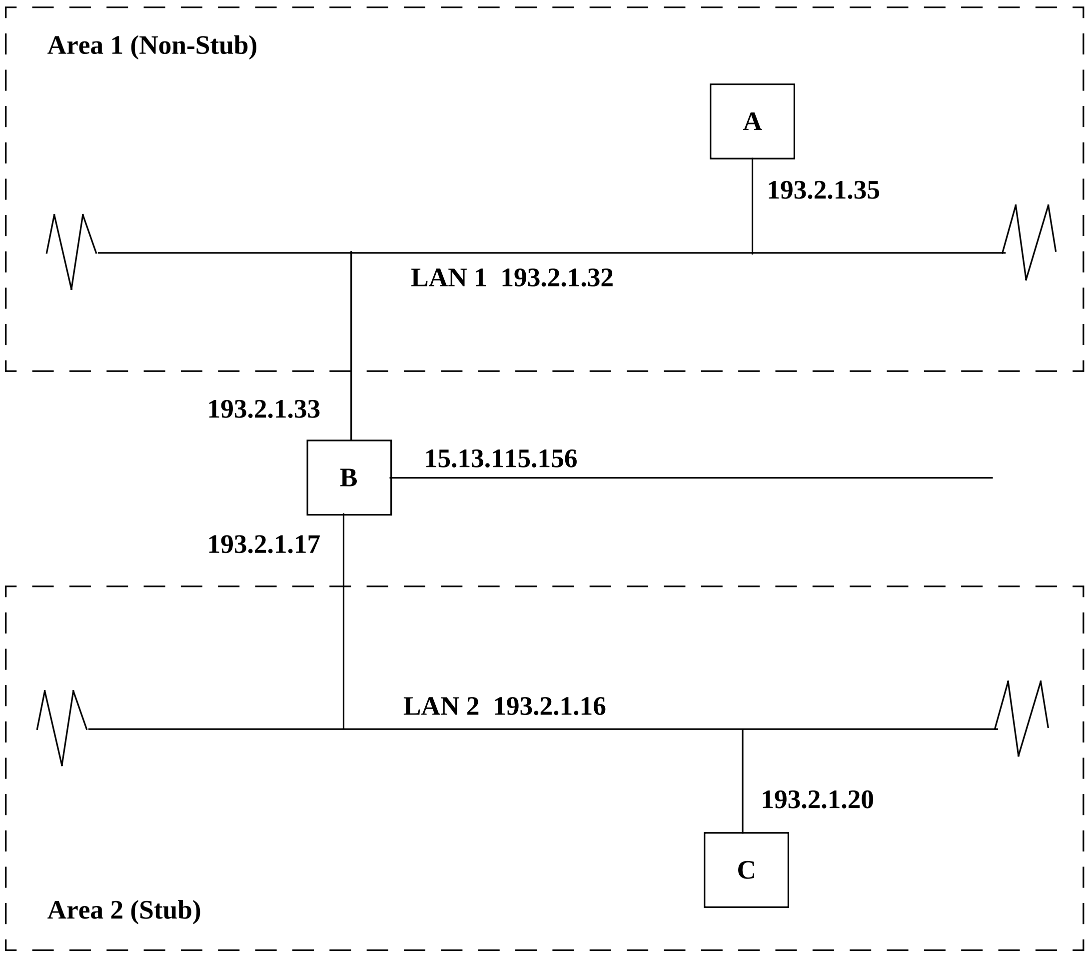


Figure 38 OSPF Sample Configuration

A: Internal Router (Non-Stub Area)

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

```
# Router A Configuration (non-stub area)
OSPF yes {
  area 0.0.0.1 {
    interface 193.2.1.35 cost 5 {
      priority 5 ;
      enable ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
    } ;
  } ;
} ;
```

Note that the configuration shown above is for a multicast interface. For an NBMA interface, the configuration in **/etc/gated.conf** would be set up as follows:

```
# Router A Configuration (non-stub area)
OSPF yes {
  area 0.0.0.1 {
    interface 193.2.1.35 nonbroadcast cost 5 {
      routers {
        193.2.1.33 eligible ;
   } ;
      priority 5 ;
      enable ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
      pollinterval 20 ;
    } ;
  } ;
} ;
```

NOTE: If you use IP multicasting in an area, every router and all intermediate network devices in that area must support IP multicasting.

B: Area Border Router

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

```
OSPF yes {
  defaults {
    cost 5 ;
  } ;
  area 0.0.0.1 {
    networks {
      193.2.1.32 mask 0xfffffff0 ;
    } ;
    interface 193.2.1.33 cost 5 {
      priority 15 ;
      enable ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
    } ;
 };

area 0.0.0.2 {
    networks {
    193.2.1.16 mask 0xfffffff0 ;
    } ;
    interface 193.2.1.17 cost 5 {
      priority 15 ;
      enable ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
    } ;
  } ;

backbone {
  interface 15.13.115.156 cost 5 {
    enable ;
    priority 10 ;
    hellointerval 5 ;
    routerdeadinterval 20 ;
    retransmitinterval 10 ;
  } ;
} ;
} ;
```

C: Internal Router (Stub Area)

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

```
OSPF yes {
  area 0.0.0.2 {
    stub cost 5 ;
    interface 193.2.1.20 cost 5 {
      priority 5 ;
      enable ;
      hellointerval 5 ;
      routerdeadinterval 20 ;
      retransmitinterval 10 ;
    } ;
  } ;
} ;
```

The routing table on node A contains routes to 193.2.1.32 and 193.2.1.16. The routing table on node C in the stub area contains routes to LAN1 only and a default router.

Accessing the OSPF MIB

HP’s **gated** also provides **ospfagt**, an OSPF Simple Management Network Protocol (SNMP) subagent that supports the OPSF MIB (Management Information Base) (see RFC 1253). The **ospfagt** subagent works with the HP SNMP Agent, **snmpdm**. If you are using an SNMP manager utility to manage your network, such as HP’s OpenView Network Node Manager, you may also want to use HP’s OSPF SNMP subagent.

To start **ospfagt** automatically at system bootup, set the environment variable **OSPFMIB** to **1** in the file **/etc/rc.config.d/netdaemons**.

To manually start **ospfagt** , enter:

```
/usr/sbin/ospfagt
```

Note that **gated** must be running before **ospfagt** can be started. Both **gated** and **ospfagt** must be running in order to retrieve OSPF MIB objects.

To load the OSPF MIB, select “Load/Unload SNMP:MIBS ...” from the Options Menu of OpenView.

Customizing Routes

gated maintains a complete routing table in the user space, and keeps the kernel routing table synchronized with this table. This section describes statements for setting up customized routes in the Static class of the **gated** configuration file, **/etc/gated.conf**. These statements can be used to specify default routers, static routes, passive interfaces, and routing metrics for interfaces.

Specifying a Default Router

A static route provides a specific destination for network packets. The static route can be a network address or host address through a router. This route is installed in the kernel's routing table. An example is shown below of a static route for the default route:

```
static {
    default gateway 15.13.114.196 retain ;
    } ;
```

The **retain** qualifier ensures that the entry is not deleted when **gated** exists.

Installing Static Routes

The **`static`** statement specifies a router or an interface in the kernel routing tables. The following is an example of a static route:

```
static {
    193.2.1.32 mask 0xfffffff0 gateway 193.2.1.30
    preference 8 retain;
    } ;
```

If you specify an **`export`** statement for the default route, the route is passed on to other routers. If only the **`static`** statement is specified and not an **`export`** statement, then the default route is not passed on as a route to other routers. This is considered a passive default route and is used only by the host that this **`gated`** is running on. The **`retain`** clause causes the route to be retained in the kernel after **`gated`** is shut down.

Setting Interface States

`gated` times out routes that pass through interfaces that are not receiving any RIP, HELLO, OSPF, BGP, or EGP packets. The **`passive`** clause in the **`interface`** statement in the Static class prevents gated from changing the preference of a route to the interface if routing information is not received for the interface. It is recommended that you use the **`passive`** clause for all interfaces in HP-UX machines.

Specifying Tracing Options

Trace options specify the desired level of tracing output from **gated**. Tracing output provides useful system information for setting up a node on the network. You can specify tracing in the following ways:

- In the Protocol class in the **/etc/gated.conf** configuration file.
- In the Trace class of the **/etc/gated.conf** configuration file.
- On the command line with the **-t** option when starting **gated**.

Trace information is appended to the trace file unless you specify **replace**. Command line options are useful for tracing events in **gated** prior to the reading of the configuration file.

Table 13 shows the valid trace options for **gated.conf** configuration files. Use trace options 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**:

```
traceoptions rip update;
tracefile "/tmp/logfile";
```

Table 13 **Trace Options for gated Configuration Files**

Option	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 options.

To find out what other trace options are available within the configuration file, type **man 4 gated.conf** at the HP-UX prompt. Tracing operations are described in the section "Troubleshooting gated" on page 345.

Specifying Route Preference

gated maintains a routing table that consists of route information learned from OSPF and from other active routing protocols, such as RIP or EGP. You can also configure static routes in the **/etc/gated.conf** file with one or more **static** clauses. (See "Installing Static Routes" on page 336.)

The **gated** routing pool can therefore contain multiple routes to a single destination. Where multiple routes exist, the route chosen by **gated** is determined by the following (in order of precedence):

1. The **preference** value associated with the route. The **preference** value is a number in the range from 0 (most preferred) to 255 (least preferred). Routes from different sources have different default preference values. For example, OSPF routes within a given AS have a preference value of 10. Table 14 shows the default preference values of various types of routes.
2. If multiple routes use the same protocol and have the same preference value, the route with the lowest metric/cost is chosen.
3. If metric/cost is the same, the router with the lowest IP address is chosen.

Table 14 **Default Preference Values of Routes**

Route Type	Preference	**/etc/gated.config** Configuration
Interface routes	0	Can be changed with **interface** statement in Interface class.
OSPF inter- and intra-areas	10	Cannot be changed.
Internal default	20	Generated by BGP or EGP when routing information is learned from a peer.
ICMP Redirect	30	Can be changed with **redirect** statement in Protocol class.
SNMP	50	Can be changed in **SNMP** statement in Protocol class.
Static Routes	60	Can be changed in **static** statement in Static class.
HELLO	90	Can be changed with **import** statement in Control class.
RIP	100	Can be changed with **import** statement in Control class.

Table 14 Default Preference Values of Routes

Route Type	Preference	**/etc/gated.config** Configuration
"Down" interface	120	Can be changed with **interface** statement in Interface class.
OSPF ASE	150	Can be changed in **defaults** statement in OSPF protocol definition and with **import** statement in Control class.
BGP	170	Can be changed with **import** statement in Control class.
EGP	200	Can be changed with **import** statement in Control class.
Kernel remnant	254	These are static routes that have been retained in the kernel after **gated** is stopped. Preference value cannot be configured.

There are several places in the **/etc/gated.conf** file where preference can be defined:

- In the static route definition in the Static class. This preference definition sets the preference for static routes. (See "Customizing Routes" on page 335.) If this option is not set, the preference values for static routes is 60.
- In **interface** statement options in the Interface class. This preference definition sets the preference for routes to this interface. (Type **man 4 gated.conf** at the HP-UX prompt.) If this option is not set, the preference value is 0.
- In a **defaults** statement in the OSPF protocol configuration. This preference definition specifies the preference value of ASE routes that are imported into OSPF. See "AS External Routes (AS Boundary Routers Only)" on page 329. ASE routes are imported into OSPF with a default preference of 150.
- In an **import** statement in the Control class of the **/etc/gated.conf** file. This preference definition overrides any preference defined in the **defaults** section of the OSPF protocol configuration. See "AS External Routes (AS Boundary Routers Only)" on page 329 and "Importing and Exporting Routes" on page 341.

Importing and Exporting Routes

The **import** and **export** control statements allow you to propagate routes from one routing protocol to another. Routes are imported into a **gated** forwarding table and exported out to the routing protocols.

Type **man 4 gated.conf** for more information on **import** and **export** statements.

import Statements

import statements restrict or control how routes are imported to the **gated** forwarding table. Once routes are imported to the **gated** forwarding table, they can be exported to the routing protocols. You can use import statements to do the following:

- Prevent routes from being imported into the **gated** forwarding table by using a **restrict** clause.
- Assign a preference value to use when comparing a route to other routes from other protocols. The route with the lowest preference available at any given route is installed in the **gated** forwarding table. The default preferences are configured by the individual protocols.

The format of **import** statements varies depending on the protocol from which you are importing routes.

With OSPF, you can apply **import** statements only to OSPF ASE routes. All OSPF intra-area and inter-area routes are imported into the **gated** forwarding table and with an assigned preference of 10.

export Statements

export statements determine which routes are exported from the **gated** forwarding table to the routing protocols. You can also restrict which routes are exported and assign metrics (values used for route selection) to be applied to the routes after they have been exported.

The format of the export statement varies according to the protocol to which you are exporting routes and the original protocol used to build the routes you are exporting.

Examples of `import` and `export` Statements

The following **`import`** statement imports an EGP route for network 195.1.1 to the **`gated`** forwarding table with a preference of 15:

```
import  proto  egp as 1 {
        195.1.1 mask 0xffffff00 preference 15 ;
        } ;
```

The following **`export`** statement exports to OSPF the ASE route that was imported to the **`gated`** forwarding table in the example above. The route was originally built by EGP and the destination of the route is network 195.1.1.

```
export proto ospfase type 1 {   /* Export an ASE route to OSPF */
        proto egp as 1 {         /* route came from EGP and AS 1 */
                        195.1.1 ; /* the route is to network 195.1.1 */
        } ;
} ;
```

Starting gated

1 Set the environment variable **GATED** to **1** in the file **/etc/rc.config.d/netconf**. This causes **gated** to start automatically whenever the system is booted.

2 Reboot your system, or issue the following command to run the **gated** startup script:

```
/sbin/init.d/gated start
```

Command line arguments for starting **gated** may be specified with the **GATED_ARGS** environment variable in the file **/etc/rc.config.d/netconf**. Table 15 lists the commonly used command line options for **gated**.

Table 15 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 **internal**, **external**, and **route** trace options automatically. When **-t** is followed by one or more trace options, only those options are turned on. (See “Specifying Tracing Options” on page 337.) Multiple trace options are separated by commas. The **-t** flag always must immediately precede the other flags.
-C	Specifies that the configuration file will be parsed for syntax errors, then **gated** will exit.
-c	Specifies that the configuration file will be parsed for syntax errors, and then **gated** will exit. A dump file is created if there are no errors. Trace options **general**, **kernel** and **nostamp** only are logged.
-n	Specifies that **gated** will not modify the kernel’s routing tables.

For more information about the options that you can specify on the command line, type **man 1M gated** at the HP-UX prompt.

To Find Out if gated is Running

Issue the following command to find out if **gated** is running:

```
/usr/bin/ps -ef | /usr/bin/grep gated
```

This command reports the process identification (PID), current time, and the command invoked (**gated**). An example output is shown below:

```
daemon   4484     1    0  Feb 18      ?      0:00 gated
user     3691  2396    2  15:08:45    ttyp2  0:00 grep 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**.

Checking for Syntax Errors in the Configuration File

After creating or modifying a **gated** configuration file, you should start **gated** from the command line with the **-C** option. This option causes the configuration file to be parsed for syntax errors.

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 the **gated -t** command line option or with the **traceoptions** statement in the **/etc/gated.conf** file. Using any of the following combinations, you can determine where the tracing output is printed and whether tracing is even done:

- If you specify trace options and a trace file, tracing output is printed to the log file.
- If you specify trace options but do not specify a trace file, tracing output is printed on the display where **gated** was started.
- If you specify a trace file but do not specify any trace options, no tracing takes place.

Once tracing is started to a file, the trace file can be rotated. Receipt of a **SIGUSR1** signal causes **gated** to stop tracing and closes the trace file. The trace file can then be moved out of the way. To send a **SIGUSR1** signal to **gated**, issue one of the following commands:

```
/usr/bin/kill -SIGUSR1 pid
```

or

```
/usr/bin/kill -USR1 pid
```

where ***pid*** is **gated**'s process ID, determined by invoking the command **ps -ef | grep gated** or **cat /var/run/gated.pid**.

A subsequent **SIGUSR1** signal starts tracing again to the same trace file. If the trace options are changed before tracing is started up again, the new options will take effect.

NOTE: You cannot use the **SIGUSR1** signal if tracing to a file has not previously been specified when starting **gated**.

gated Routing Table

Sending **gated** a **SIGINT** signal causes **gated** to write out its information in **/var/tmp/gated_dump**. The information includes the interface configurations, tasks information for each protocol, and the routing tables.

ripquery

/usr/sbin/ripquery is a support tool that can be used to query **gated** for RIP routing information. **ripquery** sends out two types of commands: a **POLL** command or a RIP **request** command. **gated** responds to a **POLL** command by listing all routes learned from RIP that are in its routing table. This does not include the interface routes for the local network or routes from other protocols that are announced via RIP. When **gated** receives a RIP **request** command, it announces routes via RIP on that interface. This includes routes from other protocols that are being imported by **gated** on the node.

You can use **ripquery** to query other non-**gated** RIP routers. To do so, you may need to use the **-p** option. This option causes **ripquery** to initially send **POLL** commands and then, if there is no response, send RIP request commands. The default query (**POLL** commands) sent by **ripquery** may not be supported by all RIP routers. Type **man 1M ripquery** at the HP-UX prompt for more information.

ospf_monitor

/usr/sbin/ospf_monitor is a support tool that can be used to query OSPF routers for information on OSPF routing tables, interfaces and neighbors, as well as data on AS external databases, link-state databases, error logs, and input/output statistics. Running the **ospf_monitor** command displays a prompt that allows you to enter interactive commands. See the **ospf_monitor** man page for details on using this tool.

Common Problems

This section covers typical problems with **gated** operation.

Problem 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. The following shows two sample configuration files, along with the trace files generated by **gated**. The node used has three interfaces: **lan0**, **lan1**, and **lan2**. In the configuration files, **lan0**, **lan1**, and **lan3** are specified. In the first configuration shown, the **strictintfs** option has been specified for the interfaces, so **gated** exits when the error is detected.

Interface Configuration with strictintfs Option Specified The following configuration references a non-existent interface. The line **options strictintfs** in the **interfaces** statement means that all configured interfaces must be present before **gated** starts.

```
tracefile "tt"  ;
traceoptions parse nostamp;
interfaces {
   options strictintfs ;
   interface lan0 lan1 lan3 passive ;
} ;
rip yes ;
```

The following is the tracing output that is produced when **gated** is started with this configuration:

```
Tracing flags enabled:  parse nostamp
parse_new_state: cc:3 old initial new interface
parse: cc:5 INTERFACE: lan0*
parse: cc:5 INTERFACE: lan1*
parse_addr_hostname: resolving lan3

parse: cc:5 Interface not found at 'lan3'

parse_new_state: cc:8 old interface new protocol

parse_parse: 2 parse errors
```

Interface Configuration without strictintfs Option Specified The following configuration references a non-existent interface, but does not include the **strictintfs** option.

```
tracefile "tt"  ;
traceoptions parse nostamp;
interfaces {
   interface lan0 lan1 lan3 passive ;
} ;
rip yes ;
```

The following is the tracing output that is produced when **gated** is started with this configuration:

```
Tracing flags enabled:  parse nostamp

parse_new_state: cc:3 old initial new interface
parse: cc:4 INTERFACE: lan0*
parse: cc:4 INTERFACE: lan1*
parse_addr_hostname: resolving lan3
parse: cc:4 INTERFACE: lan3*
parse_new_state: cc:7 old interface new protocol
inet_routerid_notify: Router ID: 15.13.115.156

***Routes are not being installed in kernel
```

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 in the interface policy statement at the bottom of the example.

```
Interfaces:

lan0   Index 4 Address 802.2 8:0:9:10:3c:al Change: <> State: <Up Broadcast Multicast>
       Refcount: 2   Up-down transitions: 0

       15.13.115.156
              Metric: 0      MTU: 1436
              Refcount: 5    Preference: 0   Down: 120
              Change: <>     Scale: <Up Broadcast Multicast Subnet Noage>
              Broadcast Address:  15.13.119.255
              Net Number:   15
              Net Mask:    255
              Subnet Number: 15.13.112      Subnet Mask: 255.255.248
              Routing protocols active: RIP
              proto:  INET   State: <AllRouters>

lan1   Index 5 Address 802.2 8:0:9:2:dd:bb  Change: <>  State: <Up Broadcast Multicast>
       Refcount: 2    Up-down transitions: 0

       193.2.1.35
              Metric: 0      MTU: 1436
              Refcount: 5   Preference: 0   Down: 120
              Change: <>     Scale: <Up Broadcast Multicast Subnet Noage>
              Broadcast Address:   193.2.1.47
              Net Number:    193.2.1
              Net Mask:    255.255.255
              Subnet Number: 193.2.1.32      Subnet Mask: 255.255.255.240
              Routing protocols active: RIP
              proto:  INET   State: <AllRouters>

lan2   Index 6 Address 802.2 8:0:9:10:d3:94 Change: <> State: <Up Broadcast Multicast>
       Refcount: 2   Up-down transitions: 0

       194.1.1.1
              Metric: 0      MTU: 1436
              Refcount: 5    Preference: 0   Down: 120
              Change: <>     Scale: <Up Broadcast Multicast>
              Broadcast Address:   194.1.1.255
              Net Number:    194.1.1
              Net Mask:    255.255.255
              Subnet Mask: 255.255.255
              Routing protocols active: RIP
              proto:  INET   State: <AllRouters>
Interface policy:
       Interface Lan0 lan1 lan3 passive
```

Note that the state recorded in lan2 does not contain the “NoAge” flag because the interface was not set to “passive” in the interface policy statement.

A common mistake is to expect **gated** to always send out RIP packets when you specify **rip yes** in a configuration file. **gated** will be an active RIP participant only if the host is a router (the host has more than one network interface).

Problem 2: gated deletes routes from the routing table

gated maintains a complete routing table in user space, and keeps the kernel routing table synchronized with its table. When **gated** starts, it reads the entries in the kernel routing table. However, if **gated** does not get confirmation from its routing protocols (RIP, OSPF, etc.) about a route, it will delete the route from its tables and the kernel routing table.

It is common to see **gated** delete the default route that many people configure in the **/etc/rc.config.d/netconf** file. To solve this problem, configure a static default route as described in the section "Installing Static Routes" on page 336.

Another common scenario occurs in networks where not all gateways implement the **gated** routing protocols. In this situation, routes that do not use **gated** gateways will not be confirmed by **gated** and **gated** will delete them unless a static statement is included in **/etc/gated.conf**.

```
static {
    13.0.0.0 mask 0xff000000 gateway 15.14.14.14 ;
};
```

The static entry in the above example ensures that the local system will include a route to network 13.0.0.0 even though the gateway to that network (15.14.14.14) is not running any of the **gated** protocols.

You may want to put **restrict** clauses in the **export** statements to keep these extra routes from being advertised.

Problem 3: gated adds routes that appear to be incorrect.

Start by looking at the routing table maintained by **gated**. Send **gated** a **SIGINT**, and look at the information output in **/var/tmp/gated_dump**. Look for the entry of the route in question. The entry shows the protocol that this route was heard over and the first-hop router. The first-hop router is likely to be the immediate source of the information.

If the route was learned over RIP, use **/usr/sbin/ripquery** to query the first-hop router for the route. That router may claim to have heard the route from a router further on. If the first-hop router 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.

Problem 4: 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.

Migrating from gated 2.0 to 3.0

The format of the configuration file used by this version of **gated** is not compatible with the format of previous versions. A conversion utility is provided to convert existing configuration files to the version 3.0 format.

To run the conversion utility, enter the following command:

```
/usr/examples/gated/conv_config oldconfig > newconfig
```

The file **/var/run/gated.pid** is used as a lock file by **gated** to prevent multiple copies of **gated** from being started. Before restarting **gated**, make sure that **gated** is not running. To stop **gated**, use the following command:

```
/sbin/init.d/gated stop
```

NOTE: In **gated** 3.0, the RIP protocol sends out RIP version 1 packets. If you want the RIP protocol to only send RIP version 2 packets, specify **version 2** in the configuration file for the RIP protocol.

10

Configuring mrouted

`mrouted`, (pronounced “M route D”), is a routing daemon that forwards IP multicast datagrams, within an autonomous network, through routers that support IP multicast addressing. The routing protocol implemented by

mrouted is the Distance-Vector Multicast Routing Protocol (DVMRP). The ultimate destination of multicast datagrams are host systems that are members of one or more multicast groups.

Multicasting enables one-to-many and many-to-many communication among hosts and is used extensively in networking applications such as audio and video teleconferencing where multiple hosts need to communicate simultaneously.

This chapter contains information about how to configure and use version 3.3 of **mrouted**. It includes the following sections:

- Overview of Multicasting
- Configuring **mrouted**
- Starting **mrouted**
- Verifying **mrouted** Operation
- Displaying **mrouted** Routing Tables
- Multicast Routing Support Tools
- Sources for Additional Information

You cannot use SAM to configure **mrouted**.

Note that **mrouted** is supported only over certain network interfaces, such as EISA Ethernet (lan2) and EISA FDDI (from a provider other than Hewlett-Packard), and that the types of interfaces will vary depending on the system platform.

mrouted is installed as part of the Internet Services software. For more information about installing **mrouted**, see “Installing the Internet Services Software” on page 28.

For additional information on **mrouted**, type **man 1m mrouted** at the HP-UX prompt.

Overview of Multicasting

DVMRP

mrouted implements the Distance-Vector Multicast Routing Protocol (**DVMRP**). DVMRP is an Interior Gateway Protocol (IGP) used for routing multicast datagrams within an autonomous network. The primary purpose of DVMRP is to maintain the shortest return paths to the source of the multicast datagrams. This is accomplished by using topological knowledge of the network to implement a multicast forwarding algorithm called Truncated Reverse Path Broadcasting (TRPB).

mrouted structures routing information in the form of a **pruned** broadcast delivery tree which contains only routing information to those subnets which have identified themselves as having members of the destination multicast group. In other words, each router determines which of its virtual network interfaces are in the shortest path tree. In this way, DVMRP can intelligently decide if an IP multicast datagram needs to be forwarded. Without such a feature, the network bandwidth can easily be saturated through forwarding of unnecessary datagrams.

Since DVMRP routes only multicast datagrams, routing of unicast or broadcast datagrams must be handled using a separate routing process.

To support multicasting across subnets that do not support IP multicasting, DVMRP provides a mechanism called **tunnelling**. Tunnelling forms a point-to-point link between pairs of **mrouted** routers by encapsulating the multicast IP datagram within a standard IP unicast datagram using the IP-in-IP protocol (IP protocol number 4). This unicast datagram, containing the multicast datagram, is then routed through the intervening routers and subnets. When the unicast datagram reaches the tunnel destination, which is another **mrouted** router, the unicast datagram is stripped away and the **mrouted** daemon forwards the multicast datagram to its destination(s).

The following figure shows a tunnel formed between a pair of **mrouted** routers. In this figure, **mrouted** router R1 receives a multicast packet from node M. Since R1 is configured as one end of a tunnel, R1 encapsulates the IP multicast packet in a standard unicast IP packet addressed to **mrouted** router R2. The packet, now treated as a normal IP packet, is sent through the intervening, non-multicast network to R2. R2 receives the packet and removes the outer IP header, thereby restoring the original multicast packet. R2 then forwards the multicast packet through its network interface to node N.

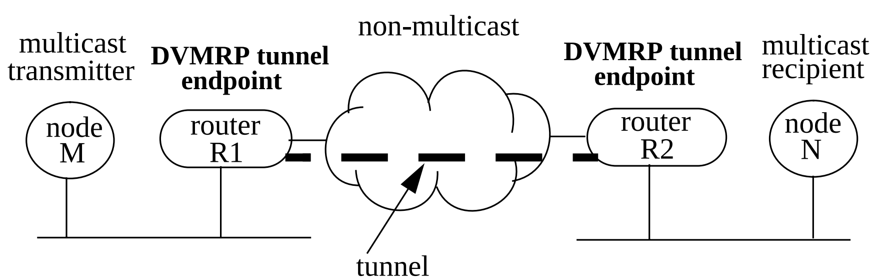


Figure 39 **Tunnel Made with mrouted Routers**

IP Multicast Addresses

IP internet address are 32 bit addresses. Each host on the internet is assigned a unique IP address. There are four classes of IP addresses identified as Class A, Class B, Class C, and Class D. Class D IP addresses are identified as IP multicast addresses. Class A, Class B, and Class C IP addresses are composed of two parts, a **netid** (network ID) and a **hostid** (host ID). Class D IP addresses are structured differently and are of the form:

Figure 40 **Class D IP multicast address format**

Bits 0 through 3 identify the address as a multicast address. Bits 4 through 31 identify the **multicast group**. Multicast addresses are in the range 224.0.0.0 through 239.255.255.255. Addresses 224.0.0.0 through 224.0.0.255 are reserved, and address 224.0.0.1 is permanently assigned to the **all hosts group**. The **all hosts group** is used to reach, on a local network, all hosts that participate in IP multicast. The addresses of other well-known permanent multicast groups are published in the “Assigned Numbers” RFC (RFC-1060, March 1990).

IP multicast addresses can only be used as destination addresses and should never appear in the source address field of a datagram. It should also be noted that ICMP (Internet Control Message Protocol) error messages are not generated for multicast datagrams.

Since IP internet addressing is a software manifestation of the underlying physical network, IP addresses must be mapped to physical addresses that are understood by the hardware comprising the network. As such, IP multicast addresses are mapped to 802.3/Ethernet multicast addresses. The IP multicasting addressing scheme, like that of Ethernet’s, uses the datagram’s destination address to indicate multicast delivery.

When mapping an IP multicast address to an Ethernet multicast address, the low-order 23 bits of the IP multicast address are placed into the low-order 23 bits of the special Ethernet multicast address. The hexadecimal value of the special Ethernet multicast address is 01-00-5E-00-00-00. The resulting Ethernet address, however, is not unique since only 23 of the 28 bits representing the multicast address are used.

Multicast Groups

A **multicast group** is comprised of hosts that have indicated their intent to join the multicast group by listening to the same IP multicast address. Group membership is dynamic in that a host may join or leave a group at any time. A host may be a member of one or more groups simultaneously. Additionally, a host is allowed to send multicast datagrams to a group without being a member of the group.

Multicast addresses are often temporary in that they are assigned to transient groups, such as when users run an application that dynamically registers to specific multicast addresses, and are then discarded when all members of the group have left. Some multicast addresses may be well-known addresses assigned to permanent groups that always exist, even when their membership is empty.

Both hosts and **`mrouted`** routers that participate in IP multicast use the Internet Group Management Protocol (IGMP) to communicate multicast group information among themselves. Hosts use IGMP to inform **`mrouted`** routers that they are joining a group. **`mrouted`** routers use IGMP to pass multicast routing information to other **`mrouted`** routers as well as to poll the hosts to determine whether the host is still an active group member.

IGMP uses IP datagrams to carry information and is a TCP/IP standard that must be present on all systems that participate in IP multicast. While IGMP defines a standard for communicating information, it does not define a standard for how the multicast information is propagated among multicast routers. Consequently, DVMRP enables multicast routers to efficiently communicate group membership information among themselves. DVMRP uses IGMP messages to carry routing and group membership information. DVMRP also defines IGMP message types that enable hosts to join and leave multicast groups and that allows multicast routers to query one another for routing information.

Configuring mrouted

When the **mrouted** daemon is started, it automatically reads the default ASCII text configuration file **/etc/mrouted.conf**. You can override the default configuration file by specifying an alternate file when invoking **mrouted**; refer to the section “Starting mrouted”. If **mrouted.conf** is changed while **mrouted** is running, you can issue the HP-UX command **kill -HUP** to signal **mrouted** to reread the configuration file.

By default, **mrouted** automatically configures itself to forward on all multicast-capable interfaces, excluding the loopback “interface”, that have the **IFF_MULTICAST** flag set. Therefore, it is not necessary to explicitly configure **mrouted**, (that is, the **mrouted.conf** file need not exist) unless you need to configure tunnel links, change the default operating parameters, or disable multicast routing over a specific physical interface.

Configuration File Commands

This section describes the statements that can be defined in the **/etc/mrouted.conf** configuration file. **mrouted** supports four configuration commands: **phyint**, **tunnel**, **cache_lifetime**, and **pruning**. Associated with each command are one or more options.

The syntax of the commands are:

```
phyint local-addr [disable] [metric m] [threshold t] [rate_limit b]
       [boundary scoped-addr/mask-len]
```

```
tunnel local-addr remote-addr [metric m] [threshold t] [srcrt] [rate_limit b]
       [boundary scoped-addr/mask-len]
```

```
cache_lifetime ct
```

```
pruning off/on
```

The **phyint** command can be used to disable multicast routing on the physical interface identified by local IP address ***local-addr*** (see figure below), or to associate a non-default **metric** or **threshold** with the specified physical interface. The local IP address ***local-addr*** may be alternatively replaced by the interface name, such as lan0.

The **tunnel** command can be used to establish a tunnel link between local IP address ***local-addr*** and remote IP address ***remote-addr***, (see figure below). It can also be used to associate a non-default **metric** or **threshold** value with that tunnel. Before a tunnel can be used, the tunnel must be set up in the **mrouted** configuration files of both **mrouted** routers participating in the tunnel. The **srcrt** option provides backwards compatibility with older versions of **mrouted** that implemented IP multicast datagram encapsulation using IP source routing.

NOTE: Any **phyint** commands *must* precede any **tunnel** commands. All the **phyint** and **tunnel** command options *must* be placed on a single line except for the **boundary** option which may begin on a separate line.

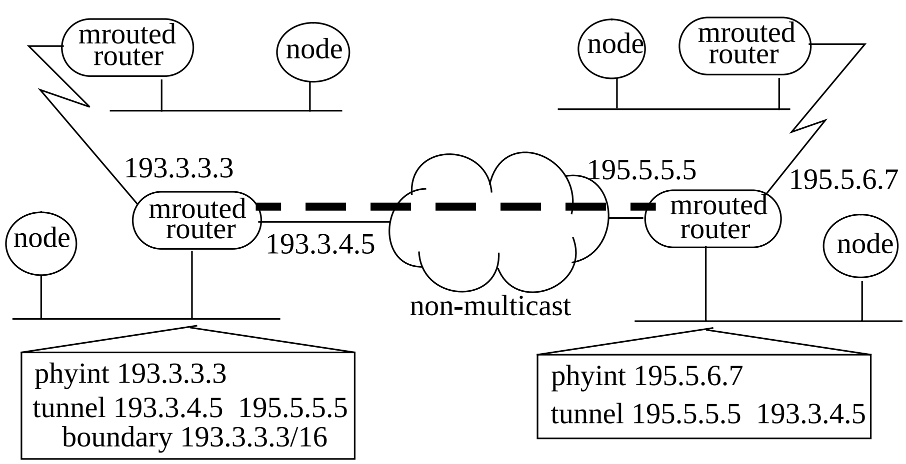


Figure 41 **Multicast Network Example Configuration**

The **cache_lifetime** value determines the amount of time that a cached multicast route remains in the kernel before timing out. This value is specified in seconds and should be between 300 (5 minutes) and 86400 (24 hours). The default value is 300.

The **pruning off** command explicitly configures **mrouted** to act as a "non-pruning" router. When pruning is **off**, IP multicast datagrams are forwarded to leaf subnets of the broadcast routing tree even when those leaf subnets do not contain members of the multicast destination group. Non-pruning mode should only be used for testing. The default mode for pruning is **on**.

The **metric** is the cost, or overhead, associated with sending a datagram on the given interface or tunnel and is used primarily to influence the choice of routes over which the datagram is forwarded; the larger the value, the higher the cost. Metrics should be kept as small as possible since **mrouted** cannot route along paths with a sum of metrics greater than 31. In general, you should use a metric value of 1 for all links unless you are specifically attempting to force traffic to take another route. In this case, the metric of the alternate path should be the sum of the metrics on the primary path + 1. The default value is 1.

The **threshold** is the minimum IP time-to-live (TTL) required for a multicast datagram to be forwarded to the given interface or tunnel. It controls the scope of multicast datagrams. If the TTL value in the datagram is less than the threshold value, the datagram is dropped; if the TTL is greater than or equal to the threshold, the packet is forwarded. The default threshold is 1.

The TTL of forwarded packets is only compared to the threshold, it is not decremented by the threshold. The TTL is set by the application that initiates the IP multicast datagram and typically represents the number of subnets, or hops, that the datagram will need to traverse to reach its destination. Every multicast router decrements the TTL by 1. It is recommended that you use the default threshold value unless you have a specific need to set it otherwise.

In general, all interfaces connected to a particular subnet or tunnel should use the same **metric** and **threshold** values for that subnet or tunnel.

The **rate_limit** option allows the network administrator to specify a certain bandwidth in Kbits/second which would be allocated to multicast traffic.

The **boundary** option allows an interface to be configured as an administrative boundary for the specified ***scoped-addr*** (scoped address). More than one boundary option may be specified in **phyint** and **tunnel** commands. Packets belonging to the scoped address, which is an IP multicast group address, will not be forwarded on this interface. ***mask-len*** indicates the number of leading 1's in the mask applied (that is, bitwise logically ANDed) to the scoped address. For example, the statement **boundary 239.2.3.3/16** would result in the mask 255.255.0.0 being logically ANDed with 239.2.3.3 to isolate the first two octets, 239.2, of the scoped address. Therefore, all IP multicast addresses beginning with 239.2 will not be forwarded on the specified interface.

The primary use of the boundary option is to allow concurrent use of the same IP multicast address(s) on downstream subnets without interfering with multicast broadcasts using the same IP multicast address(s) on subnets that are upstream from the **mrouted** gateway.

mrouted will terminate execution if it has less than two enabled virtual interfaces (vifs), where a vif is either a physical multicast-capable interface or a tunnel.

Starting mrouted

mrouted is started from the HP-UX prompt or from within a shell script by issuing the following command:

```
/etc/mrouted [-p] [-c config_file] [-d debug_level]
```

The **-p** option disables pruning by overriding a **pruning on** statement within the **/etc/mrouted.conf** configuration file. This option should only be used for testing.

The **-c** option overrides the default configuration file **/etc/mrouted.conf**. Use ***config_file*** to specify the alternate configuration file.

The **-d *debug_level*** option specifies the debug level. ***debug_level*** can be in the range 0 to 3. Refer to the “Invocation” section of the **mrouted (1m)** man pages for an explanation of the ***debug_level*** values.

Regardless of the debug level, **mrouted** always writes warning and error messages to the system log daemon. These messages can be retrieved from the system log file, **syslog.log**, usually located in the directory**/var/adm/syslog**.

Verifying mrouted Operation

You can use one or more of the following methods to verify that **`mrouted`** is operating.

- retrieve the **Virtual Interface Table** and the **Multicast Routing Table** to see if the correct virtual interfaces (vifs) are configured. Refer to the section "Displaying mrouted Routing Tables", within this chapter, for information on retrieving these tables.
- retrieve the **Routing Cache Table** to see if the routing and cache information is appropriate for your configuration of **`mrouted`**. Refer to the section "Displaying mrouted Routing Tables", within this chapter, for information on retrieving this table.
- look at the syslog file **`/var/adm/syslog/syslog.log`** and check for warning and error messages that indicate the status of **`mrouted`**. When **`mrouted`** starts, it logs a startup message that indicates the **`mrouted`** version number, such as "mrouted version 3.3".
- issue the HP-UX **`ps`** (process status) command and search, using **`grep`**, for the string "mrouted" to determine if the **`mrouted`** program is running. You can issue the **`ps`** command as follows:

```
ps -ef | grep mrouted
```

Displaying mrouted Routing Tables

There are three routing tables associated with **`mrouted`**. They are the **Virtual Interface Table**, the **Multicast Routing Table**, and the **Multicast Routing Cache Table**.

The Virtual Interface Table displays topological information for both physical and tunnel interfaces, the number of incoming and outgoing packets at each interface, and the value of specific configuration parameters, such as **`metric`** and **`threshold`**, for each virtual interface (vif).

The Multicast Routing Table displays connectivity information for each subnet from which a multicast datagram can originate.

The Multicast Routing Cache Table maintained by **`mrouted`** is a copy of the kernel forwarding cache table. It contains status information for multicast destination group-origin subnet pairs.

These tables are retrieved by sending the appropriate signal to the **mrouted** daemon. For retrieving routing tables, **mrouted** responds to the following signals:

USR1 defined as signal 16, dumps the internal routing tables (Virtual Interface Table and Multicast Routing Table) to **`/var/tmp/mrouted.dump`**.

USR2 defined as signal 17, dumps the Multicast Routing Cache Tables to /**`var/tmp/mrouted.cache`**.

QUIT dumps the internal routing tables (Virtual Interface Table and Multicast Routing Table) to **`stderr`** (only if **`mrouted`** was invoked with a non-zero debug level).

Signals can be sent to **`mrouted`** by issuing the HP-UX **`kill`** command at the HP-UX prompt. For example:

```
kill -USR1 pid
```

where *pid* is the process ID of the **`mrouted`** daemon.

Refer to the “Example” section of the **`mrouted (1m) man`** pages for an explanation of the contents of the **`mrouted`** routing tables.

Refer to the “Signals” section of the **`mrouted (1m) man`** pages for additional information about other signals to which **`mrouted`** responds.

Multicast Routing Support Tools

mrinfo

mrinfo is a multicast routing tool that requests configuration information from **mrouted** and prints the information to standard out. By default, configuration information for the local instance of **mrouted** is returned. You can override the default request to the local instance of **mrouted** by specifying an alternate router IP address or system name.

Type **man 1m mrinfo** for additional information on using **mrinfo**.

map-mbone

map-mbone is a multicast routing tool that requests multicast router connection information from **mrouted** and prints the "connection map" information to standard out. By default (no alternate router address specified), the request message is sent to all the multicast routers on the local network. If **map-mbone** discovers new neighbor routers from the replies it receives, it sends an identical request to those routers. This process continues until the list of new neighbors has been exhausted.

Type **man 1m map-mbone** for additional information on using **map-mbone**.

netstat

netstat is a tool that can be used to display multicast related information including network statistics and multicast routing table contents.

Type **man 1m netstat** for additional information on using **netstat**.

Sources for Additional Information

RFC documents

Additional information pertaining to **mrouted** and IP multicast routing can be obtained from the following RFC (Request for Comment) documents. Refer to the section “Military Standards and Request for Comment Documents” within chapter 1 of this manual for information on accessing these documents.

- RFC 1075: “Distance-Vector Multicast Routing Protocol”

  This RFC has been obsoleted and has no successor. Therefore, it is no longer a precise specification of the DVMRP implementation obtainable in the public domain or provided by Hewlett-Packard.

- RFC 1112: “Host Extensions for IP Multicasting

Other Documents

The following sources of information neither originated at Hewlett-Packard nor are maintained by Hewlett-Packard. As such, their content and availability are subject to change without notice.

- The MBONE FAQ

  The MBONE (Multicast Backbone) is a virtual network implemented on top of the physical Internet. It supports routing of IP multicast packets. It originated as a cooperative, volunteer effort to support experimentation in audio and video teleconferencing over the Internet.

  This document can be retrieved, via **ftp**, at: **isi.edu**. The location of this file is **/mbone/faq.txt**.

  An HTML-formatted version of the MBONE FAQ can be found at: **http://www.research.att.com/mbone-faq.html**.

11

Using rdist

This chapter contains information about how to use **rdist**, a program that distributes and maintains identical copies of files across multiple network hosts. System administrators can use **rdist** to install new or updated software on all machines in a network. This chapter includes the following sections:

- Overview
- Setting Up remsh
- Creating the Distfile
- Starting rdist
- Troubleshooting rdist

Overview

To use **rdist**, one system in the network is designated as the **master host**. The master host contains the master copy of source files that are distributed to remote hosts.

rdist software is installed as part of the operating system. It must reside in the **/usr/bin** directory on the master host and on the remote hosts that are to be updated. It must be owned by root and must have its access permissions set to **rwsr-xr-x**. The **rdist** process on the master host starts an **rdist** process on each remote host.

rdist uses **remsh** as the mechanism for distributing files over the network. In order to use **rdist**, you must set up **remsh** on each of the remote hosts. See “Setting Up remsh” on page 375.

A file on a remote host is updated if the size or modification time of the file differs from the master copy. Programs that are being executed on the remote host can be updated. The owner, group, mode, and modification time of the files on the master host are preserved on the remote host, if possible.

By default, the list of files updated on each remote host is printed to standard output on the master host. You can mail the list of updated files for a particular remote host to a specified mail recipient.

Figure 42 shows the distribution of source files **filea1**, **filea2**, and **filea3** from master host A to remote hosts B and C.

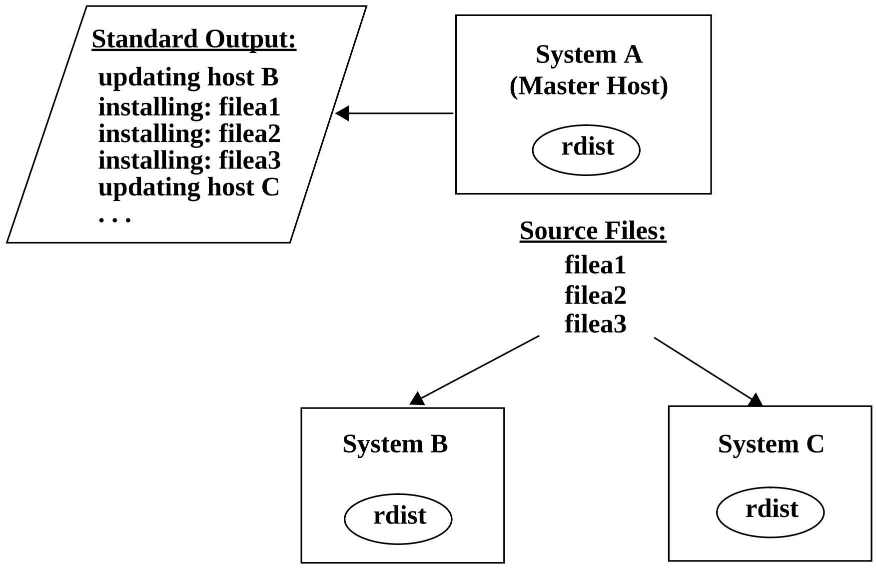


Figure 42 **Distributing Files with rdist**

Note that the **`rdist`** process does not prompt for passwords. The user on the master host who starts **`rdist`** (usually a system or network administrator) must have an account on the remote host and must be allowed remote command execution. (The working directory on the remote host is the user's home directory.) Or, you can specify a user name on a remote host for **`rdist`** to use that has the appropriate permissions for accessing files on that remote host. This is described in "Creating the Distfile" on page 376.

`rdist` on the master host reads commands from a **distfile**, an ASCII file that specifies the files or directories to be copied, the remote hosts to be updated, and the operations to be performed for the update. A distfile can be specified when invoking **`rdist`** on the master host. Otherwise, **`rdist`** looks in the current working directory for a file named **`distfile`** to use as input; if **`distfile`** does not exist in the current working directory, then **`rdist`** looks for **`Distfile`**.

The next section describes the contents of the distfile.

Setting Up `remsh`

`rdist` uses **`remsh`** as the mechanism for distributing files over the network. In order to use **`rdist`**, you must set up **`remsh`** on each of the remote hosts. Follow these steps:

1. On each of the remote hosts, create an entry for the master host in the **`$HOME/.rhosts`** file of the user who will run **`rdist`**. For example, if **`rdist`** will always be run by user root, create an entry for the master host in root's **`.rhosts`** file (**`/.rhosts`**) on each of the remote hosts.
2. On each of the remote hosts, make sure following line is uncommented in the **`/etc/inetd.conf`** file. (Make sure it is not preceded by #.)

```
shell stream tcp nowait root /usr/lbin/remshd remshd
```

3. On each of the remote hosts, issue the following command to force **`inetd`** to reread its configuration file:

```
/usr/sbin/inetd -c
```

Creating the Distfile

The distfile used by the master host contains a sequence of entries that specify the files to be copied, the destination hosts, and the operations to be performed to do the updating. Since a distfile is an ASCII file, you can create it with any text editor. If you are familiar with the **`make`** program, the structure of a distfile is somewhat similar to a makefile.

The following syntax rules apply:

- Newlines, tabs, and blanks are used as separators and are ignored.
- Comments begin with "#" and end with a newline.
- Shell meta characters ([,], {, }, *, and ?) are expanded on the master host in the same way as with the **`csh`** command. Use a backslash (\) to escape a meta character. (Type **`man 1 csh`** for more information.)
- File names that do not begin with "/" or "~" are assumed to be relative to the user's home directory on each remote host.

A distfile contains the following types of entries:

- Definitions of variables to be used with distfile commands.
- Commands that distribute files to other hosts.
- Commands to create lists of files that have been changed since a specified date.

Each of these types of entries is described in the following sections.

Variable Definitions

Variables can be used to represent a list of items, such as the names of files to be distributed or the remote hosts to be updated. Variables can be defined anywhere in the distfile, but they are usually grouped together at the beginning of the file. Variables are then used in command entries. The format for defining variables is:

```
variable_name = name_list
```

variable_name is a name by which the variable is referenced.

name_list consists of item names separated by white space, enclosed in parentheses.

Spaces or tabs immediately to the left and right of the "=" are ignored. Subsequent appearances of **${*variable_name*}** in the distfile (except in comments), are replaced by ***name_list***. (Braces can be omitted if ***variable_name*** consists of just one character.)

Variable definitions can also be specified in the command line when invoking **rdist**; variable definitions in the command line override definitions in the distfile. (See "Starting rdist" on page 382.)

The following are examples of three variable definition entries in a distfile:

```
HOSTS = ( matisse root@arpa)

FILES = ( /bin /lib /usr/bin /usr/games
        /usr/include/{*.h,{stand,sys,vax*,pascal,machine}/*.h
                  /usr/lib /usr/man/man? /usr/ucb
                  /usr/local/rdist `cat ./std-files` )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag crontab dshrc
        sendmail.cf sendmail.fc sendmail.hf sendmail.st uucp vfont )
```

The first entry defines the variable **HOSTS** to represent two remote hosts, **matisse** and **arpa**, that are to be updated. Note that if a remote host is specified in the form ***user@host***, ***user*** is the user name on ***host*** that is used to update files and directories on that host. Otherwise, the user name on the master host is used to update the remote host.

The second entry defines the variable **FILES** to represent the files and directories to be updated on the remote hosts. The shell meta characters {, }, and * in the second line of this entry are used in a "shorthand" that represents the files **/usr/include/*.h**, **/usr/include/stand/*.h**, **/usr/include/sys/*.h**, **/usr/include/vax*/*.h**, etc. The * character is used as a wildcard. Note that you can use commands, such as **cat**, within single backquotes (`) in the variable list.

The last entry defines the variable **EXLIB** to represent the files that should not be updated on the remote hosts.

Examples of how variables are used in distfile command entries are shown in the following sections.

File Distribution Commands

Distfile command entries that distribute files to a remote host are specified in the following format:

```
[label:] source_list -> destination_list command_list ;
```

***label*:** is optional and is used to group command entries. You can use labels to perform a partial update. Normally, **rdist** updates all the files and directories listed in a distfile. You can invoke **rdist** with a specific label; in this case, **rdist** executes only the entries under the specified label.

source_list specifies the directories or files on the master host that are to be used as the master copy for distributing to the remote hosts.

destination_list specifies the list of remote hosts to which ***source_list*** is to be distributed.

source_list and ***destination_list*** can consist of the following:

- Single name (for example, **matisse**).
- Variable defined previously in the distfile. Variables to be expanded begin with "$", followed by the variable name in curly braces (for example, **${HOSTS}**).
- List of names, separated by white space and enclosed in parentheses (for example, **(/usr/lib /usr/bin /usr/ucb)**.

command_list consists of one or more of the commands listed in Table 16. Each command must end in a semicolon (;).

Table 16 Distfile Commands

install	Copies source files/directories to each host in the destination list. Any of the following options can be specified:
	-b performs a binary comparison and updates files if they differ. Without this option, **rdist** updates files only if the size or modification time differs.
	-h follows symbolic links on the master host and copies the file(s) that the link points to. Without this option, **rdist** copies the name of a symbolic link.
	-i ignores unresolved links. Without this option, **rdist** tries to maintain the link structure of the files being copied and sends out warnings if any link cannot be found.
	-R removes files in the remote host's directory that do not exist in the corresponding directory on the master host.
	-v displays the files that are out of date on the remote host but does not update any files or send any mail.
	-w appends the full pathname (including directory subtree) to a destination directory name. For example, if file **/dira/filea** is copied to **dirb**, the resulting file is **/dirb/dira/filea**. Without this option, the preceding copy operation would result in **/dirb/filea**.
	-y does not update files on the remote host that are newer than the master copy.
	destpath installs the file on the remote host as the specified path name.
notify *user*[@*host*]	Mails a list of updated files and/or any errors that have occurred to a specified receiver. If **host** is not specified, the remote host name is assumed.
except *file_list*	Updates all files in the source list except for the file(s) specified in ***file_list***.

Table 16 **Distfile Commands**

except_pat *pattern*	Updates all files in the source list except for those files whose names contain the pattern ***pattern***. The characters “\” and “$” must be escaped with a backslash (\).
special [*file*] "*command*"	Specifies command(s) that are to be executed on the remote host after each specified file is updated or installed. Used to rebuild databases and configuration files after a program has been updated. If ***file*** is not specified, ***command*** is performed for every updated file. ***command*** can contain multiple commands, each separated by semicolons. The user’s home directory on the remote host is the default working directory for each command.

If there is no install command in a distfile or if the ***destpath*** option is not used with the **install** command, the name of the file on the master host is given to the remote host’s file. Parent directories in a file’s path are created on a remote host if they do not exist. **rdist** does not replace non-empty directories on a remote host. However, if the **-R** option is specified with the **install** command, a non-empty directory is removed on the remote host if the corresponding directory does not exist on the master host.

For a detailed description of commands and their options, type **man 1 rdist** at the HP-UX prompt.

The following two examples of file distribution commands use the variable definitions that were shown previously:

```
${FILES} -> ${HOSTS}
     install -R ;
     except /usr/lib/${EXLIB} ;
     except /usr/games/lib ;

srcs:
     /usr/src/bin -> arpa
     except_pat ( \\.o$ /SCCS\$ ) ;
```

The first example distributes the source files defined in the variable **FILES** to the destination hosts defined in the variable **HOSTS**. **rdist** copies the files to each remote host, removing files in the remote host’s directory that do not exist on the master directory. **rdist** does not update files in **/usr/lib/${EXLIB}** or in **/usr/games/lib**.

The second example (labeled **srcs**) distributes the directory **/usr/src/bin** to the host **arpa**; object files or files that are under SCCS control are not copied.

Changed Files List Commands

The third type of Distfile entry is used to make a list of files that have been changed on the master host since a specified date. The format for this type of entry is as follows:

```
[label:] source_list :: timestamp_file command_list ;
```

label*:** and ***source_list are specified in the same manner as in the entries to distribute files.

timestamp_file is a file on the local host, whose modification time is used as a timestamp. ***source_list*** files on the local host that are newer than the timestamp are noted in a list.

Use the **notify** command to mail the list of changed files to a specified user. The following is an example of this type of entry:

```
${FILES} :: stamp.cory
    notify root@cory ;
```

In the above example, the list of files that are newer than the timestamp in **stamp.cory** are mailed to the user **root@cory**. Note that with the notify command, if no “@” appears in the user name, the remote host name is assumed.

Starting rdist

After creating the distfile on the master host, you can start **rdist** from the command line or from a **cron** file. **rdist** must be run as root on the master host. There are two forms of the **rdist** command syntax. One form is the following:

```
/usr/bin/rdist [-b] [-h] [-i] [-n] [-q] [-R] [-v] [-w]
[-y] [-d var=value] [-f distfile] [-m host] ... [label]
```

-d *var=value* changes the value of the variable ***var*** previously defined in the distfile. ***value*** can be a list of names enclosed in parentheses and separated by white space, a single name, or an empty string.

-f *distfile* specifies ***distfile*** as the distfile to be used to update files and directories. If the distfile is not specified, **rdist** looks in the current working directory for the file **distfile**, then the file **Distfile**.

-m *host* limits the updates to ***host***, which is one of the hosts previously identified in the distfile. Multiple **-m** arguments may be specified.

label performs only the command entries specified by ***label*** in the distfile.

Other options are listed in Table 17 on page 383.

The other form of the rdist command syntax is:

```
/usr/bin/rdist [-b] [-h] [-i] [-n] [-q] [-R] [-v] [-w]
[-y] -c pathname ... [login@]host[:destpath]
```

-c *pathname* ... [*login*@]*host*[:*destpath*] updates file(s) in ***pathname*** on the remote host ***host***. (The **-c** arguments are interpreted as a distfile.) ***login*** specifies the user name used to perform the update. ***destpath*** specifies the pathname of the installed file on the remote host.

Other options are listed in Table 17 on page 383.

Table 17 **rdist Command Line Options**

-b	Performs a binary comparison and updates files if they differ. Without this option, **rdist** updates files only if the size or modification time differs.
-h	Follows symbolic links on the master host and copies the file(s) that the link points to. Without this option, **rdist** copies the name of a symbolic link.
-i	Ignores unresolved links. Without this option, **rdist** tries to maintain the link structure of the files being copied and sends out warnings if any link cannot be found.
-M	Checks that mode, ownership, and group of updated files on the remote host are the same as master copy and updates the files if they differ. This is done in addition to any other comparison that may be in effect.
-n	Prints **rdist** commands on standard output on the master host without executing them. This option is useful for debugging a distfile.
-q	Suppresses printing of files being modified to standard output on the master host.
-R	Removes files in remote host's directory that do not exist on the master directory.
-v	Displays the files that are out of date on the remote host but does not update any files or send any mail.
-w	Appends the full pathname (including directory subtree) to a destination directory name.
-y	Does not update files on the remote host that are newer than the master copy.

Example Output on the Master Host

This section shows an example of what is displayed on the standard output on the master host when **rdist** is run. An example distfile is shown below:

```
HOSTS = (lassie benji )

FILES = ( myprog.c )
${FILES} -> ${HOSTS}
   install;
   special"cc";
   notify     duncan@snoopy;
```

rdist is started with no command line options. The display on the standard output on the master host is shown below:

```
% /usr/bin/rdist
updating host lassie
installing: myprog.c
special "cc"
notify @lassie (duncan@snoopy)
updating host benji
installing: myprog.c
special "cc"
notify @benji (duncan@snoopy)
```

Troubleshooting rdist

Errors, warnings, and other messages are sent to standard output on the master host. Use the **notify** command to mail a list of files updated and errors that may have occurred to the specified users on the remote host being updated. To mail the list to a user that is not on the remote host, make sure that you specify the mail recipient as ***user@host***.

If **rdist** does not update files on the remote system, check the following:

- Use the **-n** command line option to check the operation of a distfile. This option prints out the commands to standard output on the master host without executing them.
- Make sure that the remote system is reachable by using the **ping** command.
- Source files must reside on the master host where **rdist** is executed. Make sure that the files exist on the master host.
- **rdist** aborts on files that have a negative modification time (before January 1, 1970). Make sure that the source files do not have a negative modification time.

NOTE: On NFS-mounted file systems, root may not have its usual access privileges. If **rdist** is run by root, **rdist** may fail to copy to NFS-mounted volumes.

A message that there is a mismatch of **rdist** version numbers may be caused by one of the following:

- The BSD version of the **rdist** software running on the master host is not the same as that running on the remote system. The HP-UX **rdist** software is based on BSD's version 3 of **rdist** and is compatible with other implementations of BSD's version 3 of **rdist** . Make sure the **rdist** software running on all systems is based on BSD's version 3.
- An executable version of **rdist** is not in **/usr/bin** on the remote system.

NOTE: The **-M** command line option may not be supported by non-HP **rdist** implementations.

12

Troubleshooting Internet 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
- Flowchart Format
- Troubleshooting the Internet Services
- Reporting Problems to Your Hewlett-Packard Support Contact

Troubleshooting information for DDFA is documented in the *DTC Device File Access Utilities* manual.

Troubleshooting information for the Secure Internet Services product is documented in the section “Troubleshooting the Secure Internet Services” on page 90.

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 as follows:

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 as follows:

- 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?

It is often difficult to determine whether the problem is hardware-related or software-related. The symptoms of the problem that indicate you should suspect the hardware are as follows:

- 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-related or hardware-related.

Diagnostic Tools Summary

The most frequently used diagnostic tools are listed below. These tools are documented in the link installation manuals.

Table 18 Diagnostic Tools

netstat	A nodal management command that returns statistical information regarding your network.
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 communication time between the local and remote hosts. (Type **man 1M ping** for more information.)
psidad	A utility under DUI that can help to identify problems on the PSI/800 board/card.
rlb	A diagnostic program which tests LAN connections to other HP 9000 computers. **rlb** does not test a connection to an HP 1000 computer.
x25check **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 from the X.25 device and 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 and side B communicate.

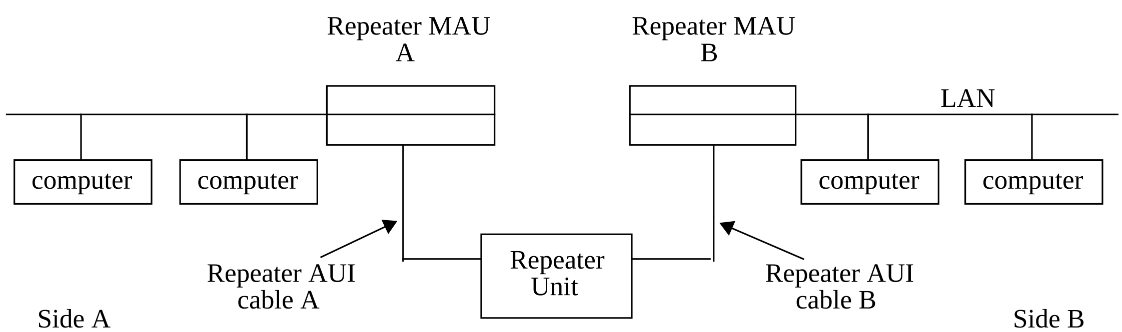


Figure 43 **Troubleshooting Networks that Use Repeaters**

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 the following:

```
netstat -r
```

- To obtain routing statistics, execute the following:

```
netstat -rs
```

The statistics could indicate a bad route, suggesting a problem with a gateway node. If so,

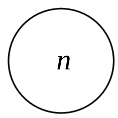
- Check with the node manager of the gateway node to ascertain proper operation of the gateway.
- You can detect problems with the X.25 line by the number of errors shown when you execute the following:

```
x25stat -f -d /devicefile
```

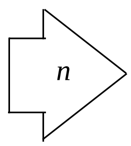
For more information on troubleshooting gateways, refer to the appropriate link manual. For information on repeaters, refer to the *HP-PB LAN Interface Controller (LANIC) Installation Manual*.

Flowchart Format

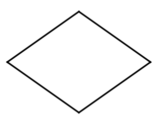
The flowcharts in this chapter each have a corresponding set of labeled explanations. You can follow the flowcharts alone or follow the flowcharts and read the explanations for more detail. The explanations are on the pages that follow each flowchart.



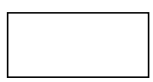
Start of flowchart *n*; reenter current flowchart



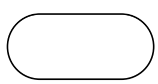
Go to and enter flowchart *n*



Make a decision



Perform an action



Exit flowchart

Troubleshooting the Internet Services

When troubleshooting problems with the Internet 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 log in 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 after a connection is completely established.

Whenever you receive an error message, follow the corrective action supplied in the man page for that service. The error message is preceded by the name of the service. Table 19 shows the appropriate man page to consult for a description of the error messages:

Table 19 **Reference Pages for Error Messages**

Service	Client	Server
telnet	**telnet**(1)	**telnetd**(1M)
ftp	**ftp**(1)	**ftpd**(1M)
rlogin	**rlogin**(1)	**rlogind**(1M)
remsh	**remsh**(1)	**remshd**(1M)
rcp	**rcp**(1)	**remshd**(1M)
ruptime	**ruptime**(1)	**rwhod**(1M)
rwho	**rwho**(1)	**rwhod**(1M)
ddfa	user application	**ocd**(1M)

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.

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.

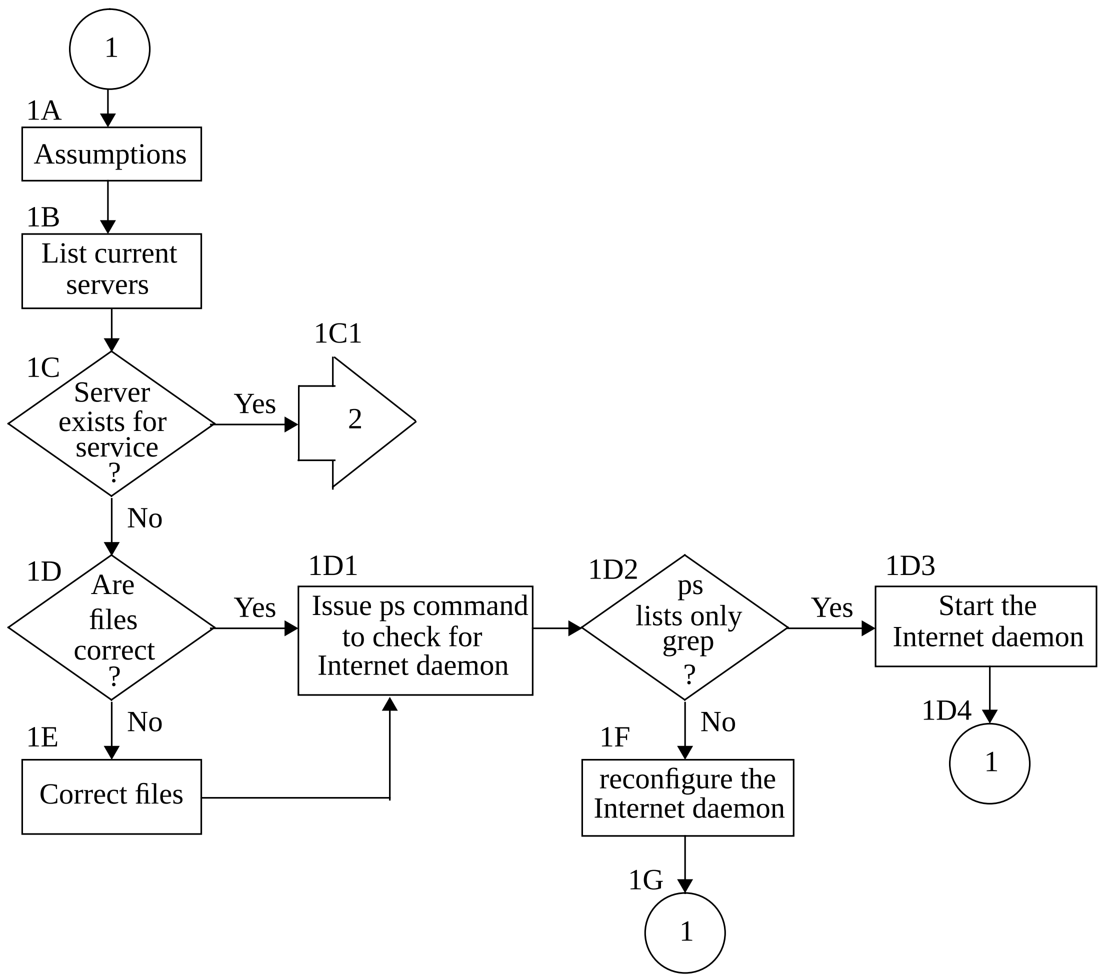


Figure 44 **Flowchart 1. Checking for a Server**

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/9000 Software*).

1B. List current servers. List the servers currently running on your system by executing the following:

```
netstat -a
```

Table 20 lists the servers required for each service.

Table 20 **Servers Required for Each Service**

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.

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 Internet Services.

1D. Are files correct? Is there an entry for the servers or services in the **/etc/inetd.conf** or **/etc/services** files?

Table 21 lists the entries that are required in the **/etc/inetd.conf** file.

Table 21 Entries Required in /etc/inetd.conf

Service Requested	**inetd.conf** Entry
ftp	**ftp stream tcp nowait root /usr/lbin/ftpd ftpd**
telnet	**telnet stream tcp nowait root /usr/lbin/telnetd telnetd**
rlogin	**login stream tcp nowait root /usr/lbin/rlogind rlogind**
remsh, **rcp**	**shell stream tcp nowait root /usr/lbin/remshd remshd**
rexec library	**exec stream tcp nowait root /usr/lbin/rexecd rexecd**
tftp	**tftp dgram udp nowait root /usr/lbin/tftpd tftpd**
bootpd	**bootps dgram udp wait root /usr/lbin/bootpd bootpd**
fingerd	**finger stream tcp nowait bin /usr/lbin/fingerd fingerd**

Check the permissions on the files in the **/usr/lbin** and **/usr/sbin** directories. The files **ftpd**, **bootpd**, **telnetd**, **rlogind**, **remshd**, **rexecd**, **rwhod**, and **inetd** must be owned and executable by root only. The file **fingerd** should be owned and executed by **bin** only. No other user should have permission to write them, although all users can read them.

Table 22 on page 401 lists the entries that are required in the **/etc/services** file.

Table 22 **Entries Required in /etc/services**

Service Requested	**/etc/services** Entry
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** **bootpc 68/udp**
fingerd	**finger 79/tcp**

If the file entries or permissions are not correct, continue with 1E.

1D1. Issue ps command to check for internet daemon. To see if the **inetd** daemon is active on the server node, log in to the server node and execute the following:

```
ps -ef | grep inetd
```

1D2. The **ps** command lists only the **grep** process for **inetd**? 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 the following as superuser:

```
/usr/sbin/inetd
```

or, if you want to start connection logging,

```
/usr/sbin/inetd -l
```

The **/sbin/init.d/inetd** shell script usually starts **inetd** at boot time. See “Installing and Configuring Internet Services” on page 25.

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 the internet daemon. To reconfigure **inetd**, execute the following as superuser:

```
/usr/sbin/inetd -c
```

and continue with 1G.

1G. Go to 1B. Repeat flowchart from 1B to check if the server exists.

Flowchart 2. Security for `telnet` and `ftp`

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.

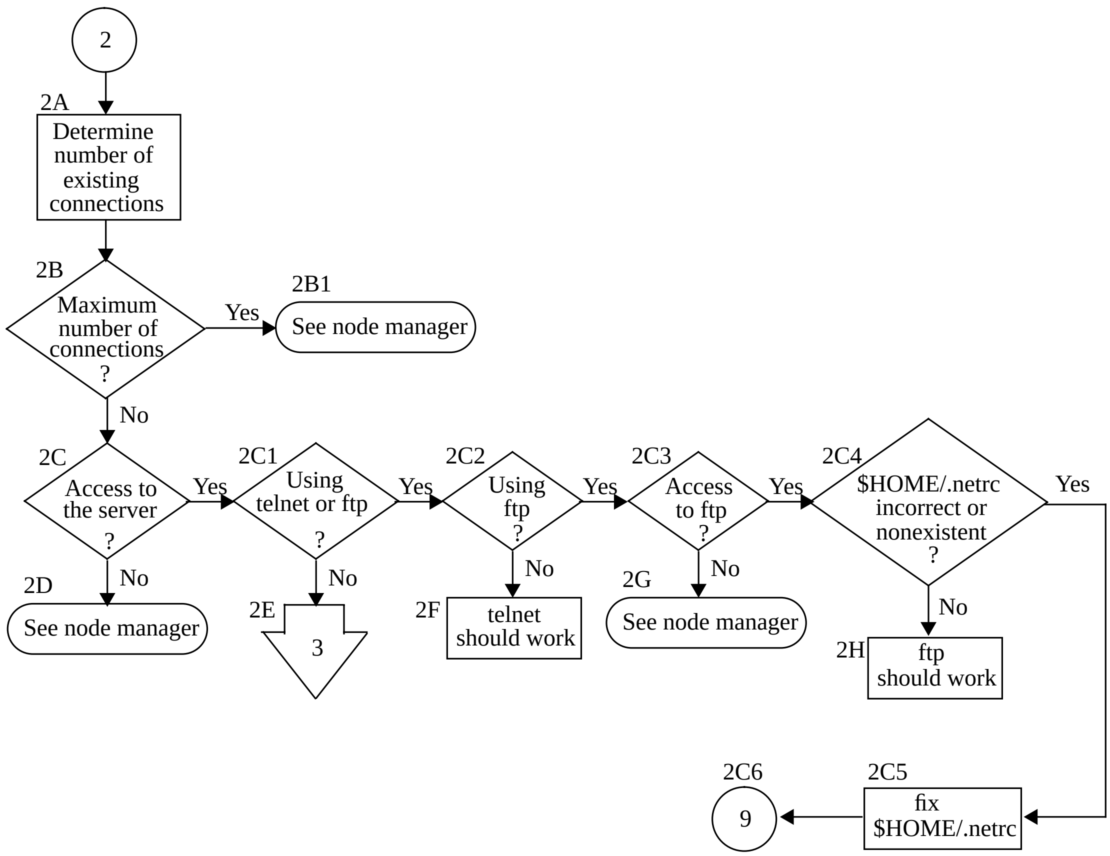


Figure 45 **Flowchart 2. Security for `telnet` and `ftp`**

NOTE: The corrections suggested in 2B1, 2C1 and 2F1 must be done by the superuser. 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, **/var/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 **/var/adm/inetd.sec** file.

2C. Access to the server? The **/var/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 **telnet** or **ftp**? There are additional security files that exist for these services that must be checked. If you are using **ftp** or **telnet** go to 2C2; otherwise, go to 2E.

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. Type **`man 4 netrc`** for more information. If it is correct, go to 2H.

2C5. Fix **`$HOME/.netrc`**. If the file is incorrect, make corrections to it 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 **`/var/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 (**`sendmail`**, BIND, **`finger`**, the **`rexec`** library, or any of the “**`r`**” 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/ftupusers`** 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 syntax and none of the error messages have solved the problem, report the problem to your Hewlett-Packard support contact.

Flowchart 3. Security for Berkeley Services

This flowchart is for troubleshooting security for the Berkeley Services: **`sendmail`**, BIND, **`finger`**, the **`rexec`** library, and those services that begin with “**`r`**”. The following information assumes an account has a password. If it does not, the security checks are not performed.

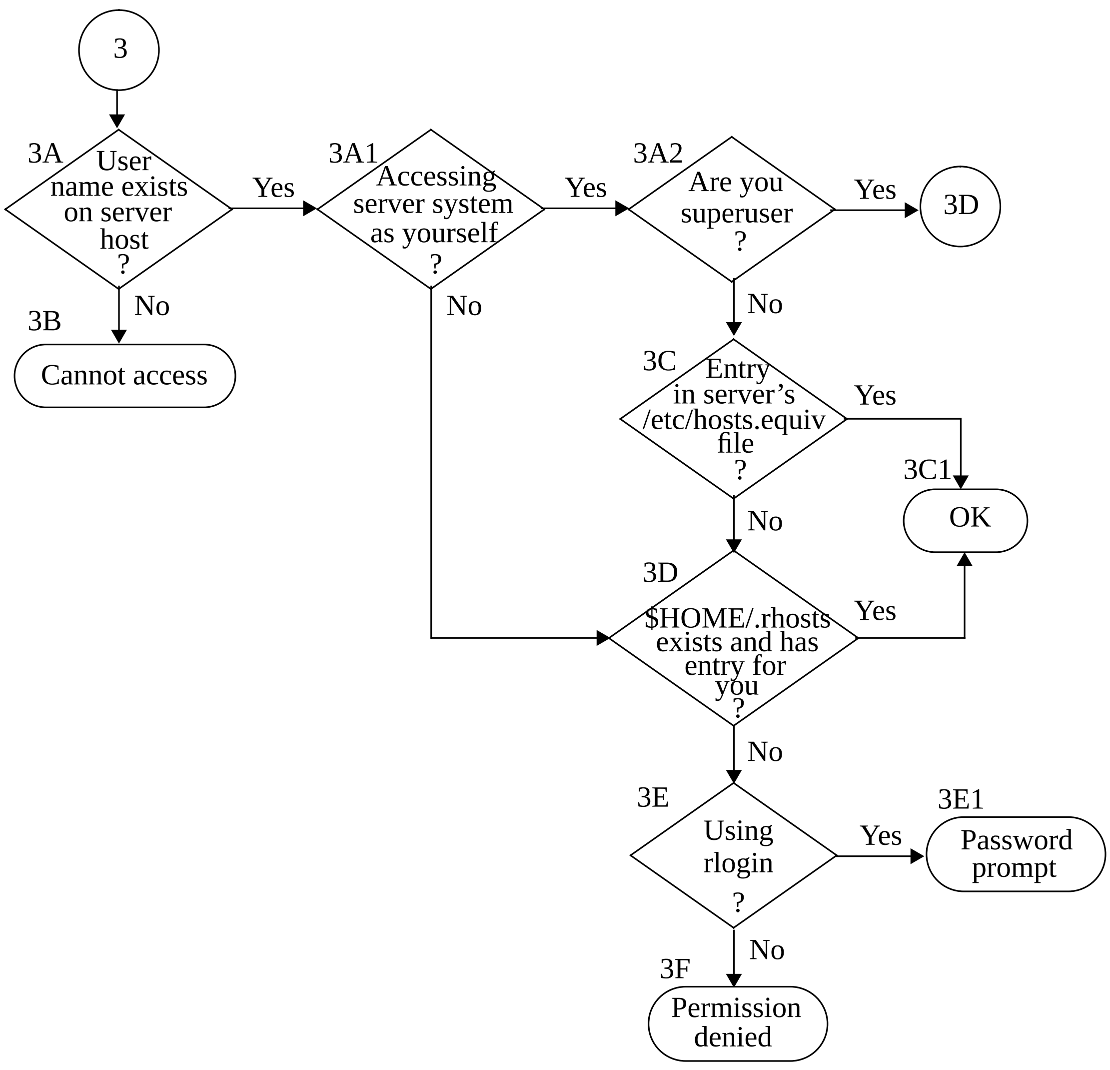


Figure 46 **Flowchart 3. Security for Berkeley Services**

3A. User name exists on server host? Does the user name that you want to log in as exists 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 log in 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 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.

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 to his or her **`.rhosts`** file.

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 the following: 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 Internet Services version, execute the **`what`** ***`service_name`*** command, where ***`service_name`*** is a network service specific to the networking product such as **`ftp`** for Internet Services.

  To check the version of your kernel, execute **`uname -r`**.

  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.
- 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 Internet 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 **`/sbin/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.

Index

Index

Index

Index

Index

Index

Index

Index

Index