System Administration Guide, Volume 3

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part Number 806-0916–10
February 2000

Copyright 2000 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, NFS, PCNFSpro, SunOS, WebNFS, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227–14(g)(2)(6/87) and FAR 52.227–19(6/87), or DFAR 252.227–7015(b)(6/95) and DFAR 227.7202–3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2000 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, Californie 94303-4900 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, NFS, PCNFSpro, SunOS, WebNFS, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Contents

Preface

System Administration Guide, Volume 3 is part of a three-volume set that covers a significant part of the Solaris™ system administration information. This book assumes that you have already installed the SunOS™ 5.8 operating system, and you have set up any networking software that you plan to use. The SunOS 5.8 operating system is part of the Solaris product family, which also includes many features, including the Solaris Common Desktop Environment (CDE). The SunOS 5.8 operating system is compliant with AT&T's System V, Release 4 operating system.

Note - The Solaris operating environment runs on two types of hardware, or platforms—SPARC™ and IA. The Solaris operating environment runs on both 64-bit and 32-bit address spaces. The information in this document pertains to both platforms and address spaces unless called out in a special chapter, section, note, bullet, figure, table, example, or code example.

Who Should Use This Book

This book is intended for anyone responsible for administering one or more systems running the Solaris 8 release. To use this book, you should have 1-2 years of UNIX® system administration experience. Attending UNIX system administration training courses might be helpful.

How the System Administration Volumes Are Organized

Here is a list of the topics covered by the three volumes of the System Administration Guides.

System Administration Guide, Volume 1

- "Managing Users and Groups Topics" in *System Administration Guide, Volume 1*
- "Managing Server and Client Support Topics" in *System Administration Guide, Volume 1*
- "Shutting Down and Booting a System Topics" in *System Administration Guide, Volume 1*
- "Managing Removable Media Topics" in *System Administration Guide, Volume 1*
- "Managing Software Topics" in *System Administration Guide, Volume 1*
- "Managing Devices Topics" in *System Administration Guide, Volume 1*
- "Managing Disks Topics" in *System Administration Guide, Volume 1*
- "Managing File Systems Topics" in *System Administration Guide, Volume 1*
- "Backing Up and Restoring Data Topics" in *System Administration Guide, Volume 1*

System Administration Guide, Volume 2

- "Managing Printing Services Topics" in *System Administration Guide, Volume 2*
- "Working With Remote Systems Topics" in *System Administration Guide, Volume 2*
- "Managing Terminals and Modems Topics" in *System Administration Guide, Volume 2*
- "Managing System Security Topics" in *System Administration Guide, Volume 2*
- "Managing System Resources Topics" in *System Administration Guide, Volume 2*
- "Managing System Performance Topics" in *System Administration Guide, Volume 2*
- "Troubleshooting Solaris Software Topics" in *System Administration Guide, Volume 2*

System Administration Guide, Volume 3

- "Network Services Topics" in *System Administration Guide, Volume 3*
- "IP Address Managament Topics" in *System Administration Guide, Volume 3*
- "Modem-Related Network Services Topics" in *System Administration Guide, Volume 3*
- "Accessing Remote File Systems Topics" in *System Administration Guide, Volume 3*
- "Mail Services Topics" in *System Administration Guide, Volume 3*
- "Monitoring Network Services Topics" in *System Administration Guide, Volume 3*

Related Books

This is a list of related documentation that is refered to in this book.

- *Solaris Naming Administration Guide*
- *Solaris Naming Setup and Configuration Guide*
- *System Administration Guide, Volume 1*
- *System Administration Guide, Volume 2*
- Anderson, Bart, Bryan Costales, and Harry Henderson. *UNIX Communications.* Howard W. Sams & Company, 1987.
- Cheswick, William R. and Steven M. Bellovin. *Firewalls and Internet Security*. Addison Wesley, 1994.
- Costales, Bryan. *sendmail, Second Edition*. O'Reilly & Associates, Inc., 1997.
- Frey, Donnalyn and Rick Adams. *!%@:: A Directory of Electronic Mail Addressing and Networks*. O'Reilly & Associates, Inc., 1993.
- Krol, Ed. *The Whole Internet User's Guide and Catalog*. O' Reilly & Associates, Inc., 1993.
- O' Reilly, Tim and Grace Todino. *Managing UUCP and Usenet*. O' Reilly & Associates, Inc., 1992.
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1, The Protocols*. Addison Wesley, 1994.

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

The docs.sun.com[SM] Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

What Typographic Conventions Mean

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls –a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename.*
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide.* These are called *class* options. You must be *root* to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	#

CHAPTER 1

Network Services Topics

Chapter 2	Provides overview information for the network services

CHAPTER 2

Network Services Overview

This chapter introduces some of the changes to some of the network services for this release. In addition, it introduces the role of the network administrator. If you are a new network administrator, the topics covered give you an idea of the tasks you might perform. The chapter then presents fundamental networking concepts that you need to know as you progress through this book. If you are an experienced network administrator, consider reviewing the first section, and skipping the rest. The following topics are explained:

What's New for the Solaris 8 Release?

The table below shows some of the new features included in this release.

TABLE 2–1 What's New for the Solaris 8 Release?

Technology	Description	For Instructions, Go To ...
DHCP	The Dynamic Host Configuration Protocol (DHCP) enables a host to get an IP address and other system configuration information without preconfiguration by a system administrator. A Java-based graphical interface for configuring and managing the DHCP server and databases has been added for the Solaris 8 release.	Chapter 8
IPsec	The IP Security Architecture (IPSec) provides protection for IP datagrams. The protection can include confidentiality, strong integrity of the data, partial sequence integrity (replay protection), and data authentication. IPSec is performed inside the IP processing, and it can be applied with or without the knowledge of an Internet application.	Chapter 18
IPv6	IPv6 is a new version of Internet Protocol (IP) designed to be an evolutionary step from the current version, IPv4. It is a natural increment to IPv4. Deploying IPv6, using defined transition mechanisms, does not disrupt current operations. In addition, IPv6 provides a platform for new Internet functionality.	Chapter 14
LLC2	The Class II logical link control driver (llc2) interfaces network software (such as NetBIOS, SNA, and OSI) running under the Solaris operating environment to a physical LAN network controlled by one of the supported communications adapters.	"New Version of Logical Link Control Driver" on page 45
NFS Logging	NFS logging adds transaction logging to the NFS server.	"NFS Server Logging" on page 540
Sendmail	The Solaris mail service is based on the 8.9.3 version of `sendmail`.	"What's New With `sendmail`" on page 651
NCA	The Solaris Network Cache and Accelerator improves web server performance.	"Solaris Network Cache and Accelerator (NCA)" on page 46

New Version of Logical Link Control Driver

The Class II logical link control driver (llc2) interfaces network software (NetBIOS, SNA, OSI, and so forth), running under the Solaris operating environment to a physical LAN network controlled by one of the supported communications adapters. The llc2 driver, which appears as a driver to the network software, resides in the kernel and is accessed by standard UNIX STREAMS functions.

This version of the llc2 driver includes support for both connectionless and connection-oriented logical link control class II llc2 operations for Ethernet, Token Ring, and FDDI adapters when accessed through the appropriate Solaris MAC layer driver. The Data Link Provider Interface (DLPI) to the llc2 driver enables multiple and different protocol stacks, (including NetBIOS and SNA), to operate simultaneously over one or more local area networks.

To start the llc2 driver by default, rename file `/etc/llc2/llc2_start.default` to `/etc/llc2/llc2_start`. This allows the `/etc/init.d/llc2/rc2.d/S40llc2` script to build up the configuration file for each ppa interface in `/etc/llc2/default/llc2.*` and start llc2 on each interface. To verify the configuration files, manually run `/usr/lib/llc2/llc2_autoconfig`.

For more information on the llc2 driver, see the IEEE standard 802.2 Logical Link Control and the `llc2`(7) man page.

The `llc2` files contain information needed by LLC2 to establish the appropriate links to the underlying MAC layer drivers as well as the parameters necessary to configure the LLC (Logical Link Control) Class II Station Component structures for that link. For more information the `llc2` configuration files, see the `llc2`(4) man page.

The `llc2_autoconfig` utility is used to generate LLC2 configuration files (`/etc/llc2/default/llc2.*`). If there is no configuration file in `/etc/llc2_default/`, it detects all the available interfaces in the system and generates corresponding default configuration files.

If configuration files exist in `/etc/llc2_default/`, it will check if those interfaces defined in the files still exist. If they do not exist in the system, it sets `llc2_on` in those files to 0. After this, it detects if there are new interfaces in the system. If there are, it generates configuration files for them. For more information on the `llc2_autoconfig` utility, see the `llc2_autoconfig`(1) man page.

The `llc2_config` utility is used to start/stop the LLC2 subsystem and to configure LLC2 interface parameters. For more information on the `llc2_config` utility, see the `llc2_config`(1) man page.

You can use the `llc2_loop` loopback diagnostics command to test the driver, adapter, and network. For more information on the `llc2_loop` command, see the `llc2_loop`(1M) man page.

The `llc2_stats` command is used to retrieve statistical information from the Host-based Logical Link Control Class 2 component of the LLC2 Driver. Statistics are kept for the station, Service Access Point (SAP), and connection components. For more information on the `llc2_stats` command, see the `llc2_stats`(1) man page.

Solaris Network Cache and Accelerator (NCA)

The Solaris Network Cache and Accelerator, NCA, increases web server performance by maintaining an in-kernel cache of web pages accessed during HTTP requests. NCA provides full HTTP support in the kernel by either handling the request or passing it to the web server for processing.

This product is intended to be run on a dedicated web server. Running other large processes on a server running NCA can cause problems.

The NCA feature requires two components.

- Kernel module, `ncakmod`
- Web server, `httpd`

`ncakmod` communicates with a web server, `httpd`, through a Solaris `door` (see *door_create(3DOOR)*). The Solaris `doors` library offers a fast reliable synchronous RPC mechanism between processes on the same host and between the kernel and a user space process.

`ncakmod` to `httpd` protocol is a synchronous, request-response protocol using the Solaris doors remote procedure call (RPC) interface. Door RPC calls originate in the kernel in the NCA and are synchronous. Data is transfered in both directions; that is, from the NCA to the http server and from the http server to the NCA in each door RPC. `ncakmod` passes HTTP requests to `httpd`. `httpd` returns a response to the request over the doors interface. This provides functionality similar to features such as *acceptx* and *sendfile*.

NCA logging writes HTTP request data to the disk in binary format. The NCA feature supports the CLF (common log format) log file format.

TABLE 2–2 Solaris NCA Administration Task Map

Task	Description	For Instructions, Go To ...
Enabling NCA	Steps to enable in-kernel caching of web pages on a web server.	"How to Enable NCA" on page 47
Disabling NCA	Steps to enable in-kernel caching of web pages on a web server.	"How to Disable NCA" on page 49
Changing NCA logging	Steps to enable or disable the NCA logging process.	"How to Enable or Disable NCA Logging" on page 50

▼ How to Enable NCA

1. **Become superuser.**

2. **Register the interfaces.**

   Enter the names of each of the physical interfaces in the `/etc/nca/nca.if` file (see the `nca.if`(4) man page for more information).

```
# cat /etc/nca/nca.if
hme0
hme1
```

   For each interface, there must be an accompanying `hostname.`*interface-name* file and an entry in `/etc/hosts` file for the contents of `hostname.`*interface-name*. To bring up the NCA feature on all interfaces, place an asterisk, `*`, in the `nca.if` file.

3. **Enable the** `ncakmod` **kernel module.**

   Change the `status` entry in `/etc/nca/ncakmod.conf` to `enabled`.

```
# cat /etc/nca/ncakmod.conf
#
# Copyright (c) 1998-1999 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)ncakmod.conf       1.1     99/08/06 SMI"
#
# NCA Kernel Module Configuration File
#
status=enabled
```

(continued)

```
httpd_door_path=/var/run/nca_httpd_1.door
```

See the `ncakmod.conf`(4) man page for more information.

4. Enable NCA logging.

Change the `status` entry in `/etc/nca/ncalogd.conf` to `enabled`.

```
# cat /etc/nca/ncalogd.conf
#
# Copyright (c) 1998-1999 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)ncalogd.conf       1.1     99/08/06 SMI"
#
# NCA Log Daemon Configuration File
#
status=enabled
logd_path_name="/var/nca/log"
logd_file_size=1000000
```

You can change the location of the log file by changing the path indicted by the `logd_path_name` entry. See the `ncalogd.conf`(4) man page for more information.

5. For IA only: Increase the virtual memory size.

Use the `eeprom` command to set the `kernelbase` of the system.

```
# eeprom kernelbase=0x900000000
# eeprom kernelbase
kernelbase=0x900000000
```

The second command verifies that the parameter has been set.

Note - This reduces the amount of virtual memory that is available to user processes to less than 3 Gbytes, which means that the system is not ABI-compliant. When the system boots, it will display a message warning you about non-compliance. Most programs do not actually need the full 3 Gbyte virtual address space. If you have a program that does, you need to run it on a system that does not have NCA enabled.

6. **Reboot the server.**

▼ How to Disable NCA

1. **Become superuser.**

2. **Disable the** `ncakmod` **kernel module.**

   Change the `status` entry in `/etc/nca/ncakmod.conf` to `disabled`.

```
# cat /etc/nca/ncakmod.conf
#
# Copyright (c) 1998-1999 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)ncakmod.conf       1.1     99/08/06 SMI"
#
# NCA Kernel Module Configuration File
#
status=disabled
httpd_door_path=/var/run/nca_httpd_1.door
```

   See the `ncakmod.conf`(4) man page for more information.

3. **Disable NCA logging.**

   Change the `status` entry in `/etc/nca/ncalogd.conf` to `disabled`.

```
# cat /etc/nca/ncalogd.conf
#
# Copyright (c) 1998-1999 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)ncalogd.conf       1.1     99/08/06 SMI"
#
```

(continued)

```
# NCA Log Daemon Configuration File
#
status=disabled
logd_path_name="/var/nca/log"
logd_file_size=1000000
```

See the `ncalogd.conf`(4) man page for more information.

4. **Reboot the server.**

▼ How to Enable or Disable NCA Logging

NCA log processing can be turned on or off as needed once NCA has been enabled (see "How to Enable NCA" on page 47 for more information.)

1. **Become superuser.**

2. **Change NCA log processing.**

   To permanently disable logging you need to change the status in `/etc/nca/ncalogd.conf` to disabled and reboot the system. See the `ncalogd.conf`(4) man page for more information.

   a. **To stop logging**:

      Type the following command.

      ```
      # /etc/init.d/ncalogd stop
      ```

   b. **To start logging**:

      Type the following command.

      ```
      # /etc/init.d/ncalogd start
      ```

NCA Files

You need several files to support the NCA feature. Many of these files are ASCII, but some of them are binary. The table below lists all of the files.

TABLE 2–3 NCA Files

File Name	Function
`/etc/hostname.*`	Lists all physical interfaces configured on the server.
`/etc/hosts`	Lists all host names associated with the server. Entries in this file must match with entries in `/etc/hostname.*` files for NCA to function.
`/etc/init.d/ncalogd`	NCA startup script run when a server is booted.
`/etc/nca/nca.if`	Lists the interfaces on which NCA is run (see the `nca.if`(4) man page).
`/etc/nca/ncakmod.conf`	Lists configuration parameters for NCA (see the `ncakmod.conf`(4) man page).
`/etc/nca/ncalogd.conf`	Lists configuration parameters for NCA logging (see the `ncalogd.conf`(4) man page).
`/usr/bin/ncab2clf`	Command used to convert data in the log file to Common Log File format (see the `ncab2clf`(1) man page).
`/var/nca/log`	Holds log file data; in binary format so do not edit this file.

Perl 5

Practical Extraction and Report Language (Perl) 5.005_03, a powerful general purpose programming language and generally available as free software, is included in this Solaris release. Perl has emerged as the standard development tool for complex system administration tasks, such as graphic, network, and world wide web programming because of its excellent process, file, and text manipulation features.

Perl 5 includes a dynamically loadable module framework, which allows the addition of new functionality for specific tasks. Many modules are freely available from the Comprehensive Perl Archive Network (CPAN), at `http://www.cpan.org`. Some of the core modules included with this Solaris Perl

installation are CGI, NDBM_File, and Getopt. These modules reside in the `/usr/perl5/5.00503` directory. The `site_perl` directory is initially empty and is intended to store your locally installed Perl 5 modules.

For more information, use the `perldoc` command to examine the Perl pod (portable documentation) in the `/usr/perl5/pod` directory, like this:

```
% cd /usr/perl5/pod
% /usr/perl5/bin/perldoc perlfaq1.pod
```

Responsibilities of the Network Administrator

As a network administrator, your tasks generally fall into the following areas:

- Designing and planning the network
- Setting up the network
- Maintaining the network
- Expanding the network

Each task area corresponds to a phase in the continuing life cycle of a network. You might be responsible for all the phases, or you might ultimately specialize in a particular area, for example, network maintenance.

Designing the Network

The first phase in the life cycle of a network involves creating its design, a task not usually performed by new network administrators. Designing a network involves making decisions about the type of network that best suits the needs of your organization. In larger sites this task is performed by a senior network architect: an experienced network administrator familiar with both network software and hardware.

Chapter 5 describes the factors involved in network design.

Setting Up the Network

After the new network is designed, the second phase of network administration begins, which involves setting up and configuring the network. This consists of installing the hardware that makes up the physical part of the network, and configuring the files or databases, hosts, routers, and network configuration servers.

The tasks involved in this phase are a major responsibility for network administrators. You should expect to perform these tasks unless your organization is very large, with an adequate network structure already in place.

Chapter 6 contains instructions for the tasks involved in this phase of the network life cycle.

Maintaining the Network

The third phase of network administration consists of ongoing tasks that typically constitute the bulk of your responsibilities. They might include:

- Adding new host machines to the network
- Administering network security
- Administering network services, such as NFS™ services, name services, and electronic mail
- Troubleshooting network problems

"Configuring Network Clients" on page 103 explains how to set up new hosts on an existing network. "General Troubleshooting Tips" on page 111 contains hints for solving network problems. For information on network services, refer to Chapter 29, Chapter 33, the *Solaris Naming Administration Guide*, and the *NIS+ Transition Guide*. For security-related tasks, refer to the *System Administration Guide, Volume 1*.

Expanding the Network

The longer a network is in place and functioning properly, the more your organization might want to expand its features and services. Initially, you can increase network population by adding new hosts and expanding network services by providing additional shared software. But eventually, a single network will expand to the point where it can no longer operate efficiently. That is when it must enter the fourth phase of the network administration cycle: expansion.

Several options are available for expanding your network:

- Setting up a new network and connecting it to the existing network using a machine functioning as a router, thus creating an internetwork
- Configuring machines in users' homes or in remote office sites and enabling these machines to connect over telephone lines to your network

- Connecting your network to the Internet, thus enabling users on your network to retrieve information from other systems throughout the world
- Configuring UUCP communications, enabling users to exchange files and electronic mail with remote machines

"Configuring Routers" on page 105 contains procedures for setting up an internetwork. "Extending Your Network With PPP" on page 401 explains how to set up networking connections for nomadic computers. Chapter 25 explains how to use UUCP to exchange information between your machine and other UUCP systems.

What is TCP/IP?

A network *communications protocol* is a set of formal rules that describe how software and hardware should interact within a network. For the network to function properly, information must be delivered to the intended destination in an intelligible form. Because different types of networking software and hardware need to interact to perform the networking function, designers developed the concept of the communications protocol.

The Solaris operating environment includes the software needed for network operations for your organization. This networking software implements the communications protocol suite, collectively referred to as *TCP/IP*. TCP/IP is recognized as a standard by major international standards organizations and is used throughout the world. Because it is a set of standards, TCP/IP runs on many different types of computers, making it easy for you to set up a heterogeneous network running the Solaris operating environment.

TCP/IP provides services to many different types of computers, operating systems, and networks. Types of networks range from local area networks, such as Ethernet, FDDI, and Token Ring, to wide-area networks, such as T1 (telephone lines), X.25, and ATM.

You can use TCP/IP to construct a network out of a number of local-area networks. You can also use TCP/IP to construct a wide-area network by way of virtually any point-to-point digital circuit.

TCP/IP and its protocol family are fully described in Chapter 4.

Types of Hardware That Make Up a Solaris Network

The term *local-area network* (LAN) refers to a single network of computers limited to a moderate geographical range, such as the floor of a building or two adjacent buildings. A local-area network has both hardware and software components. From a

hardware perspective, a basic Solaris LAN consists of two or more computers attached to some form of local-area network media.

Local-Area Network Media

The cabling or wiring used for computer networks is referred to as *network media.* Figure 2–1 shows four computers connected by means of Ethernet media. In the Solaris LAN environment, Ethernet is the most commonly used local-area network media. Other types of local-area network media used in a Solaris LAN might include FDDI or Token Ring.

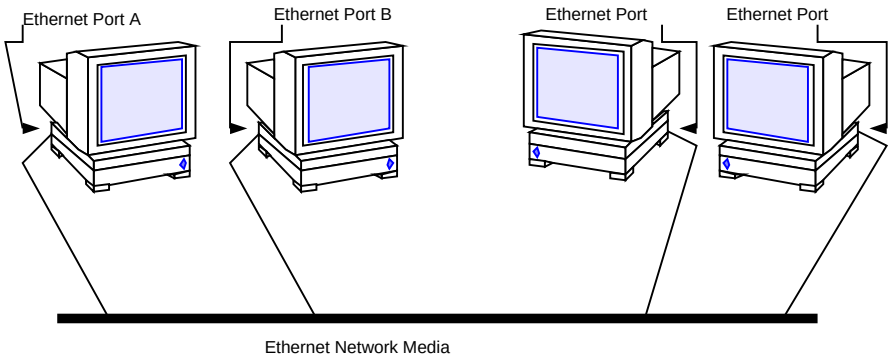


Figure 2–1 Solaris Local Area Network

Computers and Their Connectors

Computers on a TCP/IP network use two different kinds of connectors to connect to network media: serial ports, and the ports on the network interface.

Serial Ports

Each computer has at least two *serial ports*, the connectors that enable you to plug a printer or modem into the computer. The serial ports can be attached to the CPU board, or you might have to purchase them. You use these ports when attaching a modem to the system to establish a PPP or UUCP connection. PPP and UUCP actually provide wide-area network services, since they can use telephone lines as their network media.

Network Interfaces

The hardware in a computer that enables you to connect it to a network is known as a *network interface*. Many computers come with a preinstalled network interface; others can require you to purchase the network interface separately.

Each LAN media type has its own associated network interface. For example, if you want to use Ethernet as your network media, you must have an Ethernet interface installed in each host to be part of the network. The connectors on the board to which you attach the Ethernet cable are referred to as *Ethernet ports*. If you plan to use FDDI, each prospective host must have an FDDI network interface, and so on.

This book refers to the default network interface on a host as the *primary network interface*.

Note - Installing network hardware is outside the scope of this guide. Refer to *System Administration Guide, Volume 1* for instructions for configuring serial ports and manuals accompanying network media for installation instructions.

How Network Software Transfers Information

Setting up network software is an involved task. Therefore, it helps to understand how the network software you are about to set up will transfer information.

Figure 2–2 shows the basic elements involved in network communication.

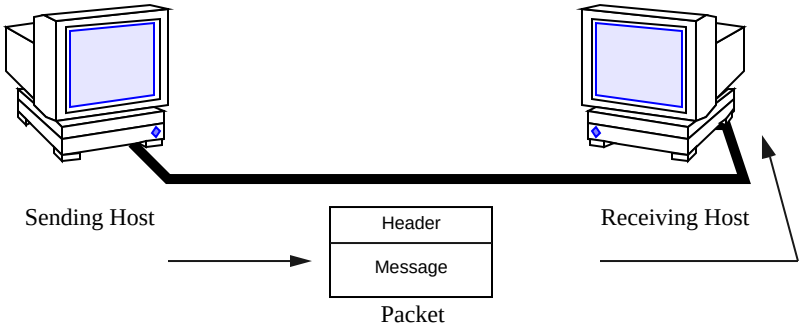


Figure 2–2 How Information Is Transferred on a Network

In this figure, a computer sends a packet over the network media to another computer attached to the same media.

How Information Is Transferred: The Packet

The basic unit of information to be transferred over the network is referred to as a *packet*. A packet is organized much like a conventional letter.

Each packet has a *header*, which corresponds to the envelope. The header contains the addresses of the recipient and the sender, plus information on how to handle the packet as it travels through each layer of the protocol suite.

The *message* part of the packet corresponds to the letter itself. Packets can only contain a finite number of bytes of data, depending on the network media in use. Therefore, typical communications such as email messages are sometimes split into packet fragments.

Who Sends and Receives Information: The Host

If you are an experienced Solaris user, you are no doubt familiar with the term "host," a word often used as a synonym for "computer" or "machine." From a TCP/IP perspective, only two types of entities exist on a network: routers and hosts.

A *router* is a machine that forwards packets from one network to another. To do this, the router must have at least two network interfaces. A machine with only one network interface cannot forward packets; it is considered a *host*. Most of the machines you set up on a network will be hosts.

It is possible for a machine to have more than one network interface but not function as a router. This type of machine is called a *multihomed* host. A multihomed host is directly connected to multiple networks through its network interfaces. However, it does not route packets from one network to another.

When a host initiates communication, it is called a *sending* host, or the sender. For example, a host initiates communications when its user types `rlogin` or sends an email message to another user. The host that is the target of the communication is called the *receiving* host, or recipient. For example, the remote host specified as the argument to `rlogin` is the recipient of the request to log in.

Each host has three characteristics that help identify it to its peers on the network. These characteristics include:

- Host name
- Internet address, or *IP address*, the form used in this book
- Hardware address

Host Name

The *host name* is the name of the local machine, combined with the name of your organization. Many organizations let users choose the host names for their machines. Programs such as `sendmail` and `rlogin` use host names to specify remote machines on a network. *System Administration Guide, Volume 1* contains more information about host names.

The host name of the machine also becomes the name of the primary network interface. This concept becomes important when you set up the network databases or configure routers.

When setting up a network, you must obtain the host names of all machines to be involved. You will use this information when setting up network databases, as described in “Naming Entities on Your Network” on page 85.

IP Address

The *IP address* is one of the two types of addresses each machine has on a TCP/IP network that identifies the machine to its peers on the network. This address also gives peer hosts a notion of *where* a particular host is located on the network. If you have installed the Solaris operating environment on a machine on a network, you might recall specifying the IP address during the installation process. IP addressing is a significant aspect of TCP/IP and is explained fully in “Designing Your IPv4 Addressing Scheme” on page 83.

Hardware Address

Each host on a network has a unique hardware address, which also identifies it to its peers. This address is physically assigned to the machine’s CPU or network interface by the manufacturer. Each hardware address is unique.

This book uses the term *Ethernet address* to correspond to the hardware address. Because Ethernet is the most commonly used network media on Solaris-based networks, the text assumes that the hardware address of your Solaris host is an Ethernet address. If you are using other network media, such as FDDI, refer to the documentation that came with your media for hardware addressing information.

Reaching Beyond the Local-Area Network—the Wide-Area Network

As your network continues to function successfully, users might need to access information available from other companies, institutes of higher learning, and other organizations not on your LAN. To obtain this information, they might need to communicate over a *wide-area network* (WAN), a network that covers a potentially vast geographic area and uses network media such as leased data or telephone lines, X.25, and ISDN services.

A prime example of a WAN is the Internet, the global public network that is the successor to the WANs for which TCP/IP was originally developed. Other examples of WANs are *enterprise networks*, linking the separate offices of a single corporation into one network spanning an entire country, or perhaps an entire continent. It is entirely possible for your organization to construct its own WAN.

As network administrator, you might have to provide access to WANs to the users on your local net. Within the TCP/IP and UNIX community, the most commonly used public network has been the Internet. Information about directly connecting to the Internet is outside the scope of this book. You can find many helpful books on this subject in a computer bookstore.

Security

Connecting a LAN to a WAN poses some security risks. You must make sure your network is protected from unauthorized use, and control access to data and resources. An overview of security issues is provided in the *System Administration Guide, Volume 1*. Further help can be found in *Firewalls and Internet Security* by William R. Cheswick and Steven M Bellovin (Addison Wesley, 1994).

You can also become informed by subscribing to `majordomo@greatcircle.com`, citing `subscribe firewalls` in the text. If you prefer the shorter version, cite `firewalls_digest` in the text.

TCP Large Window Support

TCP large windows provides the support described in RFC1323. This support is designed to improve performance over large bandwidth or delay networks such as ATM or satellite networks by using windows that exceed the normal 65535 limit.

This support expands the amount of data that can be outstanding in a TCP session from 65,535 bytes to approximately 1 Gigabyte.

TCP large window supports a number of TCP configuration parameters that allow a system administrator to enable the use of enhanced send and receive window sizes and the RFC1323 timestamp option, without having to modify the applications. These changes can be made on a system-wide basis or can be customized for particular hosts or networks. This is especially useful when using standard network utilities such as `ftp` and `rcp` which do not provide facilities to increase the buffer sizes they use.

TCP Large Window Parameters

The configuration parameters are associated with the TCP device, `/dev/tcp`, and can be inspected or modified using `ndd`(1M). Normally, these parameters would be set in one of the shell scripts executed by `init`(1M) when the system is booted (see `init.d`(4) for information on how to add a new script).

A list of the available parameters and their meanings are shown below.

tcp_xmit_hiwat — Specifies the default value for a connection's send buffer space. The default is 8K.

tcp_recv_hiwat

Specifies the default value for a connection's receive buffer space; that is, the amount of buffer space allocated for received data (and thus the maximum possible advertised receive window). The default is 8K.

tcp_wscale_always

If this parameter is nonzero, a window scale option is always sent when connecting to a remote system. Otherwise, the option is sent if-and-only-if the user has requested a receive window larger than 64K. The default is zero.

Regardless of the value of this parameter, a window scale option is always included in a connect acknowledgment if the connecting system has used the option.

tcp_tstamp_always

If this parameter is nonzero, a timestamp option is always sent when connecting to a remote system. The default is zero.

Regardless of the value of this parameter, a timestamp option is always included in a connect acknowledgment (and all succeeding packets) if the connecting system has used the option.

tcp_tstamp_if_wscale

If this parameter is nonzero, the timestamp option is sent when connecting to a remote system if the user has requested a receive window larger than 64K (that is, if a window scale option with a nonzero scale is being used). The default is zero.

tcp_max_buf

Specifies the maximum buffer size a user is allowed to specify with the SO_SNDBUF or SO_RCVBUF options. Attempts to use larger buffers fail with EINVAL. The default is 256K. It is unwise to make this parameter much larger than the maximum buffer size your applications require, since that could allow malfunctioning or malicious applications to consume unreasonable amounts of kernel memory.

tcp_host_param

This parameter is a table of IP addresses, networks, and subnetworks, along with default values for certain TCP parameters to be used on connections with the specified hosts. The table

can be displayed with the ndd command as follows:

```
example# ndd /dev/tcp tcp_host_param
Hash HSP     Address         Subnet Mask     Send       Receive    TStamp
027 fc31eea4 129.154.000.000 255.255.255.000 0000008192 0000008192      0
131 fc308244 129.154.152.000 000.000.000.000 0000032000 0000032000      0
133 fc30bd64 129.154.152.006 000.000.000.000 0000128000 0000128000      1
```

Each element in the table specifies either a host, a network (with optional subnet mask), or a subnet, along with the default send buffer space and receive buffer space, and a flag indicating whether timestamps are to be used.

The default values specified in the table are used for both active and passive connections (that is, both `connect()` and `listen()`). The most applicable match found is used; first the full host address, then the subnet, and finally the network. For subnet recognition to work properly, there must be an entry for that subnet's network that specifies the subnet mask.

The example table above specifies that:

- Connections with host 129.154.152.6 uses send and receive buffer sizes of 128000 bytes, and uses timestamps.
- Connections with other hosts on the 129.154.152 subnet uses send and receive buffer sizes of 32000 bytes.
- Connections with other hosts on the 129.154 network uses send and receive buffer sizes of 8192 bytes.

Elements are added to or removed from the table with ndd as follows:

```
ndd -set /dev/tcp tcp_host_param '<command>'
```

where <command> is either:

```
<ipaddr> [ mask <ipmask>] [ sendspace <integer> ]
    [ recvspace <integer> ] [ timestamp { 0 | 1 } ]
```

or

```
<ipaddr> delete
```

For example, the table above was created by:

```
# ndd -set /dev/tcp tcp_host_param '129.154.0.0 mask 255.255.255.0
        sendspace 8192 recvspace 8192'
# ndd -set /dev/tcp tcp_host_param '129.154.152.0 sendspace 32000
        recvspace 32000'
# ndd -set /dev/tcp tcp_host_param '129.154.152.6 sendspace 128000
        recvspace 128000 timestamp 1'
```

Note - The example commands above have been broken over two lines. Each command should be entered on one line.

It could be removed using these commands:

```
# ndd -set /dev/tcp tcp_host_param '129.154.152.6 delete'
# ndd -set /dev/tcp tcp_host_param '129.154.152.0 delete'
# ndd -set /dev/tcp tcp_host_param '129.154.0.0 delete'
```

Networks and subnets are specified by leaving the host bits zero. The same syntax used to add entries can also be used to modify existing entries.

The send and receive space values from the `tcp_host_param` table are only used if they are larger than the values set by the user (or obtained from `tcp_xmit_hiwat` and `tcp_recv_hiwat`). This is so that the user can specify larger values for improved throughput and not have them erroneously reduced.

If timestamp value in the `tcp_host_param` table is 1, the timestamp option is sent to the selected host or hosts when a connection is initiated. However, if the value is 0, the timestamp option might still be sent, depending on the settings of the `tcp_tstamp_always` and `tcp_tstamp_if_wscale` options.

TCP Selective Acknowledgment Support

The TCP selective acknowledgment (TCP SACK) provides the support described in RFC 2018 to solve the problems related to congestion and multiple packet drops especially in applications making use of TCP large windows (RFC 1323) over satellite links or transcontinental links. See RFC 2018 for complete details on TCP SACK.

The configuration parameter is associated with the TCP device, `/dev/tcp`, and can be inspected or modified using `ndd`(1M). Normally, this parameter would be set in

one of the shell scripts executed by `init`(1M) when the system is booted (see `init.d`(4) for information on how to add a new script).

The available parameter and its meaning is shown below.

tcp_sack_permitted Specifies whether SACK is permitted. The default is 1. The available options are as follows:

- **0** TCP does not accept or send SACK information.
- **1** TCP does not initiate a connection with `SACK_PERMITTED` option. If the incoming request has `SACK_PERMITTED` option, TCP responds with `SACK_PERMITTED` option.
- **2** TCP initiates and accepts connections with `SACK_PERMITTED` option.

For additional information see the `tcp`(7P) man page.

CHAPTER 3

IP Address Management Topics

Chapter 4	Provides overview information for TCP/IP
Chapter 5	Provides instructions for planning a TCP/IP network
Chapter 6	Provides step-by-step instructions for setting up and troubleshooting a TCP/IP network
Chapter 7	Provides reference information for TCP/IP
Chapter 8	Provides overview information for DHCP
Chapter 9	Provides instructions for planning for to use DHCP
Chapter 10	Provides step-by-step instructions for configuring DHCP
Chapter 11	Provides step-by-step instructions for administering DHCP
Chapter 12	Provides troubleshooting instructions for DHCP
Chapter 13	Provides background information for DHCP
Chapter 14	Provides overview information for IPv6

Chapter 15	Provides IPv6 transition strategies and mechanism
Chapter 16	Provides Solaris IPv6 implementation information
Chapter 17	Provides step-by-step instructions of IPv6 related tasks
Chapter 18	Provides overview information for IPsec
Chapter 19	Provides step-by-step instructions for setting up IPsec

CHAPTER 4

Overview of TCP/IP

This chapter introduces the Solaris implementation of the TCP/IP network protocol suite. The information is particularly geared to network administrators who are unfamiliar with TCP/IP. (For an introduction to basic network concepts, see Chapter 2.) If you are an experienced TCP/IP network administrator, consider moving on to chapters covering the tasks you want to perform. The following subjects are covered in this chapter:

Introducing the Internet Protocol Suite

This section presents an in-depth introduction to the protocols that compose TCP/IP. Although the information is conceptual, you should learn the names of the protocols and what each does. This is important because TCP/IP books explain tasks with the assumption that you understand the concepts introduced here.

TCP/IP is the commonly used nickname for the set of network protocols composing the *Internet Protocol suite*. Many texts use the term "Internet" to describe both the protocol suite and the global wide-area network. In this book, the "TCP/IP" refers specifically to the Internet protocol suite; "Internet" refers to the wide-area network and the bodies that govern it.

To interconnect your TCP/IP network with other networks, you must obtain a unique IP network number. At the time of this writing, IP network numbers are assigned by an organization known as the InterNIC.

If hosts on your network are going to participate in the Internet Domain Name system (DNS), you must obtain and register a unique domain name. The InterNIC also handles the registration of domain names under certain top-level domains such as .com (commercial), .edu (education), and .gov (government). Chapter 5 contains more information about the InterNIC. (For more information on DNS, refer to *Solaris Naming Administration Guide*.)

Protocol Layers and the OSI Model

Most network protocol suites are structured as a series of layers, sometimes referred to collectively as a *protocol stack*. Each layer is designed for a specific purpose and exists on both the sending and receiving hosts. Each is designed so that a specific layer on one machine sends or receives exactly the same object sent or received by its *peer process* on another machine. These activities take place independently from what is going on in layers above or below the layer under consideration. In other words, each layer on a host acts independently of other layers on the same machine, and in concert with the same layer on other hosts.

OSI Reference Model

Most network protocol suites are viewed as structured in layers. This is a result of the Open Systems Interconnect (OSI) Reference Model designed by the International Standards Organization (ISO). The OSI model describes network activities as having a structure of seven layers, each of which has one or more protocols associated with it. The layers represent data transfer operations common to all types of data transfers among cooperating networks.

The protocol layers of the OSI Reference Model are traditionally listed from the top (layer 7) to the bottom (layer 1) up, as shown in the following table.

TABLE 4–1 The Open Systems Interconnect Reference Model

Layer No.	Layer Name	Description
7	Application	Consists of standard communication services and applications that everyone can use
6	Presentation	Ensures that information is delivered to the receiving machine in a form that it can understand
5	Session	Manages the connections and terminations between cooperating computers
4	Transport	Manages the transfer of data and assures that received and transmitted data are identical

TABLE 4–1 The Open Systems Interconnect Reference Model *(continued)*

Layer No.	Layer Name	Description
3	Network	Manages data addressing and delivery between networks
2	Data Link	Handles the transfer of data across the network media
1	Physical	Defines the characteristics of the network hardware

The operations defined by the OSI model are conceptual and not unique to any particular network protocol suite. For example, the OSI network protocol suite implements all seven layers of the OSI Reference Model. TCP/IP uses some of OSI model layers and combines others. Other network protocols, such as SNA, add an eighth layer.

TCP/IP Protocol Architecture Model

The OSI model describes an idealized network communications protocol family. TCP/IP does not correspond to this model directly, as it either combines several OSI layers into a single layer, or does not use certain layers at all. The following table shows the layers of the Solaris implementation of TCP/IP, listed from topmost layer (application) to lowest (physical network).

TABLE 4–2 TCP/IP Protocol Stack

OSI Ref. Layer No.	OSI Layer Equivalent	TCP/IP Layer	TCP/IP Protocol Examples
5,6,7	Application, Session, Presentation	Application	NFS, NIS+, DNS, `telnet`, `ftp`, `rlogin`, `rsh`, `rcp`, RIP, RDISC, SNMP, and others
4	Transport	Transport	TCP, UDP
3	Network	Internet	IP, ARP, ICMP
2	Data Link	Data Link	PPP, IEEE 802.2
1	Physical	Physical Network	Ethernet (IEEE 802.3) Token Ring, RS-232, others

The table shows the TCP/IP protocol layers, their OSI Model equivalents, and examples of the protocols available at each level of the TCP/IP protocol stack. Each host involved in a communication transaction runs its own implementation of the protocol stack.

Physical Network Layer

The physical network layer specifies the characteristics of the hardware to be used for the network. For example, it specifies the physical characteristics of the communications media. The physical layer of TCP/IP describes hardware standards such as IEEE 802.3, the specification for Ethernet network media, and RS-232, the specification for standard pin connectors.

Data-Link Layer

The data-link layer identifies the network protocol type of the packet, in this case TCP/IP. It also provides error control and "framing." Examples of data-link layer protocols are Ethernet IEEE 802.2 framing and Point-to-Point Protocol (PPP) framing.

Internet Layer

This layer, also known as the network layer, accepts and delivers packets for the network. It includes the powerful Internet protocol (IP), the Address Resolution Protocol (ARP) protocol, and the Internet Control Message Protocol (ICMP) protocol.

IP Protocol

The IP protocol and its associated routing protocols are possibly the most significant of the entire TCP/IP suite. IP is responsible for:

- *IP addressing* - The IP addressing conventions are part of the IP protocol. (Chapter 5 describes IPv4 addressing in detail and Chapter 14 describes IPv6 addressing in detail.)
- *Host-to-host communications* - IP determines the path a packet must take, based on the receiving host's IP address.
- *Packet formatting* - IP assembles packets into units known as *IP datagrams.* Datagrams are fully described in "Internet Layer" on page 77.
- *Fragmentation* - If a packet is too large for transmission over the network media, IP on the sending host breaks the packet into smaller fragments. IP on the receiving host then reconstructs the fragments into the original packet.

Previous releases of the Solaris operating environment implemented version 4 of the Internet Protocol, which is written IPv4. However, because of the rapid growth of the

Internet, it was necessary to create a new Internet Protocol with improved capabilities, such as increased address space. This new version, known as version 6, is written IPv6. The Solaris operating environment supports both versions, which are described in this book. To avoid confusion when addressing the Internet Protocol, the following convention is used:

- When the term IP is used in a description, the description applies to both IPv4 and IPv6.
- When the term IPv4 is used in a description, the description applies only to IPv4.
- When the term IPv6 is used in a description, the description applies only to IPv6.

ARP Protocol

The Address Resolution Protocol (ARP) conceptually exists between the data link and Internet layers. ARP assists IP in directing datagrams to the appropriate receiving host by mapping Ethernet addresses (48 bits long) to known IP addresses (32 bits long).

ICMP Protocol

Internet Control Message Protocol (ICMP) is the protocol responsible for detecting network error conditions and reporting on them. ICMP reports on:

- Dropped packets (when packets are arriving too fast to be processed)
- Connectivity failure (when a destination host can't be reached)
- Redirection (which tells a sending host to use another router)

The "`ping` Command" on page 112 contains more information on the operating system commands that use ICMP for error detection.

Transport Layer

The TCP/IP transport layer protocols ensure that packets arrive in sequence and without error, by swapping acknowledgments of data reception, and retransmitting lost packets. This type of communication is known as "end-to-end." Transport layer protocols at this level are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

TCP Protocol

TCP enables applications to communicate with each other as though connected by a physical circuit. TCP sends data in a form that appears to be transmitted in a character-by-character fashion, rather than as discreet packets. This transmission

consists of a starting point, which opens the connection, the entire transmission in byte order, and an ending point, which closes the connection.

TCP attaches a header onto the transmitted data. This header contains a large number of parameters that help processes on the sending machine connect to peer processes on the receiving machine.

TCP confirms that a packet has reached its destination by establishing an end-to-end connection between sending and receiving hosts. TCP is therefore considered a "reliable, connection-oriented" protocol.

UDP Protocol

UDP, the other transport layer protocol, provides datagram delivery service. It does not provide any means of verifying that connection was ever achieved between receiving and sending hosts. Because UDP eliminates the processes of establishing and verifying connections, applications that send small amounts of data use it rather than TCP.

Application Layer

The application layer defines standard Internet services and network applications that anyone can use. These services work with the transport layer to send and receive data. There are many applications layer protocols, some of which you probably already use. Some of the protocols include:

- Standard TCP/IP services such as the `ftp`, `tftp`, and `telnet` commands
- UNIX "r" commands, such as `rlogin` and `rsh`
- Name services, such as NIS+ and Domain Name System (DNS)
- File services, such as the NFS service
- Simple Network Management Protocol (SNMP), which enables network management
- RIP and RDISC routing protocols

Standard TCP/IP Services

- *FTP and Anonymous FTP* - The File Transfer Protocol (FTP) transfers files to and from a remote network. The protocol includes the `ftp` command (local machine) and the `in.ftpd` daemon (remote machine). FTP enables a user to specify the name of the remote host and file transfer command options on the local host's command line. The `in.ftpd` daemon on the remote host then handles the requests from the local host. Unlike `rcp`, `ftp` works even when the remote computer does not run a UNIX-based operating system. A user must log in to the

remote computer to make an `ftp` connection unless it has been set up to allow anonymous FTP.

You can now obtain a wealth of materials from *anonymous FTP servers* connected to the Internet. These servers are set up by universities and other institutions to make certain software, research papers, and other information available to the public domain. When you log in to this type of server, you use the login name `anonymous`, hence the term "anonymous FTP servers."

Using anonymous FTP and setting up anonymous FTP servers is outside the scope of this manual. However, many trade books, such as *The Whole Internet User's Guide & Catalog*, discuss anonymous FTP in detail. Instructions for using FTP to reach standard machines are in *System Administration Guide, Volume 1*. The `ftp`(1) man page describes all `ftp` command options, including those invoked through the command interpreter. The `ftpd`(1M) man page describes the services provided by the daemon `in.ftpd`.

- *Telnet* - The Telnet protocol enables terminals and terminal-oriented processes to communicate on a network running TCP/IP. It is implemented as the program `telnet` (on local machines) and the daemon `in.telnet` (on remote machines). Telnet provides a user interface through which two hosts can communicate on a character-by-character or line-by-line basis. The application includes a set of commands that are fully documented in the `telnet`(1) man page.
- *TFTP* - The trivial file transfer protocol (`tftp`) provides functions similar to `ftp`, but it does not establish `ftp`'s interactive connection. As a result, users cannot list the contents of a directory or change directories. This means that a user must know the full name of the file to be copied. The `tftp`(1) man page describes the `tftp` command set.

UNIX "r" Commands

The UNIX "r" commands enable users to issue commands on their local machines that are actually carried out on the remote host that they specify. These commands include

- `rcp`
- `rlogin`
- `rsh`

Instructions for using these commands are in `rcp`(1), `rlogin`(1), and `rsh`(1) man pages.

Name Services

Two name services are available from the Solaris implementation of TCP/IP: NIS+ and DNS.

- *NIS+* - NIS+ provides centralized control over network administration services, such as mapping host names to IP and Ethernet addresses, verifying passwords, and so on. See *Solaris Naming Administration Guide* for complete details.
- *Domain Name System* - The Domain Name System (DNS) provides host names to the IP address service. It also serves as a database for mail administration. For a complete description of this service, see *Solaris Naming Administration Guide*. See also the `in.named`(1M) man page.

File Services

The NFS application layer protocol provides file services for the Solaris operating environment. You can find complete information about the NFS service in Chapter 29.

Network Administration

The Simple Network Management Protocol (SNMP) enables you to view the layout of your network, view status of key machines, and obtain complex network statistics from graphical user interface based software. Many companies offer network management packages that implement SNMP; SunNet Manager™ software is an example.

Routing Protocols

The Routing Information Protocol (RIP) and the Router Discovery Protocol (RDISC) are two routing protocols for TCP/IP networks. They are described in "Routing Protocols" on page 143.

How the TCP/IP Protocols Handle Data Communications

When a user issues a command that uses a TCP/IP application layer protocol, a chain of events is set in motion. The user's command or message passes through the TCP/IP protocol stack on the local machine, and then across the network media to the protocols on the recipient. The protocols at each layer on the sending host add information to the original data.

As the user's command makes its way through the protocol stack, protocols on each layer of the sending host also interact with their peers on the receiving host. The following figure shows this interaction.

Data Encapsulation and the TCP/IP Protocol Stack

The packet is the basic unit of information transferred across a network, consisting, at a minimum, of a header with the sending and receiving hosts' addresses, and a body with the data to be transferred. As the packet travels through the TCP/IP protocol stack, the protocols at each layer either add or remove fields from the basic header. When a protocol on the sending host adds data to the packet header, the process is called *data encapsulation*. Moreover, each layer has a different term for the altered packet, as shown in the following figure.

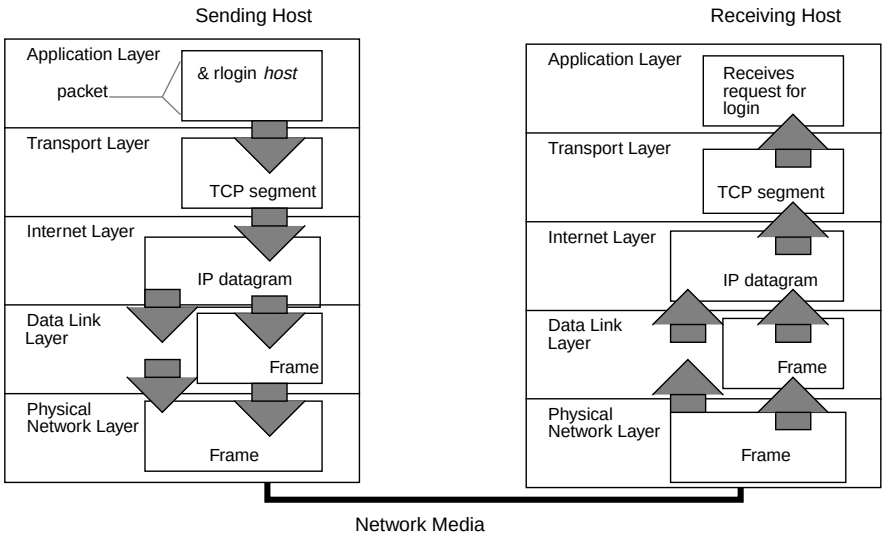


Figure 4–1 How a Packet Travels Through the TCP/IP Stack

This section summarizes the life cycle of a packet from the time the user issues a command or sends a message to the time it is received by the appropriate application on the receiving host.

Application Layer—User Initiates Communication

The packet's history begins when a user on one host sends a message or issues a command that must access a remote host. The application protocol associated with the command or message formats the packet so that it can be handled by the appropriate transport layer protocol, TCP or UDP.

Suppose the user issues an `rlogin` command to log in to the remote host, as shown in Figure 4–1. The `rlogin` command uses the TCP transport layer protocol. TCP expects to receive data in the form of a stream of bytes containing the information in the command. Therefore, `rlogin` sends this data as a TCP stream.

Not all application layer protocols use TCP, however. Suppose a user wants to mount a file system on a remote host, thus initiating the NIS+ application layer protocol.

NIS+ uses the UDP transport layer protocol. Therefore, the packet containing the command must be formatted in a manner that UDP expects. This type of packet is referred to as a *message*.

Transport Layer—Data Encapsulation Begins

When the data arrives at the transport layer, the protocols at the layer start the process of data encapsulation. The end result depends on whether TCP or UDP has handled the information.

TCP Segmentation

TCP is often called a "connection-oriented" protocol because it ensures the successful delivery of data to the receiving host. Figure 4–1 shows how the TCP protocol receives the stream from the `rlogin` command. TCP divides the data received from the application layer into segments and attaches a header to each segment.

Segment headers contain sender and recipient ports, segment ordering information, and a data field known as a *checksum*. The TCP protocols on both hosts use the checksum data to determine whether data has transferred without error.

Establishing a TCP Connection

TCP uses segments to determine whether the receiving host is ready to receive the data. When the sending TCP wants to establish connections, it sends a segment called a SYN to the peer TCP protocol running on the receiving host. The receiving TCP returns a segment called an ACK to acknowledge the successful receipt of the segment. The sending TCP sends another ACK segment, then proceeds to send the data. This exchange of control information is referred to as a *three-way handshake*.

UDP Packets

UDP is a "connectionless" protocol. Unlike TCP, it does not check to make sure that data arrived at the receiving host. Instead, UDP takes the message received from the application layer and formats it into *UDP packets*. UDP attaches a header to each packet, which contains the sending and receiving host ports, a field with the length of the packet, and a checksum.

The sending UDP process attempts to send the packet to its peer UDP process on the receiving host. The application layer determines whether the receiving UDP process acknowledges that the packet was received. UDP requires no notification of receipt. UDP does not use the three-way handshake.

Internet Layer

As shown in Figure 4–1, both TCP and UDP pass their segments and packets down to the Internet layer, where they are handled by the IP protocol. IP prepares them for delivery by formatting them into units called IP datagrams. IP then determines the IP addresses for the datagrams, so they can be delivered effectively to the receiving host.

IP Datagrams

IP attaches an *IP header* to the segment or packet's header in addition to the information added by TCP or UDP. Information in the IP header includes the IP addresses of the sending and receiving hosts, datagram length, and datagram sequence order. This information is provided in case the datagram exceeds the allowable byte size for network packets and must be fragmented.

Data-Link Layer—Framing Takes Place

Data-link layer protocols, such as PPP, format the IP datagram into a *frame*. They attach a third header and a footer to "frame" the datagram. The frame header includes a *cyclical redundancy check* (CRC) field that checks for errors as the frame travels over the network media. Then the data-link layer passes the frame to the physical layer.

Physical Network Layer—Preparing the Frame for Transmission

The physical network layer on the sending host receives the frames and converts the IP addresses into the hardware addresses appropriate to the network media. The physical network layer then sends the frame out over the network media.

How the Receiving Host Handles the Packet

When the packet arrives on the receiving host, it travels through the TCP/IP protocol stack in the reverse order from that which it took on the sender. Figure 4–1 illustrates this path. Moreover, each protocol on the receiving host strips off header information attached to the packet by its peer on the sending host. Here is what happens:

1. Physical Network Layer receives the packet in its frame form. It computes the CRC of the packet, then sends the frame to the data link layer.
2. Data-Link Layer verifies that the CRC for the frame is correct and strips off the frame header and CRC. Finally, the data link protocol sends the frame to the Internet layer.

3. Internet Layer reads information in the header to identify the transmission and determine if it is a fragment. If the transmission was fragmented, IP reassembles the fragments into the original datagram. It then strips off the IP header and passes the datagram on to transport layer protocols.
4. Transport Layer (TCP and UDP) reads the header to determine which application layer protocol must receive the data. Then TCP or UDP strips off its related header and sends the message or stream up to the receiving application.
5. Application Layer receives the message and performs the operation requested by the sending host.

TCP/IP Internal Trace Support

TCP/IP provides internal trace support by logging TCP communication when a connection is terminated by an RST packet. When an RST packet is transmitted or received, information on as many as 10 packets, which were transmitted or received immediately before on that connection, is logged with the connection information.

Finding Out More About TCP/IP and the Internet

A great deal of information about TCP/IP and the Internet has been published. If you require specific information that is not covered in this text, you can probably find what you need in the sources cited below.

Computer Trade Books

Many books about TCP/IP and the Internet are available from your local library or computer bookstore. Three highly recommended books are:

- Craig Hunt. *TCP/IP Network Administration* - This book contains some theory and much practical information for managing a heterogeneous TCP/IP network.
- W. Richard Stevens. *TCP/IP Illustrated, Volume I* - This book is an in-depth explanation of the TCP/IP protocols. It is ideal for network administrators requiring a technical background in TCP/IP and for network programmers.
- Ed Krol. *The Whole Internet User's Guide & Catalog* - This book is ideal for anyone interested in using the many tools available for retrieving information over the Internet.

RFCs and FYIs

Since 1969, developers working on the Internet protocol suite have described their protocols and related subjects in documents known as Requests for Comments (RFCs). Many RFCs are specifications for particular TCP/IP protocols and describe standards with which software implementing the protocols must comply. Other RFCs discuss the Internet, its topology, and its governing bodies. Still other RFCs explain how to manage TCP/IP applications, such as DNS.

All RFCs must be approved by the Internet Architecture Board (IAB) before they are placed in the public domain. Typically, the information in RFCs is geared to developers and other highly technical readers, though this isn't always the case.

In recent years, for your information (FYI) documents have appeared as a subset of the RFCs. The FYIs contain information that does not deal with Internet standards; rather, they contain Internet information of a more general nature. For example, FYI documents include a bibliography listing introductory TCP/IP books and papers, an exhaustive compendium of Internet-related software tools, and a glossary of Internet and general networking terms.

You'll find references to relevant RFCs throughout this guide and other books in the Solaris System Administrator set.

The InterNIC Directory and Database Service maintains the repository of RFCs. If you have a connection to the Internet, you can retrieve online RFCs as follows:

- Through `ftp`, by sending your requests to the InterNIC directory and database server `ds.internic.net`. Your request should have the format:

  `rfc/rfc.`*rfcnum*`.txt` or `rfc/rfc.`*rfcnum*`.ps`

  where *rfcnum* represents the number of the RFC you want. For example, if you wanted to retrieve RFC 1540 in PostScript™ format, you would request `rfc/rfc.1540.ps`.

- Through electronic mail, by emailing `mailserv@ds.internic.net`. This is an automatic server that expects the body of your request message to have the format:

  `document-by-name rfc`*rfcnum* or `document-by-name rfc`*rfcnum*`.ps`

- Through a World Wide Web browser, by specifying the URL `http://ds.internic.net/ds/dspg1intdoc.html`. The home page is `http://ds.internic.net`

If you need an online index of RFCs, send electronic mail to `ds.internic.net` with a message containing the request `document-by-name rfc-index`.

Note - The InterNIC information above is current as of this writing. However, the Internet is expanding at a fast pace, and the addresses listed might no longer be current by the time you read this manual.

CHAPTER 5

Planning Your TCP/IP Network

This chapter describes the issues you must resolve in order to create your network in an organized, cost-effective manner. After you have resolved these issues, you can devise a plan for your network to follow as you set it up and administer it in the future. The following topics are covered:

Designing the Network

The first phase in the life of a network—designing the network—involves making decisions about the type of network that best suits the needs of your organization. Some of the planning decisions you make will involve network hardware; for example:

- Number of host machines your network can support
- Type of network media to use: Ethernet, token ring, FDDI, and so on
- Network topology; that is, the physical layout and connections of the network hardware
- Types of hosts the network will support: standalone and dataless

Based on these factors, you can determine the size of your local-area network.

Note - Planning the network hardware is outside the scope of this manual. Refer to the manuals that came with your hardware, for assistance.

Factors Involved in Network Planning

After you have completed your hardware plan, you are ready to begin network planning, from the software perspective.

As part of the planning process you must:

1. Obtain a network number and, if applicable, register your network domain with the InterNIC.
2. Devise an IP addressing scheme for your hosts, after you receive your IP network number.
3. Create a list containing the IP addresses and host names of all machines that make up your network, which you can use as you build network databases.
4. Determine which name service to use on your network: NIS, NIS+, DNS, or the network databases in the local `/etc` directory.
5. Establish administrative subdivisions, if appropriate for your network.
6. Determine if your network is large enough to require routers, and, if appropriate, create a network topology that supports them.
7. Set up subnets, if appropriate, for your network.

The remainder of this chapter explains how to plan your network with these factors in mind.

Setting Up an IP Addressing Scheme

The number of machines you expect to support will affect several decisions you need to make at this stage of setting up a network for your site. Your organization might require a small network of several dozen standalone machines located on one floor of a single building. Alternatively, you might need to set up a network with more than 1000 hosts in several buildings. This arrangement can require you to further divide your network into subdivisions called *subnets*. The size of your prospective network will affect:

- Network class you apply for
- Network number you receive
- IP addressing scheme you use for your network

Obtaining a network number and then establishing an IP addressing scheme is one of the most important tasks of the planning phase of network administration.

Administering Network Numbers

If your organization has been assigned more than one network number, or uses subnets, appoint a centralized authority within your organization to assign network numbers. That authority should maintain control of a pool of assigned network numbers, assigning network, subnet, and host numbers as required. To prevent problems, make sure that duplicate or random network numbers do not exist in your organization. If you are planning to transition to IPv6, see Chapter 15.

Designing Your IPv4 Addressing Scheme

After you have received your network number, you can then plan how to assign the host parts of the IPv4 address.

The following table shows the division of the IPv4 address space into network and host address spaces. For each class, “range” specifies the range of decimal values for the first byte of the network number. “Network address” indicates the number of bytes of the IPv4 address that are dedicated to the network part of the address, with each byte represented by *xxx*. “Host address” indicates the number of bytes dedicated to the host part of the address. For example, in a class A network address, the first byte is dedicated to the network, and the last three are dedicated to the host. The opposite is true for a class C network.

TABLE 5–1 Division of IPv4 Address Space

Class	Range	Network Address	Host Address
A	0–127	*xxx*	*xxx.xxx.xxx*
B	128–191	*xxx.xxx*	*xxx.xxx*
C	192–223	*xxx.xxx.xxx*	*xxx*

The numbers in the first byte of the IPv4 address define whether the network is class A, B, or C and are always assigned by the InterNIC. The remaining three bytes have a range from 0–255. The numbers 0 and 255 are reserved; you can assign the numbers 1–254 to each byte *depending on the network number assigned to you*.

The following table shows which bytes of the IPv4 address are assigned to you and the range of numbers within each byte that are available for you to assign to your hosts.

TABLE 5–2 Range of Available Numbers

Network Class	Byte 1 Range	Byte 2 Range	Byte 3 Range	Byte 4 Range
A	0–127	1–254	1–254	1–254
B	128–191	Preassigned by Internet	1–254	1–254
C	192–223	Preassigned by Internet	Preassigned by Internet	1–254

How IP Addresses Apply to Network Interfaces

In order to connect to the network, a computer must have at least one network interface, as explained in “Network Interfaces” on page 56. Each network interface must have its own unique IP address. The IP address that you give to a host is assigned to its network interface, sometimes referred to as the *primary network interface*. If you add a second network interface to a machine, it must have its own unique IP number. Adding a second network interface changes the function of a machine from a host to a router, as explained in “Configuring Routers” on page 105. If you add a second network interface to a host and disable routing, the host is then considered a multihomed host.

Each network interface has a device name, device driver, and associated device file in the `/devices` directory. The network interface might have a device name, such as `le0` or `smc0`, device names for two commonly used Ethernet interfaces.

Note - This book assumes that your machines have Ethernet network interfaces. If you plan to use different network media, refer to the manuals that came with the network interface for configuration information.

Naming Entities on Your Network

After you have received your assigned network number and given IP addresses to your hosts, the next task is to assign names to the hosts and determine how you will handle name services on your network. You will use these names when you initially set up your network and, later, for expanding your network through routers or PPP.

The TCP/IP protocols locate a machine on a network by using its IP address. However, humans find it much easier to identify a machine if it has an understandable name. Therefore, the TCP/IP protocols (and the Solaris operating environment) require both the IP address and the host name to uniquely identify a machine.

From a TCP/IP perspective, a network is a set of named entities. A host is an entity with a name. A router is an entity with a name. The network is an entity with a name. A group or department in which the network is installed can also be given a name, as can a division, a region, or a company. In theory, the hierarchy of names that can be used to identify a network and its machines has virtually no limit. The term for these named entities is *domain*.

Administering Host Names

Many sites let users pick host names for their machines. Servers also require at least one host name, which is associated with the IP address of its primary network interface.

As network administrator, you must ensure that each host name in your domain is unique. In other words, no two machines on your network could both have the name “fred,” although the machine “fred” might have multiple IP addresses.

When planning your network, make a list of IP addresses and their associated host names for easy access during the setup process. The list can help you verify that all host names are unique.

Selecting a Name Service

The Solaris operating environment gives you the option of using four types of name services: local files, NIS, NIS+, and DNS. Name services maintain critical information about the machines on a network, such as the host names, IP addresses, Ethernet addresses, and so forth.

Network Databases

When you install the operating system, you supply the host name and IP address of your server, clients, or standalone machine as part of the procedure. The Solaris installation program enters this information into two *network databases* called the `hosts` and `ipnodes` databases. These databases are part of a set of network databases that contain information necessary for TCP/IP operation on your network. These databases are read by the name service you select for your network.

Setting up the network databases is a critical part of network configuration. Therefore, you need to decide which name service to use as part of the network planning process. Moreover, the decision to use name services also affects whether or not you organize your network into an administrative domain. "Network Databases and `nsswitch.conf` File" on page 134 has detailed information on the set of network databases.

Using NIS, NIS+, or DNS for Name Service

The NIS, NIS+, or DNS name services maintain network databases on several servers on the network. *Solaris Naming Setup and Configuration Guide* fully describes these name services and explains how to set them up. It also explains the "namespace" and "administrative domain" concepts in detail. If you are changing name services from NIS to NIS+, refer to *NIS+ Transition Guide*. You should refer to these manuals to help you decide whether to use these name services on your network.

Using Local Files for Name Service

If you do not implement NIS, NIS+, or DNS, the network will use *local files* to provide name service. The term "local files" refers to the series of files in the `/etc` directory that the network databases use. The procedures in this book assume you are using local files for your name service, unless otherwise indicated.

Note - If you decide to use local files as the name service for your network, you can set up another name service at a later date.

Domain Names

Many networks organize their hosts and routers into a hierarchy of administrative domains. If you are going to use NIS, NIS+, or the DNS name services, you must select a domain name for your organization that is unique worldwide. To ensure that your domain name is unique, you should register it with the InterNIC. This is especially important if you plan to use DNS.

The domain name structure is hierarchical. A new domain typically is located below an existing, related domain. For example, the domain name for a subsidiary

company could be located below the domain of the parent company. If it has no other relationship, an organization can place its domain name directly under one of the existing top-level domains.

Examples of top-level domains include:

- `.com` – Commercial companies (international in scope)
- `.edu` – Educational institutions (international in scope)
- `.gov` – U.S. government agencies
- `.fr` – France

The name that identifies your organization is one that you select, with the provision that it is unique.

Administrative Subdivisions

The question of administrative subdivisions deals with matters of size and control. The more hosts and servers you have in a network, the more complex your management task. You might want to handle such situations by setting up additional administrative divisions in the form of more additional networks of a particular class or by dividing existing networks into subnets. The decision as to whether to set up administrative subdivisions for your network hinges on the following factors:

- How large is the network?

  A single network of several hundred hosts, all in the same physical location and requiring the same administrative services, can be handled by a single administrative division. On the other hand, a network of fewer machines, divided into a number of subnets and physically scattered over an extensive geographic area, would benefit from the establishment of several administrative subdivisions.

- Do users on the network have similar needs?

  For example, you might have a network that is confined to a single building and supports a relatively small number of machines. These machines are divided among a number of subnetworks, each supporting groups of users with different needs. Such a case could call for an administrative subdivision for each subnet.

  Solaris Naming Administration Guide discusses administrative subdivisions in detail.

Registering Your Network

Before you assign IP addresses to the machines on your Solaris network, you must obtain a network number from the InterNIC. Moreover, if you plan to use administrative domains, you should register them with the InterNIC.

InterNIC and InterNIC Registration Services

The InterNIC was created in 1993 to act as a central body where users of the Internet could go for information, such as:

- What the Internet's policies are
- How to access the Internet, including training services
- What resources are available to Internet users, such as anonymous FTP servers, Usenet user groups, and so on

The InterNIC also includes the InterNIC Registration Services, the organization with which you register your TCP/IP network. The InterNIC Registration Services provide templates for obtaining a network number and for registering your domain. Two points to remember about registration are:

- The InterNIC assigns network numbers.

Note - Do not arbitrarily assign network numbers to your network, even if you do not plan to attach it to other existing TCP/IP networks.

Subnet numbers are not assigned by the InterNIC. Rather, they are composed partly of the assigned network number and numbers that you define, as explained in "What Is Subnetting?" on page 131.

- You—not InterNIC—determine the domain name for your network and then register it with the InterNIC.

How to Contact the InterNIC

You can reach the InterNIC Registration Services by:

- Mail

  Write to:

  ```
  Network Solutions
  Attn: InterNIC Registration Services
  505 Huntmar Park Drive
  Herndon, Virginia 22070
  ```

- Telephone

  The phone number is 1-703-742-4777. Phone service is available from 7 a.m. to 7 p.m. Eastern Standard Time.

- Electronic mail

  Send email regarding network registration to: `Hostmaster@rs.internic.net`

- Anonymous FTP or Telnet inquiries, through the Gopher and WAIS interfaces.

Connect to `rs.internic.net`. (Anonymous FTP and Telnet are outside the scope of this book; however, books on these subjects are available in computer bookstores.)

Adding Routers

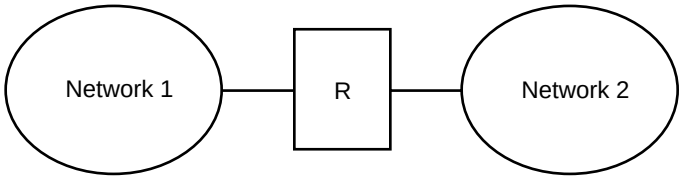
Recall that in TCP/IP, two types of entities exist on a network: hosts and routers. All networks must have hosts, while not all networks require routers. Whether you use routers should depend on the physical topology of the network. This section introduces the concepts of network topology and routing, important when you decide to add another network to your existing network environment.

Network Topology

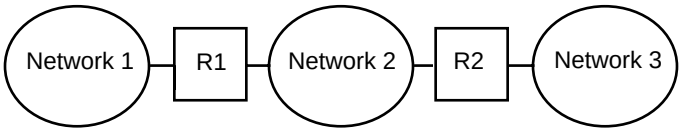
Network topology describes how networks fit together. Routers are the entities that connect networks to each other. From a TCP/IP perspective, a router is any machine that has two or more network interfaces. However, the machine cannot function as a router until properly configured, as described in “Configuring Routers” on page 105.

Two or more networks can be connected together by routers to form larger internetworks. The routers must be configured to pass packets between two adjacent networks. They also should be able to pass packets to networks that lie beyond the adjacent networks.

The following figure shows the basic parts of a network topology. The first illustration shows a simple configuration of two networks connected by a single router. The second shows a configuration of three networks, interconnected by two routers. In the first case, network 1 and network 2 are joined into a larger internetwork by router R. In the second case, router R1 connects networks 1 and 2, and router R2 connects networks 2 and 3, thus forming a network made up of networks 1, 2, and 3.



T wo Networks Connected by a Router



Three Networks Connected by T wo Routers

Figure 5–1 Basic Network Topology

Routers join networks into internetworks and route packets between them based on the addresses of the destination network. As internetworks grow more complex, each router must make more and more decisions regarding where packets are to be sent.

A step up in complexity is the case shown in the following figure. Networks 1 and 3 are directly connected by a router R3. The reason for such redundancy is reliability. If network 2 goes down, router R3 still provides a route between networks 1 and 3. Any number of networks can be interconnected and communicate as long as they all adhere to the same network protocols.

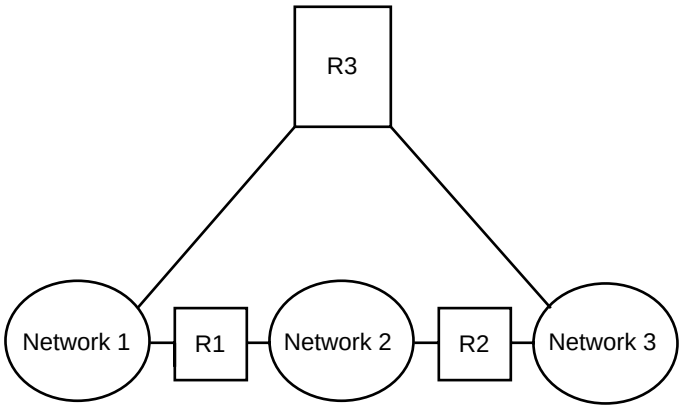


Figure 5–2 Providing an Additional Path Between Networks

How Routers Transfer Packets

Routing decisions on a network are based on the network portion of the IP address of the recipient that is contained in the packet header. If this address includes the network number of the local network, the packet goes directly to the host with that IP address. If the network number is not the local network, the packet goes to the router on the local network.

Routers maintain routing information in *routing tables*. These tables contain the IP address of the hosts and routers on the networks to which the router is connected. The tables also contain pointers to these networks. When a router gets a packet, it consults its routing table to see if it lists the destination address in the header. If the table does not contain the destination address, the router forwards the packet to another router listed in its routing table. Refer to “Configuring Routers” on page 105 for detailed information on routers.

The following figure shows a network topology with three networks connected by two routers.

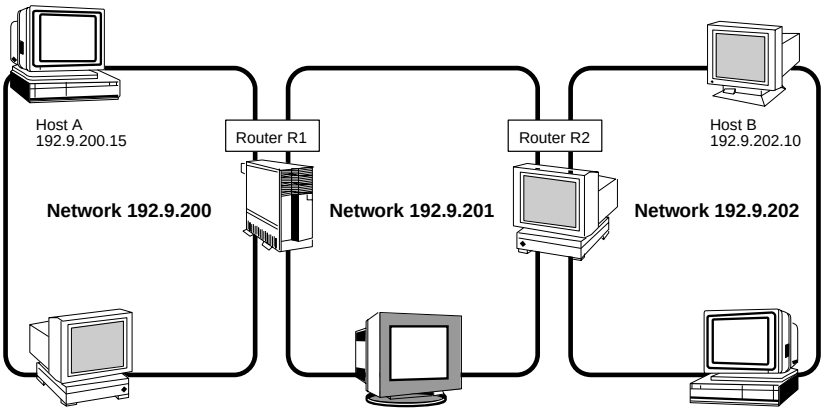


Figure 5–3 Three Interconnected Networks

Router R1 connects networks 192.9.200 and 192.9.201. Router R2 connects networks 192.9.201 and 192.9.202. If host A on network 192.9.200 sends a message to host B on network 192.9.202, this is what happens:

1. Host A sends a packet out over network 192.9.200. The packet header contains the IPv4 address of the recipient host B, 192.9.202.10.
2. None of the machines on network 192.9.200 has the IPv4 address 192.9.202.10. Therefore, router R1 accepts the packet.
3. Router R1 examines its routing tables. No machine on network 192.9.201 has the address 192.9.202.10. However, the routing tables do list router R2.
4. R1 then selects R2 as the “next hop” router and sends the packet to R2.
5. Because R2 connects network 192.9.201 to 192.9.202, it has routing information for host B. Router R2 then forwards the packet to network 192.9.202, where it is accepted by host B.

CHAPTER 6

TCP/IP Administration

This is the phase of network administration which involves setting up the network. This consists of assembling the hardware which makes up the physical part of the network, and configuring TCP/IP. This chapter explains how to configure TCP/IP, as well as how to troubleshoot TCP/IP after the network has been configured.

Before You Configure TCP/IP Task Map

Before configuring the TCP/IP software, you should have completed the tasks listed in the following table:

TABLE 6–1 Before You Configure TCP/IP Task Map

Description	For Instructions, Go To ...
Design the network topology.	See "Network Topology" on page 89.
Obtain a network number from your Internet addressing authority.	See "Designing Your IPv4 Addressing Scheme" on page 83.
Assemble the network hardware according to the topology designed and assured that the hardware is functioning.	See the hardware manuals and "Network Topology" on page 89.
Run any configuration software required by network interfaces and routers, if applicable.	See "Adding Routers" on page 89 and "Configuring Routers" on page 105 for information on routers.
Plan the IP addressing scheme for the network, including subnet addressing, if applicable.	See "Designing Your IPv4 Addressing Scheme" on page 83 and "IPv6 Addressing" on page 300.
Assign IP numbers and host names to all machines involved in the network.	See "Designing Your IPv4 Addressing Scheme" on page 83 and "IPv6 Addressing" on page 300.
Determine which name service your network uses: NIS, NIS+, DNS, or local files.	See *Solaris Naming Administration Guide*.

TABLE 6–1 Before You Configure TCP/IP Task Map *(continued)*

Description	For Instructions, Go To ...
Select domain names for your network, if applicable.	*Solaris Naming Administration Guide.*
Install the operating system on at least one machine on the prospective network.	See *Solaris Advanced Installation Guide.*

Determining Host Configuration Modes

One of your key functions as a network administrator is configuring TCP/IP to run on hosts and routers (if applicable). You can set up these machines to obtain configuration information from two sources: files on the local machine or files located on other machines on the network. Configuration information includes:

- Host name of a machine
- IP address of the machine
- Domain name to which the machine belongs
- Default router
- Netmask in use on the machine's network (if applicable)

A machine that obtains TCP/IP configuration information from local files is said to be operating in *local files* mode. A machine that obtains TCP/IP configuration information from a remote machine is said to be operating in *network client* mode.

Machines That Should Run in *Local Files* Mode

To run in *local files* mode, a machine must have local copies of the TCP/IP configuration files. These files are described in "TCP/IP Configuration Files" on page 125. The machine should have its own disk, though this is not strictly necessary.

Most servers should run in *local file* mode. This requirement includes:

- Network configuration servers
- NFS servers
- Name servers supplying NIS, NIS+, or DNS services
- Mail servers

Additionally, routers should run in *local files* mode.

Machines that exclusively function as print servers do not need to run in *local files* mode. Whether individual hosts should run in *local files* mode depends on the size of your network.

If you are running a very small network, the amount of work involved in maintaining these files on individual hosts is manageable. If your network serves hundreds of hosts, the task becomes difficult, even with the network divided into a number of administrative subdomains. Thus, for large networks, using *local files* mode is usually less efficient. On the other hand, because routers and servers must be self-sufficient, they should be configured in *local files* mode.

Network Configuration Servers

Network configuration servers are the machines that supply the TCP/IP configuration information to hosts configured in *network client* mode. These servers support three booting protocols:

- RARP – Reverse Address Resolution Protocol (RARP) maps known Ethernet addresses (48 bits) to IPv4 addresses (32 bits), the reverse of ARP. When you run RARP on a network configuration server, this enables hosts running in *network client* mode to obtain their IP addresses and TCP/IP configuration files from the server. The `in.rarpd` daemon enables RARP services. Refer to the `in.rarpd`(1M) man page for details.
- TFTP – Trivial File Transfer Protocol (TFTP) is an application that transfers files between remote machines. The `in.tftpd` daemon carries out TFTP services, enabling file transfer between network configuration servers and their network clients.
- bootparams – The bootparams protocol supplies parameters for booting that are required by clients booting off the network. The `rpc.bootparamd` daemon carries out these services.

Network configuration servers can also function as NFS file servers.

If you are going to configure any hosts as network clients, then you must also configure at least one machine on your network as a network configuration server. If your network is subnetted, then you must have at least one network configuration server for each subnet with network clients.

Machines That Are Network Clients

Any host that gets its configuration information from a network configuration server is said to be “operating” in *network client* mode. Machines configured as network clients do not require local copies of the TCP/IP configuration files.

Network client mode simplifies administration of large networks. It minimizes the number of configuration tasks that must be performed on individual hosts and assures that all machines on the network adhere to the same configuration standards.

You can configure *network client* mode on all types of computers, from fully standalone systems to dataless machines. Although it is possible to configure routers and servers in *network client* mode, *local files* mode is a better choice for these machines. Routers and servers should be as self-sufficient as possible.

Mixed Configurations

Because of the flexibility of the system, configurations are not limited to either an all-local-hosts mode or an all-network-client mode. The configuration of routers and servers typifies this, in that routers and servers should always be configured in local mode. For hosts, you can use any combination of local and network client mode you want.

Sample Network

The figure below shows the hosts of a fictional network with the network number 192.9.200. The network includes one network configuration server, the machine `sahara`. Machines `tenere` and `nubian` have their own disks and run in *local files* mode. Machine `faiyum` also has a disk but operates in *network client* mode.

Finally, the machine `timbuktu` is configured as a router. It includes two network interfaces, one named `timbuktu` on network 192.9.200 and one named `timbuktu-201` on network 192.9.201. Both networks are in the organizational domain `deserts.worldwide.com`. The domain uses local files as its name service.

Most examples in this chapter use the network shown in the following figure as their basis.

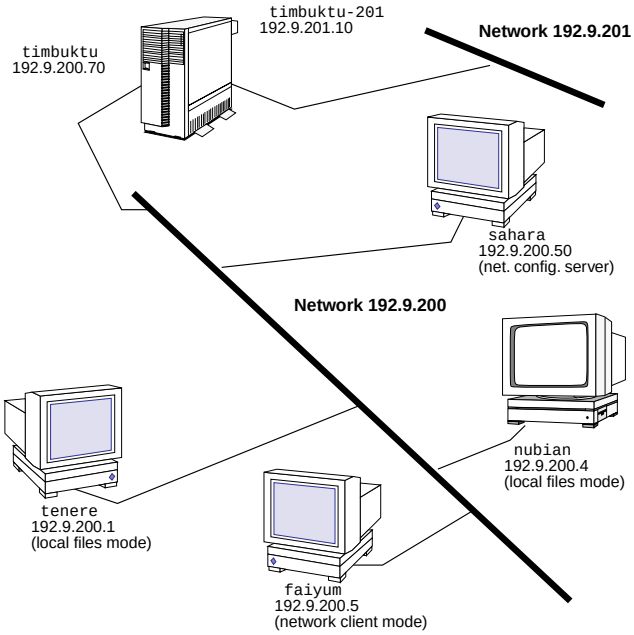


Figure 6–1 Hosts in a Sample Network

Adding a Subnet to a Network Task Map

If you are changing from a network that does not use subnets to one that is subnetted, perform the the tasks in the following table:

TABLE 6–2 Adding a Subnet to a Network Task Map

Description	For Instructions, Go To ...
1. Decide on the new subnet topology, including considerations for routers and locations of hosts on the subnets.	"Adding Routers" on page 89, "What Is Subnetting?" on page 131, and "Network Classes" on page 146.
2. Assign all subnet and host addresses.	"Setting Up an IP Addressing Scheme" on page 82 and "Parts of the IPv4 Address" on page 144.

TABLE 6–2 Adding a Subnet to a Network Task Map *(continued)*

Description	For Instructions, Go To ...
3. Modify the `/etc/inet/netmasks` file, if you are manually configuring TCP/IP, or supply the netmask to the Solaris installation program.	"`netmasks` Database" on page 131 and "Creating the Network Mask for IPv4 Addresses" on page 131.
4. Modify the `/etc/inet/hosts` and `/etc/inet/ipnodes` files on all hosts to reflect the new host addresses.	"`hosts` Database" on page 128 and "`ipnodes` Database" on page 131.
5. Reboot all machines.	

Network Configuration Procedures

Network software installation takes place along with the installation of the operating system software. At that time, certain IP configuration parameters must be stored in appropriate files so they can be read at boot time.

The procedure is a matter of creating or editing the network configuration files. How configuration information is made available to a machine's kernel depends on whether these files are stored locally (*local files* mode) or acquired from the network configuration server (*network client* mode).

Parameters supplied during network configuration are:

- IP address of each network interface on every machine
- Host names of each machine on the network. You can type the host name in a local file or a name service database.
- NIS, NIS+, or DNS domain name in which the machine resides, if applicable
- Default router addresses. You supply this only if you have a simple network topology with only one router attached to each network, or your routers don't run routing protocols such as the Router Discovery Server Protocol (RDISC) or the Router Information Protocol (RIP). (See "Routing Protocols" on page 143 for more information about these protocols.)
- Subnet mask (required only for networks with subnets)

This chapter contains information on creating and editing local configuration files. See the *Solaris Naming Administration Guide* for information on working with name service databases.

Network Configuration Task Map

TABLE 6–3 Network Configuration Task Map

Task	Description	For Instructions, Go To ...
Configure a host for local files mode	Involves editing the `nodename`, `hostname`, `hosts`, `defaultdomain`, `defaultrouter`, and `netmasks` files.	"How to Configure a Host for Local Files Mode" on page 100
Set up a network configuration server	Involves turning on the `in.tftp` daemon, and editing the `intetd.conf`, `hosts`, `ethers`, and `bootparams` files.	"How to Set Up a Network Configuration Server" on page 102
Configure a host for network client mode	Involves creating `hostname` file, editing the `hosts` file, and deleting the `nodename` and `defaultdomain` files, if they exist,	"How to Configure Hosts for Network Client Mode" on page 103
Specify a router for the network client	Involves editing the `defaultrouter` and `hosts` files.	"How to Specify a Router for the Network Client" on page 103

▼ How to Configure a Host for Local Files Mode

Use this procedure for configuring TCP/IP on a machine that runs in local files mode.

1. **Become superuser and change to the `/etc` directory.**

2. **Type the host name of the machine in the file `/etc/nodename`.**

   For example, if the name of the host is `tenere`, type `tenere` in the file.

3. **Create a file named `/etc/hostname.`*interface* for each network interface.**

   (The Solaris installation program automatically creates this file for the primary network interface.) Refer to "`/etc/hostname.`*interface* File" on page 126 for details. If you are using IPv6, see "IPv6 Network Interface Configuration File" on page 332.

4. **Type either the interface IP address or the interface name in each `/etc/hostname.`*interface* file.**

For example, create a file named `hostname.ie1`, and type either the IP address of the host's interface or the host's name.

5. **Edit the `/etc/inet/hosts` file to add:**

   a. **IP addresses that you have assigned to any additional network interfaces in the local machine, along with the corresponding host name for each interface.**

      The Solaris installation program has already created entries for the primary network interface and loopback address.

   b. **IP address or addresses of the file server, if the `/usr` file system is NFS mounted.**

   Note - The Solaris installation program creates the default `/etc/inet/hosts` for the local machine. If the file does not exist, create it as shown in "`hosts` Database" on page 128. Also, if you are using IPv6, see "`/etc/inet/ipnodes` File" on page 345.

   ---

6. **Type the host's fully qualified domain name in the `/etc/defaultdomain` file.**

   For example, suppose host `tenere` was part of the domain `deserts.worldwide.com`. Therefore, you would type: `deserts.worldwide.com` in `/etc/defaultdomain`. See "`/etc/defaultdomain` File" on page 127 for more information.

7. **Type the router's name in `/etc/defaultrouter`.**

   See "`/etc/defaultrouter` File" on page 127 for information about this file.

8. **Type the name of the default router and its IP addresses in `/etc/inet/hosts`.**

   Additional routing options are available. Refer to the discussion on routing options in "How to Configure Hosts for Network Client Mode" on page 103. You can apply these options to a local files mode configuration.

9. **If your network is subnetted, type the network number and the netmask in the file `/etc/inet/netmasks`.**

   If you have set up a NIS or NIS+ server, you can type `netmask` information in the appropriate database on the server as long as server and clients are on the same network.

10. **Reboot each machine on the network.**

▼ How to Set Up a Network Configuration Server

1. **Become superuser and change to the root directory of the prospective network configuration server.**

2. **Turn on the** `in.tftpd` **daemon by creating the directory** `/tftpboot`:

```
# mkdir /tftpboot
```

   This configures the machine as a TFTP, bootparams, and RARP server.

3. **Create a symbolic link to the directory.**

```
# ln -s /tftpboot/. /tftpboot/tftpboot
```

4. **Enable the** `tftp` **line in** `intetd.conf`**.**
   Check that the `/etc/inetd.conf` entry reads:

```
tftp dgram udp wait root /usr/sbin/in.tftpd in.tftpd -s /tftpboot
```

   This prevents `inettftpd( )` from retrieving any file other than one located in `/tftpboot`.

5. **Edit the** `hosts` **database, and add the host names and IP addresses for every client on the network.**

6. **Edit the** `ethers` **database, and create entries for every host on the network to run in network client mode.**

7. **Edit the** `bootparams` **database.**
   See “`bootparams` Database” on page 138. Use the wildcard entry or create an entry for every host that run in network client mode.

8. **Reboot the server.**

Information for setting up install servers and boot servers can be found in *Solaris Advanced Installation Guide*.

Configuring Network Clients

Network clients receive their configuration information from network configuration servers. Therefore, before you configure a host as a network client you must ensure that at least one network configuration server is set up for the network.

▼ How to Configure Hosts for Network Client Mode

Do the following on each host to be configured in network client mode:

1. **Become superuser.**

2. **Check the directory for the existence of an `/etc/nodename` file. If one exists, delete it.**

   Eliminating `/etc/nodename` causes the system to use the `hostconfig` program to obtain the host name, domain name, and router addresses from the network configuration server. See "Network Configuration Procedures" on page 99.

3. **Create the file `/etc/hostname.`*interface*, if it does not exist.**

   Make sure that the file is empty. An empty `/etc/hostname.`*interface* file causes the system to acquire the IP address from the network configuration server. If you are using IPv6, see "IPv6 Network Interface Configuration File" on page 332.

4. **Ensure that the `/etc/inet/hosts` file contains only the host name and IP address of the loopback network interface.**

   (See "Loopback Address" on page 129.) The file should not contain the IP address and host name for the local machine (primary network interface). If you are using IPv6, see "`/etc/inet/ipnodes` File" on page 345.

5. **Check for the existence of an `/etc/defaultdomain` file. If one exists, delete it.**

   The `hostconfig` program sets the domain name automatically. If you want to override the domain name set by `hostconfig`, type the substitute domain name in the file `/etc/defaultdomain`.

6. **Ensure that the search paths in the client's `/etc/nsswitch.conf` reflects the name service requirements for your network.**

▼ How to Specify a Router for the Network Client

1. **If you have only one router on the network and you want the network configuration server to specify its name automatically, ensure that the network client does not have a `/etc/defaultrouter` file.**

2. **To override the name of the default router provided by the network configuration server:**
   a. **Create** `/etc/defaultrouter` **on the network client.**
   b. **Type the host name and IP address of the machine you have designated as the default router.**
   c. **Add the host name and IP address of the designated default router to the network client's** `/etc/inet/hosts`**.**
3. **If you have multiple routers on the network, create** `/etc/defaultrouter` **on the network client, but leave it empty.**

Creating `/etc/defaultrouter` and leaving it empty causes one of the two dynamic routing protocols to run: ICMP Router Discovery protocol (RDISC), or Routing Information Protocol (RIP). The system first runs the program `in.rdisc`, which looks for routers that are running the router discovery protocol. If it finds one such router, `in.rdisc` continues to run and keeps track of the routers that are running the RDISC protocol.

If the system discovers that routers are not responding to the RDISC protocol, it uses RIP and runs the daemon `in.routed` to keep track of them.

Configuring Standard TCP/IP Services

Services such as `telnet`, `ftp`, and `rlogin` are started by the `inetd` daemon, which runs automatically at boot time. Like the name service ordering specified in `nsswitch.conf`, you can configure TCP/IP services in the file `/etc/inetd.conf` by using the `inetd` –`t` flag.

For example, you can use `inetd` to log the IP addresses of all incoming TCP connections (remote logins and `telnet`) as shown in the following procedure.

▼ How to Log the IP Addresses of All Incoming TCP Connections

1. **Become superuser.**
2. **Kill the** `inetd` **daemon.**
3. **Turn logging on by typing the following command:**

```
# /usr/sbin/inetd -t -s
```

The `t` switch turns on TCP connection-tracing in `inetd`.

Refer to the `inetd`(1M) and `inetd.conf`(4) man pages.

See *Solaris Naming Administration Guide* and *Solaris Naming Setup and Configuration Guide* for further information on name services.

Configuring Routers

TCP/IP's first requirement for a router is that the machine must have at least two network interfaces installed, as introduced in "Network Interfaces" on page 56. As long as one of the network interfaces is not disabled, the router automatically "talks" to the RDISC and RIP protocols. These protocols keep track of routers on the network and advertise the router to the hosts on the network.

After the router is physically installed on the network, configure it to operate in *local files* mode, as described in "How to Configure a Host for Local Files Mode" on page 100. This ensures that routers will boot in case the network configuration server is down. Remember that, unlike a host, a router has at least two interfaces to configure.

Configuring Routers Task Map

TABLE 6–4 Configuring Routers Task Map

Task	Description	For Instructions, Go To ...
Configure a machine as a router.	Involves creating `hostname` and `hosts` file and adding addresses.	"How to Configure a Machine as a Router" on page 106
Select static routing on a host that is a network client.	Involves adding an entry into the `defaultrouter` file.	"How to Select Static Routing on a Host That Is a Nework Client" on page 107

TABLE 6–4 Configuring Routers Task Map *(continued)*

Task	Description	For Instructions, Go To ...
Select dynamic routing on a host that is a network client.	Involves editing entries in the `defaultrouter` file.	"How to Select Dynamic Routing on a Host That Is a Nework Client" on page 107
Force a machine to be a router.	Involves creating a `gateways` file.	"How to Force a Machine to Be a Router" on page 108

Configuring Both Router Network Interfaces

Because a router provides the interface between two or more networks, you must assign a unique name and IP address to each of the router's network interface cards. Thus, each router has a host name and IP address associated with its primary network interface, plus at least one more unique name and IP address for each additional network interface.

▼ How to Configure a Machine as a Router

1. **Become superuser on the machine to be configured as a router.**

2. **Create an `/etc/hostname.`*interface* file for each network interface installed.**

   For example, create `hostname.ie0` and `hostname.ie1`. (See "`/etc/hostname.`*interface* File" on page 126 for more information.) If you are using IPv6, see "IPv6 Network Interface Configuration File" on page 332.

3. **In each file, type the host name you have selected for that interface.**

   For example, you could type the name `timbuktu` in the file `hostname.ie0`, then type the name `timbuktu-201` in the file `hostname.ie1`. Both interfaces would be located on the same machine.

4. **Type the host name and IP address of each interface into `/etc/inet/hosts`.**

   For example:

```
192.9.200.20     timbuktu       #interface for network 192.9.200
192.9.201.20     timbuktu-201   #interface for network 192.9.201
192.9.200.9      gobi
```

(continued)

```
192.9.200.10     mojave
192.9.200.110    saltlake
192.9.200.12     chilean
```

The interfaces `timbuktu` and `timbuktu-201` are on the same machine. Notice that the network address for `timbuktu-201` is different from that of `timbuktu`. That is because the medium for network 192.9.201 is connected to the `timbuktu-201` network interface while the media for network 192.9.200 is connected to the `timbuktu` interface. If you are using IPv6, see "`/etc/inet/ipnodes` File" on page 345.

5. **If the router is connected to any subnetted network, edit `/etc/inet/netmasks` and type the local network number (129.9.0.0, for example) and associated netmask number (255.255.255.0, for example).**

The startup script determines whether to start up a routing protocol (RIP or RDISC) on the machine or use static routing.

▼ How to Select Static Routing on a Host That Is a Nework Client

1. **Become superuser on the host.**
2. **Add an entry for a router on the network into the `/etc/defaultrouter` file.**

(See "`/etc/defaultrouter` File" on page 127.) A single static default route is then installed in the routing table. Under this condition, the host does not run any dynamic routing protocol (such as RIP and RDISC).

▼ How to Select Dynamic Routing on a Host That Is a Nework Client

1. **Become superuser on the host.**
2. **Ensure that the `/etc/defaultrouter` file is empty.**

   If it is empty, this forces a network client to select a dynamic routing protocol.

The type of dynamic routing used is selected according to the following criteria:

- If the `/usr/sbin/in.rdisc` program exists, the startup script starts `in.rdisc`. Any router on the network that is running RDISC then responds to any RDISC queries from the host. If at least one router responds, the host selects RDISC as its routing protocol.
- If the network router is not running RDISC or fails to respond to the RDISC queries, then `in.rdisc` on the host exits. The host then starts `in.routed`, which runs RIP.

▼ How to Force a Machine to Be a Router

You can force a machine that has only one `/etc/hostname.`*interface* file (by default a host) to be a router.

1. **Become superuser on the machine.**
2. **Create a file named `/etc/gateways` and leave it empty.**

This is important if you decide to configure PPP links, as explained in "Routing Considerations" on page 421.

Creating a Multihomed Host

By default, TCP/IP considers any machine with multiple network interfaces to be a router. However, you can change a router into a *multihomed host*—a machine with more than one network interface that does not run routing protocols or forward IP packets. You typically configure the following types of machines as multihomed hosts:

- NFS servers, particularly large data centers, can be attached to more than one network in order to share files among a large pool of users. These servers don't need to maintain routing tables.
- Database servers can have multiple network interfaces for the same reason as NFS servers—to provide resources to a large pool of users.
- Firewall gateways are machines that provide the connection between a company's network and public networks such as the Internet. Administrators set up firewalls as a security measure. When configured as a firewall, the host will not pass packets between the networks attached to it. On the other hand, it can still provide standard TCP/IP services, such as `ftp` or `rlogin`, to authorized users.

Since TCP/IP considers any machine with multiple network interfaces to be a router, you need to perform a few operations to turn it into a multihomed host.

▼ How to Create a Multihomed Host

1. **Become superuser on the prospective multihomed host.**

2. **Create an** `/etc/hostname.`***interface*** **file for each additional network interface installed in the machine.**

3. **Type:**

   % **`touch /etc/notrouter`**

   This creates an empty file called `/etc/notrouter`.

4. **Reboot the machine.**

When the machine reboots, the startup script looks for the presence of the `/etc/notrouter` file. If the file exists, the startup script does not run `in.routed -s` or `in.rdisc -r`, and does not turn on IP forwarding on all interfaces configured "up" by `ifconfig`. This happens regardless of whether an `/etc/gateways` file exists. Thus the machine is now a multihomed host.

Turning On Space-Saving Mode

Space-saving mode provides the host with a table that contains only the default routes. On a host, `in.routed` runs with space saving mode turned off by default.

If you do not want the host to have a full routing table (which provides increased protection against misconfigured routers), turn space saving mode on.

▼ How to Turn On Space-Saving Mode

1. **Become superuser on the host.**

2. **Edit the** `/etc/rc2.d/S69inet` **startup script by changing the line:**

```
/usr/sbin/in.routed -q
```

   to

```
/usr/sbin/in.routed -q -S
```

Turning Off ICMP Router Discovery

For reasons involving router reliability, you might not want your hosts to use RDISC. If the automatic selection of RIP rather than RDISC by a host is to work reliably, the routers in the network (particularly those running RDISC) must also work reliably.

If your routers are not running RDISC and you install a single Solaris router, by default all hosts connected to that router rely on it alone. To have the hosts on that network use the other routers as well, turn off RDISC on the new router.

Turning Off ICMP Router Discovery Task Map

TABLE 6–5 Turning Off ICMP Router Discovery Task Map

Task	Description	For Instructions, Go To ...
Turn off ICMP router discovery on the host.	Involves changing the name of the host's `in.rdisc` file.	"`netmasks` Database" on page 131
Turn off ICMP router discovery on the router.	Involves changing the name of the router's `in.rdisc` file.	"What Is Subnetting?" on page 131

▼ How to Turn Off ICMP Router Discovery on the Host

1. **Become superuser on the host.**
2. **Change the name of the host's `/usr/sbin/in.rdisc` to some other name, such as `/usr/sbin/in.rdisc.saved`.**
3. **Reboot the host.**

▼ How to Turn Off ICMP Router Discovery on the Router

1. **Become superuser on the router.**

2. **Change the name of the router's `/usr/bin/in.rdisc` file to some other file name.**

3. **Reboot the router.**

General Troubleshooting Tips

One of the first signs of trouble on the network is a loss of communications by one or more hosts. If a host refuses to come up at all the first time it is added to the network, the problem might lie in one of the configuration files, or in the network interface. If a single host suddenly develops a problem, the network interface might be the cause. If the hosts on a network can communicate with each other but not with other networks, the problem could lie with the router, or it could lie in another network.

You can use the `ifconfig` program to obtain information on network interfaces and `netstat` to display routing tables and protocol statistics. Third-party network diagnostic programs provide a number of troubleshooting utilities. Refer to third-party documentation for information.

Less obvious are the causes of problems that degrade performance on the network. For example, you can use tools like `ping` to quantify problems like the loss of packets by a host.

Running Software Checks

If the network has trouble, some actions that you can take to diagnose and fix software-related problems include:

1. Using the `netstat` command to display network information.
2. Checking the `hosts` database (and `ipnodes` if you are using IPv6) to make sure that the entries are correct and up to date.
3. If you are running RARP, checking the Ethernet addresses in the `ethers` database to make sure that the entries are correct and up to date.
4. Trying to connect by `telnet` to the local host.
5. Ensuring that the network daemon `inetd` is running. To do this, log in as superuser and type:

   # **`ps -ef | grep inetd`**

Here is an example of output displayed if the `inetd` daemon is running:

```
root 57 1 0 Apr 04 ? 3:19 /usr/sbin/inetd -s
root 4218 4198 0 17:57:23 pts/3 0:00 grep inetd
```

`ping` Command

Use the `ping` command to find out whether there is IP connectivity to a particular host. The basic syntax is:

`/usr/sbin/ping` *host [timeout]*

where *host* is the host name of the machine in question. The optional *timeout* argument indicates the time in seconds for `ping` to keep trying to reach the machine—20 seconds by default. The `ping`(1M) man page describes additional syntaxes and options.

When you run `ping`, the ICMP protocol sends a datagram to the host you specify, asking for a response. (ICMP is the protocol responsible for error handling on a TCP/IP network. See "ICMP Protocol" on page 71 for details.)

`ping` Command Task Map

TABLE 6–6 `ping` Command Task Map

Task	Description	For Instructions, Go To ...
Determine if a host is running.	Involves pinging the hostname.	"Network Databases and `nsswitch.conf` File" on page 134
Determine if a host is losing packets.	Involves using the `–s` option of the `ping` command.	"How Name Services Affect Network Databases" on page 134

▼ How to Determine if a Host Is Running

♦ **On the command line, type the following command.**

```
% ping hostname
```

If host *hostname* is up, this message is displayed:

```
hostname is alive
```

This indicates that *hostname* responded to the ICMP request. However, if *hostname* is down or cannot receive the ICMP packets, you receive the following response from `ping`:

```
no answer from hostname
```

▼ How to Determine if a Host Is Losing Packets

If you suspect that a machine might be losing packets even though it is running, you can use the `s` option of `ping` to try to detect the problem.

♦ **On the command line, type the following command.**

```
% ping -s hostname
```

`ping` continually sends packets to *hostname* until you send an interrupt character or a timeout occurs. The responses on your screen will resemble:

```
PING elvis: 56 data bytes
64 bytes from 129.144.50.21: icmp_seq=0. time=80. ms
64 bytes from 129.144.50.21: icmp_seq=1. time=0. ms
64 bytes from 129.144.50.21: icmp_seq=2. time=0. ms
64 bytes from 129.144.50.21: icmp_seq=3. time=0. ms
.
.
.
----elvis PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms) min/avg/max = 0/20/80
```

The packet-loss statistic indicates whether the host has dropped packets.

If `ping` fails, check the status of the network reported by `ifconfig` and `netstat`, as described in “`ifconfig` Command” on page 114 and “`netstat` Command” on page 115.

`ifconfig` Command

The `ifconfig` command displays information about the configuration of an interface that you specify. (Refer to the `ifconfig`(1M) man page for details.) The syntax of `ifconfig` is:

`ifconfig` *interface-name [protocol_family]*

`ifconfig` Command Task Map

TABLE 6–7 `ifconfig` Command Task Map

Task	Description	For Instructions, Go To ...
Get information about a specific interface.	Involves using the `ifconfig` command.	“How to Get Information About a Specific Interface” on page 114
Get information about all interfaces on a network.	Involves using the –`a` option of the `ifconfig` command.	“`nsswitch.conf` File — Specifying Which Name Service to Use” on page 136

▼ How to Get Information About a Specific Interface

1. **Become superuser.**
2. **On the command line, type the following command.**

```
# ifconfig interface
```

For an `le0` interface, your output resembles the following:

```
le0: flags=863<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 129.144.44.140 netmask ffffff00 broadcast 129.144.44.255
ether 8:0:20:8:el:fd
```

The flags section just given shows that the interface is configured “up,” capable of broadcasting, and not using “trailer” link level encapsulation. The `mtu` field tells you that this interface has a maximum transfer size of 1500 octets. Information on the second line includes the IP address of the host you are using, the netmask being currently used, and the IP broadcast address of the interface. The third line gives the machine address (Ethernet, in this case) of the host.

▼ How to Get Information About All Interfaces on a Network

A useful `ifconfig` option is `–a`, which provides information on all interfaces on your network.

1. **Become superuser.**

2. **On the command line, type the following command.**

```
# ifconfig -a interface
```

This produces, for example:

```
le0:  flags=49<UP,LOOPBACK,RUNNING> mtu 8232
     inet 127.144.44.140 netmask ff000000
le0:flags=863<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
     inet 129.144.44.140 netmask ffffff00 broadcast 129.144.44.255
ether 8:0:20:8:el:fd
```

Output that indicates an interface is not running might mean a problem with that interface. In this case, see the `ifconfig`(1M) man page.

`netstat` Command

The `netstat` command generates displays that show network status and protocol statistics. You can display the status of TCP and UDP endpoints in table format, routing table information, and interface information.

`netstat` displays various types of network data depending on the command line option selected. These displays are the most useful for system administration. The syntax for this form is:

`netstat [-m] [-n] [-s] [-i | -r] [-f` *address_family*`]`

The most frequently used options for determining network status are: `s`, `r`, and `i`. See the `netstat`(1M) man page for a description of the options.

`netstat` Command Task Map

TABLE 6–8 `netstat` Command Task Map

Task	Description	For Instructions, Go To ...
Display statistics by protocol.	Involves using the –`s` option of the `netstat` command.	"How to Display Statistics by Protocol" on page 116
Display network interface status.	Involves using the –`i` option of the `netstat` command.	"How to Display Network Interface Status" on page 117
Display routing table status.	Involves using the –`r` option of the `netstat` command.	"How to Display Routing Table Status" on page 118

▼ How to Display Statistics by Protocol

The `netstat` –`s` option displays per protocol statistics for the UDP, TCP, ICMP, and IP protocols.

♦ **On the command line, type the following command.**

```
% netstat -s
```

The result resembles the display shown in the example below. (Parts of the output have been truncated.) The information can indicate areas where a protocol is having problems. For example, statistical information from ICMP can indicate where this protocol has found errors.

```
UDP

      udpInDatagrams      =  39228     udpOutDatagrams     =  2455
      udpInErrors         =     0

TCP

      tcpRtoAlgorithm     =     4      tcpMaxConn          =    -1
      tcpRtoMax           = 60000      tcpPassiveOpens     =     2
      tcpActiveOpens      =     4      tcpEstabResets      =     1
      tcpAttemptFails     =     3      tcpOutSegs          =   315
   .
   .
IP

      ipForwarding        =     2      ipDefaultTTL        =   255
      ipInReceives        =  4518      ipInHdrErrors       =     0
   .
   .
ICMP

      icmpInMsgs          =     0      icmpInErrors        =     0
      icmpInCksumErrs     =     0      icmpInUnknowns      =     0
   .
   .

IGMP:

0 messages received
0 messages received with too few bytes
0 messages received with bad checksum
0 membership queries received
0 membership queries received with invalid field(s)
0 membership reports received
0 membership reports received with invalid field(s)
0 membership reports received for groups to which we belong
0 membership reports sent
```

▼ How to Display Network Interface Status

The `i` option of `netstat` shows the state of the network interfaces that are configured with the machine where you ran the command.

♦ **On the command line, type the following command:**

```
% netstat -i
```

Here is a sample display produced by `netstat –i`:

```
Name Mtu  Net/Dest     Address   Ipkts    Ierrs Opkts    Oerrs  Collis  Queue
le0  1500 b5-spd-2f-cm tatra     14093893 8492  10174659 1119   2314178   0
lo0  8232 loopback     localhost 92997622 5442  12451748 0      775125    0
```

Using this display, you can find out how many packets a machine thinks it has transmitted and received on each network. For example, the input packet count (`Ipkts`) displayed for a server can increase each time a client tries to boot, while the output packet count (`Opkts`) remains steady. This suggests that the server is seeing the boot request packets from the client, but does not realize it is supposed to respond to them. This might be caused by an incorrect address in the `hosts`, `ipnodes`, or `ethers` database.

On the other hand, if the input packet count is steady over time, it means that the machine does not see the packets at all. This suggests a different type of failure, possibly a hardware problem.

▼ How to Display Routing Table Status

The `–r` option of `netstat` displays the IP routing table.

- **On the command line, type the following command.**

```
% netstat -r
```

Here is a sample display produced by `netstat –r` run on machine `tenere`:

```
Routing tables
Destination   Gateway Flags Refcnt Use   Interface
temp8milptp   elvis   UGH   0      0
irmcpeb1-ptp0 elvis   UGH   0      0
route93-ptp0  speed   UGH   0      0
mtvb9-ptp0    speed   UGH   0      0
                .
mtnside       speed   UG    1      567
ray-net       speed   UG    0      0
mtnside-eng   speed   UG    0      36
mtnside-eng   speed   UG    0      558
mtnside-eng   tenere  U     33     190248  le0
```

The first column shows the destination network, the second the router through which packets are forwarded. The `U` flag indicates that the route is up; the `G` flag indicates that the route is to a gateway. The `H` flag indicates that the destination is a fully qualified host address, rather than a network.

The Refcnt column shows the number of active uses per route, and the Use column shows the number of packets sent per route. Finally, the Interface column shows the network interface that the route uses.

Logging Network Problems

If you suspect a routing daemon malfunction, you can log its actions, including all packet transfers when you start up the routed daemon.

▼ How to Log Network Problems

1. **Become superuser.**

2. **Create a log file of routing daemon actions by typing the following command at a command line prompt.**

```
# /usr/sbin/in.routed /var/logfilename
```

Caution - On a busy network, this can generate almost continuous output.

Displaying Packet Contents

You can use snoop to capture network packets and display their contents. Packets can be displayed as soon as they are received, or saved to a file. When snoop writes to an intermediate file, packet loss under busy trace conditions is unlikely. snoop itself is then used to interpret the file. For information about using the snoop command, refer to the snoop(1M) man page.

The snoop command must be run by root (#) to capture packets to and from the default interface in promiscuous mode. In summary form, only the data pertaining to the highest-level protocol is displayed. For example, an NFS packet only displays NFS information. The underlying RPC, UDP, IP, and Ethernet frame information is suppressed but can be displayed if either of the verbose options is chosen.

The snoop capture file format is described in RFC 1761. To access, use your favorite web browser with the URL: http://ds.internic.net/rfc/rfc1761.txt.

`snoop server client rpc rstatd` collects all RPC traffic between a client and server, and filters it for `rstatd`.

Displaying Packet Contents Task Map

TABLE 6–9 Displaying Packet Contents Task Map

Task	Description	For Instructions, Go To ...
Check all packets from your system.	Involves using the `netstat` and `snoop` commands and interpreting the results.	"How to Check All Packets from Your System" on page 120
Capture `snoop` results to a file.	Involves using the `–o` option of the `snoop` command.	"How to Capture `snoop` Results to a File" on page 121
Check packets between server and client.	Involves saving the results of the `snoop` command to a file and inspecting the results.	"How to Check Packets Between Server and Client" on page 122

▼ How to Check All Packets from Your System

1. **Become superuser.**

2. **Type the following command at the command line prompt to find the interfaces attached to the system.**

```
# netstat -i
```

Snoop normally uses the first non-loopback device (`le0`).

3. **Type** `snoop`**.**

Use Ctl-C to halt the process.

```
# snoop
Using device /dev/le (promiscuous mode)
     maupiti -> atlantic-82  NFS C GETATTR FH=0343
 atlantic-82 -> maupiti      NFS R GETATTR OK
     maupiti -> atlantic-82  NFS C GETATTR FH=D360
 atlantic-82 -> maupiti      NFS R GETATTR OK
```

(continued)

```
    maupiti -> atlantic-82  NFS C GETATTR FH=1A18
atlantic-82 -> maupiti      NFS R GETATTR OK
    maupiti -> (broadcast)  ARP C Who is 120.146.82.36, npmpk17a-82 ?
```

4. Interpret the results.

In the example, client `maupiti` transmits to server `atlantic-82` using NFS file handle 0343. `atlantic-82` acknowledges with OK. The conversation continues until `maupiti` broadcasts an ARP request asking who is `120.146.82.36`?

This example demonstrates the format of `snoop`. The next step is to filter `snoop` to capture packets to a file.

Interpret the capture file using details described in RFC 1761. To access, use your favorite web browser with the URL: `http://ds.internic.net/rfc/rfc1761.txt`

▼ How to Capture `snoop` Results to a File

1. Become superuser.

2. On the command line, type the following command.

```
# snoop -o filename
```

For example:

```
# snoop -o /tmp/cap
Using device /dev/le (promiscuous mode)
30 snoop: 30 packets captured
```

This has captured 30 packets in a file `/tmp/cap`. The file can be anywhere with enough disk space. The number of packets captured is displayed on the command line, enabling you to press Ctl-C to abort at any time.

`snoop` creates a noticeable networking load on the host machine, which can distort the results. To see reality at work, run `snoop` from a third system, (see the next section).

3. **On the command line, type the following command to inspect the file.**

```
# snoop -i filename
```

For example:

```
# snoop -i /tmp/cap

1  0.00000 frmpk17b-082 -> 224.0.0.2    IP  D=224.0.0.2 S=129.146.82.1 LEN=32, ID=0
2  0.56104        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
3  0.16742  atlantic-82 -> (broadcast)  ARP C Who is 129.146.82.76, honeybea ?
4  0.77247        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
5  0.80532 frmpk17b-082 -> (broadcast)  ARP C Who is 129.146.82.92, holmes ?
6  0.13462        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
7  0.94003        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
8  0.93992        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
9  0.60887        towel -> (broadcast)  ARP C Who is 129.146.82.35, udmpk17b-82 ?
10 0.86691  nimpk17a-82 -> 129.146.82.255 RIP R (1 destinations)
```

Refer to specific protocol documentation for detailed analysis and recommended parameters for ARP, IP, RIP and so forth. Searching the Web is a good place to look at RFCs.

▼ How to Check Packets Between Server and Client

1. **Establish a snoop system off a hub connected to either the client or server.**

   The third system (the snoop system) sees all the intervening traffic, so the snoop trace reflects reality on the wire.

2. **Become superuser.**

3. **On the command line, type** `snoop` **with options and save to a file.**

4. **Inspect and interpret results.**

   Look at RFC 1761 for details of the snoop capture file. To access, use your favorite web browser with the URL: `http://ds.internic.net/rfc/rfc1761.txt`

Use `snoop` frequently and consistently to get a feel for normal system behavior. For assistance in analyzing packets, look for recent white papers and RFCs, and seek the advice of an expert in a particular area, such as NFS or YP. For details on using `snoop` and its options, refer to the `snoop`(1M) man page.

Displaying Routing Information

Use the `traceroute` utility to trace the route an IP packet follows to some internet host. The `traceroute` utility utilizes the IP protocol (time to live) `ttl` field and attempts to elicit an ICMP `TIME_EXCEEDED` response from each gateway along the path, and the response `PORT_UNREACHABLE` (or `ECHO_REPLY`) from the destination host. The `traceroute` utility starts sending probes with a `ttl` of one and increases by one until it gets to the intended host or has passed through a maximum number of intermediate hosts.

The `traceroute` utility is especially useful for determining routing misconfiguration and routing path failures. If a particular host is unreachable, you can use the `traceroute` utility to see what path the packet follows to the intended host and where possible failures might occur.

The `traceroute` utility also displays the round trip time for each gateway along the path to the target host. This information can be useful for analyzing where traffic is slow between the two hosts.

▼ How to Run the Traceroute Utility

♦ **On the command line, type the following command.**

```
% traceroute destination-hostname
```

For details of the `traceroute` utility see the `traceroute`(1M) man page.

Example—`traceroute` Utility

The following sample of the `traceroute` command shows the 7–hop path a packet follows from the host `istanbul` to the host `sanfrancisco` along with the times for a packet to traverse each hop.

```
istanbul% traceroute sanfrancisco
 traceroute: Warning: Multiple interfaces found; using 172.31.86.247 @ le0
 traceroute to sanfrancisco (172.29.64.39), 30 hops max, 40 byte packets
  1  frbldg7c-86 (172.31.86.1)  1.516 ms  1.283 ms  1.362 ms
  2  bldg1a-001 (172.31.1.211)  2.277 ms  1.773 ms  2.186 ms
  3  bldg4-bldg1 (172.30.4.42)  1.978 ms  1.986 ms  13.996 ms
  4  bldg6-bldg4 (172.30.4.49)  2.655 ms  3.042 ms  2.344 ms
  5  ferbldg11a-001 (172.29.1.236)  2.636 ms  3.432 ms  3.830 ms
  6  frbldg12b-153 (172.29.153.72)  3.452 ms  3.146 ms  2.962 ms
```

(continued)

(Continuation)

```
 7  sanfrancisco (172.29.64.39)  3.430 ms  3.312 ms  3.451 ms
```

CHAPTER 7

TCP/IP Network Reference

This chapter provides TCP/IP network reference information about TCP/IP configuration files, including the types, their purpose, and the format of the file entries. The existing network databases are also described in detail.

The chapter also shows how the structure of IPv4 addresses are derived based on defined network classifications and subnet numbers.

TCP/IP Configuration Files

Each machine on the network gets its TCP/IP configuration information from the following TCP/IP configuration files and network databases:

- `/etc/hostname.`*interface* file
- `/etc/nodename` file
- `/etc/defaultdomain` file
- `/etc/defaultrouter` file (optional)
- `hosts` database

- `ipnodes` database
- `netmasks` database (optional)

The Solaris installation program creates these files as part of the installation process. You can also edit the files manually, as explained in this section. The `hosts` and `netmasks` databases are two of the network databases read by the name services available on Solaris networks. "Network Databases and `nsswitch.conf` File" on page 134 describes the concept of network databases in detail. For information on the `ipnodes` file, see "`/etc/inet/ipnodes` File" on page 345.

`/etc/hostname.`*interface* File

This file defines the network interfaces on the local host for IPv4. At least one `/etc/hostname.`*interface* file should exist on the local machine. The Solaris installation program creates this file for you. In the file name, *interface* is replaced by the device name of the primary network interface.

Note - If you add a new network interface to your system after the initial Solaris software installation, you must create an `/etc/hostname.`*interface* file for that interface, add the interface's IP address to the `/etc/inet/hosts` file, and reboot the system with the `–r` option. See substeps within "How to Configure a Host for Local Files Mode" on page 100 for instructions. Also, in order for the Solaris software to recognize and use the new network interface, you need to load the interface's device driver into the appropriate directory. Refer to the documentation that comes with the new network interface for the appropriate *interface* name and device driver instructions.

The file contains only one entry: the host name or IPv4 address associated with the network interface. For example, suppose `smc0` is the primary network interface for a machine called `tenere`. Its `/etc/hostname.`*interface* file would have the name `/etc/hostname.smc0`; the file would contain the entry `tenere`.

Files for Multiple Network Interfaces

If a machine contains more than one network interface, you must create additional `/etc/hostname.`*interface* files for the additional network interfaces. You must create these files with a text editor; the Solaris installation program does not create them for you.

For example, consider the machine `timbuktu`, shown in Figure 6–1. It has two network interfaces and functions as a router. The primary network interface `le0` is connected to network 192.9.200. Its IP address is 192.9.200.70, and its host name is `timbuktu`. The Solaris installation program creates the file `/etc/hostname.le0` for the primary network interface and enters the host name `timbuktu` in the file.

The second network interface is `le1`; it is connected to network 192.9.201. Although this interface is physically installed on machine `timbuktu`, it must have a separate IPv4 address. Therefore, you have to manually create the `/etc/hostname.le1` file for this interface; the entry in the file would be the router's name, `timbuktu-201`.

`/etc/hostname6.`*interface* File

IPv6 uses the file `/etc/hostname6.`*interface* at start-up to automatically define network interfaces in the same way IPv4 uses `/etc/hostname.`*interface*. At least one `/etc/hostname.` or `/etc/hostname6.` file should exist on the local machine. The Solaris installation program creates these files for you. In the file name, replace *interface* with the device name of the primary network interface. For more information about the `/etc/hostname6.`*interface* file, see "IPv6 Network Interface Configuration File" on page 332.

`/etc/nodename` File

This file should contain one entry: the host name of the local machine. For example, on machine `timbuktu`, the file `/etc/nodename` would contain the entry `timbuktu`.

`/etc/defaultdomain` File

This file should contain one entry, the fully qualified domain name of the administrative domain to which the local host's network belongs. You can supply this name to the Solaris installation program or edit the file at a later date.

In Figure 6–1, the networks are part of the domain `deserts.worldwide`, which was classified as a `.com` domain. Therefore, `/etc/defaultdomain` should contain the entry `deserts.worldwide.com`. For more information on network domains, refer to the *Solaris Naming Administration Guide*.

`/etc/defaultrouter` File

This file should contain an entry for each router directly connected to the network. The entry should be the name for the network interface that functions as a router between networks.

In Figure 6–1, the network interface `le1` connects machine `timbuktu` with network 192.9.201. This interface has the unique name `timbuktu-201`. Thus, the machines on network 192.9.200 that are configured in local files mode have the name `timbuktu-201` as the entry in `/etc/defaultrouter`.

hosts Database

The hosts database contains the IPv4 addresses and host names of machines on your network. If you use the NIS, NIS+, or DNS name services, the hosts database is maintained in a database designated for host information. For example, on a network running NIS+, the hosts database is maintained in the host table.

If you use local files for name service, the hosts database is maintained in the /etc/inet/hosts file. This file contains the host names and IPv4 addresses of the primary network interface, other network interfaces attached to the machine, and any other network addresses that the machine must know about.

Note - For compatibility with BSD-based operating systems, the file /etc/hosts is a symbolic link to /etc/inet/hosts.

/etc/inet/hosts File Format

The /etc/inet/hosts file uses the basic syntax that follows. (Refer to the hosts(4) man page for complete syntax information.)

IPv4-address hostname [nicknames] [#comment]

IPv4-address contains the IPv4 address for each interface that the local host must recognize.

hostname contains the host name assigned to the machine at setup, plus the host names assigned to additional network interfaces that the local host must recognize.

[nickname] is an optional field containing a nickname for the host.

[# comment] is an optional field for a comment.

Initial /etc/inet/hosts File

When you run the Solaris installation program on a machine, it sets up the initial /etc/inet/hosts file. This file contains the minimum entries that the local host requires: its loopback address, its IPv4 address, and its host name.

For example, the Solaris installation program might create the following /etc/inet/hosts file for machine tenere shown in Figure 6–1:

EXAMPLE 7–1 /etc/inet/hosts File for Machine ahaggar

```
127.0.0.1     localhost         loghost    #loopback address
192.9.200.3   tenere                       #host name
```

Loopback Address

In Example 7–1, the IPv4 address 127.0.0.1 is the *loopback address*, the reserved network interface used by the local machine to allow interprocess communication so that it sends packets to itself. The `ifconfig` command uses the loopback address for configuration and testing, as explained in "`ifconfig` Command" on page 114. Every machine on a TCP/IP network must use the IP address 127.0.0.1 for the local host.

Host Name

The IPv4 address 192.9.200.1 and the name `tenere` are the address and host name of the local machine. They are assigned to the machine's primary network interface.

Multiple Network Interfaces

Some machines have more than one network interface, because they ar eeither routers or multihomed hosts. Each additional network interface attached to the machine requires its own IPv4 address and associated name. When you configure a router or multihomed host, you must add this information manually to the router's `/etc/inet/hosts` file. (See "Configuring Routers" on page 105 for more information on setting up routers and multihomed hosts.)

Example 7–2 is the `/etc/inet/hosts` file for machine `timbuktu` shown in Figure 6–1.

EXAMPLE 7–2 `/etc/inet/hosts` File for Machine `timbuktu`

```
127.0.0.1      localhost     loghost
192.9.200.70   timbuktu      #This is the local host name
192.9.201.10   timbuktu-201  #Interface to network 192.9.201
```

With these two interfaces, `timbuktu` connects networks 192.9.200 and 192.9.201 as a router.

How Name Services Affect the `hosts` Database

The NIS, NIS+, and DNS name services maintain host names and addresses on one or more servers. These servers maintain `hosts` databases containing information for every host and router (if applicable) on the servers' network. Refer to the *Solaris Naming Administration Guide* for more information about these services.

When Local Files Provide Name Service

On a network using local files for name service, machines running in local files mode consult their individual `/etc/inet/hosts` files for IPv4 addresses and host names of other machines on the network. Therefore, their `/etc/inet/hosts` files must contain the:

- Loopback address
- IPv4 address and host name of the local machine (primary network interface)
- IPv4 address and host name of additional network interfaces attached to this machine, if applicable
- IPv4 addresses and host names of all hosts on the local network
- IPv4 addresses and host names of any routers this machine must know about, if applicable
- IPv4 address of any machine your machine wants to refer to by its host name

The figure below shows the `/etc/inet/hosts` file for machine `tenere`, a machine that runs in local files mode. Notice that the file contains the IPv4 addresses and host names for every machine on the 192.9.200 network. It also contains the IPv4 address and interface name `timbuktu-201`, which connects the 192.9.200 network to the 192.9.201 network.

A machine configured as a network client uses the local `/etc/inet/hosts` file for its loopback address and IPv4 address.

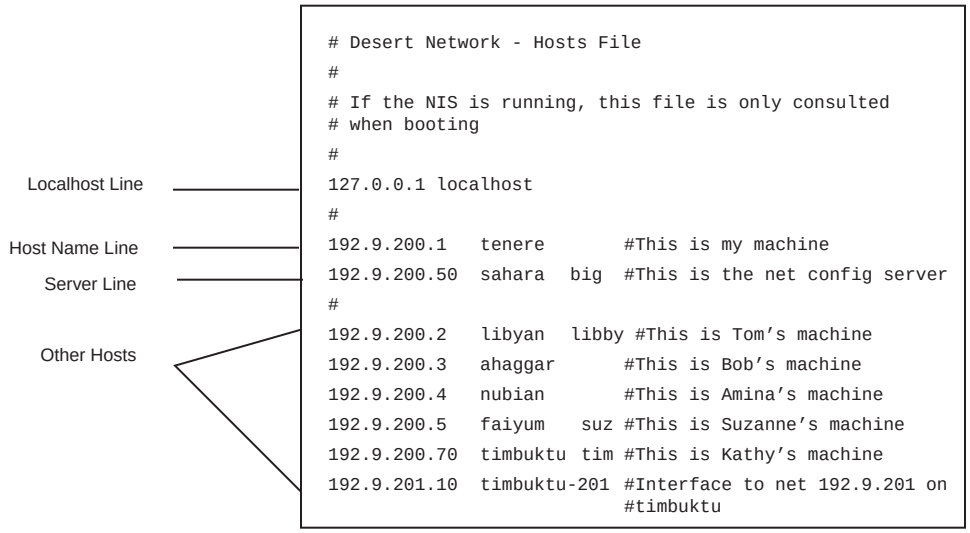


Figure 7–1 `/etc/inet/hosts` File for Machine Running in Local Files Mode

ipnodes Database

The ipnodes database contains the IPv6 addresses and host names of machines on your network. If you use the NIS, NIS+, or DNS name services, the ipnodes database is maintained in a database designated for host information. For example, on a network running NIS+, the ipnodes database is maintained in the host table. For more information about the ipnodes database, see “/etc/inet/ipnodes File” on page 345.

netmasks Database

You need to edit the netmasks database as part of network configuration *only* if you have set up subnetting on your network. The netmasks database consists of a list of networks and their associated subnet masks.

Note - When you create subnets, each new network must be a separate physical network. You cannot apply subnetting to a single physical network.

What Is Subnetting?

Subnetting is a method for getting the most out of the limited 32-bit IPv4 addressing space and reducing the size of the routing tables in a large internetwork. With any address class, subnetting provides a means of allocating a part of the host address space to network addresses, which lets you have more networks. The part of the host address space allocated to new network addresses is known as the *subnet* number.

In addition to making more efficient use of the IPv4 address space, subnetting has several administrative benefits. Routing can become very complicated as the number of networks grows. A small organization, for example, might give each local network a class C number. As the organization grows, administering a number of different network numbers could become complicated. A better idea is to allocate a few class B network numbers to each major division in an organization. For instance, you could allocate one to Engineering, one to Operations, and so on. Then, you could divide each class B network into additional networks, using the additional network numbers gained by subnetting. This can also reduce the amount of routing information that must be communicated among routers.

Creating the Network Mask for IPv4 Addresses

As part of the subnetting process, you need to select a network-wide netmask. The netmask determines how many and which bits in the host address space represent the subnet number and how many and which represent the host number. Recall that the complete IPv4 address consists of 32 bits. Depending on the address class, as

many as 24 bits and as few as 8 bits can be available for representing the host address space. The netmask is specified in the `netmasks` database.

If you plan to use subnets, you must determine your netmask before you configure TCP/IP. If you plan to install the operating system as part of network configuration, the Solaris installation program requests the netmask for your network.

As described in "Administering Network Numbers" on page 83, 32-bit IP addresses consist of a network part and a host part. The 32 bits are divided into 4 bytes. Each byte is assigned to either the network number or the host number, depending on the network class.

For example, in a class B IPv4 address, the 2 left–hand bytes are assigned to the network number, and the 2 right–hand bytes are assigned to the host number. In the class B IPv4 address 129.144.41.10, you can assign the 2 right–hand bytes to hosts.

If you are going to implement subnetting, you need to use some of the bits in the bytes assigned to the host number to apply to subnet addresses. For example, a 16-bit host address space provides addressing for 65,534 hosts. If you apply the third byte to subnet addresses and the fourth to host addresses, you can address up to 254 networks, with up to 254 hosts on each.

The bits in the host address bytes that are applied to subnet addresses and those applied to host addresses are determined by a subnet mask. Subnet masks are used to select bits from either byte for use as subnet addresses. Although netmask bits must be contiguous, they need not align on byte boundaries.

The netmask can be applied to an IPv4 address using the bit-wise logical AND operator. This operation selects out the network number and subnet number positions of the address.

It is easiest to explain netmasks in terms of their binary representation. You can use a calculator for binary-to-decimal conversion. The following examples show both the decimal and binary forms of the netmask.

If a netmask 255.255.255.0 is applied to the IPv4 address 129.144.41.101, the result is the IPv4 address of 129.144.41.0.

129.144.41.101 & 255.255.255.0 = 129.144.41.0

In binary form, the operation is:

10000001.10010000.00101001.01100101 (IPv4 address)

ANDed with

11111111.11111111.11111111.00000000 (netmask)

Now the system looks for a network number of 129.144.41 instead of a network number of 129.144. If your network has the number 129.144.41, that is what the system looks for and finds. Because you can assign up to 254 values to the third byte of the IPv4 address space, subnetting lets you create address space for 254 networks, where previously there was room for only one.

If you want to provide address space for only two additional networks, you could use a subnet mask of:

255.255.192.0

This netmask provides a result of:

11111111.11111111.1100000.00000000

This still leaves 14 bits available for host addresses. Since all 0s and 1s are reserved, at least two bits must be reserved for the host number.

The `/etc/inet/netmasks` File

If your network runs NIS or NIS+, the servers for these name services maintain `netmasks` databases. For networks that use local files for name service, this information is maintained in the `/etc/inet/netmasks` file.

Note - For compatibility with BSD-based operating systems, the file `/etc/netmasks` is a symbolic link to `/etc/inet/netmasks`.

The following example shows the `/etc/inet/netmasks` file for a class B network.

EXAMPLE 7–3 `/etc/inet/netmasks` File for a Class B Network

```
## The netmasks file associates Internet Protocol (IPv4) address
 # masks with IPv4 network numbers.
 #
 #  network-number netmask
 #
 # Both the network-number and the netmasks are specified in
 # ‘‘decimal dot’’ notation, e.g:
 #
 #        128.32.0.0   255.255.255.0
 129.144.0.0  255.255.255.0
```

If the file does not exist, create it. Use the following syntax:

network-number netmask-number

Refer to the `netmasks`(4) man page for complete details.

When creating netmask numbers, type the network number assigned by the InterNIC (not the subnet number) and netmask number in `/etc/inet/netmasks`. Each subnet mask should be on a separate line.

For example:

```
128.78.0.0     255.255.248.0
```

You can also type symbolic names for network numbers in the `/etc/inet/hosts` file. You can then use these network names instead of the network numbers as parameters to commands.

Network Databases and `nsswitch.conf` File

The network databases are files that provide information needed to configure the network. The network databases are:

- `hosts`
- `ipnodes`
- `netmasks`
- `ethers`
- `bootparams`
- `protocols`
- `services`
- `networks`

As part of the configuration process, you edit the `hosts` database and the `netmasks` database, if your network is subnetted. Two network databases, `bootparams` and `ethers`, are used to configure machines as network clients. The remaining databases are used by the operating system and seldom require editing.

Although it is not a network database, the `nsswitch.conf` file needs to be configured along with the relevant network databases. `nsswitch.conf` specifies which name service to use for a particular machine: NIS, NIS+, DNS, or local files.

How Name Services Affect Network Databases

Your network database takes a form that depends on the type of name service you select for your network. For example, the `hosts` database contains, at minimum, the host name and IPv4 address of the local machine and any network interfaces directly connected to the local machine. However, the `hosts` database could contain other IPv4 addresses and host names, depending on the type of name service on your network.

The network databases are used as follows:

- Networks that use local files for their name service rely on files in the `/etc/inet` and `/etc` directories
- NIS+ uses databases called NIS+ tables
- NIS uses databases called NIS maps
- DNS uses records with host information

Note - DNS boot and data files do not correspond directly to the network databases.

The following figure shows the forms of the `hosts` database used by these name services:

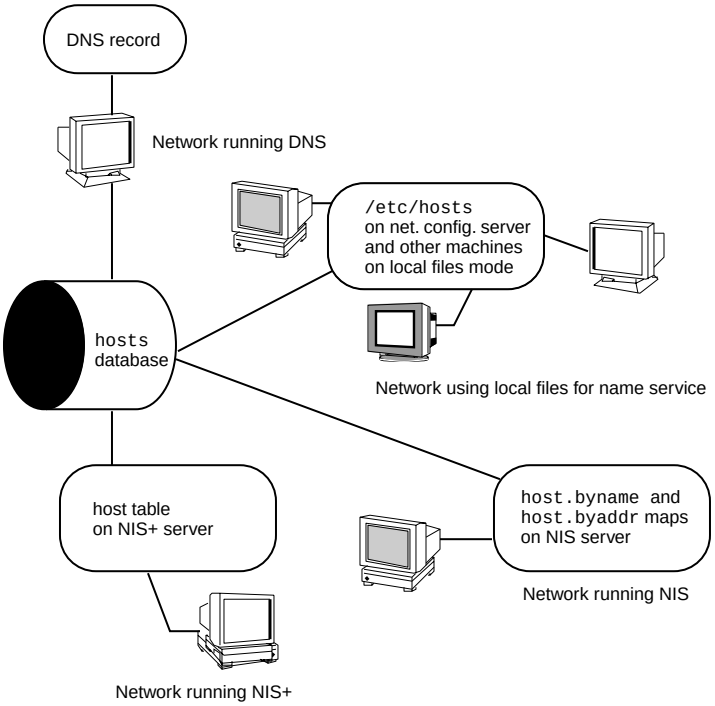


Figure 7–2 Forms of the `hosts` Database Used by Name Services

The following table lists the network databases and how they are used by local files, NIS+, and NIS.

TABLE 7–1 Network Databases and Corresponding Name Service Files

Network Database	Local Files	NIS+ Tables	NIS Maps
hosts	/etc/inet/hosts	hosts.org_dir	hosts.byaddr hosts.byname
ipnodes	/etc/inet/ipnodes	ipnodes.org_dir	ipnodes.byaddr ipnodes.byname
netmasks	/etc/inet/netmasks	netmasks.org_dir	netmasks.byaddr
ethers	/etc/ethers	ethers.org_dir	ethers.byname ethers.byaddr
bootparams	/etc/bootparams	bootparams.org_dir	bootparams
protocols	/etc/inet/protocols	protocols.org_dir	protocols.byname protocols.bynumber
services	/etc/inet/services	services.org_dir	services.byname
networks	/etc/inet/networks	networks.org_dir	networks.byaddr networks.byname

This book discusses network databases as viewed by networks using local files for name services. Information regarding the `hosts` database is in “`hosts` Database” on page 128; information regarding the `ipnodes` database is in “`/etc/inet/ipnodes` File” on page 345; information regarding the `netmasks` database is in “`netmasks` Database” on page 131. Refer to *Solaris Naming Administration Guide* for information on network databases correspondences in NIS, DNS, and NIS+.

`nsswitch.conf` File — Specifying Which Name Service to Use

The `/etc/nsswitch.conf` file defines the search order of the network databases. The Solaris installation program creates a default `/etc/nsswitch.conf` file for the local machine, based on the name service you indicate during the installation process. If you selected the “None” option, indicating local files for name service, the resulting `nsswitch.conf` file resembles the following example.

EXAMPLE 7–4 `nsswitch.conf` for Networks Using Files for Name Service

```
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file contains "switch.so" as a
# nametoaddr library for "inet" transports.

passwd:          files
group:           files
hosts:           files
networks:        files
protocols:       files
rpc:             files
ethers:          files
netmasks:        files
bootparams:      files
publickey:       files
# At present there isn't a 'files' backend for netgroup; the
# system will figure it out pretty quickly,
# and won't use netgroups at all.
netgroup:        files
automount:       files
aliases:         files
services:        files
sendmailvars:    files
```

The `nsswitch.conf`(4) man page describes the file in detail. Its basic syntax is:

database name-service-to-search

The *database* field can list one of many types of databases searched by the operating system. For example, it could indicate a database affecting users, such as `passwd` or `aliases`, or a network database. The parameter *name-service-to-search* can have the values `files`, `nis`, or `nis+` for the network databases. (The `hosts` database can also have `dns` as a name service to search.) You can also list more than one name service, such as `nis+` and `files`.

In Example 7–4, the only search option indicated is `files`. Therefore, the local machine gets security and automounting information, in addition to network database information, from files located in its `/etc` and `/etc/inet` directories.

Changing `nsswitch.conf`

The `/etc` directory contains the `nsswitch.conf` file created by the Solaris installation program. It also contains template files for the following name services:

- `nsswitch.files`
- `nsswitch.nis`

- `nsswitch.nis+`

If you want to change from one name service to another, you can copy the appropriate template to `nsswitch.conf`. You can also selectively edit the `nsswitch.conf` file, and change the default name service to search for individual databases.

For example, on a network running NIS, you might have to change the `nsswitch.conf` file on network clients. The search path for the `bootparams` and `ethers` databases must list `files` as the first option, and `nis`. The following example shows the correct search paths.

EXAMPLE 7–5 `nsswitch.conf` for a Client on a Network Running NIS

```
## /etc/nsswitch.conf:#
.
.
passwd:        files nis
group:         file nis

# consult /etc "files" only if nis is down.
hosts:         nis    [NOTFOUND=return] files
networks:      nis    [NOTFOUND=return] files
protocols:     nis    [NOTFOUND=return] files
rpc:           nis    [NOTFOUND=return] files
ethers:        files  [NOTFOUND=return] nis
netmasks:      nis    [NOTFOUND=return] files
bootparams:    files  [NOTFOUND=return] nis
publickey:     nis
netgroup:      nis

automount:     files nis
aliases:       files nis

# for efficient getservbyname() avoid nis
services:      files nis
sendmailvars:  files
```

For complete details on the name service switch, refer to *Solaris Naming Administration Guide*.

`bootparams` Database

The `bootparams` database contains information used by machines configured to boot in the network client mode. You need to edit it if your network will have network clients. (See “Configuring Network Clients” on page 103 for procedures.) The database is built from information entered into the `/etc/bootparams` file.

The `bootparams`(4) man page contains complete syntax for this database. Its basic syntax is shown in the example below:

machine-name file-key-server-name:pathname

For each network client machine, the entry might contain the following information: the name of the client, a list of keys, the names of servers, and path names.

The first item of each entry is the name of the client machine. Next is a list of keys, names of servers, and path names, separated by tab characters. All items but the first are optional. The database can contain a wildcard entry that will be matched by all clients. Here is an example:

EXAMPLE 7–6 `bootparams` Database

```
myclient   root=myserver : /nfsroot/myclient  \
swap=myserver : /nfsswap//myclient \
dump=myserver : /nfsdump/myclient
```

In this example, the term `dump=:` tells client hosts not to look for a `dump` file.

Wildcard Entry for `bootparams`

In most cases, you will want to use the wildcard entry when editing the `bootparams` database to support clients. This entry is:

```
*  root=server:/path dump=:
```

The asterisk (*) wildcard indicates that this entry applies to all clients not specifically named within the `bootparams` database.

`ethers` Database

The `ethers` database is built from information entered into the `/etc/ethers` file. It associates host names to their Ethernet addresses. You need to create an `ethers` database only if you are running the RARP daemon—that is, if you are configuring network clients.

RARP uses the file to map Ethernet addresses to IP addresses. If you are running the RARP daemon `in.rarpd`, you need to set up the `ethers` file and maintain it on all hosts running the daemon to reflect changes to the network.

The `ethers`(4) man page contains complete syntax information for this database. Its basic format is:

Ethernet-address hostname #comment

Ethernet-address is the Ethernet address of the host.

hostname is the official name of the host.

#comment is any kind of note you want to append to an entry in the file.

The equipment manufacturer provides the Ethernet address. If a machine does not display the Ethernet address when you power up, see your hardware manuals for assistance.

When adding entries to the `ethers` database, make sure that host names correspond to the primary names in the `hosts` and `ipnodes` databases, not to the nicknames, as shown below.

EXAMPLE 7–7 Entries in the `ethers` Database

```
8:0:20:1:40:16  fayoum
8:0:20:1:40:15  nubian
8:0:20:1:40:7   sahara    # This is a comment
8:0:20:1:40:14  tenere
```

Other Network Databases

The remaining network databases seldom need to be edited.

`networks` database

The `networks` database associates network names with network numbers, enabling some applications to use and display names rather than numbers. The `networks` database is based on information in the `/etc/inet/networks` file. It contains the names of all networks to which your network connects via routers.

The Solaris installation program sets up the initial `networks` database. The only time you need to update it is when you add a new network to your existing network topology.

The `networks`(4) man page contains full syntax information for `/etc/inet/networks`. Here is its basic format

network-name network-number nickname(s) #comment

network-name is the official name for the network.

network-number is the number assigned by the InterNIC.

nickname is any other name by which the network is known.

#comment is any kind of note you want to append to an entry in the file.

It is particularly important that you maintain the `networks` file. The `netstat` program uses the information in this database to produce status tables.

A sample `/etc/networks` file is shown below.

EXAMPLE 7–8 `/etc/networks` File

```
#ident "@(#)networks 1.4 92/07/14 SMI" /* SVr4.0 1.1 */
#
# The networks file associates Internet Protocol (IP) network
numbers with network names. The format of this file is:
#
#  network-name     network-number     nicnames . . .

# The loopback network is used only for intra-machine
communication
#loopback     127

# Internet networks
#
arpanet     10    arpa  # Historical
ucb-ether   46    ucbether
#
# local networks

eng   193.9.0  #engineering
acc   193.9.1  #accounting
prog  193.9.2  #programming
```

`protocols` Database

The `protocols` database lists the TCP/IP protocols installed on your system and their numbers; the Solaris installation program automatically creates it. It is rare when this file requires administrative handling.

The `protocols` database contains the names of the TCP/IP protocols installed on the system. Its syntax is completely described in the `protocols`(4) man page. An example of the `/etc/inet/protocols` file is shown below.

EXAMPLE 7–9 `/etc/inet/protocols` File

```
#
# Internet (IP) protocols
#
ip    0   IP    # internet protocol, pseudo protocol number
icmp  1   ICMP  # internet control message protocol
tcp   6   TCP   # transmission control protocol
udp  17   UDP   # user datagram protocol
```

`services` Database

The `services` database lists the names of TCP and UDP services and their wellknown port numbers; it is used by programs that call network services. The Solaris installation automatically creates the `services` database; it generally requires no administrative handling.

The `services`(4) man page contains complete syntax information. An excerpt from a typical `/etc/inet/services` file is shown below.

EXAMPLE 7–10 `/etc/inet/services` File

```
#
# Network services
#
echo      7/udp
echo      7/tcp
discard   9/udp     sink null
discard   11/tcp
daytime   13/udp
daytime   13/tcp
netstat   15/tcp
ftp-data  20/tcp
ftp       21/tcp
telnet    23/tcp
time      37/tcp    timeserver
time      37/udp    timeserver
name      42/udp    nameserver
whois     43/tcp    nickname
```

Overview of the Booting Processes

The following information is provided for your reference. It is a brief overview of the network booting processes to help you better visualize what is happening during configuration.

Note - The names of startup scripts might change from one release to another.

1. You start the operating system on a host.
2. The kernel runs `/sbin/init`, as part of the booting process.
3. `/sbin/init` runs the `/etc/rcS.d/S30rootusr.sh`. startup script.
4. The script runs a number of system startup tasks, including establishing the minimum host and network configurations for diskless and dataless operations. `/etc/rcS.d/S30rootusr.sh` also mounts the `/usr` file system.

    a. If the local database files contain the required configuration information (host name and IP address), the script uses it.

    b. If the information is not available in local host configuration files, `/etc/rcS.d/S30rootusr.sh` uses RARP to acquire the host's IP address.

5. If the local files contain domain name, host name, and default router address, the machine uses them. If the configuration information is not in local files, then the system uses the Bootparams protocol to acquire the host name, domain name, and default router address. Note that the required information must be available on a network configuration server that is located on the same network as the host. This is necessary because no internetwork communications exist at this point.
6. After `/etc/rcS/S30rootusr.sh` completes its tasks and several other boot procedures have executed, `/etc/rc2.d/S69inet` runs. This script executes startup tasks that must be completed before the name services (NIS, NIS+, or DNS) can start. These tasks include configuring the IP routing and setting the domain name.
7. At completion of the `S69inet` tasks, `/etc/rc2.d/S71rpc` runs. This script starts the NIS, NIS+, or DNS name service.
8. After `/etc/rc2.d/S71` runs, `/etc/rc2.d/S72inetsvc` runs. This script starts up services that depend on the presence of the name services. `S72inetsvc` also starts the daemon `inetd`, which manages user services such as `telnet`.

See *System Administration Guide, Volume 1* for a complete description of the booting process.

Routing Protocols

Solaris system software supports two routing protocols: Routing Information Protocol (RIP) and ICMP Router Discovery (RDISC). RIP and RDISC are both standard TCP/IP protocols.

Routing Information Protocol (RIP)

RIP is implemented by `in.routed`, the routing daemon, which automatically starts when the machine boots. When run on a router with the `s` option specified, `in.routed` fills the kernel routing table with a route to every reachable network and advertises “reachability” through all network interfaces.

When run on a host with the `q` option specified, `in.routed` extracts routing information but does not advertise reachability. On hosts, routing information can be extracted in two ways:

- Do *not* specify the `S` flag (capital “S”: “Space-saving mode”) and `in.routed` builds a full routing table exactly as it does on a router.
- Specify the `S` flag and `in.routed` creates a minimal kernel table, containing a single default route for each available router.

ICMP Router Discovery (RDISC) Protocol

Hosts used RDISC to obtain routing information from routers. Thus, when hosts are running RDISC, routers must also run another protocol, such as RIP, in order to exchange router information among themselves.

RDISC is implemented by `in.rdisc`, which should run on both routers and hosts. Normally, when `in.rdisc` runs on a host, it enters a default route for each router that is also running `in.rdisc`. A host that is running `in.rdisc` can not discover routers that are running only RIP. Furthermore, when routers are running `in.rdisc` (rather than `in.routed`), they can be configured to have a different preference, which causes hosts to select a better router. See the `rdisc`(1M) man page.

How a Machine Determines if it Is a Router

The `/etc/rc2.d/S69inet` startup script, which runs when the machine boots, determines whether a machine is a router or a host. This decision also determines whether the routing protocols (RIP and RDISC) should run in router mode or host mode.

The `/etc/rc2.d/S69inet` script concludes that a machine is a router if the following two conditions exist:

- More than one `/etc/hostname.`*interface* file exists.
- More than one interface was configured “up” by the `ifconfig` command. (See the `ifconfig`(1M) man page.)

If only one interface is found, the script concludes that the machine is a host. See “Configuring Both Router Network Interfaces” on page 106. An interface that is configured by any means other than an `/etc/hostname.`*interface* file is not taken into account.

Parts of the IPv4 Address

Each network running TCP/IP must have a unique network number, and every machine on it must have a unique IP address. It is important to understand how IP addresses are constructed before you register your network and obtain its network number. This section describes IPv4 addresses. For information on IPv6 addresses, see “IPv6 Addressing” on page 300.

The IPv4 address is a 32-bit number that uniquely identifies a network interface on a machine. An IPv4 address is typically written in decimal digits, formatted as four 8-bit fields separated by periods. Each 8-bit field represents a byte of the IPv4 address. This form of representing the bytes of an IPv4 address is often referred to as the *dotted-decimal format*.

The bytes of the IPv4 address are further classified into two parts: the network part and the host part. The following figure shows the component parts of a typical IPv4 address, 129.144.50.56.

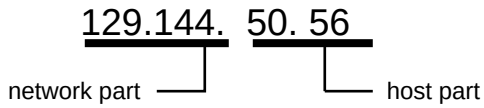


Figure 7–3 Parts of an IPv4 Address

Network Part

This part specifies the unique number assigned to your network. It also identifies the class of network assigned. In Figure 7–3, the network part takes up two bytes of the IPv4 address.

Host Part

This is the part of the IPv4 address that you assign to each host. It uniquely identifies this machine on your network. Note that for each host on your network, the network part of the address will be the same, but the host part must be different.

Subnet Number (Optional)

Local networks with large numbers of hosts are sometimes divided into subnets. If you choose to divide your network into subnets, you need to assign a *subnet number* for the subnet. You can maximize the efficiency of the IPv4 address space by using some of the bits from the host number part of the IPv4 address as a network identifier. When used as a network identifier, the specified part of the address becomes the subnet number. You create a subnet number by using a netmask, which is a bit mask that selects the network and subnet parts of an IPv4 address. (Refer to “Creating the Network Mask for IPv4 Addresses” on page 131 for details.)

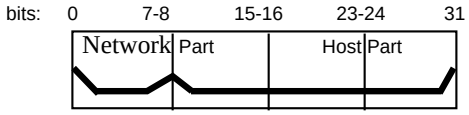
Network Classes

The first step in planning for IPv4 addressing on your network is to determine which network class is appropriate for your network. After you have done this, you can take the crucial second step: obtain the network number from the InterNIC addressing authority.

Currently there are three classes of TCP/IP networks. Each class uses the 32-bit IPv4 address space differently, providing more or fewer bits for the network part of the address. These classes are class A, class B, and class C.

Class A Network Numbers

A class A network number uses the first eight bits of the IPv4 address as its "network part." The remaining 24 bits makeup the host part of the IPv4 address, as illustrated below.



Class A Address

Figure 7–4 Byte Assignment in a Class A Address

The values assigned to the first byte of class A network numbers fall within the range 0–127. Consider the IPv4 address 75.4.10.4. The value 75 in the first byte indicates that the host is on a class A network. The remaining bytes, 4.10.4, establish the host address. The InterNIC assigns only the first byte of a class A number. Use of the remaining three bytes is left to the discretion of the owner of the network number. Only 127 class A networks can exist. Each one of these numbers can accommodate up to 16,777,214 hosts.

Class B Network Numbers

A class B network number uses 16 bits for the network number and 16 bits for host numbers. The first byte of a class B network number is in the range 128–191. In the number 129.144.50.56, the first two bytes, 129.144, are assigned by the InterNIC, and makeup the network address. The last two bytes, 50.56, make up the host address, and are assigned at the discretion of the owner of the network number. The following figure graphically illustrates a class B address.

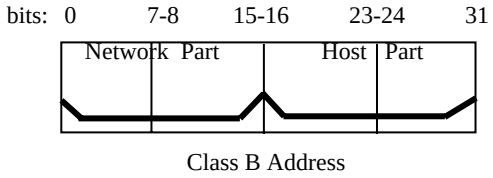


Figure 7–5 Byte Assignment in a Class B Address

Class B is typically assigned to organizations with many hosts on their networks.

Class C Network Numbers

Class C network numbers use 24 bits for the network number and 8 bits for host numbers. Class C network numbers are appropriate for networks with few hosts—the maximum being 254. A class C network number occupies the first three bytes of an IPv4 address. Only the fourth byte is assigned at the discretion of the network owners. The following figure graphically represents the bytes in a class C address.

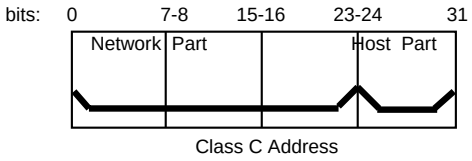


Figure 7–6 Byte Assignment in a Class C Address

The first byte of a class C network number covers the range 192–223. The second and third each cover the range 1– 255. A typical class C address might be 192.5.2.5. The first three bytes, 192.5.2, form the network number. The final byte in this example, 5, is the host number.

CHAPTER 8

Overview of DHCP

This chapter introduces the Dynamic Host Configuration Protocol (DHCP), explains the concepts underlying the protocol, and describes the advantages of using it in your network.

This chapter contains the following information:

About DHCP

DHCP is a standard developed to enable host systems in a TCP/IP network to be configured automatically for the network as they boot. DHCP uses a client/server mechanism: servers store and manage configuration information for clients, and provide that information upon a client's request. The information includes the client's IP address and information about network services available to the client.

DHCP evolved from an earlier protocol, BOOTP, which was designed for booting over a TCP/IP network. DHCP builds upon BOOTP by using the same format for messages between client and sever, while including more information in the messages. The extra information is the network configuration data for the client.

A primary benefit of DHCP is its ability to manage IP address assignments through leasing, which allows IP addresses to be reclaimed when not in use and reassigned

to other clients. This enables a site to use a smaller pool of IP address than would be needed if all clients were assigned a permanent address.

Advantages of Using Solaris DHCP

DHCP relieves the system or network administrator of some of the time-consuming tasks involved in setting up a TCP/IP network and the daily management of that network. Note that Solaris DHCP works only with IPv4.

Solaris DHCP offers the following advantages:

- **IP address management** – A primary advantage of DHCP is easier management of IP addresses. In a network without DHCP, an administrator must manually assign IP addresses, being careful to assign unique IP addresses to each client and configure each client individually. If a client moves to a different network, the administrator must make manual modifications for that client. When DHCP is enabled, the DHCP server manages and assigns IP addresses without administrator intervention. Clients can move to other subnets without manual reconfiguration because they obtain from a DHCP server new client information appropriate for the new network.
- **Centralized network client configuration** – A network administrator can create a tailored configuration for certain clients, or certain types of clients, and keep the information in one place, the DHCP data store. The administrator does not need to log in to a client to change its configuration. The administrator can make changes for multiple clients just by changing the information in the data store.
- **Support of BOOTP clients** – Both BOOTP servers and DHCP servers listen and respond to broadcasts from clients. The DHCP server can respond to requests from BOOTP clients, as well as DHCP clients. BOOTP clients receive an IP address and the information needed to boot from a server.
- **Support of local and remote clients** – BOOTP provides for the relaying of messages from one network to another. DHCP takes advantage of the BOOTP relay feature in several ways. Most network routers can be configured to act as BOOTP relay agents to pass BOOTP requests to a server that is not on the requesting client's network. DHCP requests can be relayed in the same manner because, to the router, they are indistinguishable from BOOTP requests. The Solaris DHCP server can also be configured to behave as a BOOTP relay agent, if a router that supports BOOTP relay is not available.
- **Network booting** – Clients can use DHCP to obtain the information needed to boot from a server on the network, instead of using RARP (Reverse Address Resolution Protocol) and `bootparams`. The DHCP server can give a client all the information it needs to function, including IP address, boot server, and network configuration information. Because DHCP network boot requests can be relayed

across subnets, you can deploy fewer boot servers in your network when you use DHCP network booting. RARP booting requires each subnet to have a boot server.

How DHCP Works

The DHCP server must first be installed and configured by a system administrator. During configuration, the administrator enters information about the network that clients will need for operating on the network. After this information is in place, clients are able to request and receive network information.

The sequence of events for DHCP service is shown in the following diagram. The numbers in circles correlate to the numbered items in the description following the diagram.

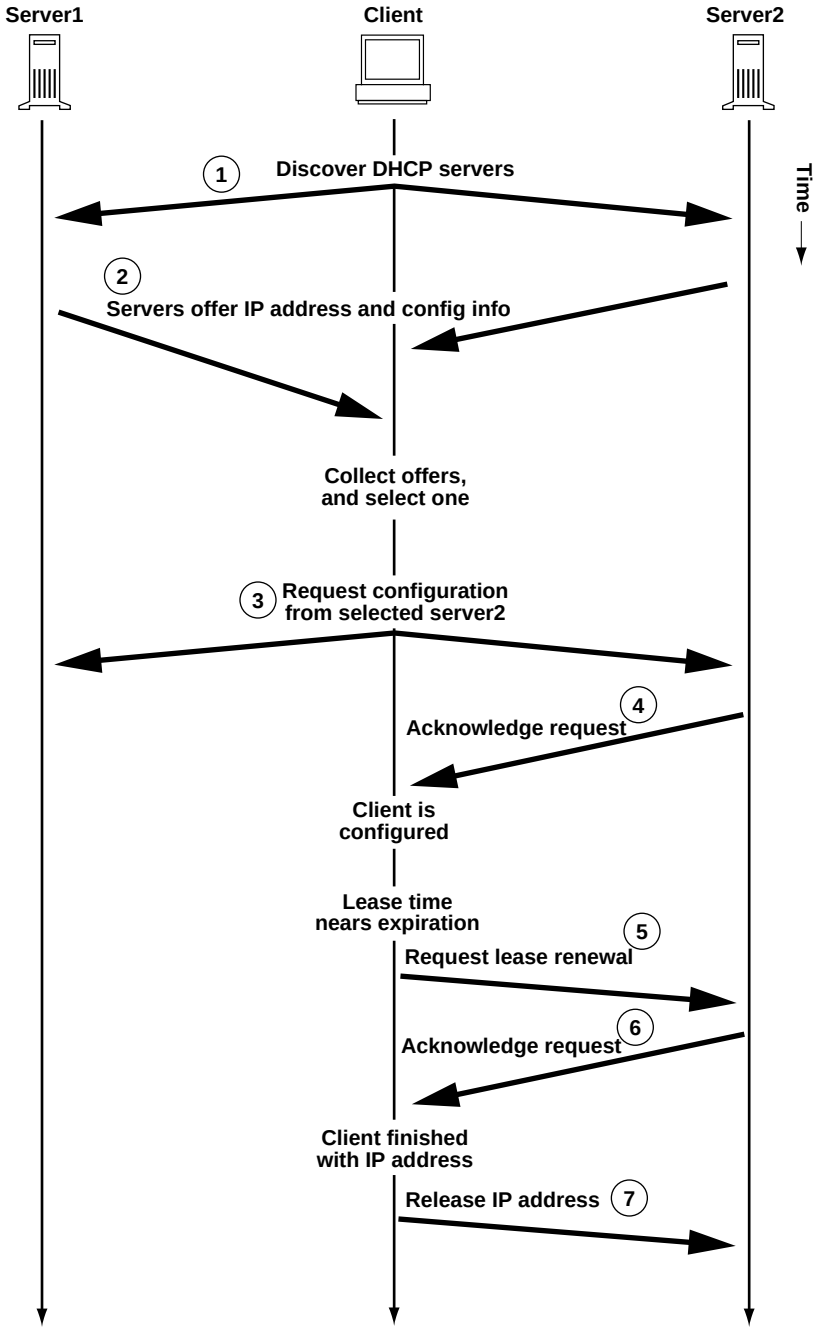


Figure 8–1 Sequence of Events for DHCP Service

LEGEND:

1. The client discovers a DHCP server by broadcasting a discover message to the limited broadcast address (255.255.255.255) on the local subnet. If a router is present and configured to behave as a BOOTP relay agent, the request is passed to other DHCP servers on different subnets. The client's broadcast includes its unique ID, which in the Solaris DHCP implementation, is derived from the client's MAC address.

   DHCP servers receiving the discover message can determine the client's network by looking at the following information:

   - Which of the server's network interfaces did the request come in on? This tells the server that the client is either on the network to which the interface is connected, or that the client is using a BOOTP relay agent connected to that network.
   - Does the request include the IP address of a BOOTP relay agent? When a request passes through a relay agent, the relay agent inserts its address in the request header. When the server detects a relay agent address, it knows that the network portion of the address indicates the client's network address because the relay agent must be connected to the client's network.
   - Is the client's network subnetted? The server consults the netmasks table, keying on the relay agent address or the address of the network interface that received the request. Once the server knows the subnet mask used, it can determine which portion of the network address is the host portion, and then select an IP address appropriate for the client. (See `netmasks`(4) for information on netmasks.)

2. After determining the client's network, DHCP servers select an appropriate IP address and verify that the address is not already in use. The DHCP servers then respond to the client by broadcasting an offer message that includes the selected IP address and information about services that can be configured for the client. Each server temporarily reserves the offered IP address until it can determine if the client will use it.
3. The client selects the best offer (based on the number and type of services offered) and broadcasts a request, specifying the IP address of the server that made the best offer. The broadcast ensures that all the responding DHCP servers know the client has chosen a server, and those servers not chosen can cancel the reservations for the IP addresses they had offered.
4. The selected server allocates the IP address for the client, storing the information in the DHCP data storage area, and sends an acknowledgement to the client. The acknowledgement message contains the network configuration parameters for the client. The client tests the IP address to make sure no other system is using it, and continues booting to join the network.
5. The client monitors the lease time, and when a set period of time has elapsed, the client sends a new message to the chosen server to increase its lease time.
6. The DHCP server receiving the request extends the lease time if it still adheres to the local lease policy set by the administrator. If the server does not respond

within 20 seconds, the client broadcasts a request so that one of the other DHCP servers can extend the lease.

7. When the client no longer needs the IP address, it sends a message notifying the server that it is releasing the IP address. This can happen during an orderly shutdown and can also be done manually.

Solaris DHCP Server

The Solaris DHCP server runs as a daemon in the Solaris operating environment on a host system. The server has two basic functions:

- **Managing IP addresses** – The server controls a range of IP addresses, and allocates them to clients, either permanently or for a defined period of time. The DHCP server uses a lease mechanism to determine how long a client can use a non-permanent address. When the address is no longer in use, it is returned to the pool and can be reassigned. The server maintains information about the binding of IP addresses to clients in its DHCP network tables, ensuring that no address is used by more than one client.
- **Providing network configuration for clients** – The server assigns an IP address and provides other information for network configuration, such as a hostname, broadcast address, network subnet mask, default gateway, name service, and potentially much more information. The network configuration information is obtained from the server's `dhcptab` database.

The Solaris DHCP server can also be configured to perform the following additional functions:

- **Responding to BOOTP client requests** – The server listens for broadcasts from BOOTP clients discovering a BOOTP server and provides them with an IP address and boot parameters. The information must have been configured statically by an administrator. The DHCP server can perform as a BOOTP server and DHCP server simultaneously.
- **Relaying requests** – The server relays BOOTP and DHCP requests to appropriate servers on other subnets. The server cannot provide DHCP or BOOTP service when configured as a BOOTP relay agent.
- **Providing network booting support for DHCP clients** – The server can provide DHCP clients with information needed to boot over the network: IP address, boot parameters, and network configuration information.

DHCP Server Management

As superuser, you can start, stop, and configure the DHCP server using the DHCP Manager, or by using command line utilities. Generally, the DHCP server is configured to start automatically when the system boots, and stop when the system is shutdown, so starting and stopping the server manually should be an infrequent occurence.

DHCP Server Data Storage

All the data used by the DHCP server is maintained in two data repositories, which you can view and manage using either the DHCP Manager or command-line utilities. The data repositories are:

- `dhcptab` – A file containing configuration information that can be passed to clients.
- **DHCP network tables** – Tables containing information about the DHCP and BOOTP clients residing on the network specified in the table name. For example, the network 134.20.0.0 would have a table named `134_20_0_0`.

The DHCP data can be stored in files on a local directory, or in a NIS+ database. “Choosing the Data Store” on page 172 discusses selecting a data storage method.

The `dhcptab` File

The `dhcptab` file contains all the information that clients can obtain from the DHCP server. The DHCP server scans the file each time it starts.

The DHCP protocol defines a number of standard items of information that can be passed to clients. These items are referred to as parameters, symbols, or options. Options are defined in the DHCP protocol by numeric codes and text labels, but without values. For example, some commonly used standard options are shown in the following table.

TABLE 8–1 Sample DHCP Standard Options

Code	Label	Description
1	Subnet	Subnet mask IP address
3	Router	IP address for router
6	DNSserv	IP address for DNS server

TABLE 8–1 Sample DHCP Standard Options *(continued)*

Code	Label	Description
12	Hostname	Text string for client hostname
15	DNSdmain	DNS domain name

Some options are automatically assigned values when the administrator provides information during server configuration. The administrator can also explicitly assign values to other options at a later time. Options and their values are passed to the client to provide configuration information. For example, the option/value pair, `DNSdmain=Georgia.Peach.COM`, sets the client's DNS domain name to `Georgia.Peach.COM`.

Options can be grouped with other options in containers known as macros, which makes it easier to pass information to a client. Some macros are created automatically during server configuration, and contain options that were assigned values during configuration. Macros can also contain other macros.

The format of the `dhcptab` file is described in `dhcptab`(4). In DHCP Manager, all the information shown in the Options and Macros tabs comes from the `dhcptab` file. See "About Options" on page 159 for more information about options, and "About Macros" on page 160 for more information about macros.

Note that the `dhcptab` file is a text file, but should not be edited manually. You should use either `dhtadm` or DHCP Manager to create, delete, or modify options and macros.

DHCP Network Tables

A DHCP network table maps client identifiers to IP addresses and the configuration parameters associated with each address. The format of the network tables is described in `dhcp_network`(4). In DHCP Manager, all the information shown in the Addresses tab is acquired from the network tables.

DHCP Manager

DHCP Manager is a graphical tool you can use to perform all management duties associated with DHCP services. You can use it to manage the server itself as well as the data the server uses. You can use DHCP Manager with the server in the following ways:

- Configure and unconfigure the DHCP server
- Start, stop, and restart the DHCP server

- Disable and enable DHCP services
- Customize server settings

DHCP Manager also allows you to manage the IP addresses, network configuration macros, and network configuration options in the following ways:

- View, add, modify, delete, and release IP addresses
- View, add, modify, and delete network configuration macros
- View, add, modify, and delete non-standard network configuration options

DHCP Manager includes extensive online help for procedures you can perform with the tool.

DHCP Command-Line Utilities

All DHCP management functions can be performed using command-line utilities. The following table lists the utilities and describes the purpose of each utility. See the man pages for the utilities for detailed information about using them by clicking on the command names in the table.

TABLE 8–2 DHCP Command-Line Utilities

DHCP Command Manual Page	Description and Purpose
`in.dhcpd`(1M)	The DHCP service daemon. It provides command-line arguments that allow you to set several runtime options.
`dhcpconfig`(1M)	Shell script that presents a text-based menu system to help you configure a DHCP server. `dhcpconfig` collects information from the server machine's network topology files to create useful information for the initial configuration. `dhcpconfig` uses the `dhtadm` and `pntadm` utilities in the background to create the initial `dhcptab` and network tables.

TABLE 8–2 DHCP Command-Line Utilities *(continued)*

DHCP Command Manual Page	Description and Purpose
`dhtadm`(1M)	Used for adding, deleting, and modifying configuration options and macros for DHCP clients. This utility lets you edit the `dhcptab` file indirectly, ensuring the format of the `dhcptab` file is correct. You should not directly edit the `dhcptab` file.
`pntadm`(1M)	Used to manage the DHCP network tables. You can use this utility to add and remove IP addresses and networks under DHCP management, modify the network configuration of specified IP addresses, and display information about IP addresses and networks under DHCP management.

DHCP Server Configuration

You configure the DHCP server the first time you run DHCP Manager on the system where you want to run the DHCP server. DHCP Manager server configuration dialogs prompt you for essential information needed to enable and run the DHCP server on one network. Some default values are obtained from existing system files. If you have not configured the system for the network, there will be no default values. DHCP Manager prompts for the following information:

- Role of the server, either DHCP server or BOOTP relay agent
- Data storage method, either local files or NIS+
- Length of lease time and whether clients should be able to renew leases
- DNS domain name and IP addresses of DNS servers
- Network address and subnet mask for the first network you want to be configured for DHCP service
- Network type, either LAN or point-to-point
- Router discovery or the IP address of a particular router
- NIS domain name and IP address of NIS servers
- NIS+ domain name and IP address of NIS+ servers

You can also configure the DHCP server using the `dhcpconfig` command. This utility gathers information from existing system files automatically in order to provide a useful initial configuration. Therefore, you must ensure that the files are correct before running `dhcpconfig`. See `dhcpconfig`(1M) for information about

the files `dhcpconfig` uses to obtain information. To make configuration changes after the initial configuration, you must make changes to the system files and rerun `dhcpconfig` so that it picks up the changes.

IP Address Allocation

The Solaris DHCP server supports the following types of IP address allocation:

- **Manual allocation** – The server provides a specific IP address chosen by the administrator for a specific DHCP client. The address cannot be reclaimed or assigned to any other client.
- **Automatic, or permanent, allocation** – The server provides an IP address that has no expiration time, making it permanently associated with the client until the administrator changes the assignment or the client releases the address.
- **Dynamic allocation** – The server provides an IP address to a requesting client, with a lease for a specific period of time. When the lease expires, the address is taken back by the server and can be assigned to another client. The period of time is determined by the lease time configured for the server.

Network Configuration Information

The administrator determines what information is provided to DHCP clients. When you configure the DHCP server you provide essential information about the network; later, you can add more information you want to provide to clients.

The DHCP server stores network configuration information in the `dhcptab` database, in the form of option/value pairs and macros. Options are keywords for network data you want to supply to clients. Values are assigned to options and passed to clients in DHCP messages. For example, the NIS server address is passed using an option called `NISservrs` that has a value (a list of IP addresses) assigned by the DHCP server. Macros provide a convenient way to group together any number of options that you want to supply to clients. You can use the DHCP Manager to assign values to the options, and create macros to group the options. If you prefer a non-graphical tool, you can use `dhtadm`, the DHCP configuration table management utility, to work with options and macros.

About Options

In Solaris DHCP, an option is a piece of network information to be passed to a client. In the DHCP literature, options are also referred to as symbols or tags. An option is defined by a numeric code and a text label, and is then assigned a value.

The DHCP protocol defines a large number of standard options for commonly specified network data: `Subnet`, `Router`, `Broadcast`, `NIS+dom`, `Hostname`, and `LeaseTim` are a few examples. A complete list of standard options is included in the DHCP Manager help. You cannot modify the standard option keywords in any way, but you can assign values to the options that are relevant to your network, and include the option/value pairs in macros.

You can create new options for data that is not represented by the standard options. Options you create must be classified in one of three categories:

- **Extended** – Reserved for options that are more recent standard DHCP options, that might not yet be included in your DHCP server implementation. You might use this if you know of a standard option that you want to use, but do not want to upgrade your DHCP server.
- **Site** – Reserved for options that are unique to your site. The system administrator creates these options.
- **Vendor** – Reserved for options that should apply only to clients of a particular class, such as hardware or vendor platform. The Solaris DHCP implementation includes a number of vendor options for Solaris clients. For example, the option `SrootIP4` is used for specifying the IP address of a server that a client booting from the network should use for its root file system.

Chapter 11 includes procedures for creating, modifying, and deleting options.

About Macros

In the Solaris DHCP service, a macro is a collection of network configuration options and the values assigned to them by the system administrator. Macros are created to group options together to be passed to specific clients or types of clients. For example, a macro intended for all clients of a particular subnet might contain option/value pairs for subnet mask, router IP address, broadcast address, NIS+ domain, and lease time.

Macro Processing by the DHCP Server

When a macro is processed by the server, the network options and values defined in the macro are placed in a DHCP message to a client. Some macros are processed automatically by the server for clients of a particular type.

In order for a macro to be processed automatically, it must be named according to one of the categories shown in the following table.

TABLE 8–3 Macro Categories for Automatic Processing

Macro Category	Description
Client class	The macro name matches a class of client, indicated by the client machine type and/or operating system. For example, if a server has a macro named `SUNW.Ultra-1`, any client that is a SUNW,Ultra-1 machine automatically receives the values in the `SUNW.Ultra-1` macro.
Network address	The macro name matches a DHCP-managed network IP address. For example, if a server has a macro named `125.53.224.0`, any client connected to the 125.53.224.0 network automatically receives the values in the `125.53.224.0` macro.
Client ID	The macro name matches some unique identifier for the client, usually derived from an Ethernet or MAC address. For example, if a server has a macro named `08002011DF32`, the client having the client ID 08002011DF32 (derived from the Ethernet address 8:0:20:11:DF:32) automatically receives the values in a macro named `08002011DF32`.

A macro with a name that does not use one of the categories listed in Table 8–3 can be processed only if one of the following is true:

- Macro is mapped to an IP address.
- Macro is included in another macro that is processed automatically.
- Macro is included in another macro that mapped to an IP address.

Note - When you configure a server, a macro that is named to match the server's name is created by default. This server macro is *not* processed automatically for any client because it is not named with one of the name types that cause automatic processing. When you later create IP addresses on the server, the IP addresses are mapped to use the server macro by default.

Order of Macro Processing

When a DHCP client requests DHCP services, the DHCP server determines which macros match the client. The server processes the macros, using the macro categories to determine the order of processing, from the more general to the specific. The macros are processed in the following order:

1. Client class macros – the most general category
2. Network address macros – more specific than Client class

3. Macros mapped to IP addresses – more specific than Network address
4. Client ID macros – the most specific category

A macro that is included in another macro is processed as part of the containing macro.

If the same option is included in more than one macro, the value set for that option in the macro with the most specific category is used because it is processed last. For example, if a Network address macro contained the lease time option with a value of 24 hours, and a Client ID macro contained the lease time option with a value of 8 hours, the client would receive a lease time of 8 hours.

Solaris DHCP Client

The term “client” is sometimes used to refer to a physical machine that is performing a client role on the network. However, the DHCP client described here is a software entity. The Solaris DHCP client is a daemon (`dhcpagent`) that runs in the Solaris operating environment on a machine that is configured to receive its network configuration from a DHCP server. DHCP clients from other vendors can also use the services of the Solaris DHCP server. However, this section describes only the Solaris DHCP client.

Notice that the description assumes one network interface. The section “DHCP Clients With Multiple Network Interfaces” on page 165 discusses issues important for hosts having two or more network interfaces.

DHCP Client Installation

The Solaris DHCP client is installed and enabled on a system during installation of the Solaris operating environment when you specify that you want to use DHCP to configure network interfaces. There is nothing further you need to do on the Solaris client to use DHCP.

If you want a machine that is already running the Solaris environment to use DHCP to configure network interfaces, see “Configuring and Unconfiguring a Solaris DHCP Client” on page 198.

DHCP Client Startup

The `dhcpagent` daemon obtains configuration information that is needed by other processes involved in booting the system. For this reason, `dhcpagent` is started

early in the boot process by the system startup scripts; booting is suspended until the network configuration information is obtained.

The presence of the file `/etc/dhcp.`*interface* (for example, `/etc/dhcp.hme0` on a Sun Enterprise Ultra™ machine) indicates to the startup scripts that DHCP is to be used on the specified interface. Upon finding a `dhcp.`*interface* file, the startup scripts start the `dhcpagent`.

After starting up, `dhcpagent` waits until it receives instructions to configure a network interface. The startup scripts issue the `ifconfig` *interface* `dhcp start`command, which instructs `dhcpagent` to start DHCP as described in "How DHCP Works" on page 151. If commands are contained within the `dhcp.`*interface* file, they are appended to the `dhcp start` option of `ifconfig`. See the `ifconfig`(1M) man page for more information about options used with `dhcp`.

How DHCP Client Manages Network Configuration Information

After the information packet is obtained from a DHCP server, `dhcpagent` configures the network interface and brings it up, controlling the interface for the duration of the lease time for the IP address. `dhcpagent` maintains the configuration data in an internal table held in memory. The system startup scripts use the `dhcpinfo` command to extract configuration option values from the `dhcpagent`'s table. The values are used in configuring the system and becoming part of the network.

The agent waits passively until a set period of time elapses, usually half the lease time, and then requests an extension of the lease from a DHCP server. If `dhcpagent` finds that the interface is down or the IP address has changed, it does not control the interface until it is instructed by `ifconfig` to do so. If `dhcpagent` finds that the interface is up and the IP address hasn't changed, it sends a request to the server for a lease renewal. If the lease cannot be renewed, `dhcpagent` takes the interface down at the end of the lease time.

DHCP Client Management

The Solaris DHCP client does not need to be managed under normal system operation. It automatically starts when the system boots, renegotiates leases, and stops when the system shuts down. You cannot manually start and stop the `dhcpagent` daemon. However, you can use the `ifconfig` command as superuser on the client machine to affect the client's management of the network interface if necessary.

`ifconfig` Commands Used With DHCP Client

The `ifconfig` command lets you:

- **Start the DHCP client**– The command `ifconfig` *interface* `dhcp start` initiates the interaction between the DHCP client and DHCP server for the purpose of obtaining an IP address and a new set of configuration options. This might be useful when you change options on the server, such as adding IP addresses or changing the subnet mask, that you want clients to use immediately.
- **Request network configuration information only** – The command `ifconfig` *interface* `dhcp inform` causes `dhcpagent` to issue a request for network configuration parameters, with the exception of the IP address. This is useful for situations where the network interface has a valid IP address, but the client system needs updated network options. For example, this might be useful if you do not use DHCP to manage IP addresses, but do use it for configuring hosts on the network.
- **Request a lease extension** – The command `ifconfig` *interface* `dhcp extend` causes `dhcpagent` to issue a request to renew the lease. This happens automatically, but you might want to use this if you change the lease time and want clients to use the new lease time immediately rather than waiting for the next attempt at lease renewal.
- **Release the IP address** – The command `ifconfig` *interface* `dhcp release` causes `dhcpagent` to relinquish the IP address being used by the network interface. This happens automatically when the lease expires. You might want to issue this command if the lease time is long and you need to take down the network interface for an extended period of time or you are removing the system from the network.
- **Drop the IP address** – The command `ifconfig` *interface* `dhcp drop` causes `dhcpagent` to take down the network interface without informing the DHCP server that it is doing so. This enables the client to use the same IP address when it reboots.
- **Ping the network interface** – The command `ifconfig` *interface* `dhcp ping` lets you test to see if the interface is under the control of DHCP.
- **View DHCP configuration status of the network interface** – The command `ifconfig` *interface* `dhcp status` displays the current state of the DHCP client. The display includes information about whether an IP address has been bound to the client; the number of requests sent, received, and declined; flags indicating whether this is the primary interface; and times indicating when the lease was obtained, expires, and when attempts to renew it will/did start. For example::

```
# ifconfig hme0 dhcp status
Interface  State         Sent  Recv  Declined  Flags
hme0       BOUND            1     1         0   [PRIMARY]
```

(continued)

```
(Began, Expires, Renew) = (07/15/1999 15:27, 07/17/1999 13:31, 07/16/
1999 15:24)
```

DHCP Client Parameter File

The file `/etc/default/dhcpagent` on the client system contains tunable parameters for `dhcpagent`. You can use a text editor to change several parameters that affect client operation. The file is well-documented so please refer to the file for more information about the parameters.

DHCP Client Shutdown

When the system running the DHCP client shuts down normally, the `dhcpagent` daemon writes the current configuration information to the file `/etc/dhcp/`*interface*`.dhc`. The lease is dropped rather than released, so the DHCP server does not know that the IP address is not in active use.

If the lease is still valid when the system is rebooted, the DHCP client sends an abbreviated request to use the same IP address and network configuration information it had used before the system was rebooted. If the DHCP server permits this, the client can use the information that it wrote to disk when the system shut down. If the server does not permit using the information, the client initiates the DHCP protocol sequence described previously and obtains new network configuration information.

DHCP Clients With Multiple Network Interfaces

The DHCP client daemon can manage several different interfaces on one system simultaneously, each with its own IP address and lease time. If more than one network interface is configured for DHCP, the client issues separate requests to configure them and maintains a separate set of network configuration options for each interface. However, although the parameters are stored separately, some of the parameters are global in nature, applying to the system as a whole, rather than to a particular network interface. Options such as hostname, NIS domain name, and timezone are global parameters and should have the same values for each interface, but it might not be the case due to errors in the information entered by the DHCP administrator. To ensure that there is only one answer to a query for a global

parameter, only the parameters for the primary network interface are requested. You can insert the word **primary** in the `/etc/dhcp.`*interface* file for the interface you want to be treated as the primary interface.

CHAPTER 9

Planning for DHCP Service

You can use DHCP services in a network you are creating, or in an existing network. If you are setting up a network, see Chapter 5 before attempting to set up DHCP services. If you have an existing network, continue in this chapter.

This chapter describes what you need to do before setting up DHCP service on your network. The planning information is targeted for use with DHCP Manager, although you can also use the command-line utility `dhcpconfig` to set up DHCP service.

This chapter contains the following information:

Preparing Your Network for DHCP

Before you can set up your network to use DHCP, you must collect information and make decisions about how you will configure the server(s). You must:

- Map out your network topology to determine which servers are the best candidates for DHCP servers, and how many servers you will need.
- Update your system files and `netmasks` table to reflect network topology accurately. If your DHCP server is to support clients on remote networks, you must also be sure the `netmasks` table entries for those networks are up to date. (See `netmasks`(4) for more information.

- Choose the data storage method you will use, local files or NIS+.
- Define a lease policy.
- Determine how router information should be obtained by clients.

Mapping Your Network Topology

If you have not already done so, you should map out the physical structure or layout of your network, indicating the location of routers and clients, and the location of servers providing network services. This mapping of your network topology can help you determine which server to use for DHCP services, and what configuration information the DHCP server can provide to clients.

See Chapter 5 for more information about planning your network.

The DHCP configuration process can look up some network information from the server's system and network files. "Updating System Files and Netmask Tables" on page 169 discusses these files. However, you might want to give clients other service information, which you must enter into the server's databases. As you examine your network topology, record the IP addresses of any servers you want your clients to know about. The following are some examples of network services you may have on your network that the DHCP configuration does not find out about:

- Time server
- Log server
- Print server
- Install server
- Boot server
- Swap server
- X window font server
- TFTP server

Network Topology to Avoid

DHCP does not work well in network environments where more than one IP network is sharing the same network hardware media, either through the use of multiple network hardware interfaces or multiple logical interfaces. When multiple IP networks run across the same physical LAN, a DHCP client's request arrives on all network hardware interfaces, making the client appear to be attached to all of the IP networks simultaneously.

DHCP must be able to determine the address of a client's network in order to assign an appropriate IP address to the client. If more than one network is present on the

hardware media, the server cannot determine the client's network, and cannot assign an IP address.

You can use DHCP on one of the networks, but not more than one. If this does not suit your needs, you must reconfigure the networks. Suggestions for reconfiguring include:

- Use variable length subnet masks (VLSM) to make better use of the IP address space you have, so you do not need to run multiple LANs on the same physical network. See RFC-1519 for more information on VLSM and Classless Inter-Domain Routing (CDIR).
- Configure the ports on your switches to assign devices to different physical LANs. This preserves the mapping of one LAN to one IP network required for Solaris DHCP. See the documentation for the switch for information about port configuration.

Determining the Number of DHCP Servers

The data store method you use has a direct effect on the number of servers you must have to support your DHCP clients. If you use the local files method one server can support a maximum of 10,000 clients. If you use NIS+ one server can support a maximum of 40,000 clients.

The section "Choosing the Data Store" on page 172 discusses data store locations.

Updating System Files and Netmask Tables

During the configuration process, DHCP Manager or the `dhcpconfig` utility scans various system files on your server for information it can use to configure the server.

You must be sure the information in the system files is current before running DHCP Manager or `dhcpconfig` to configure your server. If you notice errors after configuring the server, manually change macros on the server to update the `dhcptab` configuration table.

The following table lists some of the information gathered during DHCP server configuration, and the sources for the information. Be sure this information is set correctly on the server before you configure DHCP on it. If you make changes to the system files after configuring the server, you should rerun DHCP Manager or `dhcpconfig` to pick up the changes.

TABLE 9–1 Information for DHCP Configuration

Information	Source	Comments
Timezone	System date, timezone settings	The date and timezone are initially set during the Solaris install. You can change the date using the `date` command and change the timezone by editing the `/etc/TIMEZONE` file, which sets the TZ variable.
DNS parameters	`/etc/resolv.conf`	DHCP server uses `/etc/resolv.conf` to look up DNS parameters such as DNS domain name and DNS server addresses. See (link to "Setting Up DNS Clients" section of the Solaris Naming Setup and Configuration Guide) for more information about `resolv.conf`.
NIS+ parameters	System domain name, `nsswitch.conf`, NIS+	DHCP server uses the `domainname` command to obtain the domain name of the server machine, the `nsswitch.conf` file to determine where to look for domain-based information. If the server machine is a NIS or NIS+ client, the DHCP server queries NIS or NIS+ services to get NIS/NIS+ server IP addresses.
Default router	System routing tables, user prompt	DHCP server searches the network routing tables to find the default router for clients attached to the local network. For clients not on the same network, DHCP server must obtain the information by prompting the administrator.

TABLE 9–1 Information for DHCP Configuration *(continued)*

Information	Source	Comments
Subnet mask	Network interface, `netmasks` table	DHCP server looks to its own network interfaces to determine the netmask and broadcast address for local clients. If the request had been forwarded by a relay agent, the server looks up the subnet mask in the netmasks table on the relay agent's network.
Broadcast address	Network interface, `netmasks` table	For the local network, DHCP server obtains the broadcast address by querying the network interface. For remote networks, the server uses the BOOTP relay agent's IP address and the remote network's netmask to calculate the broadcast address for the network.

Making Decisions for Server Configuration

This section discusses some of the decisions to make before configuring the first DHCP server on your network. The topics parallel the dialogs in the DHCP Manager's Configuration Wizard, but the information is also useful if you decide to use the `dhcpconfig` utility to configure the server.

Selecting a Server for DHCP

With your network topology in mind, you can use the following guidelines to select a host on which to set up a DHCP server.

The server must:

- Run the Solaris 2.6, Solaris 7, or Solaris 8 operating environment
- Be accessible to all the networks having clients that will use DHCP, either directly on the network or through a BOOTP relay agent
- Be configured to use routing

- Have a correctly configured `netmasks` table that reflects your network topology

Choosing the Data Store

You can choose to store the DHCP data using files in a local directory or NIS+ tables in a NIS+ directory service. Because NIS+ is distributed, multiple servers can access the same database. NIS+ also provides inherently faster information retrieval. Note that the server machine must already be configured as a NIS+ client to use this option.

The files method can be used efficiently at sites having less than 10,000 DHCP clients, but it is somewhat slower than NIS+, and requires all DHCP data to be stored on one file system. The data stored in files can only be shared with multiple DHCP servers if it is exported through an NFS mount point.

Traditional NIS (as opposed to NIS+) is not offered as a data store option because it does not support fast incremental updates. If your network uses NIS, you should use files for your data store.

Setting a Lease Policy

A lease specifies the amount of time the DHCP server grants permission to a DHCP client to use a particular IP address. During the initial server configuration, you must specify a site-wide lease policy, indicating the lease time and whether clients can renew their leases. The server uses the information you supply to set option values in the default macros it creates during configuration. You can set different lease policies for specific clients or type of clients, by setting options in configuration macros you create.

The lease time is specified as a number of hours, days, or weeks for which the lease is valid. When a client is assigned an IP address (or renegotiates a lease on an IP address it is already assigned), the lease expiration date and time is calculated by adding the number of hours in the lease time to the timestamp on the client's DHCP acknowledgment. For example, if the timestamp of the DHCP acknowledgment is September 16, 1999 9:15 A.M., and the lease time is 24 hours, the lease expiration time is September 17, 1999 9:15 A.M. The lease expiration time is stored in the client's DHCP network record, viewable in the DHCP Manager or using `pntadm`.

The lease time value should be relatively small, so that expired addresses are reclaimed quickly, but large enough so that if your DHCP service becomes unavailable, the clients continue to function until the machine(s) running the DHCP service can be repaired. A rule of thumb is to specify a time that is two times the predicted down time of a server. For example, if it generally takes four hours to obtain and replace a defective part and reboot the server, you should specify a lease time of eight hours.

The lease negotiation option determines whether or not a client can renegotiate its lease with the server before the lease expires. If lease negotiation is allowed, the client tracks the time remaining in its lease, and when half the lease time is used, the client requests the DHCP server to extend its lease to the original lease time. Disallowing lease negotiation is useful for environments where there are more machines than IP addresses, so the time limit is enforced on the use of IP addresses. If there are enough IP addresses, lease negotiation should be permitted to avoid forcing a client to take down its network interface and obtain a new lease, possibly interrupting their TCP connections (such as NFS and telnet sessions). Lease negotiation can be set site-wide during the server configuration, and for particular clients or types of clients through the use of the `LeaseNeg` option in configuration macros.

Note - Systems providing services on the network should retain their IP addresses, and should not be subject to short-term leases. You can use DHCP with such machines by assigning them reserved (manual) IP addresses, rather than IP addresses with permanent leases. This enables you to detect when the machine's IP address is no longer being used.

Determining Routers for DHCP Clients

Clients use routers for any network communication beyond their local network, and they must know the IP addresses of these routers in order to use them.

During DHCP server configuration, you must provide the IP address of a router the clients can use or, if you use DHCP Manager, you can specify that clients should find routers themselves by using the router discovery protocol.

If clients on your network support router discovery, you should use router discovery protocol instead of specifying the IP address, even if there is only one router. Discovery enables a client to adapt easily to router changes in the network. For example, if a router fails and is replaced by one with a new address, clients can discover the new address automatically without having to obtain a new network configuration to get the new router address.

Making Decisions for IP Address Management

This section discusses the decisions you need to make when configuring IP addresses to be managed by DHCP. The topics parallel the dialogs of DHCP Manager's Address Wizard, but can also be used to make decisions if you use the `dhcpconfig` utility.

As part of the DHCP service setup, you determine several aspects of the IP addresses that the server is to manage. If your network needs more than one DHCP server, you must decide how to divide responsibility for the addresses so you can assign some to each server. Before you begin configuring your server you must decide on the following:

- Number or range of IP addresses that the server should manage
- Whether you want the server to automatically generate host names for clients, and the prefix to use for generated host names
- What configuration macro to use to assign clients' network configuration
- Whether IP address leases are dynamic or permanent

Number and Ranges of IP Addresses

During the initial server configuration, DHCP Manager allows you to add one block, or range, of IP addresses under DHCP management by specifying the total number of addresses and the first address in the block. DHCP Manager creates a list of contiguous addresses from this information. If you have several blocks of noncontiguous addresses, you can add the others by running DHCP Manager's Address Wizard again, after the initial configuration.

Before configuring your IP addresses, know how many addresses are in the initial block of addresses you want to add and the IP address of the first address in the range.

Client Host Name Generation

The dynamic nature of DHCP means that an IP address is not permanently associated with the host name of the system that is using it. The Solaris DHCP server can generate a client name to associate with each IP address, if you select this option. The generated client names are mapped to IP addresses in `/etc/hosts` or the NIS/ NIS+ hosts tables. The client names use a prefix, or root name, plus a dash and a number assigned by the server. For example, if the root name is `charlie`, the client names will be `charlie-1`, `charlie-2`, `charlie-3`, and so on.

By default, generated client names begin with the name of the DHCP server that manages them. This is useful in environments having more than one DHCP server because you can quickly see in the DHCP network tables which clients any given DHCP server manages. However, you can change the root name to any name you choose.

Before configuring your IP addresses, decide if you want the server to generate client names, and if so, what root name to use for the names.

Note - The client names are not automatically added to the DNS domain, thus the client names are not known outside your name service (NIS/NIS+) domain. However, you can load them into DNS manually. See "Administering DNS" in *Solaris Naming Administration Guide* and the `in.named`(1M) manual page for more information about DNS.

Default Client Configuration Macros

In Solaris DHCP, a macro is a collection of network configuration options and their assigned values. Macros are used to determine what network configuration information to send to a DHCP client.

During the initial configuration of the DHCP server, several macros are created, using information gathered from system files and from prompting the system administrator:

- Network address macro, named using the IP address of the client network, and containing information needed by any client that is part of the network, such as subnet mask, network broadcast address, default router or router discovery token, and NIS/NIS+ domain and server if the server is using NIS/NIS+. Other options applicable to your network might be included.
- Locale macro, which contains the offset (in seconds) from Universal Time to specify the time zone.
- Server macro, named using the server's host name, and containing information about the lease policy, time server, DNS domain, and DNS server, and possibly other information that the configuration program was able to obtain from system files. This macro includes the locale macro.

The network address macro is automatically processed for all clients located on that network. The locale macro is included in the server macro, so it is processed when the server macro is processed.

While configuring IP addresses for the first network, you must select a client configuration macro to be used for all DHCP clients using the addresses you are configuring. By default, the server macro is selected because it is contains information needed by all clients using this server. Clients receive the options contained in the network address macro before those in the server macro. See "Order of Macro Processing" on page 161 for more information about macro processing order.

Dynamic and Permanent Lease Type

The lease type determines whether the lease policy (lease time and negotiation) is used for the addresses you are configuring. During initial server configuration, DHCP Manager allows you to select either dynamic or permanent leases for the addresses you are adding. The `dhcpconfig` utility allows only dynamic leases.

When an address has a dynamic lease, the DHCP server can manage the address by allocating it to a client, extending the lease time, detecting when it is no longer in use, and reclaiming it. When an address has a permanent lease, the DHCP server can only allocate it to a client, after which the client owns the address until the client explicitly releases it. When the address is released, the server can assign it to another client. The address is not subject to the lease policy as long as it is configured with a permanent lease type.

When you configure a range of IP addresses, the lease type you select applies to all the addresses in the range. To get the most benefit from DHCP, you should use dynamic leases for most of the addresses. You can later modify individual addresses to make them permanent if necessary, but the total number of permanent leases should be kept to a minimum.

Reserved Addresses and Lease Type

Addresses can be reserved by manually assigning them to particular clients. A reserved address can have a permanent or dynamic lease associated with it. When a reserved address is assigned a permanent lease, the address can only be allocated to the client that is bound to the address, the DHCP server cannot allocate the address to another client, and the address cannot be reclaimed by the DHCP server.

If a reserved address is assigned a dynamic lease, the address can only be allocated to the client that is bound to the address, but the client must track lease time and negotiate for a lease extension as if the address were not reserved. This allows you to track when the client is using the address by looking at the network table.

You cannot create reserved addresses for all the IP addresses during the initial configuration because they are intended to be used sparingly for individual addresses.

Planning for Multiple DHCP Servers

If you want to configure more than one DHCP server to manage your IP addresses, consider the following guidelines:

- Divide the pool of IP addresses so that each server is responsible for a range of addresses and there is no overlap of responsibility.

- Choose NIS+ as your data store, if available. If not, choose files and specify a shared directory for the absolute path to the data store.
- Configure each server separately so that address ownership is allocated correctly and that server-based macros can be automatically created.
- Set up the servers to scan the options and macros in the `dhcptab` file at specified intervals so they each are using the latest information. You can do this by using the `–t` option with `in.dhcpd`(1M).
- Be sure that all clients can access all DHCP servers so that the servers can support one another. If a client has a valid IP address lease, and is either trying to verify its configuration or extend the lease, but the server owning the client's address is not reachable, another server can respond to the client after the client has attempted to contact the primary server for 20 seconds. If a client requests a specific address, and the server owning that address is not available, one of the other servers handles the request. The client receives a different address from the one requested.

Planning for Remote Network Configuration

After the initial configuration, you can place IP addresses in remote networks under DHCP management. However, because the system files are not local to the server, most of the information is not looked up to provide default values, so you must provide the information. Before you attempt to configure a remote network, be sure you know the following information:

- Remote network's IP address
- Subnet mask of the remote network – This can be obtained from the `netmasks` table in the name service. If the network uses local files, look in `/etc/netmasks` on a system in the network. If the network uses NIS+, use the command `niscat netmasks.org_dir`. If the network uses NIS, use the command `ypcat -k netmasks.byaddr`. Make sure the `netmasks` table contains all the topology information for all the subnets you want to manage.
- Network type – Do the clients connect to the network through a local area network (LAN) connection or using point-to-point protocol (PPP)?
- Routing – Can the client use router discovery? If not, you must find out the IP address of a router they can use.
- NIS domain and NIS servers, if applicable
- NIS+ domain and NIS+ servers, if applicable

See “Adding, Modifying, and Removing DHCP Networks” on page 218 for the procedure for adding DHCP networks.

Selecting the Tool for Configuring DHCP

After you have gathered information and made decisions, as outlined in the previous sections, you are ready to configure a DHCP server. You can use the graphical DHCP Manager or the command-line utility `dhcpconfig` to configure a server. Each tool lets you select options and type in data that is then used to create the `dhcptab` and network tables used by the DHCP server.

DHCP Manager Features

DHCP Manager, a Java-based graphical tool, provides a DHCP Configuration Wizard, which starts automatically the first time you run DHCP Manager on a system that is not configured as a DHCP server. The DHCP Configuration Wizard is a series of dialog boxes that prompt you for the essential information required to configure a server: data store, lease policy, DNS/NIS/NIS+ servers and domains, and router addresses. Some of the information is obtained by the wizard from system files, and you only need to confirm that the information is correct, and correct it if necessary.

When you progress through the dialog boxes and approve the information, and the DHCP server daemon starts on the server system, you are prompted to start the Add Addresses Wizard to configure IP addresses for the network. Only the server’s network is configured for DHCP initially, and other server options are given default values. You can run DHCP Manager again after the initial configuration is complete to add networks and modify other server options.

`dhcpconfig` Features

The `dhcpconfig` utility is a wrapper script for the `dhtadm` and `pntadm` commands. The `dhcpconfig` utility prompts you for server startup options such as the interval for reading the `dhcptab`, the timeout value for DHCP service offers, and so on. It obtains other information from the system files discussed in “Updating System Files and Netmask Tables” on page 169. You cannot view the information it obtains from system files, so it is important that the system files be updated before running `dhcpconfig`.

Note that you can rerun `dhcpconfig` after updating system files and it will properly update the DHCP data.

Comparison of DHCP Manager and `dhcpconfig`

The following table summarizes the differences between the two server configuration tools.

TABLE 9–2 Comparison of DHCP Manager and dhcpconfig

Feature	**DHCP Manager**	`dhcpconfig`
Network information gathered from system	Allows you to view the information gathered from system files, and change it if needed.	You cannot see what information `dhcpconfig` is gathering. You must look at the `dhcptab` and network tables after they are created.
Configuration experience for user	Speeds up configuration process by omitting prompts for nonessential server options by using default values for them. Allows you to change nonessential options after initial configuration.	Prompts for all server options during configuration process. To change the options later, you must rerun `dhcpconfig` or use `dhtadm` and `pntadm` commands.
User input checking	Checks validity of user input as it is entered.	Does not check validity of user input as it is entered.

The next chapter includes procedures for configuring your server using both DHCP Manager and `dhcpconfig`.

CHAPTER 10

Configuring DHCP Service

Configuring DHCP Service on your network consists largely of configuring and starting the first DHCP server. Other servers can be added later, and access the same data from a shared location. This chapter includes procedures for configuring the DHCP server and placing networks and their associated IP addresses under DHCP management. It also explains how to unconfigure a server.

This chapter also provides instructions for procedures using both DHCP Manager and `dhcpconfig` in separate sections. This chapter contains the following information:

Configuring and Unconfiguring a DHCP Server Using DHCP Manager

This section includes procedures for configuring and unconfiguring a server using DHCP Manager. Note that you must be running an X Window system such as CDE to use DHCP Manager.

The first time you run DHCP Manager on a system, the following screen is displayed to allow you to specify whether you want to configure a DHCP server or BOOTP relay agent.

Figure 10–1 Choose Server Configuration Dialog

Configuring DHCP Servers

When you configure a DHCP server, DHCP Manager starts the DHCP Configuration Wizard, which prompts you for information needed to configure the server. The initial screen of the wizard is shown in the following figure.

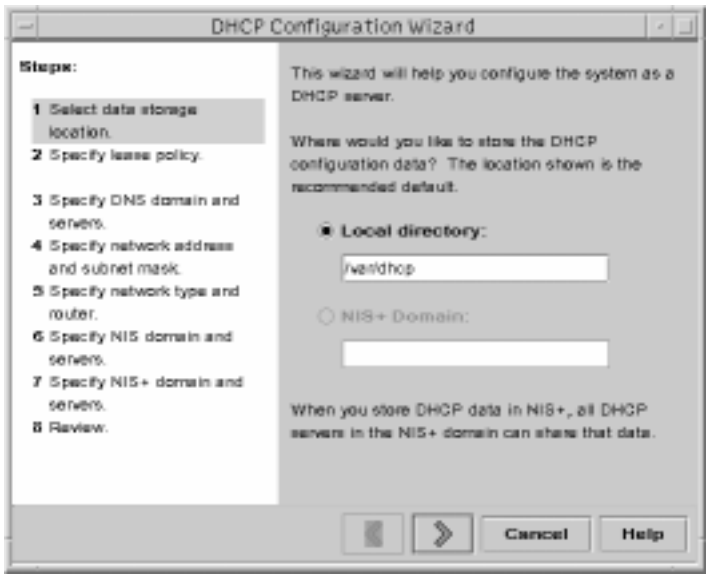


Figure 10–2 DHCP Configuration Wizard's Initial Screen

When you finish answering the wizard prompts, DHCP Manager creates the items listed in the following table.

TABLE 10–1 Items Created During DHCP Server Configuration

Item	Description	Contents
Service configuration file, `/etc/default/dhcp`	Records keywords and values for server configuration options.	Data store type and location, options used with `in.dhcpd` to start the DHCP daemon when system boots.
`dhcptab` file	DHCP Manager creates an empty file`dhcptab` if it does not already exist.	Macros and options with assigned values.
`Locale` macro, optional	Contains the local time zone's offset in seconds from Universal Time (UTC).	`UTCoffst` option
Server macro, named to match server's node name	Contains options whose values were set using input from administrator configuring the DHCP server. Options should apply to all clients using addresses owned by the server.	The `Locale` macro, plus the following options: ■ `Timeserv`, set to point to the server's primary IP address ■ `LeaseTim`, and `LeaseNeg` if you selected negotiable leases ■ `DNSdmain` and `DNSserv`, if DNS is configured ■ `Hostname`, which *must not* be assigned a value. The presence of this option indicates that the hostname must be obtained from the name service.
Network address macro, whose name is the same as the network address of client's network	Contains options whose values were set using input from administrator configuring the DHCP server. Options should apply to all clients residing on the network matching the macro name.	The following options: ■ `Subnet` ■ `Router` or `RDiscvyF` ■ `Broadcst`, if the network is a LAN ■ `MTU` ■ `NISdmain` and `NISserv`, if NIS is configured ■ `NIS+dom` and `NIS+serv`, if NIS+ is configured

TABLE 10–1 Items Created During DHCP Server Configuration *(continued)*

Item	Description	Contents
Network table for the network.	Empty table is created until you create IP address for the network.	None, until you add IP addresses.
Links to the DHCP service script, `/etc/init.d/dhcp`.	Enables the DHCP daemon to start automatically when the system boots.	The following links are made: ■ `/etc/rc0.d/K34dhcp` ■ `/etc/rc1.d/K34dhcp` ■ `/etc/rc2.d/K34dhcp` ■ `/etc/rc3.d/S34dhcp`

▼ How to Configure a DHCP Server (DHCP Manager)

1. **Select the system you want to use as a DHCP server.**
   Use the guidelines in "Making Decisions for Server Configuration" on page 171.

2. **Make decisions about your data store, lease policy, and router information.**
   Use the guidelines in "Making Decisions for Server Configuration" on page 171.

3. **Become superuser on the server system.**

4. **Type the following command**:

```
#/usr/sadm/admin/bin/dhcpmgr &
```

5. **Choose the option Configure as DHCP Server.**
   This starts the DHCP Configuration Wizard, which guides you in configuring your server.

6. **Select options or type requested information based on the decisions you made during the planning phase.**
   If you have difficulty, click Help in the Wizard window to open your web browser and display help for the DHCP Configuration Wizard.

7. **Click Finish to complete the server configuration when you have finished entering the requested information.**

8. **At the Start Address Wizard window, click Yes to begin configuring addresses for the server.**

   The Address Wizard enables you to specify which addresses to place under control of DHCP.

9. **Answer the prompts based on decisions made during the planning phase.**

   See “Making Decisions for IP Address Management” on page 173 for more information. If you have difficulty, click Help in the Wizard window to open your web browser and display help for the Add Addresses Wizard.

10. **Review your selections, and click Finish to add the addresses to the network table.**

    The network table is updated with records for each address in the range you specified.

You can add more networks to the DHCP server using the Network Wizard, as explained in “Adding DHCP Networks” on page 221.

Configuring BOOTP Relay Agents

When you configure a BOOTP relay agent, DHCP Manager takes the following actions:

- Prompts you for IP addresses of DHCP server to which requests should be relayed.
- Edits `/etc/default/dhcp`, to specify the options needed for BOOTP relay service.
- Creates the following links to `/etc/init.d/dhcp` to enable the DHCP daemon to start when the system boots:
  - `/etc/rc0.d/K34dhcp`
  - `/etc/rc1.d/K34dhcp`
  - `/etc/rc2.d/K34dhcp`
  - `/etc/rc3.d/S34dhcp`

The following figure shows the screen displayed when you choose to configure a BOOTP relay agent.

Figure 10–3 Configure BOOTP Relay Dialog Box

▼ How to Configure a BOOTP Relay Agent (DHCP Manager)

1. **Select the system you want to use as a BOOTP relay agent.**
   See “Selecting a Server for DHCP” on page 171.

2. **Become superuser on the server system.**

3. **Type the following:**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

   If the system has not been configured as a DHCP server or BOOTP relay agent, the DHCP Configuration Wizard starts. If the system has already been configured as a DHCP server, you cannot configure it as a BOOTP relay agent unless you unconfigure the server first. See “Unconfiguring DHCP Servers and BOOTP Relay Agents” on page 187.

4. **Select Configure as BOOTP Relay.**
   The Configure BOOTP Relay dialog box opens.

5. **Type the IP address or hostname of one or more DHCP servers that are configured to handle BOOTP or DHCP requests received by this BOOTP relay agent, and click Add.**
   After you click OK, notice that the DHCP Manager offers only the File menu to exit the application and the Service menu to manage the server. Other menu options are disabled because they are useful only on a DHCP server.

Unconfiguring DHCP Servers and BOOTP Relay Agents

When you unconfigure a DHCP server or BOOTP relay agent, DHCP Manager takes the following actions:

- Stops the DHCP daemon (`in.dhpcd`) process
- Removes the links that enable automatic starting when the system boots
- Removes the `/etc/default/dhcp` file, which records information about daemon startup and the data store location

The following figure shows the screen that is displayed when you choose to unconfigure a DHCP server.

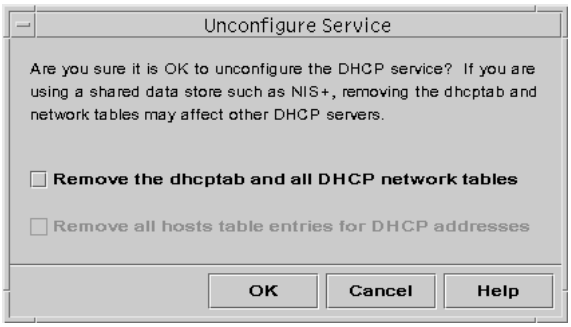


Figure 10–4 Unconfigure Service Dialog Box

When you unconfigure a DHCP server you must decide what to do with the DHCP data files: `dhcptab` and the DHCP network tables. If the data is shared among servers, you should not remove the `dhcptab` and DHCP network tables because this could render DHCP unusable across your network. Data can be shared through NIS+ or on exported local file systems. The file `/etc/default/dhcp` records the data store used and its location.

You can unconfigure a DHCP server while leaving the data intact by not selecting any of the options to remove data. Unconfiguring while leaving the data intact in effect disables the DHCP server.

If you are certain that you want to remove the data, you can select an option to remove the `dhcptab` and network tables. If you had generated client names for the DHCP addresses, you can also elect to remove those entries from `/etc/inet/hosts` or the NIS+ `hosts` table.

Before you unconfigure a BOOTP relay agent, be sure that no clients rely on this agent to forward requests to a DHCP server.

▼ How to Unconfigure a DHCP Server or BOOTP Relay Agent (DHCP Manager)

1. **Become superuser.**

2. **Type the following:**

```
#/usr/sadm/admin/bin/dhcpmgr &
```

3. **From the Services menu, choose Unconfigure.**

   The Unconfigure Service dialog box is displayed. If the server is a BOOTP relay agent, the dialog box enables you to confirm your intention to unconfigure the relay agent. If the server is a DHCP server, you must decide what to do with the DHCP data by making selections in the dialog box. See Figure 10–4.

4. **Select options to remove data.**

   If the server is using shared data (using NIS+ or files shared through NFS), do not select any options to remove the data. If the server is a DHCP server that is not using shared data, select one or both options to remove the data.

   See “Unconfiguring DHCP Servers and BOOTP Relay Agents” on page 187 for more information about removing data.

5. **Click OK to confirm.**

Configuring and Unconfiguring a DHCP Server Using `dhcpconfig`

This section includes procedures for configuring and unconfiguring a DHCP server or BOOTP relay agent using `dhcpconfig`. When you start `dhcpconfig`, the DHCP Configuration menu is displayed, as shown in the following figure.

```
***         DHCP Configuration              ***

Would you like to:

        1) Configure DHCP Service
```

(continued)

```
        2) Configure BOOTP Relay Agent
        3) Unconfigure DHCP or Relay Service
        4) Exit
Choice:
```

Figure 10–5 `dhcpconfig` Menu

▼ How to Configure a DHCP Server (`dhcpconfig`)

1. **Select the system you want to use as a DHCP server.**
   Use the guidelines in “Making Decisions for Server Configuration” on page 171.

2. **Make decisions about your data store, lease policy, and router information.**
   Use the guidelines in “Making Decisions for Server Configuration” on page 171.

3. **Become superuser.**

4. **Type the following:**

   ```
   #/usr/sbin/dhcpconfig
   ```

   The text-based DHCP Configuration menu is displayed.

5. **Type 1 and press Return to select Configure DHCP Service.**

6. **Answer the prompts listed below.**
   Use the decisions you made after reading Chapter 9. Note that the default value for each prompt is displayed in square brackets. If you want to use a default value, press Return at the prompt.

   ```
   Would you like to stop the DHCP service? (recommended) ([Y]/N)Y
   ```

   Type **Y** to stop the DHCP service. This ensures that the server does not attempt to send incomplete configuration information to clients.

```
###     DHCP Service Configuration      ###
###     Configure DHCP Database Type and  Location      ###

Enter datastore (files or nisplus) [nisplus]:
```

Type the name of the data store you have decided to use, either **files** or **nisplus**.

See the guidelines in "Choosing the Data Store" on page 172 if you need more information about the data store. Your choice is recorded in /etc/default/dhcp.

```
Enter absolute path to datastore directory [/var/dhcp]:
```

Type the path to the files or NIS+ directory that you want to use for the data store. The default location if you selected files for the data store is /var/dhcp. If you selected NIS+, the default listed is the location of the NIS+ directory that the server is already using, such as yourcompany.com.

```
Would you like to specify nondefault daemon options (Y/[N]):
```

You can successfully configure the server without specifying nondefault daemon options if you type **N** at this prompt.

However, if you type **Y** here, the following prompts are displayed.

```
Do you want to enable transaction logging? (Y/[N]):Y
```

Type **Y** if you want to enable transaction logging. See the Administering DHCP chapter for information about transaction logging. The following prompt appears only if you enable transaction logging.

```
Which syslog local facility [0-7] do you wish to log to? [0]:
```

See the Administering DHCP chapter for information about the local facility for transaction logging.

```
How long (in seconds) should the DHCP server keep outstanding OFFERs? [10]:
```

Type the number of seconds the server should cache an IP address offer to a client. The default is 10 seconds, which is adequate for most networks. You can increase this time to compensate for slow network performance.

```
How often (in minutes) should the DHCP server rescan the dhcptab? [Never]:
```

By default, the DHCP server reads the `dhcptab` only at startup or if signalled by DHCP Manager to read it. DHCP Manager enables you to update the server by reloading the `dhcptab` after you make a change to the configuration data, so automatic rescanning is not necessary if you use DHCP Manager. Generally, you should use a rescan interval only under the following circumstances:

- The data store is in NIS+ and you have more than one DHCP server on your network. Rescanning guarantees that all servers have the latest information.
- You use `dhtadm` instead of DHCP Manager to make configuration changes. The `dhtadm` utility does not offer you the option of forcing a rescan of `dhcptab` after you make a change.

If you decide to use the automatic rescan for `dhcptab`, type the interval in minutes that the server should wait to reload the client configuration information in the `dhcptab` file.

```
Do you want to enable BOOTP compatibility mode? (Y/[N]):
```

The default is to not enable BOOTP compatibility. See “Supporting BOOTP Clients with DHCP Service” on page 227 if you want to enable BOOTP compatibility.

After you finish entering information about nondefault daemon and server options, the following prompt is displayed:

```
Enter default DHCP lease policy (in days) [3]:
```

Type the number of days for the lease time. The default is three days. See “Setting a Lease Policy” on page 172 for more information.

```
Do you want to allow clients to renegotiate their leases? ([Y]/N):
```

The default is Y to allow lease negotiation. See “Setting a Lease Policy” on page 172 for more information about lease negotiation. If you type **N**, clients must give up their IP addresses when the lease expires, and then obtain a new lease and IP address.

```
Enable DHCP/BOOTP support of networks you select? ([Y]/N):
```

At this point, you can begin configuring the networks that should use DHCP. Refer to the decisions you made after reading “Making Decisions for IP Address Management” on page 173. If you are not ready to configure IP addresses, type **N**; dhcpconfig prompts you to restart the DHCP service, and returns to the initial menu. Note that DHCP is not usable until you enable DHCP/BOOTP support of at least one network.

If you are ready to configure IP addresses, type **Y** and continue to Step 4 on page 194.

▼ How to Configure a BOOTP Relay Agent (dhcpconfig)

1. **Become superuser on the system you want to configure.**

2. **Type the following:**

```
# /usr/sbin/dhcpconfig
```

The text-based DHCP Configuration menu is displayed.

3. **Type 2 and press Return to select Configure BOOTP Relay Agent.**

4. **Answer the prompts as follows:**

```
Would you like to stop the DHCP service? (recommended) ([Y]/N):Y
###     BOOTP Relay Agent Configuration ###

###     Common daemon option setup      ###

Would you like to specify nondefault daemon options (Y/[N]):Y
```

You can successfully configure the server without specifying nondefault daemon options if you type **N** at this prompt.

However, if you type **Y** here, the following prompts are displayed.

```
Do you want to enable transaction logging? (Y/[N]):Y
```

Type **Y** if you want to enable transaction logging. See the Administering DHCP chapter for information about transaction logging. The following prompt appears only if you enable transaction logging:

```
Which syslog local facility [0-7] do you wish to log to? [0]:
```

See the Administering DHCP chapter for information about the local facility for transaction logging.

```
Enter destination BOOTP/DHCP servers. Type '.' when finished.
IP address or Hostname:
```

Type one IP address or hostname for a BOOTP or DHCP server to which requests should be forwarded, and press Return. The prompt reappears to allow you to type more addresses or hostnames. Type a period and press Return when you are finished.

```
Would you like to restart the DHCP service? (recommended) ([Y]/N):Y
```

The DHCP Configuration menu is redisplayed.

5. **Type 4 to exit** `dhcpconfig`**.**

Configuring Networks Using `dhcpconfig`

This section describes the procedures for placing a network under DHCP management using `dhcpconfig`. Each procedure assumes that the server configuration is complete and you are now adding new networks to the DHCP data store.

▼ How to Configure the Local Network (`dhcpconfig`)

1. **Become superuser on the DHCP server system.**

2. **Type the following command**:

```
# /usr/sbin/dhcpconfig
```

The text-based DHCP Configuration menu is displayed.

3. **Answer the following prompts as shown to skip to the network configuration options.**

```
Would you like to stop the DHCP service? (recommended) ([Y]/N)Y
Would you like to specify nondefault daemon options (Y/[N]):N
Do you want to merge initialization data with the existing table? (Y/[N]):Y
Enable DHCP/BOOTP support of networks you select? ([Y]/N):Y
```

4. **Answer the following prompt as shown to configure the local network.**

```
Configure BOOTP/DHCP on local LAN network: 172.21.0.0? ([Y]/N):Y
```

5. **Answer the following prompts about generating client names.**

```
Do you want hostnames generated and inserted in the files hosts table? (Y/[N]):
```

The server can create host names and associate one with each IP address. See“Client Host Name Generation” on page 174 for more information.

If you type **Y**, answer the next prompt. If you type **N**, skip to step Step 6 on page 194.

```
What rootname do you want to use for generated names? [yourserver-]:
```

The default prefix, or rootname, for generated client names is the name of the DHCP server. You can accept this, or change the name to anything you like.

```
Is Rootname name_you_typed- correct? ([Y]/N):Y
```

If you made an error, type **N** here to be prompted again for the rootname.

```
What base number do you want to start with? [1]:
```

The base number indicates the first number appended to the rootname used to generate client names. For example, if you accept the default rootname and base number, the client names would be `yourserver-1`, `yourserver-2`, and so on.

6. **Answer the following prompts about the IP addresses in this network that you want to be placed under DHCP management.**

```
Enter starting IP address [172.21.0.0]:
```

The server must generate a range of IP addresses that it is to manage. Type the first address in the range that you want to be placed under DHCP management. See “Number and Ranges of IP Addresses” on page 174 for more information.

```
Enter the number of clients you want to add (x < 65535):
```

The number of clients is the number of IP addresses you want to place under DHCP management. The `dhcpconfig` program uses this information and the base number to add a contiguous block of IP addresses to be managed by DHCP.

```
The dhcp network table: 172.21.0.0 already exists.
Do you want to add entries to it? ([Y]/N):
```

You see this prompt if you are adding a block of addresses in a network for which you have already configured addresses. Type **Y** to modify the network table and add the addresses.

```
Would you like to configure BOOTP/DHCP service on remote networks? ([Y]/N)
```

If you are finished adding networks, type **N**, and restart the server when prompted.

If you want to place IP addresses on other networks under DHCP management, type **Y** at this prompt and continue with Step 5 on page 196 in the following procedure.

▼ How to Configure Remote Networks (dhcpconfig)

1. **Become superuser on the DHCP server system.**

2. **Type the following command:**

```
# /usr/sbin/dhcpconfig
```

The text-based DHCP Configuration menu is displayed.

3. **Type 1 and press Return to select Configure DHCP Service.**

4. **Answer the following prompts as shown to skip to the remote network configuration options.**

```
Would you like to stop the DHCP service? (recommended) ([Y]/N)Y
Would you like to specify nondefault daemon options (Y/[N]):N
Do you want to merge initialization data with the existing table? (Y/[N]):Y
Enable DHCP/BOOTP support of networks you select? ([Y]/N):Y
Configure BOOTP/DHCP on local LAN network: 172.21.0.0? ([Y]/N):N
```

5. Answer the following prompts as shown to configure a remote network.

```
Would you like to configure BOOTP/DHCP service on remote networks? ([Y]/N):Y
Enter Network Address of remote network, or <RETURN> if finished:
```

Type the IP address of the network you want to configure for DHCP. Remember that the network address uses 0 for the host portion of the IP address.

```
Do clients access this remote network via LAN or PPP connection? ([L]/P):
```

Indicate whether the network is a local area network (LAN) or a point-to-point protocol network (PPP) by typing **L** or **P**.

```
Do you want hostnames generated and inserted in the files hosts table? (Y/[N]):
```

The server can create hostnames for each IP address and create entries in the `/etc/inet/hosts` file or NIS+ hosts table. See “Client Host Name Generation” on page 174.

```
Enter Router (From client’s perspective), or <RETURN> if finished.
IP address:
```

Type the IP address of the router(s) the clients on this network should use. Note that you cannot specify that clients should use router discovery here.

```
Optional: Enter Remote Network’s MTU (e.g. ethernet == 1500):
```

If you know that the remote network uses a specific maximum transfer unit (MTU), type it here. Otherwise, just press Return to accept the default value.

```
Enter starting IP address [172.21.0.0]
```

Type the first IP address in the range of addresses you want to place under DHCP management. The default value is the network address.

```
Enter the number of clients you want to add (x < 65535):
```

Type the number of IP addresses you want to place under DHCP management. `dhcpconfig` uses this number and the starting IP address you entered previously to determine a block of IP addesses to place under DHCP control. The number you enter must be less than the value shown in the prompt, which is generated based on the netmask. In this example, the number must be less than 65535.

```
dhcptab macro "172.21.0.0" already exists.
Do you want to merge initialization data with the existing macro? ([Y]/
N):
```

If you have already configured this network, this message is displayed. You should merge the data into the existing macro only if the information you provided applies to all clients on the network you are adding.

```
Disable (ping) verification of 172.21.0.0 address(es)? (Y/[N]):
```

`dhcpconfig` pings the addresses you want to add to be sure they are not being used, and skips any addresses that are in use. If you type Y at this prompt, `dhcpconfig` does not ping the addresses.

```
Network: 172.21.0.0 complete.
Enter Network Address of remote network, or <RETURN>  if finished:
```

If you want to configure another remote network, enter the network address and answer the prompts for the network. When you have no other remote networks to configure, press Return at this prompt.

```
Would you like to restart the DHCP service? (recommended) ([Y]/N):
```

Type **Y** to restart the DHCP service.

Unconfiguring DHCP Servers and BOOTP Relay Agents Using `dhcpconfig`

Unconfiguring a DHCP server stops the server process and prevents it from starting automatically when the system reboots. It also removes the `/etc/default/dhcp` file, which records information about the data store location and server startup options. When you unconfigure a DHCP server you must decide what to do with the DHCP data files: `dhcptab` and the DHCP network tables. If the data is shared among

servers, you should not remove the `dhcptab` and DHCP network tables because this could render DHCP unusable across your network. Data can be shared through NIS+ or on exported local file systems. You can unconfigure a DHCP server while leaving the data intact by declining to remove the tables when prompted to do so.

▼ How to Unconfigure DHCP Servers or BOOTP Relay Agents (`dhcpconfig`)

1. **Become superuser on the server system.**

2. **Type the following:**

```
# /usr/sbin/dhcpconfig
```

   The text-based DHCP Configuration menu is displayed.

3. **Type 3 and press Return to select Unconfigure DHCP or Relay Service.**

4. **Answer the prompts as follows:**

```
Unconfigure will stop the DHCP service and remove /etc/default/dhcp.
Are you SURE you want to disable the DHCP service? ([Y]/N):
```

   Type **Y** to unconfigure the server.

```
Are you SURE you want to remove the DHCP tables? (Y/[N]):
```

   Type **Y** only if you are certain that the DHCP data is not shared with other DHCP servers. If you type **N**, the server is disabled while the data remains intact.

Configuring and Unconfiguring a Solaris DHCP Client

When you install the Solaris operating environment from CD-ROM, you are prompted to use DHCP to configure network interfaces. If you select yes, the DHCP client software is enabled on your system during Solaris installation. There is nothing further you need to do on the client machine to use DHCP.

If a client machine is already running the Solaris operating environment and not using DHCP, you must unconfigure the system and issue some commands to set up the system to use DHCP when it boots.

If you are using a non-Solaris client, please consult the documentation for that client for instructions on configuring the DHCP client.

▼ How to Configure a Solaris DHCP Client

This procedure is necessary only if DHCP was not enabled during Solaris installation.

1. **Become superuser on the client machine.**

2. **Unconfigure and shut down the system by typing the following command:**

```
# sys-unconfig
```

   See the `sys-unconfig`(1M) manual page for more information about what configuration information is removed by this command.

3. **Reboot the system when the PROM prompt appears.**

```
ok boot
```

   You are prompted for system configuration information by `sysidtool` programs when the system reboots. See the `sysidtool`(1M) manual page for more information.

4. **When prompted to use DHCP to configure network interfaces, specify Yes.**

▼ How to Unconfigure a Solaris DHCP Client

1. **Become superuser on the client machine.**

2. **Unconfigure and shut down the system by typing the following command:**

```
# sys-unconfig
```

   See the `sys-unconfig`(1M) manual page for more information about what configuration information is removed by this command.

3. **Reboot the system when the PROM prompt appears.**

```
ok boot
```

Because you unconfigured the system, you will be prompted for configuration information by `sysidtool` programs when the system reboots. See the `sysidtool`(1M) manual page for more information.

4. **When prompted to use DHCP to configure network interfaces, specify No.**

CHAPTER 11

Administering DHCP

This chapter describes tasks useful in administering the Solaris DHCP service, including tasks for the server, BOOTP relay agent, and client. Each task includes a procedure for performing the task in DHCP Manager and a procedure for performing the equivalent task with DHCP service utilities. DHCP service utilities are more fully documented in man pages.

You should have already completed the initial configuration of your DHCP service and initial network before using this chapter. Chapter 10 discusses DHCP configuration.

The chapter contains the following information:

DHCP Manager

DHCP Manager is a Java-based graphical interface you can use to perform administration tasks on the DHCP service.

The DHCP Manager Window

The DHCP Manager window's appearance differs, depending on whether the server on which it is running was configured as a DHCP server or a BOOTP relay agent.

When the server is configured as a DHCP server, DHCP Manager uses a tab-based window, in which you select a tab for the type of information you want to work with. DHCP Manager features the following tabs:

- **Addresses** – Lists all networks and IP addresses placed under DHCP management. From the Addresses tab, you can add or delete networks, and add or delete IP addresses individually or in blocks. You can also modify the properties of individual networks or IP addresses, or make the same property modifications for a block of addresses simultaneously. When you start DHCP Manager, it opens on the Addresses tab.
- **Macros** – Lists all macros available in the DHCP configuration database (`dhcptab`) and the options contained within them. From the Macros tab, you can create or delete macros, and modify them by adding options and providing values for the options.
- **Options** – Lists all options that have been defined for this DHCP server. Options listed on this tab are not the standard ones defined in the DHCP protocol; the options are extensions to the standard options, having a class of Extended, Vendor, or Site. Standard options cannot be changed in any way so they are not listed here.

The following figure shows the DHCP Manager window as it appears when you start it on a DHCP server.

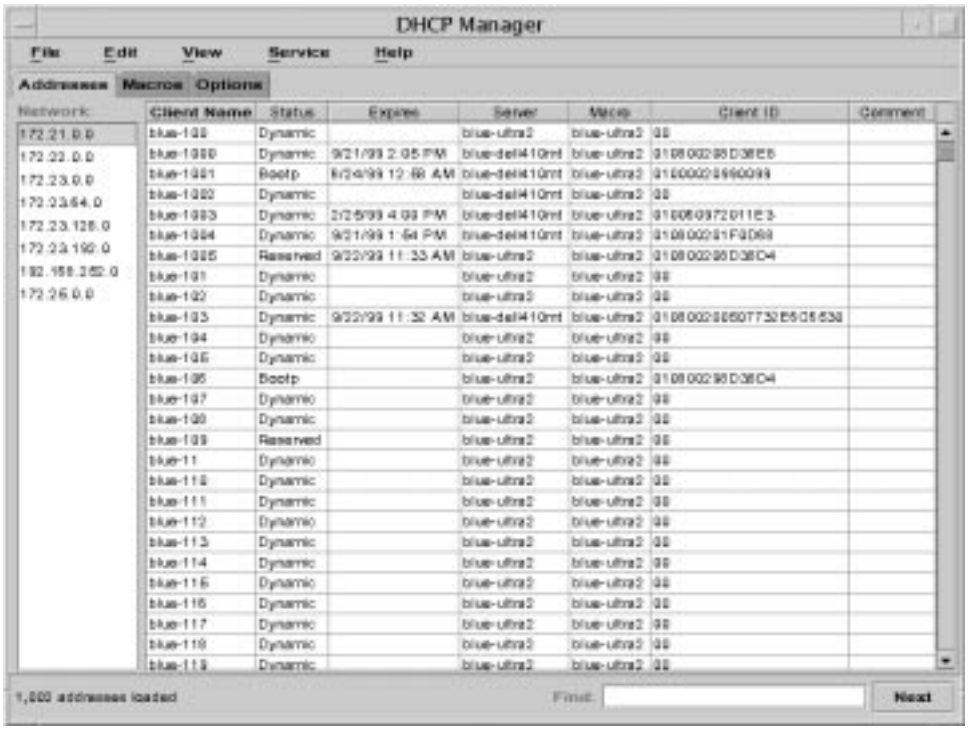


Figure 11–1 DHCP Manager Running on a DHCP Server System

When the server is configured as a BOOTP relay agent, the DHCP Manager window does not show these tabs because the BOOTP relay agent does not need any of this information. You can only modify the BOOTP relay agent's properties and stop/start the DHCP daemon with DHCP Manager. The following figure shows the DHCP Manager window as it appears when you start it on a system configured as a BOOTP relay agent.

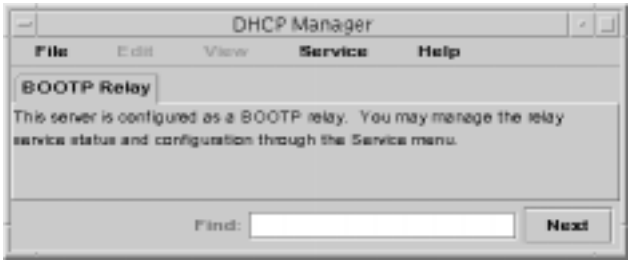


Figure 11–2 DHCP Manager Running on a BOOTP Relay Agent System

DHCP Manager Menus

DHCP Manager menus include:

- **File** – Exit DHCP Manager
- **Edit** – Perform management tasks upon networks, addresses, macros, and options
- **View** – Change the look of the tab currently selected
- **Services** – Manage the DHCP daemon
- **Help** – Open your web browser and display help for DHCP Manager

When DHCP Manager is run on a BOOTP relay agent, the Edit and View menus are disabled.

All DHCP service management activities are accomplished through the Edit and Service menus. You use the commands in the Edit menu to create, delete, and modify networks, addresses, macros, and options, depending on which tab is selected. When the Addresses tab is selected, the Edit menu also lists wizards, which are sets of dialogs that make it easy to create networks and multiple IP addresses. The Services menu lists commands for managing the DHCP daemon, enabling you to start/stop, enable/disable, modify the server configuration, and unconfigure the server.

Starting and Stopping DHCP Manager

You must run DHCP Manager on a DHCP server system as superuser, but you can display it remotely on another UNIX system using the X Windows remote display feature.

▼ How to Start DHCP Manager

1. **Become superuser on the DHCP server machine.**

2. **If you are logged in to the DHCP server machine remotely, you can display DHCP Manager on your local machine as follows.**

   a. **Type the following on the local machine:**

   ```
   # xhost +server-name
   ```

   b. **Type the following on the remote DHCP server machine:**

   ```
   # DISPLAY=local-hostname;export DISPLAY
   ```

3. **Type the following command:**

```
# /usr/sadm/admin/bin/dhcpmgr &
```

The DHCP Manager window opens, displaying the Addresses tab if the server is configured as a DHCP server, or no tabs if the server is configured as a BOOTP relay agent.

▼ How to Stop DHCP Manager

♦ **Choose Exit from the File menu.**

Starting and Stopping the DHCP Service

Starting and stopping the DHCP service encompasses several degrees of action you can take to affect the operation of the DHCP daemon. You must understand the difference between starting/stopping, enabling/disabling, and configuring/unconfiguring the DHCP service in order to select the correct procedure to obtain the result you want. The terms are explained below.

- **Starting, stopping, and restarting the DHCP service** affects the running of the daemon only at the current session. For example, if you stop the DHCP service, the currently running daemon terminates but restarts when you reboot the system. DHCP data tables are not affected by stopping the service.
- **Enabling and disabling the DHCP service** affects the running of the daemon for current and future sessions. If you disable the DHCP service, the currently running daemon terminates and does not start when you reboot the server. You must enable the DHCP daemon for the automatic start at system boot to occur. DHCP data tables are not affected.
- **Unconfiguring the DHCP service** shuts down the currently running daemon, prevents the daemon from starting on system reboot, and gives you the option of removing the DHCP data tables. Unconfiguration is described in Chapter 10.

Note - If a server has multiple network interfaces and you do not want to provide DHCP services on all the networks, see “Monitoring and Ignoring Network Interfaces for DHCP Service” on page 219.

This section provides the procedures for starting and stopping the DHCP service, and enabling and disabling it.

▼ How to Start and Stop the DHCP Service (DHCP Manager)

1. **Become superuser on the DHCP server machine.**

2. **Start DHCP Manager.**
   See "How to Start DHCP Manager" on page 204 for the procedure.

3. **Select one of the following operations:**
   a. **Choose Start from the Services menu to start the DHCP service.**

   b. **Choose Stop from the Services menu to stop the DHCP service.**
      The DHCP daemon stops until it is manually started again, or the system reboots.

   c. **Choose Restart from the Services menu to stop the DHCP service and immediately restart it.**

▼ How to Start and Stop the DHCP Service (Command Line)

1. **Become superuser on the server machine.**

2. **Choose one of the following operations:**
   a. **To start the DHCP service, type the following command:**

```
# /etc/init.d/dhcp start
```

   b. **To stop the DHCP service, type the following command:**

```
# /etc/init.d/dhcp stop
```

      The DHCP daemon stops until it is manually started again, or the system reboots.

▼ How to Enable and Disable the DHCP Service (DHCP Manager)

1. **Start DHCP Manager.**

2. **Choose one of the following operations:**
   a. **Choose Enable from the Services menu to start the DHCP service immediately and configure it for automatic startup when the system boots.**

   b. **Choose Disable from the Services menu to stop the DHCP service immediately and prevent it from starting automatically when the system boots.**

▼ How to Disable the DHCP Service (Command Line)

1. **As root, start** `dhcpconfig` **by typing the following command:**

```
# /usr/sbin/dhcpconfig
```

2. **Select Unconfigure DHCP or Relay Service by typing 3.**

3. **Type Y at the following prompt to disable DHCP:**

```
Unconfigure will stop the DHCP service and remove /etc/default/dhcp.
Are you SURE you want to disable the DHCP service? ([Y]/N): Y
```

▼ How to Enable the DHCP Service (Command Line)

This step is needed only if you previously disabled the server, keeping the data intact.

1. **Become superuser on the server system.**

2. **Start** `dhcpconfig` **by typing the following command**:

```
# /usr/sbin/dhcpconfig
```

3. **Select Configure DHCP Service or Configure BOOTP Relay Agent, as appropriate.**

4. **Press Return to accept default values for all prompts until you see the following prompt**:

```
Enable DHCP/BOOTP support of networks you select? ([Y]/N):
```

5. **Type Y at the prompt to enable the DHCP service.**

6. **Answer the following prompts as shown to avoid prompts for configuring networks.**

   If you had previously disabled the service while keeping the data intact, you should not need to reconfigure the network information.

```
###     Configure Local Networks        ###
Configure BOOTP/DHCP on local LAN network: 172.21.0.0? ([Y]/N):N
###     Configure Remote Networks       ###
Would you like to configure BOOTP/DHCP service on remote networks? ([Y]/
N):N
```

7. **Restart the DHCP service by pressing Return at the following prompt**:

```
Would you like to restart the DHCP service? (recommended) ([Y]/N):
```

Modifying DHCP Service Options

You can change values for some additional features of the DHCP service, some of which were not offered during the initial configuration with DHCP Manager. If you configured your server with `dhcpconfig`, you may have been prompted to select values for most of these options. You can use the Modify Service Options dialog box

in DHCP Manager or specify options on the `in.dhcpd` command to change service options.

The following task map shows the tasks related to service options and the procedures to use:

TABLE 11–1 Modify DHCP Service Options Task Map

Tasks...	Description	For Instructions, Go To...
Change logging options	Enable or disable verbose logging, enable or disable logging of DHCP transactions, and select a `syslog` facility to use for logging DHCP transactions.	"How to Generate Verbose DHCP Log Messages (DHCP Manager)" on page 212 "How to Generate Verbose DHCP Log Messages (Command Line)" on page 212 "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 213 "How to Enable and Disable DHCP Transaction Logging for Current Session (Command Line)" on page 213 "How to Log DHCP Transactions to a Separate `Syslog` File" on page 215
Enable or disable duplicate IP address detection	Enable or disable the DHCP server's checking that an IP address is not already in use before offering it to a client.	"How to Customize DHCP Server Performance Options (DHCP Manager)" on page 216 "How to Customize DHCP Server Performance Options (Command Line)" on page 217
Change options for DHCP server's reading of configuration information	Enable or disable automatic reading of `dhcptab` at specified intervals, or change the interval between reads	"How to Customize DHCP Server Performance Options (DHCP Manager)" on page 216 "How to Customize DHCP Server Performance Options (Command Line)" on page 217

TABLE 11–1 Modify DHCP Service Options Task Map *(continued)*

Tasks...	Description	For Instructions, Go To...
Change the number of relay agent hops	Increase or decrease the number of networks a request can travel through before being dropped by the DHCP daemon.	"How to Customize DHCP Server Performance Options (DHCP Manager)" on page 216 "How to Customize DHCP Server Performance Options (Command Line)" on page 217
Change the length of time an IP address offer is cached	Increase or decrease the number seconds that the DHCP service will reserve an offered IP address before offering to a new client	"How to Customize DHCP Server Performance Options (DHCP Manager)" on page 216 "How to Customize DHCP Server Performance Options (Command Line)" on page 217

The following figure shows DHCP Manager's Modify Services dialog box.

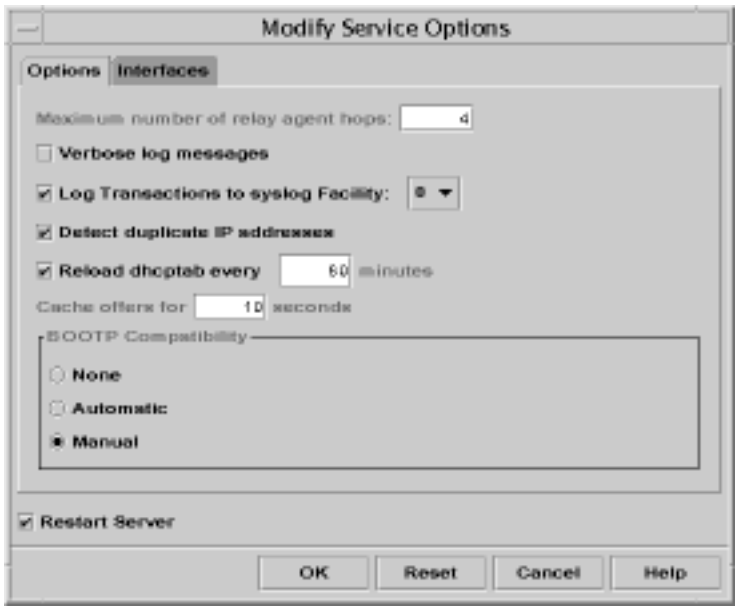


Figure 11–3 Modify Service Options Dialog Box

Changing DHCP Logging Options

The DHCP service can log DHCP service messages and DHCP transactions to `syslog`. See the `syslogd`(1M) and `syslog.conf`(4) manual pages for more information about `syslog`.

DHCP service messages logged to `syslog` include:

- Error messages, which notify the administrator of conditions that prevent the DHCP service from fulfilling a request by a client or by the administrator.
- Warnings and notices, which notify the administrator of conditions that are abnormal, but do not prevent the DHCP service from fulfilling a request.

You can increase information reporting by using the verbose option for the DHCP daemon. Verbose message output can be useful in troubleshooting DHCP problems. See “How to Generate Verbose DHCP Log Messages (DHCP Manager)” on page 212.

Another useful troubleshooting technique is transaction logging. Transactions provide information about every interchange between a DHCP server or BOOTP relay and clients. DHCP server transactions include:

- ASSIGN — IP address assignment
- EXTEND — Lease extension
- RELEASE — IP address release
- DECLINE — Client declining address assignment
- INFORM — Client requesting network configuration parameters but not an IP address
- NAK — Server does not acknowledge a client’s request to use a previously used IP address
- ICMP_ECHO — Server detects potential IP address for offering is already in use by another host.

BOOTP relay transactions include:

- RELAY-CLNT — Message being relayed from the DHCP client to a DHCP server
- RELAY–SRVR — Message being relayed from the DHCP server to the DHCP client

Transaction logging is disabled by default. When enabled, transaction logging uses the local0 `syslog` facility by default. DHCP transaction messages are generated with a `syslog` severity level of *notice*, so by default transactions are logged to the file where other notices are logged. However, because they use a local facility, the transaction messages can be logged separately from other notices if you edit the `syslog.conf` file to specify a separate log file.

You can disable or enable transaction logging, and specify a different `syslog` facility, from 0 through 7, as explained in "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 213. If you edit the server system's `syslog.conf` file, you can also instruct `syslogd` to store the DHCP transaction messages in a separate file, as explained in "How to Log DHCP Transactions to a Separate `Syslog` File" on page 215.

▼ How to Generate Verbose DHCP Log Messages (DHCP Manager)

1. **Choose Modify from the Services menu.**
2. **Select Verbose Log Messages.**
3. **Select Restart Server if it is not already selected.**
4. **Click OK.**

   Verbose mode can reduce daemon efficiency because of the time taken to display messages.

▼ How to Generate Verbose DHCP Log Messages (Command Line)

1. **Become superuser on the DHCP server system.**
2. **Type the following commands to stop the DHCP daemon and restart it in verbose mode**:

```
# /etc/init.d/dhcp stop
# /usr/lib/inet/in.dhcpd -v options
```

   where *options* are any other options you normally use to start the daemon.

   The daemon runs in verbose mode for this session only.

   Verbose mode can reduce daemon efficiency because of the time taken to display messages.

▼ How to Enable and Disable DHCP Transaction Logging (DHCP Manager)

This procedure enables/disables transaction logging for all subsequent DHCP server sessions.

1. **Choose Modify from the Service menu.**

2. **Select Log Transactions to Syslog Facility.**

   To disable transaction logging, deselect this option.

3. **Select a local facility from 0 to 7 to use for logging transactions.**

   By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `Syslog` File" on page 215.

   Message files can quickly become very large when transaction logging is enabled.

4. **Select Restart Server if it is not already selected.**

5. **Click OK.**

▼ How to Enable and Disable DHCP Transaction Logging for Current Session (Command Line)

1. **Become superuser on the DHCP server system.**

2. **Type the following commands**:

```
# /etc/init.d/dhcp stop
# /usr/lib/inet/in.dhcpd -l syslog-local-facility
```

where *syslog-local-facility* is a number from 0 through 7. If you omit this option, 0 is used by default. See "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 213.

Note - To disable transaction logging, omit the `-l` option when starting `in.dhcpd`.

By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `Syslog` File" on page 215.

Message files can quickly become very large when transaction logging is enabled.

▼ How to Enable and Disable DHCP Transaction Logging for All Sessions (Command Line)

1. **Become superuser on the DHCP server system.**

2. **Type the following to start** `dhcpconfig`:

```
# /usr/sbin/dhcpconfig
```

3. **Select Configure DHCP Service.**

4. **Press Return for the following prompts to accept the default values, which may differ from the values shown here:**

```
Would you like to stop the DHCP service? (recommended) ([Y]/N):
###     DHCP Service Configuration      ###
###     Configure DHCP Database Type and  Location      ###
Enter datastore (files or nisplus) [nisplus]:
Enter absolute path to datastore directory [dhcp.test.]:
Warning: Setting NIS_GROUP to admin.dhcp.test.
```

5. **Type Y for the following prompts as shown:**

```
###     Common daemon option setup      ###
Would you like to specify nondefault daemon options (Y/[N]):Y
```

(continued)

```
Do you want to enable transaction logging? (Y/[N]):Y
```

6. **Type a number from 0 through 7 at the following prompt:**

```
Which syslog local facility [0-7] do you wish to log to? [0]:
```

By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `Syslog` File" on page 215.

Message files can quickly become very large when transaction logging is enabled.

▼ How to Log DHCP Transactions to a Separate `Syslog` File

1. **Become superuser on the DHCP server system.**

2. **Edit the `/etc/syslog.conf` file on the server system and add a line having the following format:**

```
local<n>.notice                    path-to-logfile
```

where *n* is the `syslog` facility number you specified for transaction logging, and *path-to-logfile* is the complete path to the file to use for logging transactions.

For example, you might add the following line:

```
local0.notice              /var/log/dhcpsrvc
```

See `syslog.conf`(4) man page for more information about the `syslog.conf` file.

Customizing DHCP Service Performance Options

You can change options which affect the performance of the DHCP service. These options are described in the following table.

TABLE 11–2 Options Affecting DHCP Server Performance

Server Option	Description
Number of BOOTP relay agent hops	If a request has traveled through more than a given number of BOOTP relay agents, it is dropped. The default maximum number of relay agent hops is 4, and it is not likely that this number will be surpassed unless your network is set up to pass requests through several BOOTP relay agents before reaching a DHCP server.
Verification of IP address availability before making an offer	By default, the server pings an IP address before offering it to a requesting client to verify that it is not already in use. You can disable this feature to decrease the time it takes to make an offer, while running the risk of having duplicate IP addresses in use.
Automatic reading of `dhcptab` at specified intervals	The server can be set to automatically read the `dhcptab` at the interval you specify. If your network configuration information does not change frequently, and you do not have multiple DHCP servers, it is not necessary to reload `dhcptab` automatically. Also note that DHCP Manager gives you the option to have the server reload `dhcptab` after making a change to the data.
Length of time to reserve an IP address that has been offered	After a server offers an IP address to a client, it caches the offer, during which time the server does not offer the address again. You can change the period of time for which the offer is cached. The default is 10 seconds. On slow networks, you made need to increase the offer time.

The following procedures describe how to change these options.

▼ How to Customize DHCP Server Performance Options (DHCP Manager)

1. **Choose Modify from the Service menu.**
2. **To change the number of BOOTP relay agents a request can pass through, type the number of Maximum Number of Relay Agent Hops.**
3. **To have the DHCP server verify that an IP address is not in use before offering it to a client, select Detect Duplicate IP Addresses.**
4. **To have the DHCP server read `dhcptab` at specified intervals, select Reload dhcptab Every *n* Minutes, and type the number of minutes for the interval.**

5. **To change the length of time the server holds an IP address open after making an offer, type the number of seconds in the field Cache Offers for *n* Seconds.**

6. **Select Restart Server if it is not already selected.**

7. **Click OK.**

▼ How to Customize DHCP Server Performance Options (Command Line)

Changing options using this procedure affect only the current server session. If the DHCP server system is rebooted, the DHCP server starts using the settings specified during server configuration. If you want to change settings to apply to all future sessions, you must run `dhcpconfig` and answer the prompts as described in "How to Configure a DHCP Server (`dhcpconfig`) " on page 189.

1. **Become superuser on the DHCP server system.**

2. **Type the following command:**

```
# /etc/init.d/dhcp stop
# /usr/lib/inet/in.dhcpd options
```

where options are any of the following:

–h ***relay-hops***	Specifies the maximum number of relay agent hops that can occur before the daemon drops the DHCP/BOOTP datagram.
–n	Disables automatic duplicate IP address detection. This is not recommended.
–t ***dhcptab_rescan_interval***	Specifies the interval in minutes that the DHCP server should use to schedule the automatic rereading of the `dhcptab` information.
–o ***seconds***	Specifies the number of seconds the DHCP server should cache the offers it has extended to

discovering DHCP clients. The default setting is 10 seconds.

For example, the following command sets the hop count to 2, disables duplicate IP address detection, sets the rescan interval to 30 minutes, and sets the offer time to 20 seconds.

```
# /usr/lib/inet/in.dhcp -h 2 -n -t 30 -o 20
```

Adding, Modifying, and Removing DHCP Networks

When you configure a DHCP server, you must also configure at least one network in order to use the DHCP service. You can add more networks at any time.

This section describes:

The following task map lists tasks you need to perform when working with DHCP networks and the procedures used to carry them out.

TABLE 11–3 Working with DHCP Networks Task Map

Tasks...	Description	For Instructions, Go To
Enable or disable DHCP service on server network interfaces	The default behavior is to monitor all network interfaces for DHCP requests, but you can change this.	"How to Set Up DHCP to Ignore a Network Interface" on page 220 "How to Set Up DHCP to Monitor a Network Interface" on page 221
Add a new network to the DHCP service	Place a network under DHCP management, for the purpose of managing IP addresses on the network.	"How to Add a DHCP Network (DHCP Manager)" on page 222
Change parameters of a DHCP-managed network	Modify the information that is passed to clients of a particular network.	"How to Modify Configuration of a DHCP Network (DHCP Manager)" on page 224 "How to Modify a DHCP Network (Command Line)" on page 224
Delete a network from the DHCP service	Remove a network so that IP addresses on the network are no longer managed by DHCP	"How to Remove a DHCP Network (DHCP Manager)" on page 226 "How to Remove a DHCP Network (Command Line)" on page 226

Monitoring and Ignoring Network Interfaces for DHCP Service

By default both `dhcpconfig` and DHCP Manager's Configuration Wizard configure the DHCP server to monitor all the server system's network interfaces. If you add a new network interface to the server system, the DHCP server automatically monitors the new interface when you boot the system. You can then add any networks that will be monitored through the network interface.

However, DHCP Manager also allows you to specify which network interfaces the DHCP service should monitor and which it should ignore. You might want to ignore an interface if you do not want to offer DHCP service on that network.

If you specify that any interface should be ignored, and then install a new interface, the DHCP server ignores the new interface unless you add it to the list of monitored interfaces in DHCP Manager.

The `dhcpconfig` utility does not allow you to ignore a network interface.

This section includes procedures for ignoring a network interface and monitoring a new network interface using DHCP Manager's Modify Service Options dialog box, which is shown in the following figure.

Figure 11–4 Interfaces Tab of Modify Services Options Dialog Box

▼ How to Set Up DHCP to Ignore a Network Interface

1. **Choose Modify from the Service menu.**
   The Modify Service Options dialog box is displayed.

2. **Select the Interfaces tab.**

3. **In the Monitored Interfaces list, select the network interface that should not receive DHCP service.**

4. **Click the right arrow button to move the network interface listing to the Ignored Interfaces list.**

5. **Click OK.**

▼ How to Set Up DHCP to Monitor a Network Interface

1. **Choose Modify from the Service menu.**
   The Modify Service Options dialog box is displayed.

2. **Select the Interfaces tab.**

3. **In the Ignored Interfaces list, select the network interface that should receive DHCP service.**

4. **Click the left arrow button to move the network interface listing to the Monitored Interfaces list.**

5. **Click OK.**

Adding DHCP Networks

The first network, usually the local one on the primary interface, is configured when you configure the server with the DHCP Configuration Wizard. This section describes the procedures for placing additional networks under DHCP management using DHCP Manager's Network Wizard.

Note - For information about adding networks using command lines, see "Configuring Networks Using `dhcpconfig`" on page 193.

The following figure shows the initial dialog box for the DHCP Network Wizard.

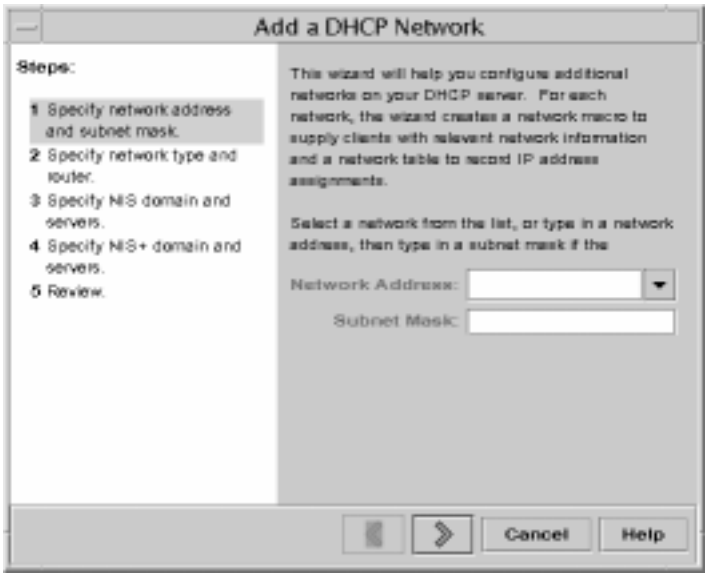


Figure 11–5 DHCP Manager's Network Wizard

When you configure a new network, DHCP Manager creates the following:

- Network table in the data store. The new network is shown in the network list on the Addresses tab of DHCP Manager.
- Network macro containing information needed by clients that reside on this network. The macro's name matches the IP address of the network.

Adding a new network with commands requires you to know the labels of the standard DHCP options, or tokens, used to pass information to the clients. See the `dhcptab`(4) manual page for information about the standard DHCP options.

▼ How to Add a DHCP Network (DHCP Manager)

1. **Click the Addresses tab in DHCP Manager.**

2. **Choose Network Wizard from the Edit menu.**

3. **Select options or type requested information based on the decisions you made during the planning phase.**

   Planning is described in "Planning for Remote Network Configuration" on page 177.

   If you have difficulty with the wizard, click Help in the wizard window to open your web browser and display help for the DHCP Network Wizard.

4. **Click Finish to complete the network configuration when you have finished entering the requested information.**

The Network Wizard creates a network macro whose name matches the IP address of the network. If you click the Macros tab and select the network macro, you can confirm that the information you provided in the wizard has been inserted as values for options contained in the macro.

The Network Wizard creates an empty network table, which is listed in the left pane of the window. You must add addresses for the network before the network's IP addresses can be managed under DHCP.

Modifying DHCP Network Configuration

After you add a network to the DHCP service, you can modify the configuration information you originally supplied only by modifying the network macro used to pass information to the clients on the network.

The following figure shows the Macros tab of the DHCP Manager.

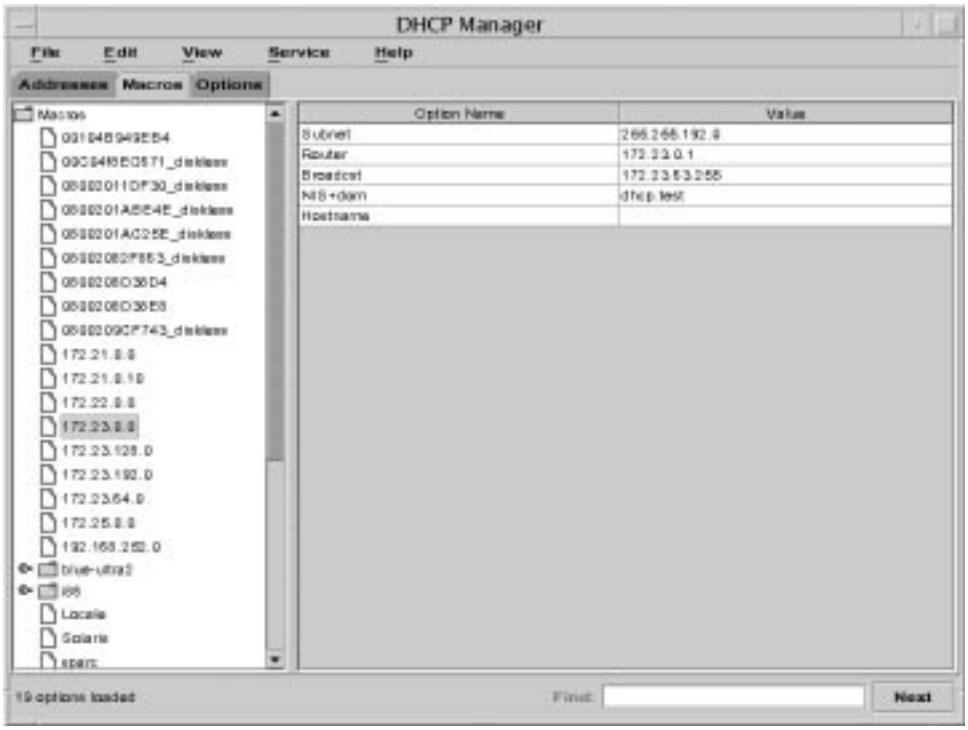


Figure 11–6 DHCP Manager's Macros Tab

▼ How to Modify Configuration of a DHCP Network (DHCP Manager)

1. **Select the Macros tab.**

   All macros defined for this DHCP server are listed in the left pane.

2. **Select the network macro matching the network whose configuration you want to change.**

   The network macro name is the network IP address.

3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box opens, displaying a table of the options included in the macro.

4. **Select the option you want to modify.**

   The option name and value are displayed in text fields near the top of the dialog box.

5. **Type the new value for the option and click Modify.**

   You can also add options here by clicking Select in the dialog box. See “Modifying DHCP Macros” on page 250 for more general information about modifying macros.

6. **Select Notify DHCP Server of Change and click OK.**

   The change is made to the `dhcptab` and the DHCP server is signaled to reread the`dhcptab` and put the changes into effect.

▼ How to Modify a DHCP Network (Command Line)

1. **Determine which macro includes information for all clients of the network.**

   The network macro’s name should match the network IP address.

   If you don’t know which macro includes this information, you can display the `dhcptab` database to list all macros, using the command `dhtadm -P`.

2. **Type a command of the following format to change the value of the option you want to change**:

```
# dhtadm -M -m macro-name -e 'symbol=value'
```

For example, to change the `172.25.62.0` macro's lease time to 57600 seconds and NIS domain to `sem.west.com`, type the following:

```
# dhtadm -M -m 172.25.62.0 -e 'LeaseTim=57600'
# dhtadm -M -m 172.25.62.0 -e 'NISdmain=sem.west.com'
```

3. **Type the following command as root to make the DHCP daemon reread** `dhcptab`:

```
# pkill -HUP in.dhcpd
```

Removing DHCP Networks

DHCP Manager enables you to remove multiple networks at once. You have the option to automatically remove the hosts table entries associated with the DHCP-managed IP addresses on those networks as well. The following figure shows DHCP Manager's Delete Networks dialog box.

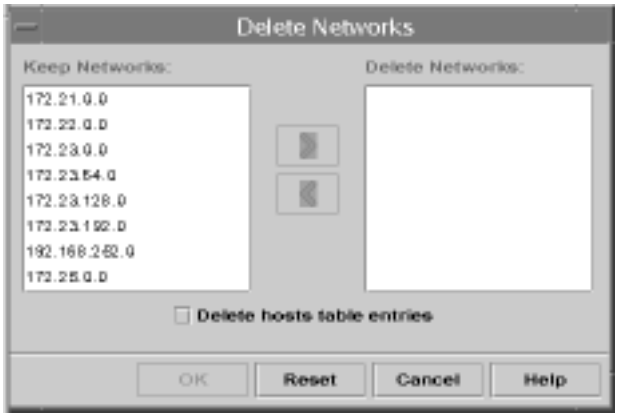


Figure 11–7 Delete Networks Dialog Box

The `pntadm` command requires you to delete each IP address entry from a network before deleting that network. You can delete only one network at a time.

▼ How to Remove a DHCP Network (DHCP Manager)

1. **Select the Addresses tab.**

2. **Choose Delete Networks from the Edit menu.**
   The Delete Networks dialog box opens.

3. **In the Keep Networks list, select the networks you want to delete.**
   Press the Control key while clicking with the mouse to select multiple networks, or press the Shift key while clicking to select a range of networks.

4. **Click the right arrow button to move the selected networks to the Delete Networks list.**

5. **If you want to remove the host table entries for the DHCP–managed addresses on this network, select Delete Host Table Entries.**

6. **Click OK.**

▼ How to Remove a DHCP Network (Command Line)

Note that this procedure deletes the addresses on the network before removing the network. This ensures that the hostnames are removed from the `hosts` file.

1. **Become superuser on the server system.**

2. **Type a command following this format to remove an IP address and its hostname from the name service:**

```
# pntadm -D -yIP-address
```

   For example, to remove address 172.25.52.1, type the following:

```
# pntadm -D -y 172.25.52.1
```

   The `-y` option specifies to delete the hostname.

3. **Repeat the `pntadm -D -y` command for each address in the network.**
   You might want to create a script to do this is you are deleting many addresses.

4. **After all addresses are deleted, type the following to delete the network from the DHCP service.**

```
# pntadm -R network-IP-address
```

For example, to remove network 172.25.52.0, type the following:

```
# pntadm -R 172.25.52.0
```

Supporting BOOTP Clients with DHCP Service

To support BOOTP clients on your DHCP server, you must set up your DHCP server to be BOOTP compatible. You can register BOOTP clients in the DHCP server's database or reserve a number of IP addresses for allocation to BOOTP clients, depending how you set up BOOTP compatibility.

You can set up support for BOOTP clients in one of the following ways:

- **Automatic BOOTP support** – Any BOOTP client on a DHCP-managed network, or on a network connected by a BOOTP relay agent to a DHCP-managed network can obtain an IP address from the server. This requires you to reserve a pool of addresses for exclusive use by BOOTP clients. This option may be more useful if the server must support a large number of BOOTP clients.
- **Manual BOOTP support** – Only those BOOTP clients that have been manually registered with the DHCP service will receive a response from the server. This requires you to bind a client's ID to a particular IP address that has been marked for BOOTP clients. This option is useful for a small number of BOOTP clients, or in the event that you want to restrict the BOOTP clients that can use the server.

Note - BOOTP addresses are permanently assigned, whether or not you explicitly assign them a permanent lease.

The following task map lists tasks you need to perform to support BOOTP clients and the procedures used to carry them out.

TABLE 11–4 BOOTP Support Task Map

Tasks....	Description	For Instructions, Go To
Set up automatic BOOTP support	Provide IP address for any BOOTP client on a DHCP-managed network, or on a network connected by a relay agent to a DHCP-managed network.	"How to Set Up Support of Any BOOTP Client (DHCP Manager)" on page 228 "How to Set Up Support for Any BOOTP Client (Command Line)" on page 230
Set up manual BOOTP support	Provide IP address for only those BOOTP clients that have been manually registered with the DHCP service.	"How to Set Up Support of Registered BOOTP Clients (DHCP Manager)" on page 229 "How to Set Up Support for Registered BOOTP Clients (Command Line)" on page 232

▼ How to Set Up Support of Any BOOTP Client (DHCP Manager)

1. **Select Modify from the Service menu.**
   The Modify Service Options dialog box opens.
2. **In the BOOTP Compatibility section of the dialog box, select Automatic.**
3. **Select Restart Server, if it is not already selected.**
4. **Click OK.**
5. **Select the Addresses tab in DHCP Manager.**
6. **Select addresses that you want to reserve for BOOTP clients.**
   Select a range of addresses by clicking the first address, pressing the Shift key, and clicking the last address.
   Select multiple non-concurrent addresses by pressing the Control key while clicking each address.
7. **Select Properties from the Edit menu.**

The Modify Multiple Addresses dialog box opens.

8. **In the BOOTP section, select Assign All Address Only to BOOTP Clients.**
   All other options should be set to Keep Current Settings.

9. **Click OK.**
   Any BOOTP client can now obtain an address from this DHCP server.

▼ How to Set Up Support of Registered BOOTP Clients (DHCP Manager)

1. **Select Modify from the Service menu.**
   The Modify Service Options dialog box opens.

2. **In the BOOTP Compatibility section of the dialog box, select Manual.**

3. **Select Restart Server if it is not already selected.**

4. **Click OK.**

5. **Select the Addresses tab in DHCP Manager.**

6. **Select an address you want to assign to a particular BOOTP client.**

7. **Choose Properties from the Edit menu.**
   The Address Properties dialog box opens.

8. **Select the Lease tab.**

9. **In the Client ID field, type the client's identifier.**
   For a BOOTP client running the Solaris operating environment on an Ethernet network, the client ID is a string derived from the client's hexadecimal Ethernet address, preceded by the ARP type for Ethernet (01). For example, a BOOTP client having the Ethernet address 8:0:20:94:12:1e would use the Client ID 0108002094121E. See Table 13–4 for the ARP types.

Tip - As superuser on the client machine, type the following to obtain the Ethernet address for the interface: `ifconfig -a`

---

10. **Select Reserved to reserve the IP address for this client.**

11. Select Assign Only to BOOTP Clients.

12. Click OK.

In the Addresses tab, BOOTP is displayed in the Status field, and the client ID you entered is listed in the Client ID field.

▼ How to Set Up Support for Any BOOTP Client (Command Line)

1. Log in to the system as root or become superuser, and type the following:

```
# /usr/sbin/dhcpconfig
```

The text-based DHCP Configuration menu is displayed.

2. Type `1` and press Return to select Configure DHCP Service.

3. Answer the following prompts as shown to skip to the BOOTP compatibility options.

If no user input is indicated, press Return to accept the default.

```
Would you like to stop the DHCP service? (recommended) ([Y]/N)
Enter datastore (files or nisplus) [files]:
Enter absolute path to datastore directory [/var/dhcp]:
Would you like to specify nondefault daemon options (Y/[N]):
Would you like to specify nondefault server options (Y/[N]):Y
How long (in seconds) should the DHCP server keep outstanding OFFERs? [10]:
How often (in minutes) should the DHCP server rescan the dhcptab? [Never]:
```

4. Answer the prompts as follows to enable BOOTP compatibility:

```
Do you want to enable BOOTP compatibility mode? (Y/[N]):Y
Do you want the server to allocate IP addresses to new BOOTP clients? ([Y]/N)
```

5. Answer the following prompts as shown to advance to prompts for creating addresses:

```
###     Initialize dhcptab table        ###
The dhcptab table already exists.
Do you want to merge initialization data with the existing table? (Y/[N]):
###     Select Networks For BOOTP/DHCP Support  ###
Enable DHCP/BOOTP support of networks you select? ([Y]/N):
```

If you want to create BOOTP addresses on the local network, continue with the next step.

If you want to create BOOTP addresses on a remote network, skip to Step 7 on page 231.

6. Answer the following prompts as shown to create the BOOTP addresses on the local network:

Note this is an example of adding four addresses for network 172.21.0.0. You should substitute appropriate responses for your network.

```
###     Configure Local Networks        ###
Configure BOOTP/DHCP on local LAN network: 172.21.0.0? ([Y]/N):
Do you want hostnames generated and inserted in the nisplus hosts table? (Y/[N]):
Enter starting IP address [172.21.0.0]: 172.21.0.15
Enter the number of clients you want to add (x < 65535): 4
BOOTP compatibility with automatic allocation is enabled.
Do you want any of your 4 addresses to be BOOTP specific? ([Y]/N):
How many (x <= 4): 4
The dhcp network table: 172.21.0.0 already exists.
Do you want to add entries to it? ([Y]/N):
dhcptab macro "172.21.0.0" already exists.
Do you want to merge initialization data with the existing macro? ([Y]/N):N
Disable (ping) verification of 172.21.0.0 address(es)? (Y/[N]):
/ 75% Complete.
Configured 4 entries for network 172.21.0.0.
```

7. If you want to create BOOTP addresses on a remote network, answer the prompts as follows:

Note this is an example of adding four addresses for network 172.23.0.0, where clients access the network via LAN connection. You should substitute appropriate responses for your network.

```
###     Configure Remote Networks       ###
Would you like to configure BOOTP/DHCP service on remote networks? ([Y]/N):
Enter Network Address of remote network, or <RETURN> if finished: 172.23.0.0
Do clients access this remote network via LAN or PPP connection? ([L]/P):
Do you want hostnames generated and inserted in the nisplus hosts table? (Y/[N]):
Enter Router (From client's perspective), or <RETURN> if finished.
IP address:
Optional: Enter Remote Network's MTU (e.g. ethernet == 1500):
Enter starting IP address [172.23.0.0]: 172.23.0.10
Enter the number of clients you want to add (x < 65535): 4
BOOTP compatibility with automatic allocation is enabled.
Do you want any of your 4 addresses to be BOOTP specific? ([Y]/N):
How many (x <= 4): 4
The dhcp network table: 172.23.0.0 already exists.
Do you want to add entries to it? ([Y]/N):
dhcptab macro "172.23.0.0" already exists.
Do you want to merge initialization data with the existing macro? ([Y]/N):N
Disable (ping) verification of 172.23.0.0 address(es)? (Y/[N]):
/ 75% Complete.
Configured 4 entries for network 172.23.0.0.
Enter Network Address of remote network, or <RETURN> if finished:
```

8. **Type 4 and press Return to exit** `dhcpconfig`**.**

▼ How to Set Up Support for Registered BOOTP Clients (Command Line)

1. **Log in to the system as root or become superuser, and type the following:**

```
# /usr/sbin/dhcpconfig
```

The text-based DHCP Configuration menu is displayed.

2. **Type 1 and press Return to select Configure DHCP Service.**

3. **Answer the following prompts as shown to skip to the BOOTP compatibility options.**

If no user input is indicated, press Return to accept the default.

```
Would you like to stop the DHCP service? (recommended) ([Y]/N)Y
Enter datastore (files or nisplus) [files]:
Enter absolute path to datastore directory [/var/dhcp]:
Would you like to specify nondefault daemon options (Y/[N]):
Would you like to specify nondefault server options (Y/[N]):Y
How long (in seconds) should the DHCP server keep outstanding OFFERs? [10]:
```

(Continuation)

```
How often (in minutes) should the DHCP server rescan the dhcptab? [Never]:
```

4. **Answer the BOOTP prompts as follows:**

```
Do you want to enable BOOTP compatibility mode? (Y/[N]):Y
Do you want the server to allocate IP addresses to new BOOTP clients? ([Y]/N):N
```

Type **N** to prevent unregistered BOOTP clients from obtaining IP addresses. This is analogous to DHCP Manager's "manual" option.

```
The dhcptab table already exists.
Do you want to merge initialization data with the existing table? (Y/[N]):N
Enable DHCP/BOOTP support of networks you select? ([Y]/N):N
```

Typing **N** here avoids prompts for adding networks to the DHCP service.

```
Would you like to restart the DHCP service? (recommended) ([Y]/N):Y
```

5. **Type 4 and press Return to exit** `dhcpconfig`**.**

6. **Modify or add addresses and reserve them for use by specific BOOTP clients by typing a command using one of the following formats:**

   a. **To modify an existing address for BOOTP:**

```
# pntadm -M ip-address -i client-id -f BOOTP -e -1 -m macro-name network-ip-address
```

For example, to modify the address 172.21.20.33, assign it to a client whose Ethernet hardware address is 8:0:20:89:a1:d2 and set the BOOTP flag, type the following:

```
# pntadm -M 172.21.20.33 -i 0108002089A1D2 -f BOOTP
```

b. To add a new BOOTP address:

```
# pntadm -A ip-address -i client-id -f BOOTP -m macro-name network-ip-address
```

For example, to add the address 172.21.20.34, assign it to a client whose Ethernet hardware address is 8:0:20:89:a1:d2, set the BOOTP flag, and to have the client receive the contents of the `blue2` macro, type the following:

```
pntadm -A 172.21.20.34 -i 0108002089A1D2 -f BOOTP -m blue2 172.21.0.0
```

You should reserve one BOOTP address for each BOOTP client on the network.

Working With IP Addresses in the DHCP Service

You can use DHCP Manager or the `pntadm` command to add IP addresses, modify their properties, and remove them from the DHCP service. Before working with IP addresses, you should refer to Table 11–6 to become familiar with IP address properties. The table provides information for users of DHCP Manager and `pntadm`.

Note - This section does not include procedures for using the `pntadm` command. However Table 11–6 includes examples of using `pntadm` to specify IP address properties while adding and modifying IP addresses. Please also refer to the `pntadm`(1M) man page for more information about `pntadm`.

The following task map lists tasks you must perform to add, modify, remove IP addresses and the procedures used to carry them out.

TABLE 11–5 IP Addresses in DHCP Task Map

Tasks	Description	For Instructions, Go To...
Add single or multiple IP addresses to DHCP service.	Add IP addresses on networks that are already managed by the DHCP service using DHCP Manager.	"How to Create a Single IP Address (DHCP Manager)" on page 240 "How to Duplicate an Existing IP Address (DHCP Manager)" on page 240 "How to Create Multiple Addresses (DHCP Manager)" on page 241
Change properties of an IP address.	Change any of the IP address properties described in Table 11–6.	"How to Enable the DHCP Service (Command Line)" on page 207
Remove IP addresses from DHCP service.	Prevent the use of specified IP addresses by DHCP.	"How to Mark Addresses Unusable (DHCP Manager)" on page 244 "How to Delete IP Addresses from DHCP Service (DHCP Manager)" on page 245
Assign consistent address to a DHCP client.	Set up a client to receive the same IP address each time it requests its configuration.	"How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager)" on page 246

The following table lists and describes the properties of IP addresses.

TABLE 11–6 IP Address Properties

Property	Description	How to Specify in `pntadm` Command
Network address	Address of the network that contains the IP address you are working with. The network address is displayed in the Networks list on the Addresses tab in DHCP Manager.	The network address must be the last argument on the `pntadm` command line used to create, modify, or delete an IP address. For example, to add an IP address to network 172.21.0.0 **`pntadm -A` *`ip-address options`* `172.21.0.0`**
IP address	Address you are working with, whether you are creating, modifying, or deleting it. The IP address is displayed in the first column of the DHCP Manager's Addresses tab.	The IP address must accompany the –`A`, –`M`, and –`D` options to the `pntadm` command. For example, to modify IP address 172.21.5.12 **`pntadm -M 172.21.5.12` *`options`* `172.21.0.0`**
Client name	Host name mapped to the IP address in the hosts table. This name may be automatically generated by DHCP Manager or `dhcpconfig` when addresses are created. If you create a single address, you can supply the name.	Specify the client name with the –`h` option. For example, to specify client name carrot12 for 172.21.5.12: **`pntadm -M 172.21.5.12 -h carrot12 172.21.0.0`**
Owning server	DHCP server that manages the IP address and is responsible for responding to the DHCP client's request for IP address allocation.	Specify the owning server name with the –`s` option. For example to specify server blue2 to own 172.21.5.12: **`pntadm -M 172.21.5.12 -s blue2 172.21.0.0`**

TABLE 11–6 IP Address Properties *(continued)*

Property	Description	How to Specify in `pntadm` Command
Configuration macro	Macro the DHCP server uses to obtain network configuration options from the `dhcptab` database. Several macros are created automatically when you configure a server and add networks. See “About Macros” on page 160 for more information about macros. The server macro is selected by default when DHCP Manager or `dhcpconfig` create addresses.	Specify the macro name with the `–m` option. For example, to assign the server macro blue2 to address 172.21.5.12 **`pntadm -M 172.21.5.12 -m blue2 172.21.0.0`**
Client ID	Text string derived from the client’s hexadecimal hardware address, preceded by the ARP code for the type of network, such as 01 for Ethernet. See Table 13–4 for a full list of ARP hardware codes. For example, a client having the hexadecimal Ethernet address 8:0:20:94:12:1e would use the Client ID 0108002094121E. The client ID is listed in DHCP Manager and `pntadm` when a client is currently using an address. If you specify a client ID when modifying the properties of an IP address, you manually bind the address to that client for its exclusive use. **Tip:** As superuser on the client machine, type the following to obtain the Ethernet address for the interface: `ifconfig -a`	Specify the client ID with the `–i` option. For example, to assign client ID 08002094121E to address 172.21.5.12 **`pntadm -M 172.21.5.12 -i 0108002094121E 172.21.0.0`**
Reserved	Setting that specifies the address is reserved exclusively for the client indicated by the client ID, and the DHCP server cannot reclaim the address. If you choose this option, you manually assign the address to the client.	Specify that the address is reserved, or manual, with the `–f` option. For example, to specify that IP address 172.21.5.12 is reserved for a client: **`pntadm -M 172.21.5.12 -f MANUAL 172.21.0.0`**

TABLE 11–6 IP Address Properties *(continued)*

Property	Description	How to Specify in `pntadm` Command
Lease type	Setting that determines how DHCP manages the use of the IP address by clients. A lease may be dynamic or permanent. See “Dynamic and Permanent Lease Type” on page 176 for a complete explanation.	Specify that the address would be permanently assigned with the `–f` option. Addresses are dynamically leased by default. For example, to specify that IP address 172.21.5.12 has a permanent lease **`pntadm -M 172.21.5.12 -f PERMANENT 172.21.0.0`**
Lease expiration time	Date and time when the lease expires, applicable only when a dynamic lease is specified. The date is specified in *mm/dd/yyyy* format, and is calculated by the DHCP server.	Specify an absolute lease expiration time with `–e`. For example, to specify an expiration time of Jaunary 1, 2000: **`pntadm -M 172.21.5.12 -e 01/01/2000 172.21.0.0`**
BOOTP setting	Setting that marks the address as reserved for BOOTP clients. See “Supporting BOOTP Clients with DHCP Service” on page 227 for more information about supporting BOOTP clients.	Reserve an address for BOOTP clients with `–f`. For example, to reserve IP address 172.21.5.12 for BOOTP clients: **`pntadm -M 172.21.5.12 -f BOOTP 172.21.0.0`**
Unusable setting	Setting that marks the address so it cannot be assigned to any client.	Mark an address unusable with `–f`. For example, to mark IP address 172.21.5.12 unusable: **`pntadm -M 172.21.5.12 -f UNUSABLE 172.21.0.0`**

Adding Addresses to the DHCP Service

Before you add addresses, you must add the network that owns them to the DHCP service. See “Adding DHCP Networks” on page 221 for information about adding networks.

You can add addresses using DHCP Manager or `dhcpconfig`. If you want to use commands to add addresses, use `dhcpconfig` as explained in “Configuring Networks Using `dhcpconfig`” on page 193.

You can add addresses on networks that are already managed by the DHCP service in several ways using DHCP Manager:

- **Create a single IP address** – Place one new IP address under DHCP management.
- **Duplicate an existing IP address** – Copy the properties of an existing IP address managed by DHCP, and supply a new IP address and client name.
- **Create a range of multiple IP addresses** – Use the Address Wizard to place a series of IP addresses under DHCP management.

The following figure shows the Create Address dialog box. The Duplicate Address dialog box is identical to the Create Address dialog box, except that the text fields display the values for an existing address.

Figure 11–8 Create Address Dialog Box

The following figure shows the first dialog of the Address Wizard, used to create a range of IP addresses.

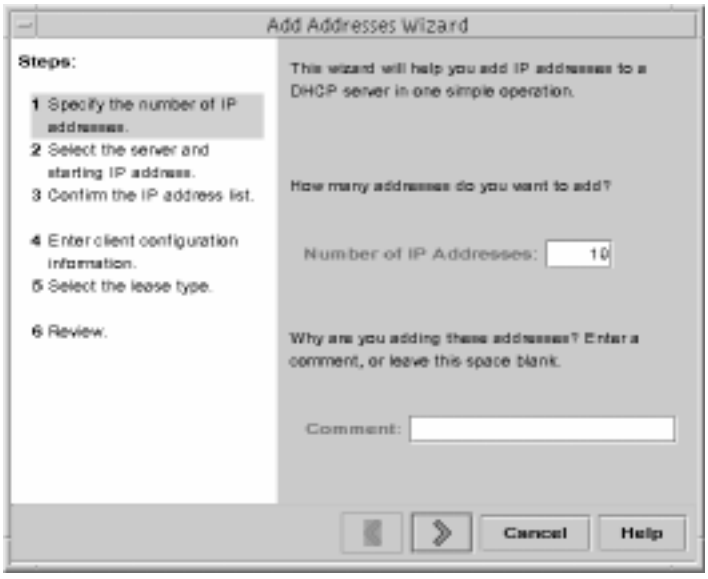


Figure 11–9 Address Wizard

▼ How to Create a Single IP Address (DHCP Manager)

1. **Select the Addresses tab.**
2. **Select the network where the new IP address is to be added.**
3. **Choose Create from the Edit menu.**
   The Create Address dialog box opens.
4. **Select or type values for the address settings on the Address and Lease tabs.**
   See Table 11–6 for information about the settings.
5. **Click OK.**

▼ How to Duplicate an Existing IP Address (DHCP Manager)

1. **Select the Addresses tab.**
2. **Select the network where the new IP address is located.**

3. **Select the address whose properties you want to duplicate.**

4. **Choose Duplicate from the Edit menu.**

5. **Change the IP address and client name for the address.**
   Most other options should remain the same, but you can change them if necessary.

6. **Click OK.**

▼ How to Create Multiple Addresses (DHCP Manager)

1. **Select the Addresses tab.**

2. **Select the network where the new IP addresses are to be added.**

3. **Choose Address Wizard from the Edit menu.**
   The Address Wizard starts, prompting you to provide values for the IP address properties. See Table 11–6 for more information about the properties. “Making Decisions for IP Address Management” on page 173 includes more extensive information.

4. **Click the right arrow button as you finish entering information in each screen, and click Finish on the last screen.**
   The Addresses tab is updated with the new addresses.

Modifying IP Addresses in the DHCP Service

After adding IP addresses to the DHCP service, you can modify any of the properties described in Table 11–6 using DHCP Manager or the `pntadm -M` command. See the `pntadm`(1M) man page for more information about using `pntadm -M`.

The following figure shows the Address Properties dialog box that you use to modify IP address properties.

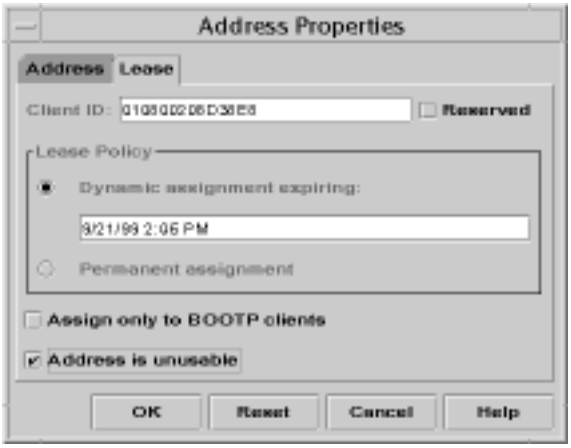


Figure 11–10 Address Properties Dialog Box

The following figure shows the Modify Multiple Addresses dialog box that you use to modify multiple IP addresses.

Figure 11–11 Modify Multiple Addresses Dialog Box

▼ How to Modify IP Address Properties (DHCP Manager)

1. **Select the Addresses tab.**

2. **Select the IP address's network.**

3. **Select one or more IP addresses you want to modify.**

   If you want to modify more than one address, press the Control key while clicking with the mouse to select multiple addresses. You can also press the Shift key while clicking to select a block of addresses.

4. **Choose Properties from the Edit menu.**

   The Modify Addresses dialog box or the Modify Multiple Address dialog box opens.

5. **Change the appropriate properties.**

   Refer to Table 11–6 for information about the properties.

6. **Click OK.**

Removing Addresses From DHCP Service

At times you might want the DHCP service to stop managing a particular address or group of addresses. The method you use to remove an address from DHCP depends on whether you want the change to be temporary or permanent.

- To temporarily prevent the use of addresses, you can mark them unusable in the Address Properties dialog box as described in "Marking IP Addresses Unusable by the DHCP Service" on page 243.
- To permanently prevent the use of addresses by DHCP clients, delete the addresses from the DHCP network tables, as described in "Deleting IP Addresses from DHCP Service" on page 244.

Marking IP Addresses Unusable by the DHCP Service

You can use the `pntadm -M` command with the `-f UNUSABLE` option to mark addresses unusable using the command line.

In DHCP Manager, you use the Address Properties dialog box, shown in Figure 11–10, to mark individual addresses, and the Modify Multiple Addresses dialog box, show in Figure 11–11, to mark multiple addresses, as described in the following procedure.

Deleting IP Addresses from DHCP Service

You should delete IP addresses from the DHCP service database if you no longer want the address to be managed by DHCP. You can use the `pntadm -D` command or DHCP Manager's Delete Address dialog box.

The following figure shows the Delete Address dialog box.

Figure 11–12 Delete Address Dialog Box

▼ How to Mark Addresses Unusable (DHCP Manager)

1. **Select the Addresses tab.**

2. **Select the IP address's network.**

3. **Select one or more IP addresses you want to mark unusable.**

   If you want to mark more than one address unusable, press the Control key while clicking with the mouse to select multiple addresses. You can also press the Shift key while clicking to select a block of addresses.

4. **Choose Properties from the Edit menu.**

   The Modify Addresses dialog box or the Modify Multiple Address dialog box opens.

5. **If you are modifying one address, select the Lease tab.**

6. **Select Address is Unusable.**

   If you are editing multiple addresses, select Mark All Addresses Unusable.

7. **Click OK.**

▼ How to Delete IP Addresses from DHCP Service (DHCP Manager)

1. **Select the Addresses tab.**

2. **Select the IP address's network.**

3. **Select one or more IP addresses you want to delete.**

   If you want to delete more than one address, press the Control key while clicking with the mouse to select multiple addresses. You can also press the Shift key while clicking to select a block of addresses.

4. **Choose Delete from the Edit menu.**

   The Delete Address dialog box opens listing the address you selected so you can confirm the deletion.

5. **If you want to delete the host names from the hosts table, select Delete From Hosts Table.**

   If the host names were generated by DHCP Manager or `dhcpconfig`, you might want to delete the names from the hosts table.

6. **Click OK.**

Setting Up DHCP Clients for a Consistent IP Address

The Solaris DHCP service attempts to provide the same IP address to a client that has previously obtained an address through DHCP. However, it is not always possible when dynamic leasing is in use.

Routers, NIS/NIS+, DNS servers, and other hosts critical to the functioning of a network should not use DHCP because they should not rely on the network to obtain their IP addresses. Clients such as print or file servers should have consistent IP addresses as well, but can be set up to receive their network configurations through DHCP.

You can set up a client to receive the same IP address each time it requests its configuration by reserving, or manually assigning, the client's ID to the address you want it to use. You can set up the reserved address to use a dynamic lease to make it easy to track the use of the address, or a permanent lease if you do not require use tracking. However, permanent leases are not recommended because once a client obtains a permanent lease, it does not contact the server again and cannot obtain updated configuration information unless it releases the IP address and restarts the DHCP lease negotiation.

You can use `pntadm -M` command or DHCP Manager's Address Properties dialog box.

The following figure shows the Lease tab of the Address Properties dialog box used to modify the lease.

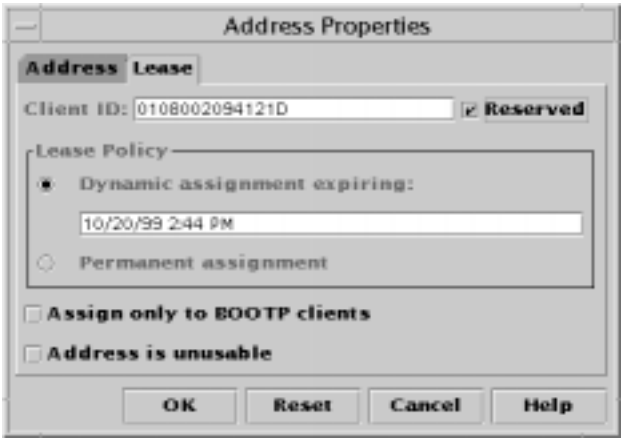


Figure 11–13 Address Properties Lease Tab

▼ How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager)

1. **Determine the client ID for the client you want to have a permanent IP address.**

   See the entry for client ID in Table 11–6 for information about determining the client ID.

2. **Select the Addresses tab in DHCP Manager.**

3. **Select the appropriate network.**

4. **Double-click the IP address you want to the client to use.**

   The Address Properties window opens.

5. **Select the Lease tab.**

6. **In the Client ID field, type the client ID you determined from the client's hardware address.**

7. **Select the Reserved option to prevent the IP address from being reclaimed by the server.**

8. **In the Lease Policy area of the window, select Dynamic or Permanent assignment.**

   Select Dynamic if you want the client to negotiate to renew leases, and thus be able to track when the address is being used. (Because you selected Reserved, the address cannot be reclaimed even when using dynamic leasing.) You do not need to enter an expiration date for this lease; the DHCP server calculates the expiration date based on the lease time.

   Selecting Permanent is not recommended because you cannot track the use of the IP address unless you enable transaction logging.

Working With DHCP Macros

DHCP macros are containers of DHCP options. The Solaris DHCP service uses macros to gather together options that should be passed to clients. DHCP Manager and `dhcpconfig` create a number of macros automatically when you configure the server. See "About Macros" on page 160 for background information about macros, and Chapter 10 for information about macros created by default.

You might find that when changes occur on your network, you need to make changes to the configuration information passed to clients. To do this, you need to work with macros by adding, modifying, duplicating, or deleting them.

Working with macros requires knowledge of DHCP standard options, which are described in the `dhcptab`(4) man page.

The following task map lists tasks for viewing, modifying, adding, and deleting DHCP macros.

TABLE 11–7 DHCP Macros Task Map

Tasks	Description	For Instructions, Go To...
View DHCP macros.	Display a list of all the macros defined on the DHCP server.	"How to View Macros Defined on a DHCP Server (DHCP Manager)" on page 249
Add DHCP macros.	Create new macros to support DHCP clients.	"How to Add a DHCP Macro (DHCP Manager)" on page 253

TABLE 11–7 DHCP Macros Task Map *(continued)*

Tasks	Description	For Instructions, Go To...
Modify values passed in macros to DHCP clients.	Change macros by modifying existing options, adding options to macros, removing options from macros.	"How to Change Values for Options in a DHCP Macro (DHCP Manager)" on page 250 "How to Add Options to a DHCP Macro (DHCP Manager)" on page 251 "How to Delete Options from a DHCP Macro (DHCP Manager)" on page 252
Delete DHCP macros.	Remove DHCP macros that are no longer used.	"How to Delete a DHCP Macro (DHCP Manager)" on page 254

The following figure shows the Macros tab in the DHCP Manager window.

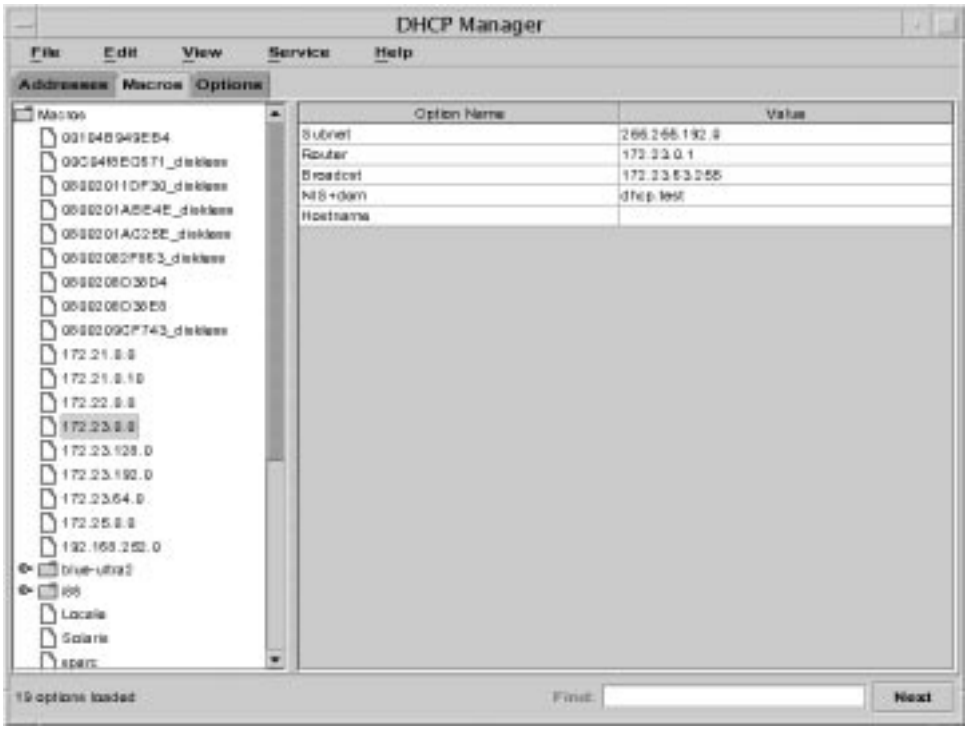


Figure 11–14 DHCP Manager's Macros Tab

▼ How to View Macros Defined on a DHCP Server (DHCP Manager)

You can use DHCP Manager or `dhtadm -P` to display all the macros defined on a DHCP server.

1. **Select the Macros tab.**

   The Macros area on the left side of the window displays, in alphabetical order, all macros defined on the server. Macros preceded by a folder icon include references to other macros, while macros preceded by a document icon do not reference other macros.

2. **To open a macro folder, click the open/close widget to the left of the folder icon.**

   The macros included in the selected macro are listed.

3. **To view the contents of a macro, click the macro name and look at the area on the right side of the window.**

   Options and their assigned values are displayed.

Modifying DHCP Macros

You might need to modify macros when some aspect of your network changes and one or more clients need to know about the change. For example, you might add a router or a NIS server, create a new subnet, or decide to change the lease policy.

When you modify a macro, you must know the name of the DHCP option that corresponds to the parameter you want to change, add, or delete. The standard DHCP options are listed in the DHCP Manager help and in the `dhcptab`(4) man page.

You can use the `dhtadm -M -m` command or DHCP Manager to modify macros. See the `dhtadm`(1M) man page for more information about `dhtadm`.

The following figure shows DHCP Manager's Macro Properties dialog box.

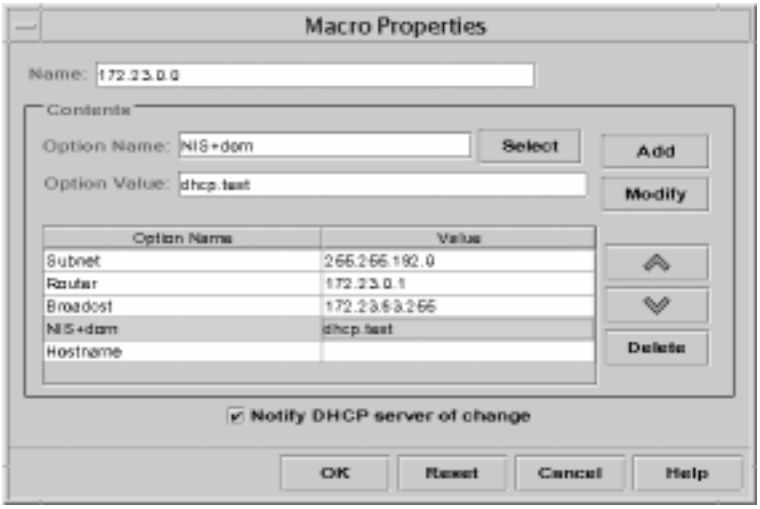


Figure 11–15 Macro Properties Dialog Box

▼ How to Change Values for Options in a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**
2. **Select the macro you want to change.**
3. **Choose Properties from the Edit menu.**
   The Macro Properties dialog box opens.
4. **In the table of Options, select the option you want to change.**

The option's name and value are displayed in the Option Name and Option Value fields.

5. **In the Option Value field, select the old value and type the new value for the option.**

6. **Click Modify.**

   The new value is displayed in the options table.

7. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` to put the change into effect immediately after you click OK.

8. **Click OK.**

▼ How to Add Options to a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**

2. **Select the macro you want to change.**

3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box opens.

4. **In the Option Name field, specify the name of an option using one of the following methods:**

   a. **Click the Select button next to the Option Name field and select the option you want to add to the macro.**

      The Select Option dialog box displays an alphabetized list of names of Standard category options and descriptions. If you want to add an option that is not in the Standard category, use the Category list to select the category you want.

      See "About Macros" on page 160 for more information about macro categories.

   b. **Type `Include` if you want to include a reference to an existing macro in the new macro.**

5. **Type the value for the option in the Option Value field.**

If you typed **Include** as the option name, you must specify the name of an existing macro in the Option Value field.

6. **Click Add.**

   The option is added to the bottom of the list of options displayed for this macro. If you want to change the option's position in the list, select the option and click the arrow keys next to the list to move the option up or down.

7. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` to put the change into effect immediately after you click OK.

8. **Click OK.**

▼ How to Delete Options from a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**

2. **Select the macro you want to change.**

3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box opens.

4. **Select the option you want to remove from the macro.**

5. **Click Delete.**

   The option is removed from the list of options for this macro.

6. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` to put the change into effect immediately after you click OK.

7. **Click OK.**

Adding DHCP Macros

You may want to add new macros to your DHCP service to support clients with specific needs. You can use the `dhtadm -A -m` command or DHCP Manager's Create Macro dialog box to add macros. See the `dhtadm`(1M) man page for more information about `dhtadm`.

The following figure shows DHCP Manager's Create Macro dialog box.

Figure 11–16 Create Macro Dialog Box

▼ How to Add a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**

2. **Choose Create from the Edit menu.**

   The Create Macro dialog box opens.

3. **Type a unique name for the macro.**

   If you use a name that matches a vendor class identifier, network address, or client ID, the macro will be processed automatically for appropriate clients. If you use a different name, the macro can only be processed if it is assigned to a specific IP address or included in another macro that is processed. See "Macro Processing by the DHCP Server" on page 160 for more detailed information.

4. **Click the Select button next to the Option Name field.**

   The Select Option dialog box opens, displaying an alphabetized list of names of Standard category options and their descriptions.

5. **If you want to an option that is not in the Standard category, use the Category list to select the category you want.**

   See "About Options" on page 159 for more information about option categories.

6. **Select the option you want to add to the macro and click OK.**

The Macro Properties dialog box displays the selected option in the Option Name field.

7. **Type the value for the option in the Option Value field.**

8. **Click Add.**

   The option is added to the bottom of the list of options displayed for this macro. If you want to change the option's position in the list, select the option and click the arrow keys next to the list to move the option up or down.

9. **Repeat Step 6 on page 253 through Step 8 on page 254 for each option you want to add to the macro.**

   If you want to rearrange the order of the options, select option names and click the arrow keys to move the names up and down in the list of options.

10. **Select Notify DHCP Server of Change when you are finished adding options.**

    This selection tells the DHCP server to reread the `dhcptab` to put the change into effect immediately after you click OK.

11. **Click OK.**

Deleting DHCP Macros

You might want to delete a macro from the DHCP service. For example, if you delete a network from the DHCP service, you can also delete the associated server macro because it will no longer be used.

You can use the `dhtadm -D -m` command or DHCP Manager to delete macros.

▼ How to Delete a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**

2. **Select the macro you want to delete.**

   The Delete Macro dialog box opens, prompting you to confirm that you want to delete the specified macro.

3. **Select Notify DHCP Server of Change.**

4. **Click OK.**

Working With DHCP Options

Options are keywords for network configuration parameters that the DHCP server can pass to clients. In the Solaris DHCP service, the only options that you can create, delete, or modify are those that are not specified as standard options in the Solaris DHCP service. For this reason, when you first set up your DHCP service, the Options tab in DHCP Manager is empty until you create options for your site.

If you create options on the DHCP server, you must also add information about the options on the DHCP client. For the Solaris DHCP client, you must edit the `/etc/dhcp/inittab` file to add entries for the new options. If you have non-Solaris DHCP clients, refer to the documentation for those clients for information about adding new options or symbols. See “About Options” on page 159 for more information about options in Solaris DHCP.

You can use either DHCP Manager or the `dhtadm` command to create, modify, or delete options.

Note - Options are called *symbols* in the DHCP literature. The `dhtadm` command and man page also refer to options as symbols.

The following task map lists tasks you must perform to create, modify, and delete DHCP options and the procedures needed to carry them out.

TABLE 11–8 DHCP Options Task Map

Tasks	Description	For Instructions, Go To...
Create DHCP options.	Add new options for information not covered by a standard DHCP option.	"How to Create DHCP Options (DHCP Manager)" on page 259 "How to Create DHCP Options (Command Line)" on page 259 "Modifying the Solaris DHCP Client's Option Information" on page 263
Modify DHCP options.	Change properties of DHCP options you have created.	"How to Modify DHCP Option Properties (DHCP Manager)" on page 261 "How to Modify DHCP Option Properties (Command Line)" on page 261
Delete DHCP options.	Remove DHCP options you have created.	"How to Delete DHCP Options (DHCP Manager)" on page 262 "How to Delete DHCP Options (Command Line)" on page 263

Before creating options, you should be familiar with the option properties listed in the following table.

TABLE 11–9 DHCP Option Properties

Option Properties	Description
Category	The category of an option must be one of the following: Vendor – Options specific to a client's vendor platform, either hardware or software. Site – Options specific to your site. Extend - Newer options that have been added to the DHCP protocol, but not yet implemented as standard options in Solaris DHCP.
Code	The code is a unique number you assign to an option; the same code cannot be used for any other option within its option category. The code must be appropriate for the option category: Vendor – Code values of 1–254 for each vendor class Site – Code values of 128–254 Extend – Code values of 77-127
Data type	The data type specifies what kind of data can be assigned as a value for the option. Valid data types are: ASCII – Text string value. BOOLEAN – No value is associated with the Boolean data type. The presence of the option indicates a condition is true, while the absence of the option indicates false. For example, the Hostname option (which is a Standard option and cannot be modified) is a Boolean. If it is included in a macro, it tells the DHCP server that it should consult name services to see if there is a host name associated with the assigned address. IP – One or more IP addresses, in dotted decimal format (*xxx.xxx.xxx.xxx*). NUMBER – Unsigned number. For example, the MTU option accepts numbers such as 1500. OCTET – Uninterpreted hexadecimal ASCII representation of binary data. For example, a client ID uses the octet data type.
Granularity	Specifies how many "instances" of the data type are needed to represent a complete option value. For example, a data type of IP and a granularity of 2 would mean that the option value must contain two IP addresses. A data type of NUMBER may specify a granularity of 1, 2, 4, or 8 octets each.

TABLE 11–9 DHCP Option Properties *(continued)*

Option Properties	Description
Maximum	The maximum number of values that can be specified for the option. Building on the previous example, a maximum of 2, with a granularity of 2 and a data type of IP Address would mean that the option value could contain a maximum of two pairs of IP addresses.
Vendor client classes	This option is available only when the option category is Vendor. It identifies the client class(es) with which the Vendor option is associated. The Class is an ASCII string representing the client machine type and/or operating system, for example, `SUNW.Javastation`. This type of option makes it possible to define configuration parameters that are passed to all clients of the same class, and *only* clients of that class. You can specify multiple client classes. Only those DHCP clients with a client class value matching one you specify will receive the options scoped by that class. For IA32–based machines, the Vendor client class is always `SUNW.i86pc`. For Sparc-based machines, the Vendor client class can be obtained by typing `uname --i` on the client. To specify the Vendor client class, substitute periods for any commans in the string returned by the `uname` command. For example, if the string `SUNW,Ultra-1` is returned by the `uname --i` command, you should specify the Vendor client class as `SUNW.Ultra-1`.

Creating DHCP Options

If you need to pass clients information for which there is not already an existing option in the DHCP protocol, you can create an option. See `dhcptab`(4) for a list of all the options that are defined in Solaris DHCP before creating your own.

You can use the `dhtadm -A -s` command or DHCP Manager's Create Option dialog box to create new options.

The following figure shows DHCP Manager's Create Option dialog box.

Figure 11–17 Create Option Dialog Box

▼ How to Create DHCP Options (DHCP Manager)

1. **Select the Options tab.**

2. **Choose Create from the Edit menu.**
   The Create Options dialog box opens.

3. **Type a short descriptive name for the new option.**
   The name may contain up to eight alphanumeric characters and no spaces.

4. **Type or select values for each setting in the dialog box.**
   Refer to Table 11–9 for information about each setting.

5. **Select Notify DHCP Server of Change if you are finished creating options.**

6. **Click OK.**
   You can now add the option to macros and assign a value to the option to pass to clients.

▼ How to Create DHCP Options (Command Line)

1. **Become superuser on the DHCP server system.**

2. **Type a command using the following format:**

```
# dhtadm -A -s option-name -d 'category,code,data-type,granularity,maximum'
```

option-name	is an alphanumeric string of eight characters or less.
category	is `Site`, `Extend`, or `Vendor=`*list-of-classes*, and *list-of-classes* is a space-separated list of vendor client classes to which the option applies. See Table 11–9 for a information about determining the vendor client class.
code	is a numeric value appropriate to the option category, as explained in Table 11–9
data-type	is `ASCII`, `IP`, `BOOLEAN`, `NUMBER`, or `OCTET`
granularity	is a non-negative number, as explained in Table 11–9
maximum	is a non-negative number, as explained in Table 11–9

The following two commands are two examples:

```
# dhtadm -A -s NewOpt -d 'Site,130,NUMBER,1,1'
# dhtadm -A -s NewServ -d 'Vendor=SUNW.Ultra-1 SUNW.SPARCstation10,200,IP,1,1'
```

Modifying DHCP Options

If you have created options for your DHCP service, you can change the properties for an option using either DHCP Manager or the `dhtadm` command.

You can use the`dhtadm -M -s` command or DHCP Manager's Option Properties dialog box to modify options.

Note that you should modify the Solaris DHCP client's option information to reflect the same modification you make to the DHCP service. See "Modifying the Solaris DHCP Client's Option Information" on page 263.

The following figure shows DHCP Manager's Option Properties dialog box.

Figure 11–18 Option Properties Dialog Box

▼ How to Modify DHCP Option Properties (DHCP Manager)

1. **Select the Options tab.**
2. **Select the option whose properties you want to change.**
3. **Choose Properties from the Edit menu.**
   The Option Properties dialog box opens.
4. **Edit the properties as needed.**
   See Table 11–9 for information about the properties.
5. **Select Notify Server of Change when you are finished with options.**
6. **Click OK.**

▼ How to Modify DHCP Option Properties (Command Line)

1. **Become superuser on the DHCP server system.**
2. **Type a command using the following format:**

```
# dhtadm -M -s option-name-d 'category,code,data-type,granularity,maximum'
```

option-name is the option name whose definition you want to change.

category is `Site`, `Extend`, or `Vendor=`*list-of-classes*, and *list-of-classes* is a space-separated list of vendor client classes to which the option applies. For example, `SUNW.Ultra-1 SUNW.i86pc`.

code is a numeric value appropriate to the option category, as explained in Table 11–9.

data-type is `ASCII`, `IP`, `BOOLEAN`, `NUMBER`, or `OCTET`.

granularity is a non-negative number, as explained in Table 11–9.

maximum is a non-negative number, as explained in Table 11–9.

Note that you must specify all of the DHCP option properties with the `-d` switch, not just the properties you are changing.

The following two commands are two examples:

```
# dhtadm -M -s NewOpt -d 'Site,135,NUMBER,1,1'
# dhtadm -M -s NewServ -d 'Vendor=SUNW.Ultra-1 SUNW.i86pc,200,IP,1,1'
```

Deleting DHCP Options

You cannot delete standard DHCP options, but if you have defined options for your DHCP service, you can delete them using DHCP Manager or the `dhtadm` command.

▼ How to Delete DHCP Options (DHCP Manager)

1. **Select the Options tab.**
2. **Choose Delete from the Edit menu.**

   The Delete Options dialog box opens.
3. **Confirm the deletion by clicking OK.**

▼ How to Delete DHCP Options (Command Line)

1. **Become superuser on the DHCP server system.**

2. **Type a command using the following format:**

```
# dhtadm -D -s option-name
```

Modifying the Solaris DHCP Client's Option Information

If you add a new DHCP option to your DHCP server, you must add a complementary entry to each DHCP client's option information. If you are using a DHCP client other than the Solaris DHCP client, please refer to that client's documentation for information about adding options or symbols.

On a Solaris DHCP client, you must edit the `/etc/default/inittab` file and add an entry for each option that you add to the DHCP server. If you later modify the option on the server, you must modify the entry in the client's `/etc/default/inittab` file accordingly.

Please refer to the `dhcp_inittab`(4) man page for detailed information about the syntax of the `/etc/default/inittab` file.

Note - If you added DHCP options to the `dhcptags` file in a previous release of Solaris DHCP, you must add the options to the `/etc/default/inittab` file. See "DHCP Option Information" on page 291 for more information.

Supporting Solaris Network Install Clients with the DHCP Service

You can use DHCP to install the Solaris operating environment on certain client machines on your network. Only Sun Enterprise Ultra machines and Intel machines meeting the hardware requirements for running the Solaris operating environment can use this feature.

The following task map shows the high-level tasks that must be done to allow clients to obtain install parameters using DHCP.

TABLE 11–10 DHCP Network Install Task Map

Task	Description	For Instructions, Go To...
Set up an install server.	Set up a Solaris server to support clients that want to install the Solaris operating environment from the network.	"Preparing to Install Solaris Software Over the Network" in *Solaris 8 Advanced Installation Guide*
Set up client systems for Solaris installation over the network using DHCP.	Use `add_install_client -d` to add DHCP network install support for a class of client (such as those of a certain machine type) or a particular client ID.	"Preparing to Install Solaris Software Over the Network" in *Solaris 8 Advanced Installation Guide* `add_install_client`(1M)
Create DHCP options for install parameters and macros including the options.	Use DHCP Manager or `dhtadm` to create new Vendor options and macros which the DHCP server can use to pass installation information to the clients.	"Creating DHCP Options and Macros for Solaris Install Parameters" on page 264

Creating DHCP Options and Macros for Solaris Install Parameters

When you add clients using the `add_install_client -d` script on the install server, the script reports DHCP configuration information to standard output. This information can be used when you create the options and macros needed to pass network installation information to clients.

To support clients needing to install from the network, you must create Vendor category options to pass information that is needed to correctly install the Solaris operating environment. The following table shows the options you need to create and the properties needed to create them.

TABLE 11–11 Values for Creating Vendor Category Options for SUNW Clients

Name	Code	Data Type	Granularity	Maximum	Vendor Client Classes *	Description
SrootOpt	1	ASCII text	1	0	SUNW.Ultra–1, SUNW.Ultra-30, SUNW.i86pc	NFS mount options for the client's root file system
SrootIP4	2	IP address	1	1	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	IP address of root server
SrootNM	3	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Host name of root server
SrootPTH	4	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to the client's root directory on the root server
SswapIP4	5	IP address	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	IP address of swap server
SswapPTH	6	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to the client's swap file on the swap server
SbootFIL	7	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to the client's boot file
Stz	8	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Time zone for client
SbootRS	9	NUMBER	2	1	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	NFS read size used by stand-alone boot program when loading kernel
SinstIP4	10	IP address	1	1	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	IP address of Jumpstart Install server
SinstNM	11	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Host name of install server

TABLE 11–11 Values for Creating Vendor Category Options for SUNW Clients *(continued)*

Name	Code	Data Type	Granularity	Maximum	Vendor Client Classes *	Description
SinstPTH	12	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to installation image on install server
SsysidCF	13	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to `sysidcfg` file, in the format *server:/path*
SjumpsCF	14	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to JumpStart configuration file in the format *server:/path*
Sterm	15	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Terminal type

* The vendor client classes determine what classes of client can use the option. Vendor client classes listed here are suggestions only. You should specify client classes that indicate the actual clients in your network that will be installing from the network. See Table 11–9 for information about determining a client's vendor client class.

When you have created the options, you can create macros that include those options. The following table lists suggested macros you can create to support Solaris installation for clients.

TABLE 11–12 Suggested Macros to Support Network Install Clients

Macro Name	Contains These Options and Macros
Solaris	SrootIP4, SrootNM, SinstIP4, SinstNM, Sterm
sparc	SrootPTH, SinstPTH
sun4u	Solaris and sparc macros
i86pc	Solaris macro, SrootPTH, SinstPTH, SbootFIL
SUNW.i86pc *	i86pc macro

TABLE 11–12 Suggested Macros to Support Network Install Clients *(continued)*

Macro Name	Contains These Options and Macros
SUNW.Ultra-1 *	sun4u macro, SbootFIL
SUNW.Ultra-30 *	sun4u macro, SbootFIL macro
xxx.xxx.xxx.xxx (network address macros)	BootSrvA option could be added to existing network address macros. The value of BOOTSrvA should indicate the tftboot server.

* These macro names match the Vendor client classes of the clients that will install from the network. These names are examples of clients you might have on your network. See Table 11–9 for information about determining a client's vendor client class.

You can create these options and macros using the dhtadm command or DHCP Manager. If you use dhtadm, it might be easiest to create the options and macros by writing a script that uses the dhtadm command repeatedly. This is the recommended approach.

The following section, "Writing a Script That Uses dhtadm to Create Options and Macros" on page 267, shows a sample script using the dhtadm command. If you prefer to use DHCP Manager, see "Using DHCP Manager to Create Install Options and Macros" on page 269.

Writing a Script That Uses dhtadm to Create Options and Macros

You can create a Korn shell script by adapting the example below to create all the options listed in Table 11–11 and some useful macros. Be sure to change all IP addresses and values contained in quotes to the correct IP addresses, server names, and paths for your network. You should also edit the Vendor= key to indicate the class of clients you have. Use the information reported by add_install_client -d to obtain the data needed to adapt the script.

CODE EXAMPLE 11–1 Sample Script for Adding Options and Macros to Support Network Installation

```
# Load the Solaris vendor specific options. We'll start out supporting
# the Ultra-1, Ultra-30, and i86 platforms. Changing -A to -M would replace
# the current values, rather than add them.
dhtadm -A -s SrootOpt -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,1,ASCII,1,0'
dhtadm -A -s SrootIP4 -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,2,IP,1,1'
```

(continued)

(Continuation)

```
dhtadm -A -s SrootNM -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,3,ASCII,1,0'
dhtadm -A -s SrootPTH -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,4,ASCII,1,0'
dhtadm -A -s SswapIP4 -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,5,IP,1,0'
dhtadm -A -s SswapPTH -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,6,ASCII,1,0'
dhtadm -A -s SbootFIL -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,7,ASCII,1,0'
dhtadm -A -s Stz -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,8,ASCII,1,0'
dhtadm -A -s SbootRS -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,9,NUMBER,2,1'
dhtadm -A -s SinstIP4 -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,10,IP,1,1'
dhtadm -A -s SinstNM -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,11,ASCII,1,0'
dhtadm -A -s SinstPTH -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,12,ASCII,1,0'
dhtadm -A -s SsysidCF -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,13,ASCII,1,0'
dhtadm -A -s SjumpsCF -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,14,ASCII,1,0'
dhtadm -A -s Sterm -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,15,ASCII,1,0'
# Load some useful Macro definitions
# Define all Solaris-generic options under this macro named Solaris.
dhtadm -A -m Solaris -d \
':SrootIP4=172.21.0.2:SrootNM="blue2":SinstIP4=172.21.0.2:SinstNM="red5":Sterm="xterm":'
# Define all sparc-platform specific options under this macro named sparc.
dhtadm -A -m sparc -d ':SrootPTH="/export/sparc/root":SinstPTH="/export/sparc/install":'
# Define all sun4u architecture-specific options under this macro named sun4u. (Includes
# Solaris and sparc macros.)
dhtadm -A -m sun4u -d ':Include=Solaris:Include=sparc:'
# Solaris on IA32-platform-specific parameters are under this macro named i86pc.
dhtadm -A -m i86pc -d \
':Include=Solaris:SrootPTH="/export/i86pc/root":SinstPTH="/export/i86pc/install"\
:SbootFIL="/platform/i86pc/kernel/unix":'
# Solaris on IA32 machines are identified by the "SUNW.i86pc" class. All
# clients identifying themselves as members of this class will see these
# parameters in the macro called SUNW.i86pc, which includes the i86pc macro.
dhtadm -A -m SUNW.i86pc -d ':Include=i86pc:'
# Ultra-1 platforms identify themselves as part of the "SUNW.Ultra-1" class.
# By default, we boot these machines in 32bit mode. All clients identifying
# themselves as members of this class will see these parameters.
dhtadm -A -m SUNW.Ultra-1 -d ':SbootFIL="/platform/sun4u/kernel/unix":Include=sun4u:'
# Ultra-30 platforms identify themselves as part of the "SUNW.Ultra-30" class.
# By default, we will boot these machines in 64bit mode. All clients
# identifying themselves as members of this class will see these parameters.
dhtadm -A -m SUNW.Ultra-30 -d ':SbootFIL="/platform/sun4u/kernel/sparcv9/
unix":Include=sun4u:'
# Add our boot server IP to each of the network macros for our topology served
# by our DHCP server. Our boot server happens to be the same machine running DHCP server.
dhtadm -M -m 172.20.64.64 -e BootSrvA=172.21.0.2
dhtadm -M -m 172.20.64.0 -e BootSrvA=172.21.0.2
dhtadm -M -m 172.20.64.128 -e BootSrvA=172.21.0.2
dhtadm -M -m 172.21.0.0 -e BootSrvA=172.21.0.2
dhtadm -M -m 172.22.0.0 -e BootSrvA=172.21.0.2
# Make sure we return hostnames to our clients.
dhtadm -M -m DHCP-servername -e Hostname=_NULL_VALUE_
# The client with this MAC address is a diskless client. Override the root
# settings which at the network scope setup for Install with our client's
# root directory.
dhtadm -A -m 0800201AC25E -d \
```

(continued)

(Continuation)

```
':SrootIP4=172.23.128.2:SrootNM="orange-svr-2":SrootPTH="/export/root/172.23.128.12":'
```

Execute the script as superuser to add the options and macros to your `dhcptab`. When you have done this, network client classes that are listed in the `Vendor=` string can install from the network using DHCP.

Using DHCP Manager to Create Install Options and Macros

You can create the options listed in Table 11–11 and the macros listed in Table 11–12 using DHCP Manager.

See Figure 11–17 and Figure 11–16 for illustrations of the dialog boxes you use to create options and macros.

▼ How to Create Options to Support Solaris Installation (DHCP Manager)

1. **Select the Options tab in DHCP Manager.**

2. **Choose Create from the Edit menu.**

   The Create Option dialog box opens.

3. **Type the option name for the first option and type values appropriate for that option.**

   Use Table 11–11 to look up the option names and values for options you must create. Notice that the Vendor Client Classes are only suggested values. You should create classes to indicate the actual client types that will install using DHCP. See Table 11–9 for information about determining a client's vendor client class.

4. **Click OK when you have entered all the values.**

5. **In the Options tab, select the option you just created.**

6. **Select Duplicate from the Edit menu.**

   The Duplicate Option dialog box opens.

7. **Type the name of another option and modify other values appropriately.**

   The values for code, data type, granularity, and maximum are most likely to need modification. See Table 11–11 for the values.

8. **Repeat Step 4 on page 269 through Step 7 on page 269 until you have created all the options.**

   You can now create macros to pass the options to network install clients, as explained in the following procedure.

▼ How to Create Macros to Support Solaris Installation (DHCP Manager)

1. **Select the Macros tab in DHCP Manager.**

2. **Choose Create from the Edit menu.**

   The Create Macro dialog box opens.

3. **Type the name of a macro.**

   See Table 11–12 for macro names you might use.

4. **Click the Select button.**

   The Select Option dialog box opens.

5. **Select Vendor in the Category list.**

   The Vendor options you created are listed.

6. **Select an option you want to add to the macro and click OK.**

7. **Type a value for the option.**

   See Table 11–11 for the option's data type and refer to the information reported by `add_install_client -d`.

8. **Repeat Step 4 on page 270 through Step 7 on page 270 for each option you want to include.**

   To include another macro, type `Include` as the option name and type the macro name as the option value.

9. **Click OK when the macro is complete.**

CHAPTER 12

Troubleshooting DHCP

This chapter provides information to help you solve problems you might encounter when configuring a DHCP server or client, or problems in using DHCP after configuration is complete.

The chapter includes the following information:

Troubleshooting DHCP Server Problems

The problems you might encounter while configuring the server generally fall into the following categories:

- NIS+, if you choose to use NIS+ for your data store
- IP address allocation

NIS+ Problems

If you decide to use NIS+ as the DHCP data store, problems you might encounter can be categorized as follows:

- Cannot select NIS+ as a data store
- NIS+ is not adequately configured
- NIS+ access problems due to insufficient permissions and credentials

Cannot Select NIS+ as a Data Store

If you try to use NIS+ as your data store, you might find that DHCP Manager does not offer it as a choice for data store, or `dhcpconfig` returns a message saying NIS+ does not appear to be installed and running. This means that NIS+ has not been configured for this server, although NIS+ might be in use on the network. Before you can select NIS+ as a data store, the server machine must be configured as an NIS+ client.

Before you set up the server as an NIS+ client the domain must have already been configured and its master server must be running. The master server of the domain's tables should be populated, and the hosts table must have an entry for the new client machine (the DHCP server machine). "Configuring NIS+ Clients" in *Solaris Naming Setup and Configuration Guide* provides detailed information about configuring an NIS+ client.

NIS+ Not Adequately Configured

After you are successfully using NIS+ with DHCP, you might encounter errors if changes are made to NIS+ at a later point, introducing configuration problems. Use the following table to help you determine the cause of configuration problems.

TABLE 12–1 NIS+ Configuration Problems

Possible problem	To determine if this is the problem...	What to do if this is the problem...
Root object does not exist in the NIS+ domain.	Enter the command `/usr/lib/nis/nisstat` This command displays statistics for the domain. If the root object does not exist, no statistics are returned.	Set up the NIS+ domain using the *Solaris Naming Setup and Configuration Guide.*
NIS+ is not used for `passwd` and `publickey` information.	Enter this command to view the name service switch configuration file. `cat /etc/nsswitch.conf` Check the `passwd` and `publickey` entries for the "nisplus" keyword.	Refer to the *Solaris Naming Setup and Configuration Guide* for information about configuring the name service switch.

TABLE 12–1 NIS+ Configuration Problems *(continued)*

Possible problem	To determine if this is the problem...	What to do if this is the problem...
The domain name is empty.	Enter the command: `domainname` If the command lists an empty string, no domain name has been set for the domain.	Use local files for your data store, or set up an NIS+ domain for your network. Refer to *Solaris Naming Setup and Configuration Guide*.
The `NIS_COLD_START` file does not exist.	Enter the following command on the server system to determine if the file exists: `cat /var/nis/NIS_COLD_START`	Use local files for your data store, or create an NIS+ client. Refer to *Solaris Naming Setup and Configuration Guide*.

NIS+ Access Problems

NIS+ access problems might cause you to receive error messages about incorrect DES credentials, or inadequate permissions to update NIS+ objects or tables. Use the following table to determine the cause of NIS+ errors you receive.

TABLE 12–2 NIS+ Access Problems

Possible problem	To determine if this is the problem...	What to do if this is the problem...
The DHCP server machine does not have create access to the `org_dir` object in the NIS+ domain.	Enter the command `nisls -ld org_dir` The access rights are listed in the form `r---rmcdrmcdr---`, where the permissions apply respectively to nobody, owner, group, and world. The owner of the object is listed next.	Use the `nischmod` command to change the permissions for `org_dir`. For example, to add create access to the group, type `nischmod g+c org_dir`
	Normally the `org_dir` directory object provides full (read, modify, create, and destroy) rights to both the owner and the group, while providing only read access to the world and nobody classes.	See the `nischmod`(1) man page for more information.

TABLE 12–2 NIS+ Access Problems *(continued)*

Possible problem	To determine if this is the problem...	What to do if this is the problem...
	The DHCP server name must either be listed as the owner of the `org_dir` object, or be listed as a principal in the group, and that group must have create access. List the group with the command: `nisls -ldg org_dir`	
The DHCP server does not have access rights to create a table under the `org_dir` object. Usually, this means the server machine's principal name is not a member of the owning group for the `org_dir` object, or no owning group exists.	Enter this command to find the owning group name: `niscat -o org_dir` Look for a line similar to `Group : "admin.myco.com."` List the principal names in the group using the command: `nisgrpadm -l` *groupname*	Add the server machine's name to the group using the `nisgrpadm` command. For example, to add the server name `pacific` to the group `admin.myco.com`, type the following: `nisgrpadm -a admin.myco.com pacific.myco.com`
	For example, `nisgrpadm -l admin.myco.com` The server machine's name should be listed as an explicit member of the group or included as an implicit member of the group.	See the `nisgrpadm`(1) man page for more information.
The DHCP server does not have valid Data Encryption Standard (DES) credentials in the NIS+ cred table.	If this is the problem, an error message states that the user does not have DES credentials in the NIS+ name service.	Use the `nisaddcred` command to add security credentials for the DHCP server machine. The following example shows how to add DES credentials for the system `mercury` in the domain `Faxco.COM`: `nisaddcred -p unix.mercury@Faxco.COM \ -P mercury.Faxco.COM. DES Faxco.COM`. The command prompts for the root password (which is required to generate an encrypted secret key). See the `nisaddcred`(1M) man page for more information.

IP Address Allocation Errors

When a client attempts to obtain or verify an IP address, you might see the following problems logged to `syslog` or in server debug output.

TABLE 12–3 IP Address Allocation and Leasing Problems

Error Message	Explanation	Solution
`There is no` *n.n.n.n* `dhcp-network table for DHCP client's network.`	A client is requesting a specific IP address or seeking to extend a lease on its current IP address but the DHCP server cannot find the DHCP network table for that address.	The DHCP network table might have been deleted mistakenly. You can recreate the network table using DHCP Manager or `dhcpconfig`.
`ICMP ECHO reply to OFFER candidate:` *n.n.n.n*`, disabling`	The IP address being considered for offering to a DHCP client is already in use. This might occur if more than one DHCP server owns the address, or if an address was manually configured for a non-DHCP network client.	Determine the proper ownership of the address and correct either the DHCP server database or the host's network configuration.
`ICMP ECHO reply to OFFER candidate:` *n.n.n.n*`. No corresponding dhcp network record.`	The IP address being considered for offering to a DHCP client does not have a record in a network table. This might occur if the IP address record is deleted from the DHCP network table after the address was selected but before the duplicate address check was completed.	Use DHCP Manager or `pntadm` to view the DHCP network table, and if the IP address is missing, create it with DHCP Manager (choose Create from the Edit menu on the Address tab) or `pntadm`.
`DHCP network record for` *n.n.n.n* `is unavailable, ignoring request.`	The record for the requested IP address is not in the DHCP network table, so the server is dropping the request.	Use DHCP Manager or `pntadm` to view the DHCP network table, and if the IP address is missing, create it with DHCP Manager (choose Create from the Edit menu on the Address tab) or `pntadm`.
n.n.n.n `currently marked as unusable.`	The requested IP address cannot be offered because it has been marked in the network table as unusable.	You can use DHCP Manager or `pntadm` to make the address usable.

TABLE 12–3 IP Address Allocation and Leasing Problems *(continued)*

Error Message	Explanation	Solution
n.n.n.n was manually allocated. No dynamic address will be allocated.	The client's ID has been assigned a manually allocated address, and that address is marked as unusable. The server cannot allocate a different address to this client.	You can use DHCP Manager or `pntadm` to make the address usable, or manually allocate a different address to the client.
Manual allocation (*n.n.n.n, client ID* has *n* other records. Should have 0.	The client having the specified client ID has been manually assigned more than one IP address. There should be only one. The server selects the last manually assigned address it finds in the network table.	Use DHCP Manager or `pntadm` to remove the additional manual allocations.
No more IP addresses on *n.n.n.n* network.	All IP addresses currently managed by DHCP on the specified network have been allocated.	Use DHCP Manager or `pntadm` to create new IP addresses for this network.
Client: *clientid* lease on *n.n.n.n* expired.	The lease was not negotiable, and timed out.	Client should automatically restart the protocol to obtain a new lease.
Offer expired for client: *n.n.n.n*	The server made an IP address offer to the client, but the client took too long to respond and the offer expired.	The client should automatically issue another discover message. If this also times out, increase the cache offer timeout for the DHCP server. In DHCP Manager, choose Modify from the Service menu.
Client: *clientid* REQUEST is missing requested IP option.	The client's request did not specify the offered IP address, so the DHCP server ignores the request. This might occur if the client is a not compliant with the updated DHCP protocol, RFC 2131.	Update client software.

TABLE 12–3 IP Address Allocation and Leasing Problems *(continued)*

Error Message	Explanation	Solution
`Client:` *clientid* `is trying to renew` *n.n.n.n*`, an IP address it has not leased.`	The IP address recorded in the DHCP network table for this client does not match the IP address that the client specified in its renewal request. The DHCP server does not renew the lease.	This problem occurs if you delete a client's record while the client is still using the IP address. Use DHCP Manager or `pntadm` to examine the network table, and correct if necessary.
`Client:` *clientid* `is trying to verify unrecorded address:` *n.n.n.n*`, ignored.`	The specified client has not been registered in the DHCP network table with this address, so the request is ignored by this DHCP server.	Another DHCP server on the network might have assigned this client the address. However, you might also have deleted the client's record while the client was still using the IP address. Use DHCP Manager or `pntadm` to examine the network table on this server and any other DHCP servers on the network and correct if necessary.

Troubleshooting DHCP Client Configuration Problems

The problems you might encounter with a DHCP client generally fall into the following categories:

- "Problems Communicating With DHCP Server" on page 277
- "Problems with Inaccurate DHCP Configuration Information" on page 287

Problems Communicating With DHCP Server

This section describes problems you might encounter as you add DHCP clients to the network.

After you enable the client software and reboot the machine, the client tries to reach the DHCP server to obtain its network configuration. If the client fails to reach the server, you might see error messages such as:

```
DHCP or BOOTP server not responding
```

Before you can determine the problem you must gather diagnostic information from both the client and the server and analyze the information. To gather information you can:

1. Run the client in debug mode.
2. Run the server in debug mode.
3. Start snoop to monitor network traffic.

You can do these things separately or concurrently.

Using the information you gather, you can determine if the problem is on the client or server machine, or on a relay agent, and find a solution.

▼ How to Run the DHCP Client in Debug Mode

If you are running a client other than the Solaris DHCP client, refer to the client's documentation for information about running the client in debug mode.

If you are running the Solaris DHCP client, use the following steps.

1. **Become superuser on the client system.**
2. **Kill the DHCP client daemon and restart it in debug mode using the following commands:**

```
# pkill -x dhcpagent
# /sbin/dhcpagent -d1 -f &
# ifconfig interface dhcp start
```

When run in debug mode, the client daemon displays messages to your screen as it performs DHCP requests. See "DHCP Client Debug Output" on page 280 for information about client debug output.

▼ How to Run the DHCP Server in Debug Mode

1. **Become superuser on the server system.**

2. **Kill and restart the DHCP server daemon in debug mode using the following commands:**

```
# pkill -x in.dhcpd
# /usr/lib/inet/in.dhcpd -d -v
```

You should also use any `in.dhcpd` command-line options that you normally use when running the daemon. For example, if you run the daemon as a BOOTP relay agent, include the `-r` option with the `in.dhcpd -d -v` command.

When run in debug mode, the daemon displays messages to your screen as it processes DHCP/BOOTP requests. See "DHCP Server Debug Output" on page 281 for information about server debug output.

▼ How to Use `snoop` to Monitor DHCP Network Traffic

1. **Become superuser on the DHCP server system.**

2. **Start `snoop` to begin tracing network traffic across the server's network interface.**

```
# /usr/sbin/snoop -d interface -o snoop-output-filename udp port 67 or udp port 68
```

For example:

```
# /usr/sbin/snoop -d le0 -o /tmp/snoop.output udp port 67 or udp port 68
```

Note that `snoop` continues to monitor the interface until you stop it explicitly by pressing Control-C after you have the information you need.

3. **Boot the client system, or restart the `dhcpagent` on the client system.**

Restarting the client is described in "How to Run the DHCP Client in Debug Mode" on page 278.

4. **On the server system, use `snoop` to display the output file with the contents of network packets:**

```
# /usr/sbin/snoop -i snoop-output-filename -x0 -v
```

For example:

```
# /usr/sbin/snoop -i /tmp/snoop.output -x0 -v
```

The `-d` switch with the `dhcpagent` command puts the client in debug mode with level 1 verbosity, and the `-f` switch causes output to be sent to the console instead of to `syslog`. Replace *interface* in the `ifconfig` command line with the name of the network interface of the client (for example, `le0`).

See "DHCP `snoop` Output" on page 285 for information about interpreting the output.

DHCP Client Debug Output

The following example shows normal debug output when a DHCP client sends its DHCP request and receives its configuration information from a DHCP server.

CODE EXAMPLE 12–1 Normal DHCP Client Debug Output

```
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x27fc8 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface le0
/sbin/dhcpagent: debug: insert_ifs: le0: sdumax 1500, optmax 1260, hwtype 1, hwlen 6
/sbin/dhcpagent: debug: insert_ifs: inserted interface le0
/sbin/dhcpagent: debug: register_acknak: registered acknak id 5
/sbin/dhcpagent: debug: unregister_acknak: unregistered acknak id 5
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x26018 (ARP reply filter)
/sbin/dhcpagent: info: setting IP netmask on le0 to 255.255.192.0
/sbin/dhcpagent: info: setting IP address on le0 to 102.23.3.233
/sbin/dhcpagent: info: setting broadcast address on le0 to 102.23.63.255
/sbin/dhcpagent: info: added default router 102.23.0.1 on le0
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x28054 (blackhole filter)
/sbin/dhcpagent: debug: configure_if: bound ifsp->if_sock_ip_fd
/sbin/dhcpagent: info: le0 acquired lease, expires Tue Aug 10 16:18:33 1999
/sbin/dhcpagent: info: le0 begins renewal at Tue Aug 10 15:49:44 1999
/sbin/dhcpagent: info: le0 begins rebinding at Tue Aug 10 16:11:03 1999
```

If the client cannot reach the DHCP server, you might see debug output similar to the following example.

CODE EXAMPLE 12–2 DHCP Client Debug Output When Client Does Not Receive Server Reply

```
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x27fc8 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface le0
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
```

If you see this message, the server's response is not getting to the client. This can mean that the request never reached the server, or that the server cannot send a response to the client. Run snoop on the server as described in "How to Use snoop to Monitor DHCP Network Traffic" on page 279 to determine if packets from the client have reached the server.

DHCP Server Debug Output

Normal server debug output shows server configuration information followed by information about each network interface as the daemon starts. Thereafter, the debug output shows information about requests the daemon processes. The following examples show sample output for a DHCP server and a BOOTP relay agent. Code Example 12–3 shows debug output for a DHCP server that has just started and then extends the lease for a client using an address owned by another DHCP server that is not responding.

CODE EXAMPLE 12–3 Debug Output for DHCP Server

```
Daemon Version: 3.1
Maximum relay hops: 4
Transaction logging to console enabled.
Run mode is: DHCP Server Mode.
Datastore: nisplus
Path: org_dir.dhcp.test..:dhcp.test..:$
DHCP offer TTL: 10
Ethers compatibility enabled.
BOOTP compatibility enabled.
ICMP validation timeout: 1000 milliseconds, Attempts: 2.
Monitor (0005/hme0) started...
Thread Id: 0005 - Monitoring Interface: hme0 *****
MTU: 1500      Type: DLPI
Broadcast: 102.21.255.255
Netmask: 255.255.0.0
Address: 102.21.0.2
Monitor (0006/nf0) started...
Thread Id: 0006 - Monitoring Interface: nf0 *****
MTU: 4352      Type: DLPI
Broadcast: 102.22.255.255
```

(continued)

(Continuation)

```
Netmask: 255.255.0.0
Address: 102.22.0.1
Monitor (0007/qe0) started...
Thread Id: 0007 - Monitoring Interface: qe0 *****
MTU: 1500      Type: DLPI
Broadcast: 102.23.63.255
Netmask: 255.255.192.0
Address: 102.23.0.1
Read 33 entries from DHCP macro database on Tue Aug 10 15:10:27 1999
Datagram received on network device: qe0
Client: 0800201DBA3A is requesting verification of address owned by 102.21.0.4
Datagram received on network device: qe0
Client: 0800201DBA3A is requesting verification of address owned by 102.21.0.4
Datagram received on network device: qe0
Client: 0800201DBA3A is requesting verification of address owned by 102.21.0.4
Datagram received on network device: qe0
Client: 0800201DBA3A maps to IP: 102.23.3.233
Unicasting datagram to 102.23.3.233 address.
Adding ARP entry: 102.23.3.233 == 0800201DBA3A
DHCP EXTEND 0934312543 0934316143 102.23.3.233 102.21.0.2
          0800201DBA3A SUNW.SPARCstation-10 0800201DBA3A
```

The following example shows debug output from a DHCP daemon that starts up as a BOOTP relay agent and relays requests from a client to a DHCP server, and relays the servers responses to the client.

CODE EXAMPLE 12–4 Sample Debug Output for BOOTP Relay

```
Relay destination: 102.21.0.4 (blue-servr2)          network: 102.21.0.0
Daemon Version: 3.1
Maximum relay hops: 4
Transaction logging to console enabled.
Run mode is: Relay Agent Mode.
Monitor (0005/hme0) started...
Thread Id: 0005 - Monitoring Interface: hme0 *****
MTU: 1500      Type: DLPI
Broadcast: 102.21.255.255
Netmask: 255.255.0.0
Address: 102.21.0.2
Monitor (0006/nf0) started...
Thread Id: 0006 - Monitoring Interface: nf0 *****
MTU: 4352      Type: DLPI
Broadcast: 102.22.255.255
Netmask: 255.255.0.0
Address: 102.22.0.1
Monitor (0007/qe0) started...
Thread Id: 0007 - Monitoring Interface: qe0 *****
MTU: 1500      Type: DLPI
Broadcast: 102.23.63.255
```

(continued)

(Continuation)

```
Netmask: 255.255.192.0
Address: 102.23.0.1
Relaying request 0800201DBA3A to 102.21.0.4, server port.
BOOTP RELAY-SRVR 0934297685 0000000000 0.0.0.0 102.21.0.4 0800201DBA3A N/A 0800201DBA3A
Packet received from relay agent: 102.23.0.1
Relaying reply to client 0800201DBA3A
Unicasting datagram to 102.23.3.233 address.
Adding ARP entry: 102.23.3.233 == 0800201DBA3A
BOOTP RELAY-CLNT 0934297688 0000000000 102.23.0.1 102.23.3.233 0800201DBA3A N/
A 0800201DBA3A
Relaying request 0800201DBA3A to 102.21.0.4, server port.
BOOTP RELAY-SRVR 0934297689 0000000000 0.0.0.0 102.21.0.4 0800201DBA3A N/A 0800201DBA3A
Packet received from relay agent: 102.23.0.1
Relaying reply to client 0800201DBA3A
Unicasting datagram to 102.23.3.233 address.
Adding ARP entry: 102.23.3.233 == 0800201DBA3A
```

If there is a problem, the debug output might display warnings or error messages. Use the following table to find the error message or condition and find a solution.

TABLE 12–4 DHCP Server Error Messages

Message	Explanation	Solution
`ICMP ECHO reply to OFFER candidate:` *ip_address* `disabling`	Before the DHCP server offers an IP address to a client, it verifies that the address is not in use by pinging the address. If a client replies, the address is in use.	Make sure the addresses you configured are not already in use.
`No more IP addresses on` *network_address* `network.`	No available IP addresses in the client's per network table.	Allocate more IP addresses using DHCP Manager or `pntadm`. If the DHCP daemon is monitoring multiple subnets, be sure the additional addresses are for the subnet where the client is located.

TABLE 12–4 DHCP Server Error Messages *(continued)*

Message	Explanation	Solution
`No more IP addresses for` *network_address* `network` when you are running the DHCP daemon in BOOTP compatibility mode (`-b` option).	BOOTP does not use a lease time, so the DHCP server looks for free addresses with the BOOTP flag set to allocate to BOOTP clients.	Use DHCP Manager to allocate BOOTP addresses.
`Request to access nonexistent per network database:` *database_name* `in datastore:` *datastore.*	During configuration of the DHCP server, a DHCP network table for a subnet was not created.	Use DHCP Manager or the `pntadm` to create the DHCP network table and new IP addresses.
`There is no` *table_name* `dhcp-network table for DHCP client's network.`	During configuration of the DHCP server, a DHCP network table for a subnet was not created.	Use DHCP Manager or the `pntadm` to create the DHCP network table and new IP addresses.
`Client using non_RFC1048 BOOTP cookie.`	A device on the network is trying to access an unsupported implementation of BOOTP.	Ignore this message, unless you need to configure this device.

`Client` *client_id* `is trying to verify unrecorded address` *ip_address*`, ignored.`	The IP address and client ID in the `/etc/dhcp/`*interface*`.dhc` file on the client do not match the IP address and client ID in the DHCP network database that is checked by the DHCP server.
	
	This can happen when you use local files as the DHCP data store and have multiple DHCP servers (not sharing information) or have changed the DHCP network table
		Restart the DHCP protocol on the client by typing the following commands:
		
		`ifconfig` *interface* `dhcp release`
		
		`ifconfig` *interface* `dhcp start`

DHCP snoop Output

In the snoop output, you should see that packets are exchanged between the DHCP client machine and DHCP server machine. The IP addresses for each machine (and any relay agents or routers in between) is indicated in each packet. If you do not see packets being exchanged, the client machine might not be able to contact the server machine at all, and the problem is at a lower level. See “General Troubleshooting Tips” on page 111 for information about general troubleshooting methods.

To evaluate snoop output, you should know what the expected behavior is (such as if the request should be going through a BOOTP relay agent). You should also know the MAC addresses and IP address of the systems involved (and those of the network interfaces, if there is more than one) so that you can determine if those values are as expected. The following example shows normal snoop output for a DHCP acknowledgement message sent from the DHCP server on `blue-servr2` to a client whose MAC address is `8:0:20:8e:f3:7e`, assigning it the IP address `192.168.252.6` and the host name `white-6`. A number of standard network options and several vendor-specific options are passed to the client as well.

CODE EXAMPLE 12–5 Sample snoop Output for One Packet

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 26 arrived at 14:43:19.14
ETHER:  Packet size = 540 bytes
ETHER:  Destination = 8:0:20:8e:f3:7e, Sun
ETHER:  Source      = 8:0:20:1e:31:c1, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 526 bytes
IP:   Identification = 64667
IP:   Flags = 0x4 IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 254 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 157a
IP:   Source address = 102.21.0.4, blue-servr2
IP:   Destination address = 192.168.252.6, white-6
IP:   No options
IP:   UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
```

(continued)

(Continuation)

```
UDP:  Length = 506
UDP:  Checksum = 5D4C
UDP:
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP:
DHCP: Hardware address type (htype) =  1 (Ethernet (10Mb))
DHCP: Hardware address length (hlen) = 6 octets
DHCP: Relay agent hops = 0
DHCP: Transaction ID = 0x2e210f17
DHCP: Time since boot = 0 seconds
DHCP: Flags = 0x0000
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 192.168.252.6
DHCP: Next server address (siaddr) = 102.21.0.2
DHCP: Relay agent address (giaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:11:E0:1B
DHCP:
DHCP: ----- (Options) field options -----
DHCP:
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 102.21.0.4
DHCP: Subnet Mask = 255.255.255.0
DHCP: Router at = 192.168.252.1
DHCP: Broadcast Address = 192.168.252.255
DHCP: NISPLUS Domainname = dhcp.test
DHCP: IP Address Lease Time = 3600 seconds
DHCP: UTC Time Offset = -14400 seconds
DHCP: RFC868 Time Servers at = 102.21.0.4
DHCP: DNS Domain Name = sem.west.dor.com
DHCP: DNS Servers at = 102.21.0.1
DHCP: Client Hostname = white-6
DHCP: Vendor-specific Options (166 total octets):
DHCP:   (02) 04 octets  0x8194AE1B (unprintable)
DHCP:   (03) 08 octets  "pacific"
DHCP:   (10) 04 octets  0x8194AE1B (unprintable)
DHCP:   (11) 08 octets  "pacific"
DHCP:   (15) 05 octets  "xterm"
DHCP:   (04) 53 octets  "/export/s28/base.s28s_nxt/latest/Solaris_8/Tools/
Boot"
DHCP:   (12) 32 octets  "/export/s28/base.s28s_nxt/latest"
DHCP:   (07) 27 octets  "/platform/sun4m/kernel/unix"
DHCP:   (08) 07 octets  "EST5EDT"
  0: 0800 208e f37e 0800 201e 31c1 0800 4500    .. .ó~.. .1...E.
 16: 020e fc9b 4000 fe11 157a ac15 0004 c0a8    ....@....z......
 32: fc06 0043 0044 01fa 5d4c 0201 0600 2e21    ...C.D..]L.....!
 48: 0f17 0000 0000 0000 0000 c0a8 fc06 ac15    ................
 64: 0002 0000 0000 0800 2011 e01b 0000 0000    ........ .......
 80: 0000 0000 0000 0000 0000 0000 0000 0000    ................
 96: 0000 0000 0000 0000 0000 0000 0000 0000    ................
112: 0000 0000 0000 0000 0000 0000 0000 0000    ................
128: 0000 0000 0000 0000 0000 0000 0000 0000    ................
144: 0000 0000 0000 0000 0000 0000 0000 0000    ................
160: 0000 0000 0000 0000 0000 0000 0000 0000    ................
```

(continued)

```
176: 0000 0000 0000 0000 0000 0000 0000 0000    ................
192: 0000 0000 0000 0000 0000 0000 0000 0000    ................
208: 0000 0000 0000 0000 0000 0000 0000 0000    ................
224: 0000 0000 0000 0000 0000 0000 0000 0000    ................
240: 0000 0000 0000 0000 0000 0000 0000 0000    ................
256: 0000 0000 0000 0000 0000 0000 0000 0000    ................
272: 0000 0000 0000 6382 5363 3501 0536 04ac    ......c.Sc5..6..
288: 1500 0401 04ff ffff 0003 04c0 a8fc 011c    ................
304: 04c0 a8fc ff40 0964 6863 702e 7465 7374    .....@.dhcp.test
320: 3304 0000 0e10 0204 ffff c7c0 0404 ac15    3...............
336: 0004 0f10 736e 742e 6561 7374 2e73 756e    ....sem.west.dor
352: 2e63 6f6d 0604 ac15 0001 0c07 7768 6974    .com........whit
368: 652d 362b a602 0481 94ae 1b03 0861 746c    e-6+.........pac
384: 616e 7469 630a 0481 94ae 1b0b 0861 746c    ific.........pac
400: 616e 7469 630f 0578 7465 726d 0435 2f65    ific...xterm.5/e
416: 7870 6f72 742f 7332 382f 6261 7365 2e73    xport/s28/base.s
432: 3238 735f 776f 732f 6c61 7465 7374 2f53    28s_nxt/latest/S
448: 6f6c 6172 6973 5f38 2f54 6f6f 6c73 2f42    olaris_8/Tools/B
464: 6f6f 740c 202f 6578 706f 7274 2f73 3238    oot. /export/s28
480: 2f62 6173 652e 7332 3873 5f77 6f73 2f6c    /base.s28s_nxt/l
496: 6174 6573 7407 1b2f 706c 6174 666f 726d    atest../platform
512: 2f73 756e 346d 2f6b 6572 6e65 6c2f 756e    /sun4m/kernel/un
528: 6978 0807 4553 5435 4544 54ff              ix..EST5EDT.
```

Problems with Inaccurate DHCP Configuration Information

If a DHCP client is receiving inaccurate information in its network configuration information, such as the wrong NIS domain name, or incorrect router IP address, you must look at the values of options in the macros being processed by the DHCP server for this client.

Use the following general guidelines to help you determine where the inaccurate information is being passed to the client.

- Look at the macros defined on the server as described in “How to View Macros Defined on a DHCP Server (DHCP Manager)” on page 249. Review the information in “Order of Macro Processing” on page 161 and determine which macros are being processed automatically for this client.
- Look at the network table to determine what macro (if any) is assigned to the client’s IP address as the configuration macro. See “Working With IP Addresses in the DHCP Service” on page 234 for more information.
- Take note of any options that occur in more than one macro and make sure that the value you want for an option is set in the last processed macro.
- Edit the appropriate macro(s) to assure that the correct value is passed to the client. See “Modifying DHCP Macros” on page 250.

CHAPTER 13

DHCP Reference

This chapter lists information useful regarding DHCP. It explains the relationships between files and the commands that use the files, but does not explain how to use the commands. The commands are linked to their man pages, so you can find information on a command by clicking on it.

DHCP Commands

The following table lists the commands you might find useful in managing DHCP on your network.

TABLE 13–1 Commands Used in DHCP

Command Man Page	Command Description
`dhtadm`(1M)	Used to make changes to the options and macros in the `dhcptab` file. This command is most useful in scripts that you create to automate changes you need to make to your DHCP information. Use `dhtadm` with the `-P` option and pipe it through the `grep` command for a quick way to search for particular option values in the data base.
`pntadm`(1M)	Used to make changes to the DHCP network tables that map client IDs to IP addresses and optionally associate configuration information with IP addresses.
`dhcpconfig`(1M)	Used to configure and unconfigure DHCP servers and BOOTP relay agents. `dhcpconfig` uses `dhtadm` and `pntadm` to create and make changes to `dhcptab` and DHCP network tables.

TABLE 13–1 Commands Used in DHCP *(continued)*

Command Man Page	Command Description
`in.dhcpd`(1M)	The DHCP server daemon. This command is used by `/etc/init.d/dhcp`, the DHCP service startup and shutdown script. You can start `in.dhcpd` with non-default options, such as `-d` for debugging.
`dhcpmgr`(1M)	The DHCP Manager, a graphical tool used to configure and manage the DHCP service. DHCP Manager is the recommended Solaris DHCP management tool.
`ifconfig`(1M)	Used at system boot to assign IP addresses to network interfaces, configure network interface parameters or both. On a Solaris DHCP client, `ifconfig` starts DHCP to get the parameters (including the IP address) needed to configure a network interface.
`dhcpinfo`(1)	Used by system startup scripts on client machines to obtain information (such as hostname) from the DHCP client daemon (`dhcpagent`) . You can also use `dhcpinfo` in scripts or at the command line to obtain specified parameter values.
`snoop`(1M)	Used to capture and display the contents of packets being passed across the network. `snoop` is useful for troubleshooting problems with the DHCP service.
`dhcpagent`(1M)	The DHCP client daemon, which implements the client side of the DHCP protocol.

DHCP Files

The following table lists files associated with Solaris DHCP.

TABLE 13–2 Files Used by DHCP Daemons and Commands

File Man Pages	**Description**
`dhcptab`(4)	Table of configuration information recorded as options with assigned values, which are then grouped into macros. The `dhcptab` file's location is determined by the data store you use for DHCP information.
`dhcp_network`(4)	Maps IP addresses to client IDs and configuration options. DHCP network tables are named according to the IP address of the network, such as 172.21.32.0. There is no file called `dhcp_network`. The location of DHCP network tables is determined by the data store you use for DHCP information.

TABLE 13–2 Files Used by DHCP Daemons and Commands *(continued)*

File Man Pages	Description
`dhcp`(4)	Records DHCP daemon startup options and the location and type of data store used by DHCP service. The file is located in the `/etc/default` directory.
`nsswitch.conf`(4)	Specifies the location of name service databases and the order in which to search them when looking up various kinds of information. The `nsswitch.conf` file is consulted when you configure a DHCP server in order to obtain accurate configuration information. The file is located in the `/etc` directory.
`resolv.conf`(4)	Contains information used by the DNS resolver. During DHCP server configuration, this file is consulted for information about the DNS domain and DNS server. The file is located in the `/etc` directory.
`dhcp.`*interface*	Indicates that DHCP is to be used on the client's network interface specified in the file name, such as `dhcp.qe0`. The `dhcp.`*interface* file might contain commands that are passed as options to the `ifconfig` *interface* `dhcp start` *option* command used to start DHCP on the client. The file is located in the `/etc` directory.
interface`.dhc`	Contains the configuration parameters obtained from DHCP for the given network interface. The client caches the current configuration information in `/etc/dhcp/`*interface*`.dhc` when the interface's IP address lease is dropped. The next time DHCP starts on the interface, the client requests to use the cached configuration if the lease has not expired. If the DHCP server denies the request, the client begins the standard DHCP lease negotiation process.
`dhcpagent`	Sets parameter values for the `dhcpagent` client daemon. The path to the file is `/etc/default/dhcpagent`. See the file itself or the daemon's `dhcpagent`(1M) man page for information about the parameters.
`dhcp_inittab`(4)	Defines aspects of DHCP option codes, such as the data type, and assigns mnemonic labels. The information in the `/etc/dhcp/inittab` file is used by `dhcpinfo` to provide more meaningful information to human readers of the information. This file replaces the `/etc/dhcp/dhcptags` file. "DHCP Option Information" on page 291 provides more information about this replacement.

DHCP Option Information

Historically, DHCP option information has been stored in several places in Solaris DHCP, including the server's `dhcptab` file, the client's `dhcptags` file, and internal tables of `in.dhcpd`, `snoop`, `dhcpinfo`, and `dhcpmgr`. In an effort to begin

consolidating option information, the Solaris 8 DHCP product has introduced the `/etc/dhcp/inittab` file. See `dhcp_inittab`(4) for detailed information about the file.

The `inittab` file is currently used on the Solaris DHCP client as a replacement for the `dhcptags` file. The `dhcptags` file was used to obtain information about option codes received in its DHCP packet. The `inittab` file now provides this information. In future releases, the `in.dhcpd`, `snoop`, and `dhcpmgr` programs on the DHCP server might use this file as well.

Note - Most sites using Solaris DHCP are *not* affected by this change. Your site is affected only if you are upgrading to Solaris 8, you previously created new DHCP options and modified the `/etc/dhcp/dhcptags` file, and want to retain the changes. When you upgrade, the upgrade log will notify you that your `dhcptags` file was modified and that you should make changes to the `inittab` file.

Differences Between `dhcptags` and `inittab`

The `inittab` file contains more information than the `dhcptags` file, and the syntax used is different.

A sample `dhcptags` entry is:

```
33 StaticRt - IPList Static_Routes
```

where `33` is the numeric code that is passed in the DHCP packet, `StaticRt` is the option name, `IPList` indicates the expected data is a list of IP addresses, and `Static_Routes` is a more descriptive name.

The `inittab` file consists of one-line records describing each option. The format is similar to the format for defining symbols in `dhcptab`. The syntax of the `inittab` is described in the following table.

TABLE 13–3 DHCP `inittab` Syntax

Option	Description
option-name	Name of the option. The option name must be unique within its option category, and not overlap with other option names in the Standard, Site, and Vendor categories. For example, you cannot have two Site options with the same name, and you should not create a Site option with the same name as a Standard option.
category	Identifies the namespace in which the option belongs. Must be one of Standard, Site, Vendor, Field, or Internal.

TABLE 13–3 DHCP `inittab` Syntax *(continued)*

Option	Description
code	Identifies the option when it is sent over the network. In most cases, the code uniquely identifies the option, without a category. However, in the case of internal categories like Field or Internal, a code might be used for other purposes and thus might not be globally unique. The code should be unique within the option's category, and not overlap with codes in the Standard and Site fields.
type	Describes the data associated with this option. Valid types are IP, Ascii, Octet, Number, Bool, Unumber8, Unumber16, Unumber32, Snumber8, Snumber16, and Snumber32. For numbers, a preceding 'U' or 'S' indicates whether the number is unsigned or signed, and the trailing number indicates the number of bits in the number.
granularity	Describes how many units of data make up a whole value for this option. In the case of Number, granularity describes the number of bytes in the number.
maximum	Describes how many whole values are allowed for this option. 0 indicates an infinite number.
consumers	Describes which programs can use this information. This should be set to `sdmi`, where: s – `snoop` d – `in.dhcpd` m – `dhcpmgr` i – `dhcpinfo`

A sample `inittab` entry is:

```
StaticRt Standard, 33, IP, 2, 0, sdmi
```

This entry describes an option named `StaticRt`, which is in the Standard category and is option code 33. The expected data is a potentially infinite number of pairs of IP addresses because the type is `IP`, granularity is `2`, and maximum is infinite (`0`). The consumers of this option are `sdmi`: `snoop`, `in.dhcpd`, `dhcpmgr`, and `dhcpinfo`.

Converting `dhcptags` Entries to `inittab` Entries

If you previously added entries to your `dhcptags` file, you must add corresponding entries to the new `inittab` file. The following example shows how a sample `dhcptags` entry might be expressed in `inittab` format.

Suppose you had added the following `dhcptags` entry for fax machines connected to the network:

```
128 FaxMchn - IP  Fax_Machine
```

The code `128` means that it must be in the site category, the option name is `FaxMchn`, the data type is `IP`.

The corresponding `inittab` entry might be:

```
FaxMchn SITE, 128, IP, 1, 1, sdmi
```

The granularity of 1 and maximum of 1 indicate that one IP address is expected for this option.

ARP Assignments for Network Hardware

The ARP (Address Resolution Protocol) assignment for the network hardware is specified at the beginning of a client ID in DHCP network tables. Use the following table to determine what code to use in your client ID.

The codes are assigned by the Internet Assigned Numbers Authority (IANA) in the ARP registrations section of the Assigned Numbers standard at `http://www.iana.com/numbers.html`.

TABLE 13–4 ARP Types

Code	Hardware Type
1	Ethernet (10Mb)
2	Experimental Ethernet (3Mb)
3	Amateur Radio AX.25
4	Proteon ProNET Token Ring
5	Chaos
6	IEEE 802
7	ARCNET
8	Hyperchannel
9	Lanstar

TABLE 13–4 ARP Types *(continued)*

Code	Hardware Type
10	Autonet Short Address
11	LocalTalk
12	LocalNet (IBM PCNet or SYTEK LocalNET)
13	Ultra link
14	SMDS
15	Frame Relay
16	Asynchronous Transmission Mode (ATM)
17	HDLC
18	Fibre Channel
19	Asynchronous Transmission Mode (ATM)
20	Serial line
21	Asynchronous Transmission Mode (ATM)
22	MIL-STD-188-220
23	Metricom
24	IEEE 1394.1995
25	MAPOS
26	Twinaxial
27	EUI-64
28	HIPARP

CHAPTER 14

Overview of IPv6

The Internet Protocol, version 6 (IPv6), is a new version of Internet Protocol (IP) designed to be an evolutionary step from the current version, IPv4. It is a natural increment to IPv4. Deploying IPv6, using defined transition mechanisms, does not disrupt current operations. IPv6 adds increased address space and improves Internet functionality using a simplified header format, support for authentication and privacy, autoconfiguration of address assignments, and new quality-of-service capabilities.

IPv6 Features

The changes between IPv4 to IPv6 fall primarily into the following categories:

- **Expanded routing and addressing capabilities** – IPv6 increases the IP address size from 32 bits to 128 bits to support more levels of addressing hierarchy, provide a much greater number of addressable nodes, and employ simpler auto-configuration of addresses.

The addition of a *scope* field improves the scalability of multicast routing to multicast addresses.

A new type of address, called an *anycast* address, is defined. It identifies sets of nodes, where a packet sent to an anycast address is delivered to one of the nodes. The use of anycast addresses in the IPv6 source route allows nodes to control the path over which their traffic flows.

- **Header format simplification** – Some IPv4 header fields have been dropped or made optional. This reduces the common-case processing cost of packet handling and keeps the bandwidth cost of the IPv6 header as low as possible, despite the increased size of the addresses. Even though the IPv6 addresses are four times longer than the IPv4 addresses, the IPv6 header is only twice the size of the IPv4 header.
- **Improved support for options** – Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.
- **Quality-of-service capabilities** – A new capability is added to enable the labeling of packets belonging to particular traffic *flows* for which the sender requests special handling, such as non-default quality of service or *real-time* service.
- **Authentication and privacy capabilities** – IPv6 includes the definition of extensions that provide support for authentication, data integrity, and confidentiality.

IPv6 Header and Extensions

The IPv6 protocol defines a set of headers, including the basic IPv6 header and the IPv6 extension headers.

Header Format

The following figure shows the elements that appear in the IPv6 header and the order in which they appear.

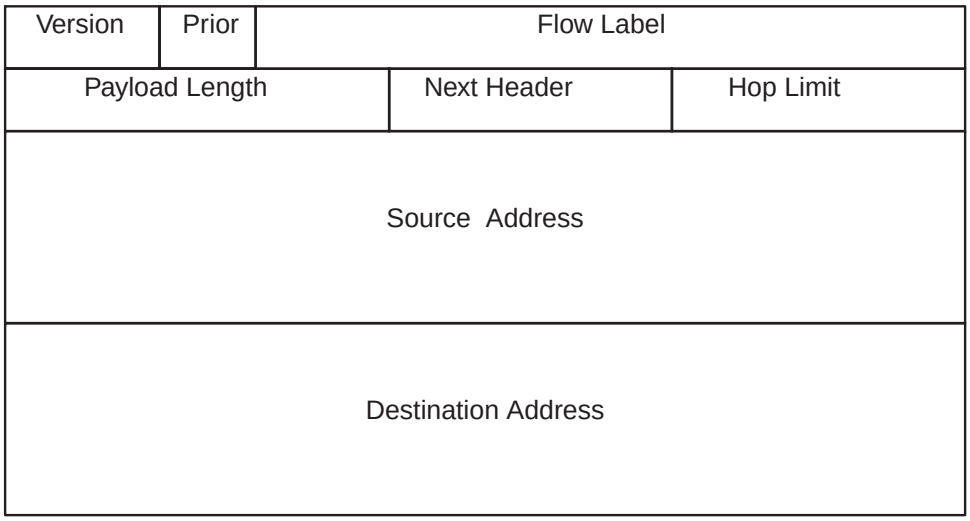


Figure 14–1 IPv6 Header Format

The following list describes the function of each header field.

- **Version** – 4-bit Internet protocol version number = 6
- **Priority** – 4-bit priority value (see “Priority” on page 318)
- **Flow Label** – 24-bit field (see “IPv6 Quality-of-Service Capabilities” on page 316)
- **Payload Length** – 16-bit unsigned integer, which is the rest of the packet following the IPv6 header, in octets
- **Next Header** – 8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 protocol field (see “Extension Headers” on page 299).
- **Hop Limit** – 8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero.
- **Source Address** – 128 bits. The address of the initial sender of the packet (see “IPv6 Addressing” on page 300).
- **Destination Address** – 128 bits. The address of the intended recipient of the packet (not necessarily the recipient if an optional Routing Header is present)

Extension Headers

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

containing options. In IPv4, the presence of any options requires the router to examine all options.

The other improvement is that, unlike IPv4 options, IPv6 extension headers can be of arbitrary length and the total amount of options carried in a packet is not limited to 40 bytes. This feature, plus the manner in which they are processed, permits IPv6 options to be used for functions that were not practical in IPv4. A good example of this is the IPv6 authentication and security encapsulation options.

To improve performance when handling subsequent option headers (and the transport protocol that follows), IPv6 options are always an integer multiple of 8 octets long so that alignment of subsequent headers is retained.

The following IPv6 extension headers are currently defined.

- **Routing** – Extended routing (like IPv4 loose source route)
- **Fragmentation** – Fragmentation and reassembly
- **Authentication** – Integrity and authentication; security
- **Encapsulation** – Confidentiality
- **Hop-by-Hop Option** – Special options that require hop-by-hop processing
- **Destination Options** – Optional information to be examined by the destination node

IPv6 Addressing

IPv6 addresses are 128-bits long and are identifiers for individual interfaces and sets of interfaces. IPv6 addresses of all types are assigned to interfaces, not nodes (hosts and routers). Because each interface belongs to a single node, any of that node's interfaces' unicast addresses can be used as an identifier for the node. A single interface can be assigned multiple IPv6 addresses of any type.

The three types of IPv6 addresses are: unicast, anycast, and multicast.

- Unicast addresses identify a single interface.
- Anycast addresses identify a set of interfaces in such a way that a packet sent to an anycast address is delivered to a member of the set.
- Multicast addresses identify a group of interfaces in such a way that a packet sent to a multicast address is delivered to all of the interfaces in the group.

IPv6 has no broadcast addresses: multicast addresses took over.

IPv6 supports addresses that are four times the number of bits as IPv4 addresses (128 vs. 32). This is 4 billion times 4 billion times the size of the IPv4 address space. Realistically, the assignment and routing of addresses requires the creation of hierarchies that reduce the efficiency of address space usage, thus reducing the

number of available addresses. Nonetheless, IPv6 provides enough address space to last into the foreseeable future.

The leading bits in the address specify the type of IPv6 address. The variable-length field containing these leading bits is called the format prefix (FP). The following table shows the initial allocation of these prefixes.

TABLE 14–1 Format Prefix Allocations

Allocation	**Prefix (binary)**	**Fraction of Address Space**
Reserved	0000 0000	1/256
Unassigned	0000 0001	1/256
Reserved for NSAP Allocation	0000 001	1/128
Reserved for IPX Allocation	0000 010	1/128
Unassigned	0000 011	1/128
Unassigned	0000 1	1/32
Unassigned	0001	1/16
Aggregate Global Unicast Address	001	1/8
Unassigned	010	1/8
Unassigned	011	1/8
Reserved for Neutral-Interconnect-Based Unicast Addresses	100	1/8
Unassigned	101	1/8
Unassigned	110	1/8
Unassigned	1110	1/16
Unassigned	1111 0	1/32
Unassigned	1111 10	1/64
Unassigned	1111 110	1/128
Unassigned	1111 1110 0	1/512

TABLE 14–1 Format Prefix Allocations *(continued)*

Allocation	Prefix (binary)	Fraction of Address Space
Link Local Use Addresses	1111 1110 10	1/1024
Site Local Use Addresses	1111 1110 11	1/1024
Multicast Addresses	1111 1111	1/256

The allocations support the direct allocation of aggregate global unicast addresses, local-use addresses, and multicast addresses. Space is reserved for NSAP (Network Service Access Point) addresses, IPX (Internetwork Packet Exchange Protocol) addresses, and neutral-interconnect addresses. The remainder of the address space is unassigned for future use. This remaining address space can be used for expansion of existing use (for example, additional aggregate global unicast addresses) or new uses (for example, separate locators and identifiers). Notice that anycast addresses are not shown here because they are allocated out of the unicast address space.

Approximately fifteen percent of the address space is initially allocated. The remaining 85% is reserved for future use.

Unicast Addresses

IPv6 unicast address assignment consists of the following forms:

- Aggregate global unicast address
- Neutral-interconnect unicast address
- NSAP address
- IPX hierarchical address
- Site-local-use address
- Link-local-use address
- IPv4-capable host address

Additional address types can be defined in the future.

Aggregate Global Unicast Addresses

Aggregate global unicast addresses are used for global communication. They are similar in function to IPv4 addresses under CIDR (classless interdomain routing). The following table shows their format.

TABLE 14–2 Aggregate Global Unicast Addresses Format

3 bits	13 bits	8 bits	24 bits	16 bits	64 bits
FP	TLA ID	RES	NLA ID	SLA ID	Interface ID

Where:

FP	Format Prefix (001)
TLA ID	Top-Level Aggregation identifier
RES	Reserved for future use
NLA ID	Next-Level Aggregation identifier
SLA ID	Site-Level Aggregation identifier
INTERFACE ID	Interface identifier

The first 48 bits represent the public topology. The next 16 bits represent the site topology.

The first 3 bits identify the address as an aggregate global unicast address. The next field, TLA ID, is the top level in the routing hierarchy. The next 8 bits are reserved for future use. The NLA ID field is used by organizations assigned a TLA ID to create an addressing hierarchy and to identify sites.

The SLA ID field is used by an individual organization to create its own local addressing hierarchy and to identify subnets. This is analogous to subnets in IPv4 except that each organization has a much greater number of subnets. The 16 bit SLA ID field supports 65,535 individual subnets. The Interface ID is used to identify interfaces on a link. They are required to be unique on that link. They can also be unique over a broader scope. In many cases, an interface identifier is the same or is based on the interface's link-layer address.

Local-Use Addresses

A local-use address is a unicast address that has only local routability scope (within the subnet or within a subscriber network), and can have a local or global uniqueness scope. These addresses are intended for use inside of a site for *plug and play* local communication and for bootstrapping up to the use of global addresses.

Two types of local-use unicast addresses are defined. These are link-local and site-local. The Link-Local-Use is for use on a single link and the Site-Local-Use is for use on a single site. The following table shows the Link-Local-Use address format.

TABLE 14–3 Link-Local-Use Addresses Format

10 bits	*n* bits	118-*n* bits
1111111010	0	Interface ID

Link-Local-Use addresses are used for addressing on a single link for purposes such as auto-address configuration.

The following table shows the Site-Local-Use address format.

TABLE 14–4 Site-Local-Use Addresses

10 bits	*n* bits	*m* bits	118-(*n*+*m*) bits
1111111011	0	Subnet ID	Interface ID

For both types of local-use addresses, the interface ID is an identifier that must be unique in the domain in which it is being used. In most cases these will use a node's IEEE-802 48 bit address. The Subnet ID identifies a specific subnet in a site. The combination of the Subnet ID and the interface ID to form a local-use address allows a large private internet to be constructed without any other address allocation.

Local-use addresses allow organizations that are not yet connected to the global Internet to operate without the need to request an address prefix from the global Internet address space. If the organization later connects to the global Internet, it can use its Subnet ID and Interface ID in combination with a global prefix (for example, Registry ID + Provider ID + Subscriber ID) to create a global address. This is a significant improvement over IPv4, which requires sites that use private (non-global) IPv4 addresses to manually renumber when they connect to the Internet. IPv6 automatically does the renumbering.

IPv6 Addresses With Embedded IPv4 Addresses

The IPv6 transition mechanisms include a technique for hosts and routers to tunnel IPv6 packets dynamically under IPv4 routing infrastructure. IPv6 nodes that utilize this technique are assigned special IPv6 unicast addresses that carry an IPv4 address

in the low-order 32-bits. This type of address is called an *IPv4-compatible IPv6 address* and its format is shown in the following table.

TABLE 14–5 IPv4–compatible IPv6 Address Format

80 bits	16 bits	32 bits
0000.....0000	0000	IPv4 Address

A second type of IPv6 address that holds an embedded IPv4 address is also defined. This address is used to represent an IPv4 address within the IPv6 address space. It is mainly used internally within the implementation of applications, APIs, and the operating system. This type of address is called an *IPv4-mapped IPv6 address* and its format is shown in the following table.

TABLE 14–6 IPv4–mapped IPv6 Address Format

80 bits	16 bits	32 bits
0000.....0000	FFFF	IPv4 Address

Anycast Addresses

An IPv6 anycast address is an address that is assigned to more than one interface (typically belonging to different nodes), where a packet sent to an anycast address is routed to the *nearest* interface having that address, according to the routing protocol's measure of distance.

Anycast addresses, when used as part of a route sequence, permit a node to select which of several Internet service providers it wants to carry its traffic. This capability is sometimes called *source selected policies*. You implement this by configuring anycast addresses to identify the set of routers belonging to Internet service providers (for example, one anycast address per Internet service provider). You can use these anycast addresses as intermediate addresses in an IPv6 routing header, to cause a packet to be delivered by a particular provider or sequence of providers. You can also use anycast addresses to identify the set of routers attached to a particular subnet or the set of routers providing entry into a particular routing domain.

You can locate anycast addresses from the unicast address space by using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. When you assign a unicast address to more than one interface, that is, turning it into an anycast address, you must explicitly

configure the nodes to which the address is assigned in order to know that it is an anycast address.

Multicast Addresses

An IPv6 multicast address is an identifier for a group of interfaces. An interface can belong to any number of multicast groups. The following table shows the multicast address format.

TABLE 14–7 Multicast Address Format

8 bits	4 bits	4 bits	112 bits
11111111	FLGS	SCOP	Group ID

11111111 at the start of the address identifies the address as a multicast address. FLGS is a set of 4 flags: 0,0,0,T.

The high-order 3 flags are reserved and must be initialized to 0.

- **T=0** – Indicates a permanently assigned (*well-known*) multicast address, assigned by the global Internet numbering authority.
- **T=1** – Indicates a non-permanently assigned (*transient*) multicast address.

SCOP is a 4-bit multicast scope value used to limit the scope of the multicast group. The following table shows the SCOP values.

TABLE 14–8 SCOP Values

0	Reserved	8	Organization-local scope
1	Node-local scope	9	(unassigned)
2	Link-local scope	A	(unassigned)
3	(unassigned)	B	(unassigned)
4	(unassigned)	C	(unassigned)
5	Site-local scope	D	(unassigned)
6	(unassigned)	E	Global scope
7	(unassigned)	F	Reserved

TABLE 14–8 SCOP Values *(continued)*

Group ID identifies the multicast group, either permanent or transient, within the given scope.

IPv6 Routing

Routing in IPv6 is almost identical to IPv4 routing under CIDR, except that the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. With very straightforward extensions, all of IPv4's routing algorithms (such as OSPF, RIP, IDRP, IS-IS) can be used to route IPv6.

IPv6 also includes simple routing extensions that support powerful new routing functionality. These capabilities include:

- Provider selection (based on policy, performance, cost, and so on)
- Host mobility (route to current location)
- Auto-readdressing (route to new address)

You obtain the new routing functionality by creating sequences of IPv6 addresses using the IPv6 routing option. An IPv6 source uses the routing option to list one or more intermediate nodes (or topological group) to be *visited* on the way to a packet's destination. This function is very similar in function to IPv4's loose source and record route option.

In order to make address sequences a general function, IPv6 hosts are required, in most cases, to reverse routes in a packet it receives (if the packet was successfully authenticated using the IPv6 authentication header) containing address sequences in order to return the packet to its originator. This approach makes IPv6 host implementations support the handling and reversal of source routes. This is the key that allows them to work with hosts that implement the new features, such as provider selection or extended addresses.

IPv6 Neighbor Discovery

IPv6 solves a set of problems related to the interaction between nodes attached to the same link. It defines mechanisms for solving each of the following problems.

- **Router discovery** – Hosts locate routers that reside on an attached link.

- **Prefix discovery** – Hosts discover the set of address prefixes that define which destinations are attached to the link (sometimes referred to as on-link). (Nodes use prefixes to distinguish destinations that reside on a link from those only reachable through a router.)
- **Parameter discovery** – A node learns link parameters, such as the link MTU (maximum transmission unit), or Internet parameters, such as the hop limit value, to place in outgoing packets.
- **Address autoconfiguration** – Nodes automatically configure an address for an interface.
- **Address resolution** – Nodes determine the link-layer address of a neighbor (an on-link destination) given only the destinations's IP address.
- **Next-hop determination** – Algorithm determines mapping for an IP destination address into the IP address of the neighbor to which traffic for the destination should be sent. The next-hop can be a router or the destination itself.
- **Neighbor unreachability detection** – Nodes determine that a neighbor is no longer reachable. For neighbors used as routers, alternate default routers can be tried. For both routers and hosts, address resolution can be performed again.
- **Duplicate address detection** – Node determines that an address it wants to use is not already in use by another node.
- **Redirect** – A router informs a host of a better first-hop node to reach a particular destination.

Neighbor discovery defines five different ICMP (Internet Control Message Protocol) packet types: a pair of router solicitation and router advertisement messages, a pair of neighbor solicitation and neighbor advertisements messages, and a redirect message. The messages serve the following purpose:

- **Router solicitation** – When an interface becomes enabled, hosts can send out router solicitations that request routers to generate router advertisements immediately, rather than at their next scheduled time.
- **Router advertisement** – Routers advertise their presence together with various link and Internet parameters, either periodically, or in response to a router solicitation message. Router advertisements contain prefixes that are used for on-link determination or address configuration, a suggested hop limit value, and so on.
- **Neighbor solicitation** – Sent by a node to determine the link-layer address of a neighbor, or to verify that a neighbor is still reachable by a cached link-layer address. Neighbor solicitations are also used for duplicate address detection.
- **Neighbor advertisement** – A response to a Neighbor Solicitation message, node can also send unsolicited neighbor advertisements to announce a link-layer address change.
- **Redirect** – Used by routers to inform hosts of a better first hop for a destination, or that the destination is on-link.

Router Advertisement

On multicast-capable links and point-to-point links, each router periodically multicasts a router advertisement packet announcing its availability. A host receives router advertisements from all routers, building a list of default routers. Routers generate router advertisements frequently enough that hosts will learn of their presence within a few minutes, but not frequently enough to rely on an absence of advertisements to detect router failure; a separate neighbor unreachability detection algorithm provides failure detection.

Router Advertisement Prefixes

Router advertisements contain a list of prefixes used for on-link determination or autonomous address configuration; flags associated with the prefixes specify the intended uses of a particular prefix. Hosts use the advertised on-link prefixes to build and maintain a list that is used in deciding when a packet's destination is on-link or beyond a router. A destination can be on-link even though it is not covered by any advertised on-link prefix. In such cases a router can send a redirect informing the sender that the destination is a neighbor.

Router advertisements (and per-prefix flags) allow routers to inform hosts how to perform address autoconfiguration. For example, routers can specify whether hosts should use stateful (DHCPv6) or autonomous (stateless) address configuration.

Router Advertisement Messages

Router advertisement messages also contain Internet parameters, such as the hop limit that hosts should use in outgoing packets and, optionally, link parameters such as the link MTU. This facilitates centralized administration of critical parameters that can be set on routers and automatically propagated to all attached hosts.

Nodes accomplish address resolution by multicasting a neighbor solicitation that asks the target node to return its link-layer address. Neighbor solicitation messages are multicast to the solicited-node multicast address of the target address. The target returns its link-layer address in a unicast neighbor advertisement message. A single request-response pair of packets is sufficient for both the initiator and the target to resolve each other's link-layer addresses; the initiator includes its link-layer address in the neighbor solicitation.

Neighbor Solicitation and Unreachability

Neighbor solicitation messages can also be used to determine if more than one node has been assigned the same unicast address.

Neighbor unreachability detection detects the failure of a neighbor or the failure of the forward path to the neighbor. Doing so requires positive confirmation that packets sent to a neighbor are actually reaching that neighbor and being processed properly by its IP layer. Neighbor unreachability detection uses confirmation from two sources. When possible, upper-layer protocols provide a positive confirmation that a connection is making *forward progress*, that is, previously sent data is known to have been delivered correctly (for example, new TCP acknowledgments were received recently). When positive confirmation is not forthcoming through such hints, a node sends unicast neighbor solicitation messages that solicit neighbor advertisements as reachability confirmation from the next hop. To reduce unnecessary network traffic, probe messages are sent only to neighbors to which the node is actively sending packets.

In addition to addressing the above general problems, neighbor discovery also handles the following situations.

- **Link-layer address change** – A node that knows its link-layer address has been changed can multicast a few (unsolicited) neighbor advertisement packets to all nodes to update cached link-layer addresses that have become invalid. The sending of unsolicited advertisements is a performance enhancement only. The neighbor unreachability detection algorithm ensures that all nodes will reliably discover the new address, though the delay might be somewhat longer.
- **Inbound load balancing** – Nodes with replicated interfaces might want to load-balance the reception of incoming packets across multiple network interfaces on the same link. Such nodes have multiple link-layer addresses assigned to the same interface. For example, a single network driver could represent multiple network interface cards as a single logical interface having multiple link-layer addresses.

  Load balancing is handled by allowing routers to omit the source link-layer address from router advertisement packets, thereby forcing neighbors to use neighbor solicitation messages to learn link-layer addresses of routers. Returned neighbor advertisement messages can then contain link-layer addresses that differ, depending on who issued the solicitation.
- **Anycast addresses** – Anycast addresses identify one of a set of nodes providing an equivalent service, and multiple nodes on the same link can be configured to recognize the same anycast address. Neighbor discovery handles anycasts by setting nodes to expect to receive multiple neighbor advertisements for the same target. All advertisements for anycast addresses are tagged as being non-override advertisements. This invokes specific rules to determine which of potentially multiple advertisements should be used.
- **Proxy advertisements** – A router willing to accept packets on behalf of a target address that is unable to respond to neighbor solicitations can issue non-override neighbor advertisements. There is currently no specified use of proxy, but proxy advertising could potentially be used to handle cases like mobile nodes that have moved off-link. However, it is not intended as a general mechanism to handle nodes that, for example, do not implement this protocol.

Comparison With IPv4

The IPv6 neighbor discovery protocol corresponds to a combination of the IPv4 protocols ARP (Adress Resolution Protocol), ICMP Router Discovery, and ICMP Redirect. In IPv4 there is no generally agreed upon protocol or mechanism for neighbor unreachability detection, although host requirements do specify some possible algorithms for dead gateway detection (a subset of the problems that neighbor unreachability detection tackles).

The neighbor discovery protocol provides a multitude of improvements over the IPv4 set of protocols.

- Router discovery is part of the base protocol set; there is no need for hosts to *snoop* the routing protocols.
- Router advertisements carry link-layer addresses; no additional packet exchange is needed to resolve the router's link-layer address.
- Router advertisements carry prefixes for a link; there is no need to have a separate mechanism to configure the *netmask*.
- Router advertisements enable address autoconfiguration.
- Routers can advertise an MTU for hosts to use on the link, ensuring that all nodes use the same MTU value on links lacking a well-defined MTU.
- Address resolution multicasts are spread over 4 billion (2^32) multicast addresses, greatly reducing address-resolution-related interrupts on nodes other than the target. Moreover, non-IPv6 machines should not be interrupted at all.
- Redirects contain the link-layer address of the new first hop; separate address resolution is not needed upon receiving a redirect.
- Multiple prefixes can be associated with the same link. By default, hosts learn all on-link prefixes from router advertisements. However, routers can be configured to omit some or all prefixes from router advertisements. In such cases, hosts assume that destinations are off-link and send traffic to routers. A router can then issue redirects as appropriate.
- Unlike IPv4, the recipient of an IPv6 redirect assumes that the new next-hop is on-link. In IPv4, a host ignores redirects specifying a next-hop that is not on-link, according to the link's network mask. The IPv6 redirect mechanism is analogous to the XRedirect facility. It is expected to be useful on non-broadcast and shared media links in which it is undesirable or not possible for nodes to know all prefixes for on-link destinations.
- Neighbor unreachability detection is part of the base significantly improving the robustness of packet delivery in the presence of failing routers, partially failing or partitioned links, and nodes that change their link-layer addresses. For instance, mobile nodes can move off-link without losing any connectivity due to stale ARP caches.
- Unlike ARP, neighbor discovery detects half-link failures (using neighbor unreachability detection) and avoids sending traffic to neighbors with which two-way connectivity is absent.

- Unlike in IPv4 router discovery, the router advertisement messages do not contain a preference field. The preference field is not needed to handle routers of different stability; the neighbor unreachability detection detect dead routers and switch to a working one.
- The use of link-local addresses to uniquely identify routers (for router advertisement and redirect messages) makes it possible for hosts to maintain the router associations in the event of the site renumbering to use new global prefixes.
- Because neighbor discovery messages have a hop limit of 255 upon receipt, the protocol is immune to spoofing attacks originating from off-link nodes. In contrast, IPv4 off-link nodes can send both ICMP (Internet Control Message Protocol) redirects and router advertisement messages.
- Placing address resolution at the ICMP layer makes the protocol more media-independent than ARP and makes it possible to use standard IP authentication and security mechanisms as appropriate.

IPv6 Stateless Address Autoconfiguration

A host takes several steps to decide how to autoconfigure its interfaces in IPv6. The autoconfiguration process includes creating a link-local address and verifying its uniqueness on a link, determining what information should be autoconfigured (addresses, other information, or both), and, in the case of addresses, whether they should be obtained through the stateless mechanism, the stateful mechanism, or both. This section describes the process for generating a link-local address, the process for generating site-local and global addresses by stateless address autoconfiguration, and the duplicate address detection procedure.

Stateless Autoconfiguration Requirements

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

Stateful Autoconfiguration Model

In the stateful autoconfiguration model, hosts obtain interface addresses or configuration information and parameters from a server. Servers maintain a database that keeps track of which addresses have been assigned to which hosts. The stateful autoconfiguration protocol allows hosts to obtain addresses, other configuration information, or both, from a server. Stateless and stateful autoconfiguration complement each other. For example, a host can use stateless autoconfiguration to configure its own addresses, but use stateful autoconfiguration to obtain other information.

When to Use Stateless and Stateful Approaches

The stateless approach is used when a site is not particularly concerned with the exact addresses hosts use, so long as they are unique and properly routable. The stateful approach is used when a site requires tighter control over exact address assignments. Both stateful and stateless address autoconfiguration can be used simultaneously. The site administrator specifies which type of autoconfiguration to use through the setting of appropriate fields in router advertisement messages.

IPv6 addresses are leased to an interface for a fixed (possibly infinite) length of time. Each address has an associated lifetime that indicates how long the address is bound to an interface. When a lifetime expires, the binding (and address) become invalid and the address can be reassigned to another interface elsewhere. To handle the expiration of address bindings gracefully, an address goes through two distinct phases while assigned to an interface. Initially, an address is preferred, meaning that its use in arbitrary communication is unrestricted. Later, an address becomes *deprecated* in anticipation that its current interface binding will become invalid. While in a deprecated state, the use of an address is discouraged, but not strictly forbidden. New communication (for example, the opening of a new TCP connection) should use a preferred address when possible. A deprecated address should be used only by applications that have been using it and would have difficulty switching to another address without a service disruption.

Duplicate Address Detection Algorithm

To ensure that all configured addresses are likely to be unique on a given link, nodes run a *duplicate address detection* algorithm on addresses before assigning them to an interface. The duplicate address detection algorithm is performed on all addresses, independent of whether they are obtained by stateless or stateful autoconfiguration.

The autoconfiguration process specified in this document applies only to hosts and not routers. Because host autoconfiguration uses information advertised by routers, routers need to be configured by some other means. However, it is expected that routers will generate link-local addresses using the mechanism described in this

document. In addition, routers are expected to pass successfully the duplicate address detection procedure on all addresses, prior to assigning them to an interface.

IPv6 Protocol Overview

This section provides an overview of the typical steps that take place when an interface autoconfigures itself. Autoconfiguration is performed only on multicast-capable links and begins when a multicast-capable interface is enabled, for example, during system startup. Nodes (both hosts and routers) begin the autoconfiguration process by generating a link-local address for the interface. A link-local address is formed by appending the interface's identifier to the well-known link-local prefix.

Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this tentative address is not already in use by another node on the link. Specifically, it sends a neighbor solicitation message containing the tentative address as the target. If another node is already using that address, it returns a neighbor advertisement saying so. If another node is also attempting to use the same address, it sends a neighbor solicitation for the target as well. The number of neighbor solicitation transmissions or retransmissions, and the delay between consecutive solicitations, are link-specific and can be set by system management.

If a node determines that its tentative link-local address is not unique, autoconfiguration stops and manual configuration of the interface is required. To simplify recovery in this case, it should be possible for an administrator to supply an alternate interface identifier that overrides the default identifier in such a way that the autoconfiguration mechanism can then be applied using the new (presumably unique) interface identifier. Alternatively, link-local and other addresses will need to be configured manually.

After a node ascertains that its tentative link-local address is unique, it assigns it to the interface. At this point, the node has IP-level connectivity with neighboring nodes. The remaining autoconfiguration steps are performed only by hosts.

Obtaining Router Advertisement

The next phase of autoconfiguration involves obtaining a router advertisement or determining that no routers are present. If routers are present, they send router advertisements that specify what sort of autoconfiguration a host should perform. If no routers are present, stateful autoconfiguration is invoked.

Routers send router advertisements periodically, but the delay between successive advertisements are generally longer than a host performing autoconfiguration wants to wait. To obtain an advertisement quickly, a host sends one or more router solicitations to the all-routers multicast group. Router advertisements contain two flags indicating what type of stateful autoconfiguration (if any) should be performed.

A *managed address configuration* flag indicates whether hosts should use stateful autoconfiguration to obtain addresses. An *other stateful configuration* flag indicates whether hosts should use stateful autoconfiguration to obtain additional information (excluding addresses).

Prefix Information

Router advertisements also contain zero or more prefix information options that contain information used by stateless address autoconfiguration to generate site-local and global addresses. It should be noted that the stateless and stateful address autoconfiguration fields in router advertisements are processed independently of one another, and a host can use both stateful and stateless address autoconfiguration simultaneously. One prefix information option field, the *autonomous address-configuration* flag, indicates whether or not the option even applies to stateless autoconfiguration. If it does, additional option fields contain a subnet prefix with lifetime values, indicating how long addresses created from the prefix remain preferred and valid.

Because routers generate router advertisements periodically, hosts continually receive new advertisements. Hosts process the information contained in each advertisement as described above, adding to and refreshing information received in previous advertisements.

Address Uniqueness

For safety, all addresses must be tested for uniqueness prior to their assignment to an interface. In the case of addresses created through stateless autoconfiguration, however, the uniqueness of an address is determined primarily by the portion of the address formed from an interface identifier. Thus, if a node has already verified the uniqueness of a link-local address, additional addresses created from the same interface identifier need not be tested individually. In contrast, all addresses obtained manually or by stateful address autoconfiguration should be tested individually for uniqueness. To accommodate sites that believe the overhead of performing duplicate address detection outweighs its benefits, the use of duplicate address detection can be disabled through the administrative setting of a per-interface configuration flag.

To speed up the autoconfiguration process, a host can generate its link-local address (and verify its uniqueness) in parallel with waiting for a router advertisement. Because a router might delay responding to a router solicitation for a few seconds, the total time needed to complete autoconfiguration can be significantly longer if the two steps are done serially.

IPv6 Mobility Support

Because routing is based on the subnet prefix in a packet's destination IP address, packets destined for a mobile node, host or router, do not reach the node when unattached to its home link (the link where its home IPv6 subnet prefix exists). In order to continue communication, regardless of a node's movement, a mobile node could change its IP address each time it moves to a new link. However, the mobile node would not maintain transport and higher-layer connections when it changes location. Consequently, IPv6 mobility support is particularly important when recognizing that mobile computers might account for a significant population of the Internet in the future.

IPv6 mobility support solves this problem. It enables a mobile node to move from one link to another without changing the mobile node's IP address. It accomplishes this through the assignment of an IP address to the mobile node within its home subnet prefix on its home link. This is known as the node's *home address.*

Thus, packets routed to the mobile node's *home address* reach their destination regardless of the mobile node's current point of attachment to the Internet. The mobile node can continue to communicate with other nodes (stationary or mobile) after moving to a new link.

Though IPv6 mobility support solves the problem of transparently routing packets to and from mobile nodes while away from home, it does not solve all the problems related to the use of mobile computers or wireless networks. In particular, it does not attempt to solve the following problems:

- Handling links with partial reachability, such as typical wireless networks (though a movement detection procedure addresses some aspect)
- Access control on a link being visited by a mobile node

IPv6 Quality-of-Service Capabilities

A host can use the flow label and the priority fields in the IPv6 header to identify those packets for which it requests special handling by IPv6 routers, such as non-default quality of service or real-time service. This important capability enables the support of applications that require some degree of consistent throughput, delay, or jitter. These types of applications are known as *multi-media* or *real-time* applications.

Flow Labels

A source can use the 24-bit flow label field in the IPv6 header to label those packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or real-time service. This aspect of IPv6 is still experimental and subject to change as the requirements for flow support in the Internet become clearer. Hosts or routers that do not support the functions of the flow label field are required to set the field to zero when originating a packet, forward the field unchanged when forwarding a packet, and ignore the field when receiving a packet.

What Is a Flow?

A flow is a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers. The nature of that special handling might be conveyed to the routers by a control protocol, such as a resource reservation protocol, or by information within the flow's packets themselves, for example, in a hop-by-hop option.

Active flows from a source to a destination can be multiple and can contain traffic that is not associated with any flow. The combination of a source address and a non-zero flow label uniquely identifies a flow. Packets that do not belong to a flow carry a flow label of zero.

The flow's source node assigns a flow label to a flow. New flow labels must be chosen (pseudo-)randomly and uniformly from the range 1 to FFFFFF hex. This random allocation makes any set of bits within the flow label field suitable for use as a hash key by routers for looking up the state associated with the flow.

Packets Belonging to the Same Flow

All packets belonging to the same flow must be sent with the same source address, same destination address, and same non-zero flow label. If any of those packets includes a hop-by-hop options header, then they must all be originated with the same hop-by-hop options header contents (excluding the next header field of the hop-by-hop options header). If any of those packets includes a routing header, then they must all be originated with the same contents in all extension headers up to and including the routing header (excluding the next header field in the routing header). The routers or destinations are permitted, but not required, to verify that these conditions are satisfied. If a violation is detected, it should be reported to the source by an ICMP parameter problem message, Code 0, pointing to the high-order octet of the flow label field (that is, offset 1 within the IPv6 packet).

Routers are free to *opportunistically* set up the flow-handling state for any flow, even when no explicit flow establishment information has been provided to them from a control protocol, a hop-by-hop option, or other means. For example, upon receiving a packet from a particular source with an unknown, non-zero flow label, a router can

process its IPv6 header and any necessary extension headers as if the flow label were zero. That processing would include determining the next-hop interface, and possibly other actions, such as updating a hop-by-hop option, advancing the pointer and addresses in a routing header, or deciding how to queue the packet based on its Priority field. The router can then choose to *remember* the results of those processing steps and cache that information, using the source address plus the flow label as the cache key. Subsequent packets with the same source address and flow label can then be handled by referring to the cached information rather than examining all those fields that, according to the requirements of the previous paragraph, can be assumed unchanged from the first packet seen in the flow.

Priority

The 4-bit priority field in the IPv6 header enables a source to identify the desired delivery priority of its packets, relative to other packets from the same source. The priority values are divided into the following two ranges:

- Values 0 through 7 specify the priority of traffic for which the source provides congestion control, that is, traffic that *backs off* in response to congestion, such as TCP traffic.
- Values 8 through 15 specify the priority of traffic that does not back off in response to congestion, for example, real-time packets being sent at a constant rate.

For congestion-controlled traffic, the priority values are recommended for particular application categories. The following table shows the priority values.

TABLE 14–9 Priority Values

Priority	**Meaning**
0	Uncharacterized traffic
1	"Filler" traffic (for example, netnews)
2	Unattended data transfer (for example, email)
3	(Reserved)
4	Attended bulk transfer (for example, FTP, HTTP).
5	(Reserved)
6	Interactive traffic (for example, telnet, X)
7	Internet control traffic (for example, routing protocols, SNMP)

TABLE 14–9 Priority Values *(continued)*

For non-congestion-controlled traffic, you should use the lowest priority value (8) for those packets that the sender is most willing to have discarded under conditions of congestion (for example, high-fidelity video traffic). You should use the highest value (15) for those packets that the sender is least willing to have discarded (for example, low-fidelity audio traffic). There is no relative ordering implied between the congestion-controlled priorities and the non-congestion-controlled priorities.

IPv6 Security Improvements

The current Internet has a number of security problems and lacks effective privacy and authentication mechanisms below the application layer. IPv6 remedies these shortcomings by having two integrated options that provide security services. You can use these two options either individually or together to provide differing levels of security to different users. Different user communities have different security needs.

The first option, and extension header called the *IPv6 Authentication Header*, provides authentication and integrity (without confidentiality) to IPv6 datagrams. While the extension is algorithm-independent and supports many different authentication techniques, the use of keyed MD5 is proposed to help ensure interoperability within the worldwide Internet. This eliminates a significant class of network attacks, including host masquerading attacks. When using source routing with IPv6, the IPv6 authentication header becomes important because of the known risks in IP source routing. Its placement at the Internet layer helps provide host origin authentication to those upper layer protocols and services that currently lack meaningful protections.

The second option, an extension header called the *IPv6 Encapsulating Security Header*, provides integrity and confidentiality to IPv6 datagrams. Though simpler than some similar security protocols (for example, SP3D, ISO NLSP), it remains flexible and algorithm-independent. To achieve interoperability within the global Internet, DES CBC is being used as the standard algorithm for use with the IPv6 Encapsulating Security Header.

The IPv6 Authentication Header and IPv6 Encapsulating Security Header are features of the new Internet Protocol Security (IPsec). For an overview of IPsec, see Chapter 18. For a description of how you implement IPsec, see Chapter 19.

CHAPTER 15

Transitioning From IPv4 to IPv6

As hosts and routers are upgraded to support IPv6, they must be able to interoperate over the network with the nodes (hosts and routers) that support only IPv4. This chapter provides an overview of the approach and the standardized solutions to transitioning from IPv4 to IPv6. RFC 1933 also provides detailed solutions to the transition problem.

Transition Requirements

The transition does not require any global coordination. Instead, it allows sites and Internet Service Providers (ISPs) to transition at their own pace. Furthermore, an effort has been made to minimize the number of dependencies during the transition. For instance, the transition does not require that routers be upgraded to IPv6 prior to upgrading hosts.

Different sites have different constraints when transitioning. Also, early adopters of IPv6 are likely to have different concerns than production users of IPv6. RFC 1933 defines the transition tools currently available. The driving force for transition is either the lack of IPv4 address space or required use of new features in IPv6, or both. The IPv6 specification requires 100% compatibility for the existing protocols and applications during the transition.

To understand the transition approaches, the following terms have been defined.

- **IPv4-only node** – A host or router that implements only IPv4. An IPv4-only node does not understand IPv6. The installed base of IPv4 hosts and routers existing before the transition begins are IPv4-only nodes.
- **IPv6/IPv4 node** – A host or router that implements both IPv4 and IPv6, also known as *dual stack*
- **IPv6-only node** – A host or router that implements IPv6, and does not implement IPv4
- **IPv6 node** – Any host or router that implements IPv6. IPv6/IPv4 and IPv6-only nodes are both IPv6 nodes.
- **IPv4 node** – Any host or router that implements IPv4. IPv6/IPv4 and IPv4-only nodes are both IPv4 nodes.
- **Site** – Piece of the private topology of the Internet, that is, topology that does not carry transit traffic for anybody and everybody. The site can span a large geographic area. For instance, the private network on a multinational corporation is one site.

Standardized Transition Tools

RFC 1933 defines the following transition mechanisms:

- When you upgrade your hosts and routers to IPv6, they retain their IPv4 functionality to provide compatibility for all IPv4 protocols and applications. These hosts and routers are known as *dual stack.*
- They use the name service (for example, DNS) to carry information about which nodes are IPv6 capable.
- IPv6 address formats can contain IPv4 addresses.
- You can tunnel IPv6 packets in IPv4 packets as a means of crossing routers that have not been upgraded to IPv6.

Implementing Dual Stack

The dual stack term normally refers to a complete duplication of all levels in the protocol stack from applications to the network layer. An example of this would be OSI and TCP/IP protocols running on the same machine. However, in the context of IPv6 transition, it means a protocol stack contains both IPv4 and IPv6, with the rest of the stack being identical. Consequently, the same transport protocols (TCP, UDP, and so on) and the same applications will run over both IPv4 and IPv6.

The following figure illustrates dual stack protocols through the OSI layers.

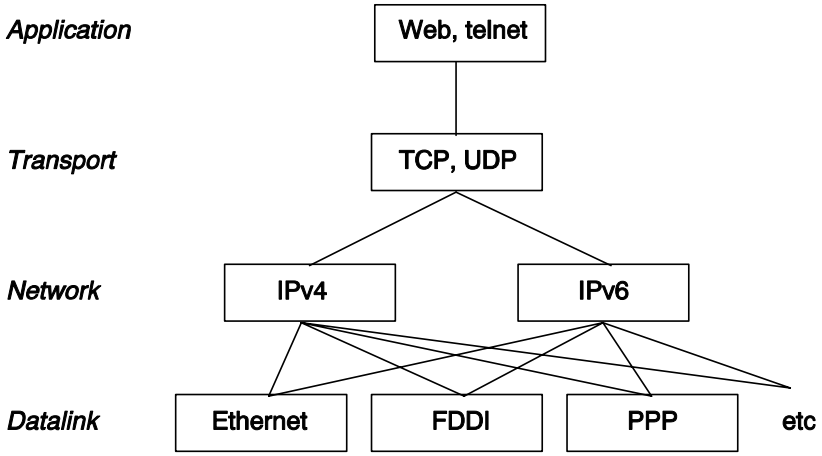


Figure 15–1 Dual Stack Protocols

In the dual stack approach, subsets of both hosts and routers are upgraded to support IPv6, in addition to IPv4. This ensures that the upgraded nodes can always interoperate with IPv4-only nodes by using IPv4. Thus, upgrading from IPv4 to dual stack does not break anything.

Configuring Name Services

A dual node must determine if the peer can support IPv6 or IPv4 in order to know which IP version to use when transmitting. Controlling what information goes in the name service accomplishes this. You define an IPv4 node's IP address and the IPv6 node's IP address in the name service. Thus, a dual node has both addresses in the name service.

However, the presence of an IPv6 address in the name service also signifies that the node is reachable, using IPv6 from all nodes that get information from that name service. For example, placing an IPv6 address in NIS implies that the IPv6 host is reachable using IPv6 from all IPv6 and dual nodes that belong to that NIS domain. Placing an IPv6 address in global DNS requires that the node is reachable from the Internet IPv6 *backbone*. This is no different than in IPv4 where, for example, mail delivery and HTTP proxy operation depend on there being only IPv4 addresses for nodes that can be reached using IPv4. When no reachability exists in IPv4, for instance, due to firewalls, the name service must be partitioned into an *inside firewall* and *outside firewall* database so that IPv4 addresses are visible only where they are reachable.

The protocol used to access the name service (DNS, NIS, NIS+, or something else) is independent of the type of address that can be retrieved from the name service. This name service support, coupled with dual stacks, allows a dual node to use IPv4 when communicating with IPv4-only nodes and use IPv6 when communicating with IPv6 nodes, provided that there is an IPv6 route to the destination.

Using IPv4 Compatible Address Formats

In many cases you can represent a 32-bit IPv4 address as an 128-bit IPv6 address. The transition mechanism defines the following two formats.

- **IPv4 compatible address**

000 ... 000	IPv4 Address

- **IPv4 mapped address**

000 ... 000	0xffff	IPv4 Address

The compatible format is used to represent an IPv6 node. This format enables you to configure an IPv6 node to use IPv6 without having a *real* IPv6 address. This address format lets you experiment with different IPv6 deployments because you can use automatic tunneling to cross IPv4–only routers. However, you cannot configure these addresses using the IPv6 stateless address autoconfiguration mechanism. This mechanism requires existing IPv4 mechanisms such as DHCPv4 or static configuration files.

The mapped address format is used to represent an IPv4 node. The only currently defined use of this address format is part of the socket API. It is convenient for an application to have a common address format for both IPv6 addresses and IPv4 addresses by representing an IPv4 address as a 128-bit mapped address. However, these addresses can also be used when there are IPv4 to IPv6 protocol translators.

Tunneling Mechanism

To minimize any dependencies during the transition, all the routers in the path between two IPv6 nodes do not need to support IPv6. This mechanism is called *tunneling*. Basically, IPv6 packets are placed inside IPv4 packets, which are routed through the IPv4 routers. The following figure illustrates the tunneling mechanism through routers (R) using IPv4.

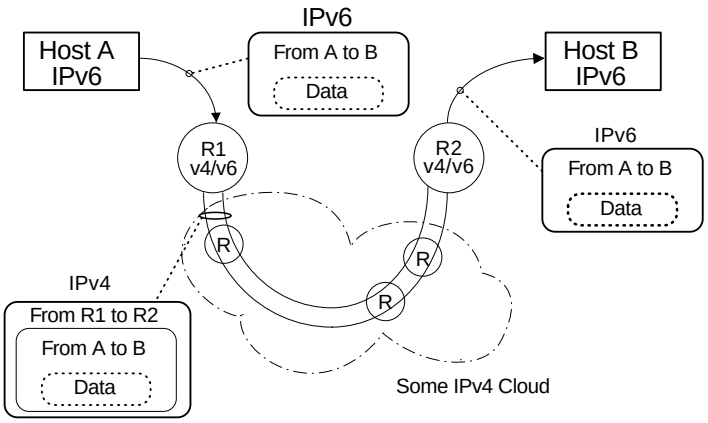


Figure 15–2 Tunneling Mechanism

Different uses of tunneling in the transition are:

- Configured tunnels between two routers (as in the figure above)
- Automatic tunnels that terminate at the dual hosts

A configured tunnel is currently used in the Internet for other purposes, for example, the MBONE (the IPv4 multicast backbone). Operationally, it consists of configuring two routers to have a virtual point-to-point link between them over the IPv4 network. This kind of tunnel is likely to be used on some parts of the Internet for the foreseeable future.

Automatic Tunnels

The automatic tunnels have a more limited use during early experimental deployment. They require IPv4 compatible addresses and can be used to connect IPv6 nodes when there are no IPv6 routers available. These tunnels can originate either on a dual host or on a dual router (by configuring an automatic tunneling network interface), and they always terminate on the dual host. These tunnels work by dynamically determining the destination IPv4 address (the endpoint of the tunnel) by extracting it from the IPv4 compatible destination address.

Interaction With Applications

Even on a node that has been upgraded to IPv6, the use of IPv6 is dependent on the applications. An application might not use a networking API that asks the name service for IPv6 addresses, either because the application uses an API (such as sockets) that requires changes in the application, or the provider of the API (such as an implementation of the `java.net` class) has no support for IPv6 addresses. In either case the node only sends and receives IPv4 packets like an IPv4 node.

The following names have become standard terminology within the Internet community:

- **IPv6–unaware**—This application cannot handle IPv6 addresses; that is, it cannot communicate with nodes that do not have an IPv4 address.
- **IPv6–aware**—This application can communicate with nodes that do not have an IPv4 address, that is, the application can handle the larger IPv6 addresses. In some cases this might be transparent to the application, for instance, when the API hides the content and format of the actual address.
- **IPv6–enabled**—This application can, in addition to being IPv6–aware, take advantage of some IPv6 specific feature such as flow labels. The enabled applications can still operate over IPv4, though in a degraded mode.
- **IPv6–required**—This application requires some IPv6 specific feature and cannot operate over IPv4.

IPv4 and IPv6 Interoperability

During the gradual transition phase from IPv4 to IPv6, existing IPv4 applications must continue to work with newer IPv6 enabled applications. Initially, vendors provide host and router platforms that are running a dual stack, that is, both an IPv4 protocol stack and an IPv6 protocol stack. IPv4 applications will continue to run on a dual stack that is also IPv6 enabled with at least one IPv6 interface. No changes need to be made to these applications (no porting required).

IPv6 applications running on a dual stack can also use the IPv4 protocol. This is possible by using an IPv4-mapped IPv6 address. Because of the design of IPv6, separate applications (IPv4 and IPv6) are not needed. For example, you do not need an IPv4 client on a dual host to talk with a server on an IPv4-only host and a separate IPv6 client to talk with an IPv6 server. Implementors only need to port their IPv4 client application to the new IPv6 API. The client can communicate with both IPv4 only servers as well as IPv6 servers running on either a dual host or an IPv6 only host.

The address the client gets back from name server determines if IPv6 or IPv4 is used. For example, if the name server has an IPv6 address for a server, this means that the server runs IPv6.

The following table summarizes the interoperability between IPv4 and IPv6 clients and servers. It also assumes that the dual-stack host has both an IPv4 and IPv6 address in the respective name service database.

TABLE 15–1 Client-Server Applications: IPv4 and IPv6 Interoperability

Type of Application (Type of Node)	**IPv6-unaware server (IPv4-only node)**	**IPv6-unaware server (IPv6-enabled node)**	**IPv6-aware server (IPv6-only node)**	**IPv6-aware server (IPv6-enabled node)**
IPv6-unaware client (IPv4-only node)	IPv4	IPv4	X	IPv4
IPv6-unaware client (IPv6-enabled node)	IPv4	IPv4	X	IPv4
IPv6-aware client (IPv6-only node)	X	X	IPv6	IPv6
IPv6-aware client (IPv6-enabled node)	IPv4	(IPv4)	IPv6	IPv6

X means that it is not possible to communicate between the respective server and client.

(IPv4) denotes that the interoperability depends on the address chosen by the client. Choosing an IPv6 address fails. However, choosing an IPv4 address, which is returned to the client as an IPv4–mapped IPv6 address, causes an IPv4 datagram to be sent and succeeds.

In the first phase of IPv6 deployment, most implementations of IPv6 are on dual-stack nodes. Initially, most vendors do not release IPv6–only implementations.

Site Transition Scenarios

Each site and ISP has different issues and requires different steps during the transition phase. This section provides some examples of site transition scenarios.

The first step in transitioning a site is to upgrade the name services to support IPv6 addresses. For DNS, this consists of upgrading to a DNS server that supports the new AAAA (quad-A) records such as BIND 4.9.4 and later. Two new NIS maps and a new NIS+ table have been introduced for storing IPv6 addresses. The new NIS maps and NIS+ table can be created and administered on any Solaris system. See “IPv6 Extensions to Solaris Name Services” on page 345 for details on the new databases.

After the name service is able to hand out IPv6 addresses, you can start transitioning hosts. You can transition hosts in the following ways:

- Upgrading one host at a time using IPv4 compatible addresses and automatic tunneling. No routers need to be upgraded. This can be used for initial *experimental* transition and offers only a subset of the IPv6 benefits. It does not offer stateless

address autoconfiguration or IP multicast. You can use this scenario to verify that applications work over IPv6 and that the application can use IPv6 IP layer security.

- Upgrading one subnet at a time using configured tunnels between the routers. In this scenario at least one router per subnet is upgraded to dual and the dual routers in the site are tied together using configured tunnels. Then hosts on those subnets can use all the IPv6 features. As more routers become upgraded in this incremental scheme, you can remove the configured tunnels.
- Upgrading all the routers to dual before any host is upgraded. Though this approach appears orderly, it does not provide any IPv6 benefits until all the routers have been upgraded. This scenario constrains the incremental deployment approach.

Other Transition Mechanisms

The mechanisms specified previously handle interoperability between dual nodes and IPv4 nodes, where the dual nodes have an IPv4 address. They do not handle interoperability between IPv6-only nodes (or dual nodes that have no IPv4 address) and IPv4-only nodes. While most implementations could be made dual (duality is only an issue of memory footprint for the code), the real issue is whether there will be enough IPv4 address space to assign one address for every node that needs to interoperate with IPv4-only nodes.

Several possibilities enable you to accomplish this interoperability without requiring any new transition mechanisms.

- Use application layer gateways (ALG) that sit at the boundary between the IPv6-only nodes and the rest of the Internet. Examples of ALGs in use today are HTTP proxies and mail relays.
- Companies are already selling NAT boxes (network address translators) for IPv4 that translate between the private IP addresses (for example, network 10—see RFC 1918) on the *inside* and other IP addresses on the *outside*. It is likely that these companies will upgrade their NAT boxes to also support IPv6 to IPv4 address translation.

Unfortunately, both ALG and NAT solutions create single points of failure. Using these result in a much less robust Internet. The IETF is working on a better solution for IPv6-only interoperability with IPv4-only nodes. One proposal is to use header translators with a way to allocate IPv4 compatible addresses on demand. Another proposal is to allocate IPv4 compatible addresses on demand and use IPv4 in IPv6 tunneling to bridge the IPv6-only routers.

The stateless header translator would translate between IPv4 and IPv6 header formats as long as the IPv6 addresses in use can be represented as IPv4 addresses (that is, they have to be IPv4-compatible or IPv4-mapped addresses). The support for these translators has been built into the IPv6 protocol so that, barring any encrypted

packets or rarely used features like source routing, the translation can occur without any information loss.

CHAPTER 16

Managing IPv6

The Solaris implementation of IPv6 consists primarily of changes to the TCP/IP stack, both at the kernel and user level. New modules were added to enable tunneling, router discovery, and stateless address autoconfiguration. This chapter describes the concepts associated with the Solaris implementation of IPv6.

Overview of the Solaris IPv6 Implementation

As a part of the IPv4 to IPv6 transition, IPv6 specifies methods for encapsulating IPv6 packets within IPv4 packets, as well as IPv6 packets encapsulated within IPv6 packets. Consequently, a new module, `tun`(7M), which performs the actual packet encapsulation, has been added. This module, known as the tunneling module, is plumbed and configured using the `ifconfig` utility the same as any physical

interface. It enables the tunneling module to be pushed between IP device and IP module. Tunneling devices also have entries in the system interface list.

The `ifconfig`(1M) utility was also modified to create the IPv6 stack and support new parameters, which are described in this chapter.

The `in.ndpd`(1M) daemon was added to perform router dicscovery and stateless address autoconfiguration.

IPv6 Network Interface Configuration File

IPv6 uses the file `/etc/hostname6.`*interface* at start-up to automatically define network interfaces in the same way IPv4 uses `/etc/hostname.`*interface*. At least one `/etc/hostname.*` or `/etc/hostname6.*` file should exist on the local machine. The Solaris installation program creates these files for you. In the file name, replace *interface* with the device name of the primary network interface.

The file name has the following syntax:

```
hostname.interface
hostname6.interface
```

Interface has the following syntax:

```
dev[.Module[.Module ...]]PPA
```

Dev	A network interface device. The device can be a physical network interface, such as `le`, `qe`, and so on, or a logical interface, such as a tunnel (see “Solaris Tunneling Interfaces for IPv6” on page 343 for more details).
Module	The list of one or more streams modules to be pushed onto the device when it is plumbed
PPA	The physical point of attachment

The syntax [.[.]] is also accepted.

Examples of valid file names include:

```
hostname6.le0
hostname6.ip.tun0
hostname.ip.tun0
```

IPv6 Interface Configuration File Entry

Since autoconfiguration of interfaces in IPv6 allows a node to compute its own link-local address based on its link-layer address, the IPv6 interface configuration file might not have an entry. In this case the startup scripts configure an interface. The node then learns of other addresses and prefixes through the neighbor discovery daemon, `in.ndpd`. If it is required for an interface to have static addresses (which is less common in IPv6), you can still add them using the command interface of the `ifconfig` utility. Consequently, the address or hostname is stored in `/etc/hostname6.`*interface* (or `/etc/hostname.`*interface*) and the content is passed to `ifconfig` when configuring an interface.

In this case, the file contains only one entry: the host name or IP address associated with the network interface. For example, suppose `smc0` is the primary network interface for a machine called `ahaggar`. Its `/etc/hostname6.*` file would have the name `/etc/hostname6.smc0` and the file would contain the entry `ahaggar`.

The networking start-up script examines the number of interfaces and the existence of the `/etc/inet/ndpd.conf` file to start routing daemons and packet forwarding (see “How to Configure a Solaris IPv6 Router” on page 352).

IPv6 Extensions to the `ifconfig` Utility

The `ifconfig` utility was changed to allow for plumbing IPv6 interfaces as well as the tunneling module. The `ifconfig`(1M) utility uses an extended set of ioctls to configure both IPv4 and IPv6 network interfaces. The following table shows the set of options added to this utility. See “How to Display Interface Address Assignments” on page 356 for a description of useful diagnostic procedures using this utility.

TABLE 16–1 New `ifconfig` Utility Options

Option	Description
`index`	Set the interface index.
`tsrc/tdst`	Set tunnel source/destination.

TABLE 16–1 New `ifconfig` Utility Options *(continued)*

Option	Description
`addif`	Create the next available logical interface.
`removeif`	Delete a logical interface with a given IP address.
`destination`	Set the point-to-point destination address for an interface.
`set`	Set an address, netmask, or both for an interface.
`subnet`	Set the subnet address of an interface.
`xmit`/`-xmit`	Enable/disable packet transmission on an interface.

"Enabling IPv6 Nodes" on page 350 provides IPv6 configuration procedures.

Examples—New `ifconfig` Utility Options

The following usage of the `ifconfig` command creates the `hme0:3` logical interface to the 1234::5678/64 IPv6 address and enables it with the `up` option, reports its status, disables it, then deletes the interface.

EXAMPLE 16–1 Examples—Using `addif` and `removeif`

```
# ifconfig hme0 inet6 addif 1234::5678/64 up
Created new logical interface hme0:3

# ifconfig hme0:3 inet6
hme0:3: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
  inet6 1234::5678/64

# ifconfig hme0:3 inet6 down

# ifconfig hme0 inet6 removeif 1234::5678
```

The following usage of the `ifconfig` command opens the device associated with the physical interface name and sets up the streams needed for TCP/IP to use the device, reports its status, configures the tunnels source and destination address, and reports its new status after the configuration.

EXAMPLE 16–2 Examples—Using `tsrc`/`tdst` and `index`

```
# ifconfig ip.tun0 inet6 plumb index 13

# ifconfig ip.tun0 inet6
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu
1480 index 13
  inet tunnel src 0.0.0.0
  inet6 fe80::/10 --> ::

# ifconfig ip.tun0 inet6 tsrc 120.46.86.158 tdst 120.46.86.122

# ifconfig ip.tun0 inet6
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu
1480 index 13
  inet tunnel src 120.46.86.158  tunnel dst 120.46.86.122
  inet6 fe80::8192:569e/10 --> fe80::8192:567a
```

Nodes With Multiple Network Interfaces

If a node contains more than one network interface, you must create additional `/etc/hostname.`*interface* files for the additional network interfaces.

IPv4 Behavior

For example, consider the machine `timbuktu`, shown in the Figure 6–1 figure. It has two network interfaces and functions as a router. The primary network interface `le0` is connected to network `192.9.200`. Its IP address is `192.9.200.70` and its host name is `timbuktu`. The Solaris installation program creates the file `/etc/hostname.le0` for the primary network interface and enters the host name `timbuktu` in the file.

The second network interface is `le1`; it is connected to network `192.9.201`. Although this interface is physically installed on machine `timbuktu`, it must have a separate IP address. Therefore, you have to manually create the `/etc/hostname.le1` file for this interface; the entry in the file is the router‘s name, `timbuktu-201`.

IPv6 Behavior

If IPv6 is to be configured, all that is necessary is that the interfaces for `/etc/hostname6.le0` and `/etc/hostname6.le1` exist. Each interface address is configured automatically when the system is started.

IPv6 Daemons

This section describes the following IPv6 daemons:

- `in.ndpd` – Daemon for IPv6 autoconfiguration
- `in.ripngd` – Network routing daemon for IPv6
- `inetd` – Internet services daemon

`in.ndpd` Daemon

This daemon implements router discovery and auto-address configuration for IPv6. The following table shows the supported options.

TABLE 16–2 `in.ndpd` Daemon Options

Option	Description
`–d`	Turns on debugging for all events
`–D`	Turns on specific debugging
`–f`	File to read configuration from (instead of default file)
`–I`	Prints per interface related information
`–n`	Does not loop back router advertisements
`–r`	Ignores received packets

TABLE 16–2 `in.ndpd` Daemon Options *(continued)*

Option	Description
`–v`	Verbose mode (reports various types of diagnostic messages)
`–t`	Turns on packet tracing

Parameters set in the `/etc/inet/ndpd.conf` configuration file and the `/var/inet/ndpd_state.`*interface* startup file (if they exist) control the actions of `in.ndpd`.

When `/etc/inet/ndpd.conf` exists, it is parsed and used to configure a node as a router. The following table lists the valid keywords that might appear in this file. When a host is booted, routers might not be immediately available, or advertised packets by the router might be dropped and not reach the host. The `/var/inet/ndpd_state.`*interface* file is a state file that is maintained and updated periodically by each node so that when it fails and is restarted it can configure its interfaces in the absence of routers. This file contains information such as: the interface address, the time it was updated, and how long it is valid, along with other parameters learned from previous router advertisements.

Note - You do not need to alter the contents of the state files. The `in.ndpd` daemon automatically maintains them.

TABLE 16–3 `/etc/inet/ndpd.conf` Keywords

Keywords	Description
`ifdefault`	Specifies router behavior for all interfaces. Use the following syntax to set router parameters and corresponding values: `ifdefault` [*variable value*]
`prefixdefault`	Specifies prefix advertisement default behavior. Use the following syntax to set router parameters and corresponding values: `prefixdefault` [*variable value*]

TABLE 16–3 `/etc/inet/ndpd.conf` Keywords *(continued)*

Keywords	Description
`if`	Sets per interface parameters. Use the following syntax: `if` *interface* [*variable value*]
`prefix`	Advertises per interface prefix information. Use the following syntax: `prefix` *prefix/length interface* [*variable value*]

Note - The `ifdefault`/`prefixdefault` entries must precede the `if` and `prefix` entries in the configuration file.

See the `in.ndpd`(1M) man page and also see the `ndpd.conf`(4) man page for a list of configuration variables and allowable values.

Example—`/etc/inet/ndpd.conf` File

The following example provides a template (commented lines) and also shows an example of how the keywords and configuration variables are used.

```
# ifdefault       [variable value]*
# prefixdefault [variable value]*
# if ifname    [variable value]*
# prefix prefix/length ifname
#
#  Per interface configuration variables
#
#DupAddrDetectTransmits
#AdvSendAdvertisements
#MaxRtrAdvInterval
#MinRtrAdvInterval
#AdvManagedFlag
#AdvOtherConfigFlag
#AdvLinkMTU
#AdvReachableTime
#AdvRetransTimer
#AdvCurHopLimit
#AdvDefaultLifetime
#
# Per Prefix:  AdvPrefixList configuration varialbes
#
#
#AdvValidLifetime
#AdvOnLinkFlag
#AdvPreferredLifetime
```

(continued)

```
#AdvAutonomousFlag
#AdvValidExpiration
#AdvPreferredExpiration

ifdefault AdvReachableTime 30000 AdvRetransTimer 2000
prefixdefault AdvValidLifetime 240m AdvPreferredLifetime 120m

if qe0 AdvSendAdvertisements 1
prefix 2:0:0:56::/64 qe0
prefix fec0:0:0:56::/64 qe0

if qe1 AdvSendAdvertisements 1
prefix 2:0:0:55::/64 qe1
prefix fec0:0:0:56::/64 qe1

if qe2 AdvSendAdvertisements 1
prefix 2:0:0:54::/64 qe2
prefix fec0:0:0:54::/64 qe2
```

in.ripngd Daemon

The in.ripngd daemon implements the RIPng routing protocol for IPv6 routers. RIPng defines the IPv6 equivalent of RIP, a widely used IPv4 routing protocol based on the Bellman-Ford distance vector algorithm. The following table shows the supported options.

TABLE 16–4 in.ripngd Daemon Options

Option	Description
–p *n*	*n* specifies the alternate port number used to send/receive RIPNG packets
–q	Suppresses routing information
–s	Forces routing information whether or not it is acting as a router
–P	Suppresses use of poison reverse
–S	If in.ripngd does not act as a router, it will enter only a default route for each router.

inetd Internet Services Daemon

An IPv6–enabled server is one that can handle IPv4 or IPv6 addresses depending on what the corresponding client is using. The /etc/inet/inetd.conf file contains the list of servers that inetd(1M) invokes when it receives an Internet request over a socket. Each socket-based Internet server entry is composed of a single line that uses the following syntax:

```
service_name  socket_type  proto  flags  user  server_pathname  args
```

See the inetd.conf(4) man page for a description of the possible values for each field. For the Solaris operating environment, to specify a service as IPv6-enabled in the /etc/inet/inetd.conf file, you must specify the *proto* field as tcp6 or udp6. If the service is IPv4-only, the *proto* field must be specified as tcp or udp. By specifying a *proto* value of tcp6 or udp6 for a service, inetd will pass the given daemon an AF_INET6 socket.

The following entry in the inetd.conf file depicts a udp server (myserver) that can communicate with both IPv4 and IPv6 client applications.

EXAMPLE 16–3 Server Communicating With Both IPv4 and IPv6 Client Applications

```
myserver   dgram   udp6 wait root /usr/sbin/myserver myserver
```

If an IPv6–enabled server is written in such a way that it can inherit an AF_INET (IPv4 only) or an AF_INET6 (IPv6 and IPv4) socket from inetd, the *proto* value for the service is specified as either tcp6 (udp6) or tcp (udp). For these type of servers you can also specify two inetd.conf entries: one with *proto* as tcp, and one with *proto* as tcp6.

Note - Because AF_INET6 sockets work with either the IPv4 or IPv6 protocols, specifying a *proto* value of tcp6 (udp6) is sufficient.

See *Network Interface Guide* for details on writing various types of IPv6–enabled servers.

All Solaris bundled servers require only one inetd entry that specifies *proto* as tcp6 or udp6. However, the remote shell server (shell) and the remote execution server (exec) must have an entry for both the tcp and tcp6 *proto* values. The following example shows the inetd entries for rlogin, telnet, shell, and exec.

EXAMPLE 16–4 inetd.conf Entries for Some Solaris Bundled Servers

```
login stream tcp6 nowait root /usr/sbin/in.rlogind in.rlogind
telnet stream tcp6 nowait root /usr/sbin/in.telnetd in.telnetd
shell stream tcp nowait root /usr/sbin/in.rshd in.rshd
```

(continued)

```
shell stream tcp6 nowait root /usr/sbin/in.rshd in.rshd
exec stream tcp nowait root /usr/sbin/in.rexecd in.rexecd
exec stream tcp6 nowait root /usr/sbin/in.rexecd in.rexecd
```

If you specify *TCP Wrappers* (a public domain utility used for monitoring and filtering incoming requests for various network services, such as telnet) as the *server_pathname* for any of these utilities, you must ensure that *TCP Wrappers* is IPv6 capable. Otherwise, you must specify *proto* as `tcp` or `udp` for those services that are being used with *TCP Wrappers*.

In addition, if you replace a bundled Solaris utility with another implementation, you must verify if the implementation of that service supports IPv6. If it does not, then you must specify the *proto* value as either `tcp` or `udp`.

Note - If you specify *proto* as `tcp` or `udp` only, the service uses only IPv4. You need to specify proto as `tcp6` or `udp6` to enable either IPv4 or IPv6 connections. If the service does not support IPv6, then do not specify `tcp6` or `udp6`.

See IPv6 extensions to the Socket API in *Network Interface Guide* for more details on writing IPv6 enabled servers using sockets.

IPv6 Extensions to Existing Utilities

User-level interface changes also include extensions to the following utilities:

- `netstat`(1M)
- `snoop`(1M)
- `route`(1M)
- `ping`(1M)
- `traceroute`(1M)

The `ifconfig`(1M) utility has also changed. See “IPv6 Extensions to the `ifconfig` Utility” on page 333 for a description.

`netstat`(1M)

In addition to displaying IPv4 network status, `netstat` can display IPv6 network status as well. You can choose what protocol information to display by setting the

DEFAULT_IP value in the /etc/default/inet_type file and the –f command-line option. With a permanent setting of DEFAULT_IP, you can make sure netstat displays only IPv4 information. You can override this setting with the –f option. For more information on the inet_type file, see the inet_type(4) man page.

The new –p option displays the net-to-media table, which is the ARP table for IPv4 and neighbor cache for IPv6. See netstat(1M) man page for details. See "How to Display Network Status" on page 357 for descriptions of procedures using this command.

snoop(1M)

The snoop command can capture both IPv4 and IPv6 packets. It can display IPv6 headers, IPv6 extension headers, ICMPv6 headers, and neighbor discovery protocol data. By default, the snoop command displays both IPv4 and IPv6 packets. By specifying the ip or ip6 protocol keywords, the snoop command displays only IPv4 or IPv6 packets. The IPv6 filter option enables you to filter through all packets (both IPv4 and IPv6), displaying only the IPv6 packets. See the snoop(1M) man page for details. See "How to Monitor Only IPv6 Network Traffic" on page 362 for a description of procedures using this command.

route(1M)

This utility now operates on both IPv4 and IPv6 routes. By default, route operates on IPv4 routes. If you use the option –inet6 on the command line immediately following the route command, operations are performed on IPv6 routes. See the route(1M) man page for details.

ping(1M)

The ping command can use both IPv4 and IPv6 protocols to probe target hosts. Protocol selection depends on the addresses returned by the name server for the given target host. By default, if the name server returns an IPv6 address for the target host, the ping command uses the IPv6 protocol. If the server returns only an IPv4 address, it uses the IPv4 protocol. You can override this action by using the –A command-line option to specify what protocol to use.

Additionally, you can ping all the addresses of a multihomed target host by using the –a command-line option. See the ping(1M) man page for details. See "How to Probe All Multihomed Host Addresses" on page 363 for a description of a procedure using this command.

`traceroute`(1M)

You can use the `traceroute` command to trace both the IPv4 and IPv6 routes to a given host. As to which protocol to use, `traceroute` uses the same algorithm as `ping`. Use the –A command-line option to override this selection. You can trace each individual route to every address of a multihomed host by using the –a command-line option. See the `traceroute`(1M) man page for details.

Controlling Display Output

You can control how the `netstat` and `ifconfig` commands display output:

- Use keywords added to the command line to specify either `inet` or `inet6` addresses.
- Set the configuration variable DEFAULT_IP in the `/etc/default/inet_type` file.

You can set the value of DEFAULT_IP to IP_VERSION4, IP_VERSION6, or BOTH. If you do not create this file specifying the DEFAULT_IP, then `netstat` and `ifconfig` displays both versions.

Note - The `inet` or `inet6` keyword options used as part of the command-line argument override the value set in the `inet_type` file (if it exists) when using the `netstat` and `ifconfig` commands.

See “How to Control the Display Output of IPv6 Related Commands” on page 361 for a description of related procedures.

Solaris Tunneling Interfaces for IPv6

Tunneling interfaces have the following format:

```
ip.tun ppa
```

where *ppa* is the physical point of attachment.

Note - The Solaris software does not yet support encapsulating packets within IPv6 packets.

At system startup the tunneling module (`tun`) is pushed (by `ifconfig`) on top of IP to create a virtual interface. This is accomplished by creating the appropriate `hostsname6.*` file.

For example, to create a tunnel to encapsulate IPv6 packets over an IPv4 network (IPv6 over IPv4), you create a file named:

```
/etc/hostname6.ip.tun0
```

The content of this file is passed to `ifconfig`(1M) after the interfaces have been plumbed and the content becomes the parameters necessary to configure a point-to-point tunnel.

The following listing is an example of entries in `hostname6.ip.tun0` file:

EXAMPLE 16–5 `hostname6.`*interface* Entries

```
tsrc 120.68.100.23 tdst 120.68.7.19 up
addif 1234:1234::1 5678:5678::2 up
```

In this example, the IPv4 source and destination addresses are used as tokens to autoconfigure source and destination IPv6 link-local addresses for the `ip.tun0` interface. Two interfaces are configured, the `ip.tun0` interface mentioned, and a logical interface (`ip.tun0:1`) having the source and destination IPv6 addresses given by the `addif` command.

As mentioned previously, the contents of these configuration files are passed to `ifconfig` unmodified when the system is started as multiuser. The previous example is equivalent to:

```
# ifconfig ip.tun0 inet6 plumb
# ifconfig ip.tun0 inet6 tsrc 120.68.100.23 tdst 120.68.7.19 up
# ifconfig ip.tun0 inet6 addif 1234:1234::1 5678:5678::2 up
```

The output of `ifconfig –a` for this tunnel is:

```
ip.tun0: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480
index 6
        inet tunnel src 120.68.100.23  tunnel dst 120.68.7.19
        inet6 fe80::c0a8:6417/10 --> fe80::c0a8:713
ip.tun0:1: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480
index 5
        inet6 1234:1234::1/128 --> 5678:5678::2
```

You can configure more logical interfaces by adding lines to the configuration file using the following syntax:

```
addif IPv6-source IPv6-destination up
```

Note - If either end of the tunnel is an IPv6 router advertising one or more prefixes over the tunnel, you do not need `addif` commands in the tunnel configuration files, that is, just `tsrc` and `tdst` might be required because all other addresses are autoconfigured.

In some cases, specific source and destination link-local addresses need to be manually configured for a given tunnel. To do this, change the first line of the configuration file to include these link-local addresses. For example:

```
tsrc 120.68.100.23 tdst 120.68.7.19 fe80::1/10 fe80::2 up
```

Notice that the source link-local address has a prefix length of 10. In this example, the `ip.tun0` interface looks like the following:

```
ip.tun0: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480
index 6
        inet tunnel src 120.68.100.23  tunnel dst 120.68.7.19
        inet6 fe80::1/10 --> fe80::2
```

For specific information about `tun`, see the `tun`(7M) man page. For a general description of tunneling concepts during the transition to IPv6, see “Tunneling Mechanism” on page 324. For a description of a procedure for configuring tunnels see “How to Configure IPv6 Over IPv4 Tunnels” on page 364.

IPv6 Extensions to Solaris Name Services

This section describes naming changes resulting from the implementation of IPv6 introduced in the Solaris 8 release. You can store IPv6 addresses in any of the Solaris naming services (NIS, NIS+, DNS, and files), and also use NIS and NIS+ over IPv6 RPC transports to retrieve any NIS or NIS+ data.

`/etc/inet/ipnodes` File

The `/etc/inet/ipnodes` file stores both IPv4 and IPv6 addresses. It serves as a local database that associates the names of hosts with their IPv4 and IPv6 addresses. You should not store host names and their addressess in static files, such as `/etc/inet/ipnodes`. However, for testing purposes, it might be useful to store

IPv6 addresses in a file in the same way that IPv4 addresses are stored in `/etc/inet/hosts`. The `ipnodes` file uses the same format convention as the `hosts` file. See “Network Databases” on page 86 for a description of the `hosts` file. See `ipnodes`(4) man page for a description of the `ipnodes` file.

IPv6–aware utilities use the new `/etc/inet/ipnodes` database. The existing `/etc/hosts` database, which contains only IPv4 addresses, remains as it is to facilitate existing applications. If the `ipnodes` database does not exist, IPv6–aware utilities use the existing `hosts` database.

Note - If you need to add addresses, you must add IPv4 addresses to both the `hosts` and `ipnodes` files. You add only IPv6 addresses to the `ipnodes` file.

Example—`/etc/inet/ipnodes` File

```
#
# Internet IPv6 host table
# with both IPv4 and IPv6 addresses
#
::1     localhost
2::9255:a00:20ff:fe78:f37c      fripp.guitars.com fripp fripp-v6
fe80::a00:20ff:fe78:f37c        fripp-11.guitars.com frippll
120.46.85.87         fripp.guitars.com fripp fripp-v4
2::9255:a00:20ff:fe87:9aba      strat.guitars.com strat strat-v6
fe80::a00:20ff:fe87:9aba        strat-11.guitars.com stratll
120.46.85.177        strat.guitars.com strat strat-v4 loghost
```

Note - You must group host name addresses by the host name as shown in the above example.

NIS Extensions for IPv6

Two new maps have been added for NIS: `ipnodes.byname` and `ipnodes.byaddr`. Similar to `/etc/inet/ipnodes`, these maps contain both IPv4 and IPv6 information. The existing `hosts.byname` and `hosts.byaddr` maps, which contain only IPv4 information, remain as they are to facilitate existing applications.

NIS+ Extensions for IPv6

A new table has been added for NIS+ named `ipnodes.org_dir`. It contains both IPv4 and IPv6 addresses for a host. The existing `hosts.org_dir` table, which contains only IPv4 addresses for a host, remains as it is to facilitate existing applications.

DNS Extensions for IPv6

A new resource record defined as an AAAA record has been specified by RFC 1886. This AAAA record maps a host name into an 128–bit IPv6 address. The existing PTR record is still used with IPv6 to map IP addresses into host names. The 32 4–bit nibbles of the 128–bit address are reversed for an IPv6 address. Each nibble is converted to its corresponding hexadecimal ASCII value with `ip6.int` appended to it.

Changes to the `nsswitch.conf` File

In addition to the capability of looking up IPv6 addresses through `/etc/inet/ipnodes`, IPv6 support has been added to the NIS, NIS+, and DNS name services. Consequently, the `nsswitch.conf`(4) file has been modified to support IPv6 lookups. The addition of an ipnodes line to the `/etc/nsswitch.conf` file enables you to perform lookups in the new databases for each of the Solaris Name Services (NIS, NIS+, DNS, and files). For example:

```
hosts:  files dns nisplus [NOTFOUND=return]
ipnodes: files dns nisplus [NOTFOUND=return]
```

Note - Before changing the `/etc/nsswitch.conf` file to search ipnodes in multiple name services, populate these ipnodes databases with IPv4 and IPv6 addresses. Otherwise, unnecessary delays can result in the resolution of host addresses (including possible boot timing delays).

The following diagram shows the new relationship between the `nsswitch.conf` file and the new name services databases for applications using the `gethostbyname`() and `getipnodebyname`() commands. Items in italics are new. The `gethostbyname`() command looks only for IPv4 addresses stored in `/etc/inet/hosts`. The `getipnodebyname`() command consults the database specified in the `ipnodes` entry in the `nsswitch.conf` file. If the lookup fails, then it consults the database specified in the `hosts` entry in the `nsswitch.conf` file.

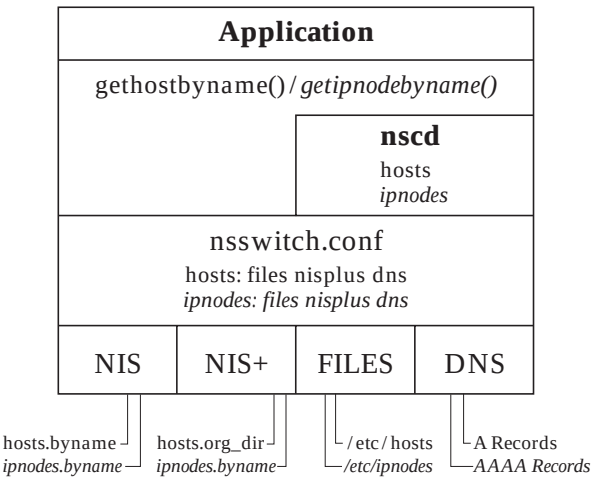


Figure 16–1 Relationship Between `nsswitch.conf` and Name Services

For more information on Naming Services, see *Solaris Naming Setup and Configuration Guide*.

Changes to Name Service Commands

To support IPv6, you can look up IPv6 addresses with the existing name service commands. For example, the `ypmatch` command works with the new NIS maps. The `nismatch` command works with the new NIS+ tables. The `nslookup` command can look up the new AAAA records in DNS. For a description of the changes to the name services see “NIS Extensions for IPv6” on page 346, “NIS+ Extensions for IPv6” on page 347, and “DNS Extensions for IPv6” on page 347.

For a description of procedures using these commands, see “Displaying IPv6 Name Service Information” on page 366.

NFS and RPC IPv6 Support

NFS and RPC software support IPv6 in a seamless manner. There are no changes to existing commands related to NFS services. Most RPC applications will also run over IPv6 without any change. Some advanced RPC applications with transport knowledge might require updates.

CHAPTER 17

Implementing IPv6

This chapter provides procedures for enabling IPv6, IPv6 routers, configuring IPv6 addresses for DNS, NIS, and NIS+; creating tunnels between routers; running IPv6 additions to commands for diagnostics; and displaying IPv6 name service information.

This is a list of the step-by-step instructions in this chapter.

For ...	See ...
Overview information about IPv6	Chapter 14
Transition information about transitioning from IPv4 to IPv6	Chapter 15
Conceptual information related to the procedures in this chapter	Chapter 16

Enabling IPv6 Nodes

This section provides procedures you might need to configure IPv6 nodes on your network.

Note - The term *node* in this context refers either to a Solaris server or client workstation.

Enabling IPv6 Nodes Task Map

TABLE 17–1 Enabling IPv6 Nodes Task Map

Task	Description	For Instructions, Go to ...
Enable IPv6 on a node	Involves touching `hostname6.`*interface* file, displaying addresses, and entering them in the `/etc/inet/ipnodes` file. (See note.)	"How to Enable IPv6 on a Node" on page 351
Configure a Solaris IPv6 router	Involves adding entries to the `indp.conf` file.	"How to Configure a Solaris IPv6 Router" on page 352

TABLE 17–1 Enabling IPv6 Nodes Task Map *(continued)*

Task	Description	For Instructions, Go to ...
Add IPv6 addresses to NIS and NIS+	Involves adding entries to the `/etc/ipnodes` file.	"How to Add IPv6 Addresses to NIS and NIS+" on page 353
Add IPv6 addresses to DNS	Involves adding AAAA records to the DNS zone and reverse zone file.	"How to Add IPv6 Addresses to DNS" on page 354

Note - You can enable IPv6 on a system when you install the Solaris software. If you answered *yes* to enable IPv6 during the installation process, you can omit the following IPv6 enabling procedure.

▼ How to Enable IPv6 on a Node

1. **Become superuser on the system where you want to enable IPv6.**

2. **On a command line, type the following for each interface.**

```
# touch /etc/hostname6.interface
```

Interface — Interface name, such as `le0`, `le1`.

3. **Reboot.**

Note - Rebooting sends out router discovery packets and the router responds with a prefix, enabling the node to configure the interfaces with an IP address. Rebooting also restarts key networking daemons in IPv6 mode.

4. **On a command line, type the following command to display the IPv6 addresses.**

```
# ifconfig -a
```

5. **Add the IPv6 address to the appropriate name service as follows:**

a. **For NIS and NIS+, see "How to Add IPv6 Addresses to NIS and NIS+" on page 353.**

b. **For DNS, see "How to Add IPv6 Addresses to DNS" on page 354.**

▼ How to Configure a Solaris IPv6 Router

1. **Become superuser on the system that will act as a router.**

2. **Edit the file `/etc/inet/ndpd.conf` with subnet prefixes by adding one or more of the following entries.**

   See the `in.ndpd`(1M) man page for a list of variables and allowable values. For more information about the `ndpd.conf` file, see the `ndpd.conf`(4) man page.

   a. **Add entries specifying router behavior for all interfaces.**

   ```
   ifdefault variable value
   ```

   b. **Add entries specifying prefix advertisement default behaviors.**

   ```
   prefixdefault variable value
   ```

   c. **Add sets per interface parameter entries.**

   ```
   if interface variable value
   ```

   d. **Add advertises per interface prefix information entries.**

   ```
   prefix prefix/length interface variable value
   ```

3. **Reboot the system.**

Note - Neighbor discovery (`in.ndpd`) relays to the hosts their subnet address prefixes. Also, the RIPng routing protocol (`in.ripngd`) runs automatically.

Example—ndpd.conf Router Configuration File

```
# Send router advertisements out all NICs
ifdefault AdvSendAdvertisements on
# Advertise a global prefix and a
# site local prefix on three interfaces.
# 0x9255 = 146.85
prefix 2:0:0:9255::0/64   hme0
prefix fec0:0:0:9255::0/64  hme0
# 0x9256 = 146.86
prefix 2:0:0:9256::0/64  hme1
prefix fec0:0:0:9256::0/64 hme1
# 0x9259 = 146.89
prefix 2:0:0:9259::0/64  hme2
prefix fec0:0:0:9259::0/64 hme2
```

▼ How to Add IPv6 Addresses to NIS and NIS+

A new table has been added for NIS+ named `ipnodes.org_dir`. It contains both IPv4 and IPv6 addresses for a host. The existing `hosts.org_dir` table, which contains only IPv4 addresses for a host, remains as it is to facilitate existing applications. You must keep both the `hosts.org_dir` and `ipnodes.org_dir` tables consistent with the IPv4 addresses. This does not happen automatically. See “IPv6 Extensions to Solaris Name Services” on page 345 for an overview.

Administration of the new `ipnodes.org_dir` table is similar to administering the `hosts.org_dir`. The same tools and utilities that were used in administering the previous NIS+ tables are valid for `ipnodes.org_dir`. See *Solaris Naming Administration Guide* for complete details on how to manipulate the NIS+ table.

The following procedure merges the entries from `/etc/inet/ipnodes` into the `ipnodes.org_dir` table (in verbose mode). The NIS+ table was probably created by `nistbladm`(1), `nissetup`(1M), or `nisserver`(1M).

- **On a command line, type the following command:**

```
% nisaddent -mv -f /etc/inet/ipnodes ipnodes
```

Use the following procedure to display the `ipnodes.org_dir` table.

- **On a command line, type the following command:**

```
% nisaddent -d ipnodes
```

Two new maps have been added for NIS: `ipnodes.byname` and `ipnodes.byaddr`. These maps contain both IPv4 and IPv6 hostname and address associations. The existing `hosts.byname` and `hosts.byaddr` maps, which contain only IPv4 hostname and address associations, will remain as they are to facilitate existing applications. Administration of the new maps is similar to the maintenance of the older `hosts.byname` and `hosts.byaddr` maps. Again, it is important that when you update the *hosts* maps with IPv4 addresses that the new ipnode maps are also updated with the same information.

Note - IPv6 aware tools will use the new NIS and NIS+ maps and tables exclusively.

▼ How to Add IPv6 Addresses to DNS

1. **Become superuser on system that has DNS.**

2. **Edit the appropriate DNS zone file by adding AAAA records for the IPv6–enabled host, using the following format.**

```
host-name   IN    AAAA   host-address
```

3. **Edit the DNS reverse zone file and add PTR records, using the following format.**

```
host-address IN    PTR    host-name
```

See RFC 1886 for more information about AAAA and PTR records.

Example—DNS Zone File

```
vallejo  IN  AAAA   2::9256:a00:20ff:fe12
IN  AAAA   fec0::9256:a00:20ff:fe12:528
```

Example—DNS Reverse Zone File

```
$ORIGIN ip6.int.
8.2.5.0.2.1.e.f.f.f.9.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0 \
 IN  PTR  vallejo.Eng.apex.COM.
8.2.5.0.2.1.e.f.f.f.9.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.c.e.f \
 IN  PTR  vallejo.Eng.apex.COM.
```

Monitoring IPv6

The following commands were modified to accommodate the Solaris implementation of IPv6.

- `ifconfig`(1M)
- `netstat`(1M)
- `snoop`(1M)
- `ping`(1M)
- `traceroute`(1M)

You can use the new additions to conduct diagnostics. For conceptiual descriptions of these commands, see “IPv6 Extensions to the `ifconfig` Utility” on page 333 and “IPv6 Extensions to Existing Utilities” on page 341.

Monitoring IPv6 Task Map

TABLE 17–2 Monitoring IPv6 Task Map

Task	Description	For Instructions, Go to
Display interface address assignments	Displays all, or just IPv4, or just IPv6 address assignments using `ifconfig` command.	“How to Display Interface Address Assignments” on page 356
Display network status	Displays all sockets and routing table entries, inet address family for IPv4, inet6 address family for IPv6, and statistics per interface IPv6/ICMPv6 counters using `netstat` command.	“How to Display Network Status” on page 357

TABLE 17–2 Monitoring IPv6 Task Map *(continued)*

Task	Description	For Instructions, Go to
Control the display output of IPv6 related commands	Controls the output of the `ping`, `netstat`, `ifconfig`, and `traceroute` commands by creating a file named `inet_type` and setting the DEFAULT_IP variable in it.	"How to Control the Display Output of IPv6 Related Commands" on page 361
Monitor only IPv6 network traffic	Displays all IPv6 packets using the `snoop` command.	"How to Monitor Only IPv6 Network Traffic" on page 362
Probe all multihomed host addresses	Checks all addresses using the `ping` command.	"How to Probe All Multihomed Host Addresses" on page 363
Trace all routes	Uses the `traceroute` command.	"How to Trace All Routes" on page 363

▼ How to Display Interface Address Assignments

You can use the `ifconfig` command to display all address assignments as well as just IPv4 or IPv6 address assignments.

♦ **On the command line, type the following command.**

```
% ifconfig [option]
```

For more information on the `ifconfig` command, see the `ifconfig`(1M) man page.

Example—Displaying Addressing Information for All Interfaces

```
% ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.146.86.255
```

(continued)

```
        ether 8:0:73:56:a8
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:20ff:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

Example—Displaying Addressing Information for All IPv4 Interfaces

```
% ifconfig -a4
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
```

Example—Displaying Addressing Information for All IPv6 Interfaces

```
% ifconfig -a6
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:20ff:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

▼ How to Display Network Status

These procedures enable you to display the following network data structure formats using the `netstat` command:

- All sockets and routing table entries
- Inet address family for IPv4

- Inet6 address family for IPv6
- Statistics per interface—IPv6/ICMPv6 counters

♦ **On the command line, type the following command.**

```
% netstat [option]
```

For more information on the `netstat` command, see the `netstat`(1M) man page.

Example—Displaying All Sockets and Routing Table Entries

```
% netstat -a
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.*                                  Unbound
      *.apexrpc                             Idle
      *.*                                  Unbound
   .
   .
UDP: IPv6
   Local Address                      Remote Address                   State
If
--------------------------------- --------------------------------- -------
      *.*                                                          Unbound
      *.time                                                       Idle
      *.echo                                                       Idle
      *.discard                                                    Idle
      *.daytime                                                    Idle
      *.chargen                                                    Idle

TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*               0      0     0      0 IDLE
      *.apexrpc            *.*               0      0     0      0 LISTEN
      *.*                  *.*               0      0     0      0 IDLE
      *.ftp                *.*               0      0     0      0 LISTEN
localhost.427              *.*               0      0     0      0 LISTEN
      *.telnet             *.*               0      0     0      0 LISTEN
tn.apex.COM.telnet is.Eng.apex.COM         8760      0   8760     0 ESTABLISHED
tn.apex.COM.33528 np.apex.COM.46637        8760      0   8760     0 TIME_WAIT
tn.apex.COM.33529 np.apex.COM.apexrpc      8760      0   8760     0 TIME_WAIT
TCP: IPv6
   Local Address     Remote Address   Swind Send-Q  Rwind Recv-Q   State   If
----------------- -----------------   ----- ------  ----- ------   -----
      *.*                  *.*             0      0      0      0   IDLE
      *.ftp                *.*             0      0      0      0   LISTEN
      *.telnet             *.*             0      0      0      0   LISTEN
      *.shell              *.*             0      0      0      0   LISTEN
      *.smtp               *.*             0      0      0      0   LISTEN
```

(continued)

(Continuation)

```
  .
  .
 2::56:8.login     something.1023    8640      0   8640      0 ESTABLISHED
 fe80::a:a8.echo   fe80::a:89        8640      0   8640      0 ESTABLISHED
 fe80::a:a8.ftp    fe80::a:90        8640      0   8640      0 ESTABLISHED
```

Example—Displaying Inet Address Family for IPv4

```
% netstat -f inet
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
tn.apex.COM.telnet    is.apex.COM.35388    8760      0  8760   0  ESTABLISHED
tn.apex.COM.1022      alive-v4.nfsd        8760      0  8760   0  ESTABLISHED
tn.apex.COM.1021      sl.apex.COM.nfsd     8760      0  8760   0  ESTABLISHED
.
.
tn.apex.COM.33539     np.apex.COM.apexrpc  8760      0  8760   0  TIME_WAIT
```

Example—Displaying Inet6 Address Family for IPv4

```
% netstat -f inet6
TCP: IPv6
   Local Address        Remote Address     Swind Send-Q Rwind Recv-Q   State      If
-------------------- -------------------- ----- ------ ----- ------ ----------- -----
2::56:a8.login        something.1023       8640      0  8640      0 ESTABLISHED
fe80::a0:a8.echo      fe80::a0:de.35389    8640      0  8640      0 ESTABLISHED
.
.
fe80::a0:a8.ftp-data  fe80::a0:de.35394   25920      0 25920      0 TIME_WAIT
```

Example—Displaying Statistics Per Interface: IPv6/ICMPv6 Counters

```
% netstat -sa
RAWIP
        rawipInDatagrams    =  1407     rawipInErrors       =     0
        rawipInCksumErrs    =     0     rawipOutDatagrams   =     5
        rawipOutErrors      =     0

UDP
        udpInDatagrams      =  7900     udpInErrors         =     0
        udpOutDatagrams     =  7725     udpOutErrors        =     0

TCP     tcpRtoAlgorithm     =     4     tcpRtoMin           =   200
        tcpRtoMax           = 60000     tcpMaxConn          =    -1
    .
    .
IPv4    ipForwarding        =     2     ipDefaultTTL        =   255
        ipInReceives        =406345     ipInHdrErrors       =     0
        ipInAddrErrors      =     0     ipInCksumErrs       =     0
    .
    .
IPv6 for lo0
        ipv6Forwarding      =     2     ipv6DefaultHopLimit =     0
        ipv6InReceives      =     0     ipv6InHdrErrors     =     0
    .
    .
IPv6 for le0
        ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      =   885     ipv6InHdrErrors     =     0
    .
    .
IPv6    ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      =   885     ipv6InHdrErrors     =     0
    .
    .
ICMPv4  icmpInMsgs          =   618     icmpInErrors        =     0
        icmpInCksumErrs     =     0     icmpInUnknowns      =     0
        icmpInDestUnreachs  =     5     icmpInTimeExcds     =     0
    .
    .
ICMPv6 for lo0
        icmp6InMsgs         =     0     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
    .
    .
ICMPv6 for le0
        icmp6InMsgs         =   796     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
        icmp6InTimeExcds    =     0     icmp6InParmProblems =     0
    .
    .
ICMPv6  icmp6InMsgs         =   796     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
    .
    .
```

(continued)

```
IGMP:
       2542 messages received
          0 messages received with too few bytes
          0 messages received with bad checksum
       2542 membership queries received
    .
    .
```

▼ How to Control the Display Output of IPv6 Related Commands

You can control the output of the `netstat` and `ifconfig` commands by creating a file named `inet_type` in the `/etc/default` directory and specifying the value of the DEFAULT_IP variable. For more information about the `inet_type`, see the `inet_type`(4) man page.

1. **Create the `/etc/default/inet_type` file.**

2. **Make one of the following entries, as needed.**
   a. **To display IPv4 information only, enter:**

```
DEFAULT_IP=IP_VERSION4
```

   b. **To display both IPv4 and IPv6 information, enter:**

```
DEFAULT_IP=BOTH
```

   or

```
DEFAULT_IP=IP_VERSION6
```

Note - The –4 and –6 flags in `ifconfig` and –`f` flag in `netstat` override the value set in the `inet_type` file (if it exists).

Examples—Controlling Output to Select IPv4 and IPv6 Information

- When you specify the DEFAULT_IP=BOTH or DEFAULT_IP=IP_VERSION6 variable in the `inet_type` file:

```
% ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:a00:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

- When you specify the DEFAULT_IP=IP_VERSION4 variable in the `inet_type` file:

```
% ifconfig -a
lo0: flags=849 mtu 8232
        inet 120.10.0.1 netmask ff000000
le0: flags=843 mtu 1500
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
```

▼ How to Monitor Only IPv6 Network Traffic

In this procedure you use the `snoop` command to display all IPv6 packets.

1. **Become superuser.**

2. **On the command line, type the following command.**

```
# snoop ip6
```

For more information on the `snoop` command, see the `snoop`(1M) man page.

Example—Displaying Only IPv6 Network Traffic

```
# snoop ip6
Using device /dev/le (promiscuous mode)
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=104
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=104
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=104
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=152
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=72
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=72
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=72
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=72
```

▼ How to Probe All Multihomed Host Addresses

In this procedure you use the `ping` command to check all addresses.

♦ **On the command line, type the following command.**

```
% ping -a ipng11
ipng11 (2::102:a00:fe79:19b0) is alive
ipng11 (fec0::102:a00:fe79:19b0) is alive
ipng11 (190.68.10.75) is alive
```

For more information on the `ping` command, see the `ping`(1M) man page.

▼ How to Trace All Routes

In this procedure you use the `traceroute` command to trace all routes.

♦ **On the command line, type the following command.**

```
% traceroute -a <hostname>
```

For more information on the `traceroute` command, see the `traceroute`(1M) man page.

Example—Tracing All Routes

```
% traceroute -a ipng11
traceroute: Warning: Multiple interfaces found; using 2::56:a0:a8 @ le0:2
traceroute to ipng11 (2::102:a00:fe79:19b0),30 hops max, 60 byte packets
 1  ipng-rout86 (2::56:a00:fe1f:59a1)  35.534 ms  56.998 ms *
 2  2::255:0:c0a8:717  32.659 ms  39.444 ms *
 3  ipng61.Eng.apex.COM (2::103:a00:fe9a:ce7b)  401.518 ms  7.143 ms *
 4  ipng12-00 (2::100:a00:fe7c:cf35)  113.034 ms  7.949 ms *
 5  ipng11 (2::102:a00:fe79:19b0)  66.111 ms *  36.965 ms

traceroute: Warning: Multiple interfaces found; using fec0::56:a8 @ le0:1
traceroute to ipng11 (fec0::10:b0), 30 hops max, 60 byte packets
 1  ipng-rout86 (fec0::56:a00:fe1f:59a1) 96.342 ms  78.282 ms 88.327 ms
 2  ipng8-tun1 (fec0::25:0:0:c0a8:717)  268.614 ms  508.416 ms  438.774 ms
 3  ipng61.Eng.apex.COM (fec0::103:a00:fe9a:ce7b)  6.356 ms * 713.166 ms
 4  ipng12-00 (fec0::100:a00:fe7c:cf35)  7.409 ms * 122.094 ms
 5  ipng11 (fec0::102:a00:fe79:19b0)  10.620 ms * *

traceroute to ipng11.eng.apex.com (190.68.10.75),30 hops max,40 byte packets
 1  rmpj17c-086.Eng.apex.COM (120.46.86.1)  4.360 ms  3.452 ms  3.479 ms
 2  flrmpj17u.Eng.apex.COM (120.46.17.131)  4.062 ms  3.848 ms  3.505 ms
 3  ipng8.Eng.apex.COM (120.68.7.23)  4.773 ms *  4.294 ms
 4  ipng61.Eng.apex.COM (120.68.10.104)  5.128 ms  5.362 ms *
 5  ipng12-20.Eng.apex.COM (120.68.10.62)  7.298 ms  5.444 ms *
 6  ipng11.Eng.apex.COM (120.68.10.75)  8.053 ms  6.394 ms *
```

Configuring IPv6 Over IPv4 Tunnels

This section describes how you configure IPv6 over IPv4 tunnels.

For conceptual descriptions of tunnels, see “Solaris Tunneling Interfaces for IPv6” on page 343 and “Tunneling Mechanism” on page 324.

▼ How to Configure IPv6 Over IPv4 Tunnels

1. **Become superuser.**

2. **Create the file `/etc/hostname6.ip.tun`*n* (where *n* is 0, 1, 2, and so on) and add entries using the following steps.**

   a. **Add the tunnel source and tunnel destination addresses.**

   ```
   tsrc IPv4-source-addr tdst IPv4-destination-addr up
   ```

   b. **(Optional) Add a logical interface for the source and destination IPv6 addresses.**

```
addif IPv6-source-address   IPv6-destination-address up
```

Omit this step if you want the address autoconfigured for this interface. You do not need to configure link-local addresses for your tunnel because they are automatically configured.

When you finish configuring the tunnels, you must reboot.

Note - You must perform the same steps at the other end of the tunnel for bidirectional communication to occur.

If your system is to be configured as a router, you must also configure your router to advertise over tunneling interfaces before rebooting (see "How to Configure Your Router to Advertise Over Tunneling Interfaces" on page 365).

Example—IPv6 Configuration File Entry to Autoconfigure IPv6 Addresses

This is an example of a tunnel for which all IPv6 addresses are autoconfigured.

```
tsrc 129.146.86.138 tdst 192.168.7.19 up
```

Example—IPv6 Configuration File Entry for Manually Configured Addresses

This is an example of a tunnel for which global and site local source and destination addresses are manually configured.

```
tsrc 120.46.86.138 tdst 190.68.7.19 up
addif fec0::1234:a00:fe12:528 fec0::5678:a00:20ff:fe12:1234 up
addif 2::1234:a00:fe12:528 2::5678:a00:20ff:fe12:1234 up
```

▼ How to Configure Your Router to Advertise Over Tunneling Interfaces

Do the following steps for each tunnel.

1. **Become superuser.**

2. **Edit the file** `/etc/inet/ndpd.conf` **and add entries using the following steps.**
   a. **Enable router advertisement over the tunneling interface.**

```
if ip.tunn AdvSendAdvertisements 1
```

b. Add prefixes as needed.

```
prefix interface-address ip.tunn
```

3. Reboot.

Displaying IPv6 Name Service Information

This section provides procedures to display IPv6 Name Service information.

Displaying IPv6 Name Service Information Task Map

TABLE 17–3 Displaying IPv6 Name Service Information Task Map

Task	Description	For Instructions, Go to
Display IPv6 name service information	Displays IPv6 Name Service Information using the `nslookup` command.	"How to Display IPv6 Name Service Information" on page 367
Verify that DNS IPv6 PTR records were updated correctly	Displays the DNS IPv6 PTR records using the `nslookup` command and the `set q=PTR` parameter to display.	"How to Verify That DNS IPv6 PTR Records Were Updated Correctly" on page 368
Display IPv6 information through NIS	Displays the IPv6 information through NIS using the `ypmatch` command.	"How to Display IPv6 Information Through NIS" on page 369

TABLE 17–3 Displaying IPv6 Name Service Information Task Map *(continued)*

Task	Description	For Instructions, Go to
Display IPv6 information through NIS	Displays the IPv6 information through NIS+ using the `nismatch` command.	"How to Display IPv6 Information Through NIS+" on page 369
Display IPv6 information independent of name service	Displays the IPv6 information using the `getent` command.	"How to Display IPv6 Information Independent of Name Service" on page 370

▼ How to Display IPv6 Name Service Information

In this procedure, you use the `nslookup` command to display IPv6 name service information.

1. **On the command line, type the following command:**

```
% /usr/sbin/nslookup
```

   The default server name and address appear, followed by the `nslookup` command angle bracket prompt.

2. **To see information about a particular host, type the following commands at the angle bracket prompt:**

```
>set q=any
>host-name
```

3. **To see only AAAA records, type the following command at the angle bracket prompt:**

```
>set q=AAAA
```

4. **Quit the command by typing exit.**

Example—Using nslookup to Display IPv6 Information

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  120.46.168.78
> set q=any
> vallejo
Server:  space1999.Eng.apex.COM
Address:  120.46.168.78

vallejo.ipv6.eng.apex.com      IPv6 address = fec0::9256:a00:fe12:528
vallejo.ipv6.eng.apex.com      IPv6 address = 2::9256:a00:fe12:528
> exit
```

▼ How to Verify That DNS IPv6 PTR Records Were Updated Correctly

In this procedure you use the nslookup command to display DNS IPv6 PTR records.

1. **On the command line, type the following command:**

```
% /usr/sbin/nslookup
```

   The default server name and address display, followed by the nslookup command angle bracket prompt.

2. **To see the PTR records, type the following command at the angle bracket prompt:**

```
>set q=PTR
```

3. **Quit the command by typing exit.**

Example—Using nslook to Display PTR Records

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  120.46.168.78
> set q=PTR
> 8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int
```

(continued)

```
8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int name =
vallejo.ipv6.Eng.apex.COM
ip6.int nameserver = space1999.Eng.apex.COM
> exit
```

▼ How to Display IPv6 Information Through NIS

In this procedure you use the `ypmatch` command to display IPv6 information through NIS.

- **On the command line, type the following command:**

```
% ypmatch host-name ipnodes.byname
```

The information about *host-name* displays.

EXAMPLE 17–1 Example—Using `ypmatch` to Display IPv6 Information Through NIS

```
% ypmatch vallejo ipnodes.byname
fec0::9256:a00:20ff:fe12:528    vallejo
2::9256:a00:20ff:fe12:528       vallejo
```

▼ How to Display IPv6 Information Through NIS+

In this procedure you use the `nismatch` command to display IPv6 information through NIS.

- **On the command line, type the following command:**

```
% nismatch host-name ipnodes.org-dir
```

The information about *host-name* displays.

CODE EXAMPLE 17–1 Example—Using `nismatch` to Display IPv6 Information Through NIS+

```
% nismatch vallejo ipnodes.org_dir
vallejo vallejo fec0::9256:a00:20ff:fe12:528
vallejo vallejo 2::9256:a00:20ff:fe12:528
```

▼ How to Display IPv6 Information Independent of Name Service

♦ **On the command line, type the following command:**

```
% getent ipnodes host-name
```

The information about *host-name* displays.

CODE EXAMPLE 17–2 Example—Using `getent` to Display IPv6 Information Independent of Name Service

```
% getent ipnodes vallejo
2::56:a00:fe87:9aba        vallejo vallejo
fec0::56:a00:fe87:9aba     vallejo vallejo
```

CHAPTER 18

Overview of IPsec

The IP Security Architecture (IPsec) provides protection for IP datagrams. The protection can include confidentiality, strong integrity of the data, partial sequence integrity (replay protection), and data authentication. IPsec is performed inside the IP process, and it can be applied with or without the knowledge of an Internet application. While IPsec is an effective tool in securing network traffic, it does not eliminate security problems.

Introduction to IPsec

IPsec provides security associations that include secure datagram authentication and encryption mechanisms within IP. When you invoke IPsec, it applies the security mechanisms to IP datagrams that you have enabled in the IPsec global policy file.

The following figure shows how an IP addressed packet, as part of an IP datagram, proceeds when IPsec has been invoked on an outbound packet. As you can see from the flow diagram, authentication header (AH) and encapsulated security payload

(ESP) entities can be applied to the packet. Subsequent sections describe how you apply these entities, as well as athentication and encryption algorithms.

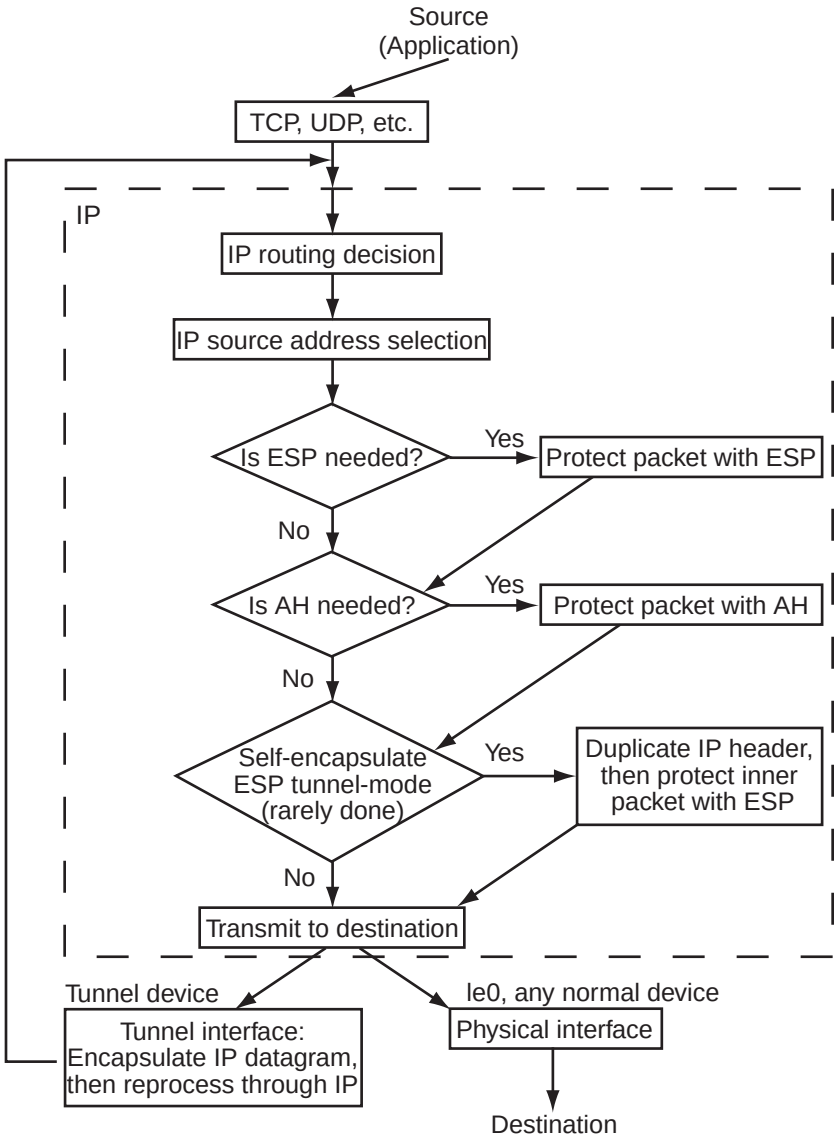


Figure 18–1 IPsec Applied to Outbound Packet Process

The following figure shows the IPsec inbound process.

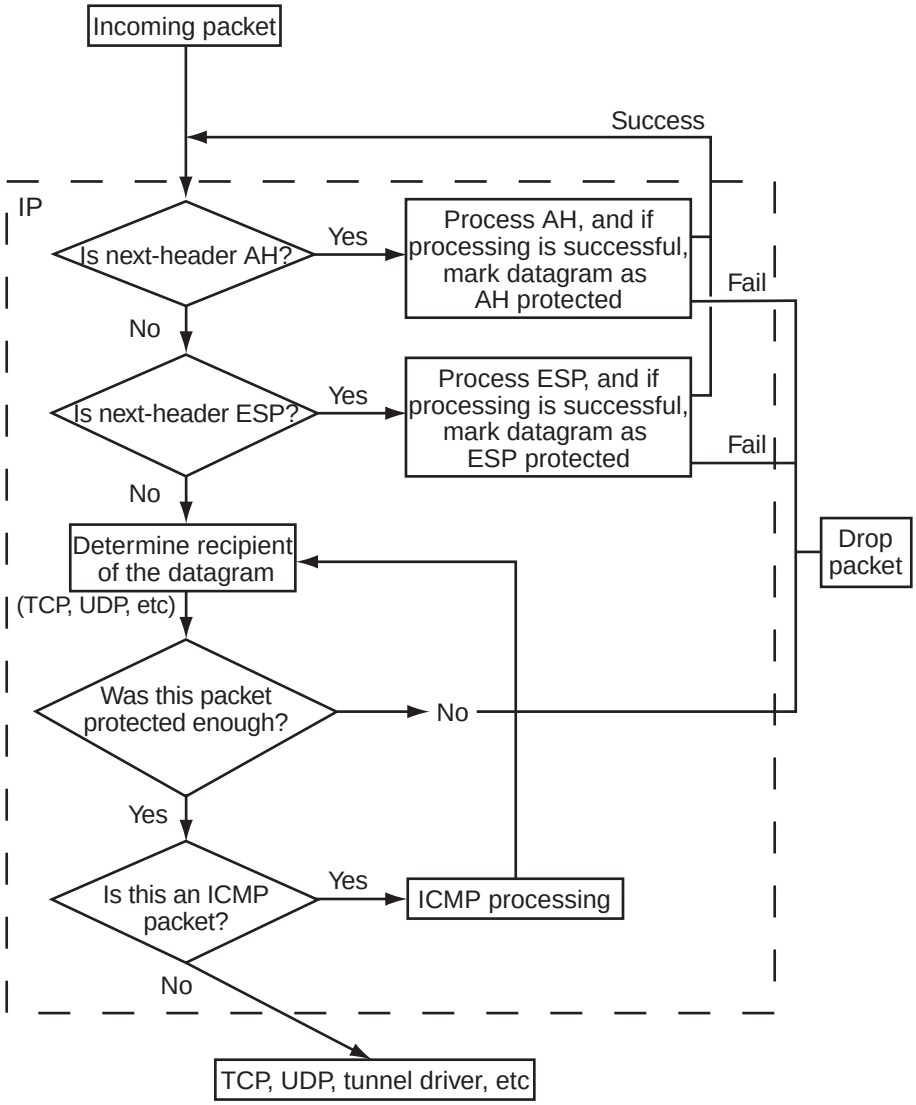


Figure 18–2 IPsec Applied to Inbound Packet Process

Security Associations

Security associations (SA) specify security properties from one host to another. Two communicating systems need at least two SAs to communicate securely, unless they

use multicast, in which case they can use the same multicast SA. The `pf_key`(7P) interface manages Security Associations. IPsec does not currently support automatic SA management, but you can use `ipseckey`(1M) as a command-line front-end. The AH or ESP, destination IP address, and security parameters index (SPI) identifies an IPsec SA. The security parameters index, an arbitrary 32-bit value, is transmitted with an AH or ESP packet. See `ipsecah`(7P) or `ipsecesp`(7P) man pages for an explanation about where the SPI resides in a protected packet.

Key Management

A security association contains keying information, algorithm choices, endpoint identities, and other parameters. Managing SAs is called key management. Currently, you must manually do key management.

Protection Mechanisms

IPsec provides two mechanisms for protecting data:

- Authentication Header (AH)
- Encapsulating Security Payload (ESP)

Both mechanisms use security associations.

Authentication Header

The authentication header, a new IP header, provides strong integrity, partial sequence integrity (replay protection), and data authentication to IP datagrams. AH protects as much of the IP datagram as it can. AH cannot protect fields that change nondeterministically between sender and receiver. For example, the IP TTL field is not a predictable field and, consequently, not protected by AH. AH is inserted between the IP header and the transport header. The transport header can be TCP, UDP, ICMP, or another IP header when tunnels are being used. See the `tun`(7M) man page for details on tunneling.

Authentication Algorithms and the AH Device

IPsec implements AH as a module that is automatically pushed on top of IP. The `/dev/ipsecah` entry tunes AH with `ndd`(1M), in addition to allowing future authentication algorithms to be loaded on top of AH. Current authentication algorithms include HMAC-MD5 and HMAC-SHA-1. Each authentication algorithm

has its own key size and key format properties. See the `authmd5h`(7M) and `authsha1`(7M) man pages for details.

Security Considerations

Without replay protection enabled, all replay attacks jeopardize AH. AH does not protect against eavesdropping. Adversaries can still see data protected with AH.

Encapsulating Security Payload

The ESP provides confidentiality over what it encapsulates, as well as the services that AH provides, but only over that which it encapsulates. ESP's authentication services are optional. These services enable you to use ESP and AH together on the same datagram without redundancy. Because ESP uses encryption-enabling technology, it falls under U.S. export control laws.

ESP encapsulates its data, so it only protects the data that follows its beginning in the datagram. In a TCP packet, ESP encapsulates only the TCP header and its data. If the packet is an IP in IP datagram, ESP protects the inner IP datagram. Per-socket policy allows *self-encapsulation*, so ESP can encapsulate IP options, when it needs to. Unlike the authentication header (AH) , ESP allows multiple kinds of datagram protection. Using only a single form of datagram protection can expose the datagram to vulnerabilities. For example, you can use only ESP to provide confidentiality, but protecting confidentiality alone exposes vulnerabilities in both replay attacks and cut-and-paste attacks. Similarly, if ESP protects only integrity, and does not fully protect against eavesdropping, it could provide weaker protection than AH.

Algorithms and the ESP Device

IPsec ESP implements ESP as a module that is automatically pushed on top of IP. Use the `/dev/ipsecesp` entry to tune ESP with `ndd`(1M), as well as to allow future algorithms to be loaded on top of ESP. ESP allows encryption algorithms to be pushed on top of it, in addition to the authentication algorithms used in AH. Encryption algorithms include United States Data Encryption Standard (DES) and Triple-DES (3DES). Each encryption algorithm has its own key size and key format properties. Because of export laws in the United States, not all encryption algorithms are available outside of the United States.

Security Considerations

ESP without authentication exposes vulnerabilities to cut-and-paste cryptographic attacks, as well as eavesdropping attacks. When you use ESP without confidentiality, its vulnerability to replay is similar to AH. Because of United States export control

laws, the encryption strength available on ESP is weaker for versions of the SunOS sold outside the United States.

Authentication and Encryption Algorithms

IPsec uses two types of algorithms:

- Authentication
- Encryption

Authentication Algorithms

Authentication algorithms produce an integrity checksum value or *digest* based on the data and a key. The authentication algorithm man pages describe the size of both the digest and key (see, for example, `authmd5h`(7M) and `authsha1`(7M) man pages).

Encryption Algorithms

Encryption algorithms encrypt data with a key. Encryption algorithms operate on data in units of a *block size*. The encryption algorithm man pages describe the size of both the block size and the key size (see, for example, `encrdes`(7M) and `encr3des`(7M) man pages).

Protection Policy and Enforcement Mechanisms

IPsec separates protection policy and enforcement mechanisms. You can enforce IPsec policies in the following places:

- On a system-wide level
- On a per-socket level

You use the `ipsecconf`(1M) command to configure system-wide policy.

IPsec applies system-wide policy to incoming and outgoing datagrams. You can apply some additional rules to outgoing datagrams, because of the additional data known by the system. Inbound datagrams can be either accepted or dropped. The decision to drop or accept an inbound datagram is based on several criteria, which sometimes overlap or conflict. Resolving conflict resolution depends on which rule is parsed first; except when a policy entry states that traffic should bypass all other policy, it is

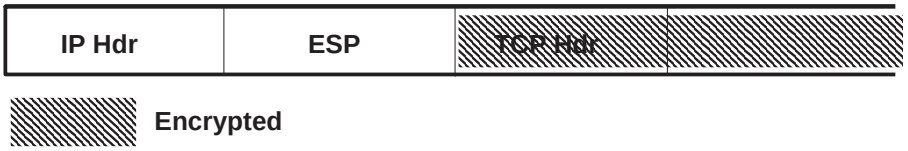
automaticaly accepted. Outbound datagrams are either sent with protection or without protection. If protection is applied, the algorithms are either specific or non-specific. If policy normally protects a datagram, it can be bypassed by either an exception in system-wide policy, or by requesting a bypass in per-socket policy.

For intra-system traffic, policies are enforced, but actual security mechanisms are not applied. Instead, the outbound policy on an intra-system packet translates into an inbound packet that has had those mechanisms applied.

Transport and Tunnel Modes

When you invoke ESP or AH after the IP header to protect a datagram, this is referred to as transport mode. For example, if a packet starts off as:

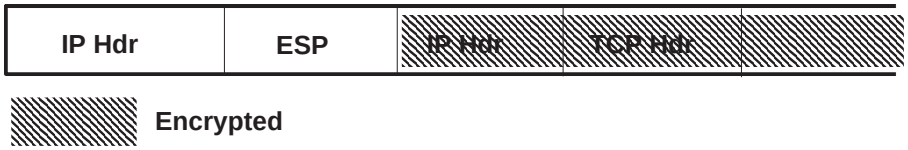
Then in transport mode, ESP protects the data as follows:



AH, in transport mode, protects the data as follows:

AH actually covers the data before it appears in the datagram. Consequently, the protection provided by AH, even in transport mode, does cover some of the IP header.

When an entire datagram is *inside* the protection of an IPsec header, this is referred to as tunnel mode. Since AH covers most of its preceding IP header, tunnel mode is usually performed only on ESP. The previous example datagram would be protected in tunnel mode as follows:



Often, in tunnel mode, the outer (unprotected) IP header has different source and destination addresses from the inner (protected) IP header. The inner and outer IP headers can match if, for example, an IPsec-aware network program uses self-encapuslation with ESP. This is done in case of an IP header option that needs to be protected with ESP.

The Solaris implementation of IPsec is primarily a transport mode IPsec implementation, which implements the tunnel mode as a special case of the transport mode. This is accomplished by treating IP-in-IP tunnels as a special transport provider. When you use `ifconfig`(1M) configuration options to set tunnels, they are nearly identical to the options available to socket programmers when enabling per-socket IPsec. Also, tunnel mode can be enabled in per-socket IPsec.

Tunneling Module for IPsec Tunnels

A configured tunnel is a point-to-point interface. It enables an IP packet to be encapsulated within an IP packet. Configuring a tunnel requires both a tunnel source and tunnel destination. See the `tun`(7M) man page and “Solaris Tunneling Interfaces for IPv6” on page 343 for more information.

A tunnel creates an apparent physical interface to IP. The physical link’s integrity depends on the underlying security protocols. If you set up the security associations securely, then you can trust the tunnel; that is, packets that come off the tunnel came from the peer specified in the tunnel destination. If this trust exists, you can use per-interface IP forwarding to create a virtual private network.

Enabling Virtual Private Networks

You can use IPsec to construct virtual private networks (VPN). You do this by constructing an Intranet using the Internet infrastructure. For example, an organization that has separate offices (with separate networks), and uses VPN technology to connect their offices, can deploy IPsec to secure traffic between the two offices.

The following figure illustrates how two offices use the Internet to form their VPN with IPsec deployed on their network systems.

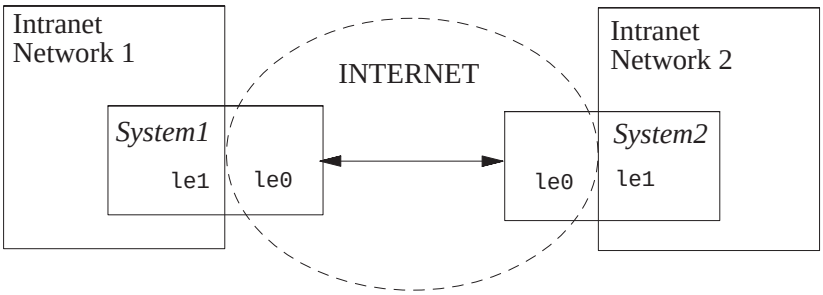


Figure 18–3 Virtual Private Network

See "How to Set Up a Virtual Private Network" on page 390 for a description of the setup procedure.

Managing IPsec

This section describes the IPsec initialization configuration file and various commands that enable you to manage IPsec within your network. See Chapter 19 for a description of procedures you can use to manage IPsec.

IPsec Initialization Configuration File

To invoke IPsec security policies when you start the Solaris operating environment, you need to create an IPsec initialization configuration file with your specific IPsec entries. You should name the file, `/etc/inet/ipsecinit.conf`. See the `ipsecconf`(1M) man page for details about policy entries and their format.

Example—`ipsecinit.conf` File

The Solaris software includes a sample `ipsecinit.conf` file that you can use as a template to create your own `ipsecinit.conf` file. This sample file is named `ipsecinit.sample` and it contains the following entries:

```
#
#ident "@(#)ipsecinit.sample 1.4 99/04/28 SMI"
#
# Copyright (c) 1999 by Sun Microsystems, Inc.
# All rights reserved.
#
# This file should be copied to /etc/inet/ipsecinit.conf to enable IPsec.
# Even if this file has no entries, IPsec will be loaded if
```

(continued)

(Continuation)

```
# /etc/inet/ipsecinit.conf exists.
#
# Add entries to protect the traffic using IPSEC. The entries in this
# file are currently configured using ipsecconf from inetinit script
# after /usr is mounted.
#
# For example,
#
#  {dport 23} apply {encr_algs des encr_auth_algs md5 sa shared}
#  {sport 23} permit {encr_algs des encr_auth_algs md5}
#
# will protect the telnet traffic to/from the host with ESP using DES and
# MD5.  Also:
#
#  {daddr 10.5.5.0/24} apply {auth_algs any sa shared}
#  {saddr 10.5.5.0/24} permit {auth_algs any}
#
# will protect traffic to or from the 10.5.5.0 subnet with AH
# using any available algorithm.
#
#
# WARNING: This file is read before default routes are established, and
#  before any naming services have been started.  The
#  ipsecconf(1M) command attempts to resolve names, but it will
#  fail unless the machine uses files, or DNS and the DNS server
#  is on-subnet (i.e. reachable without a default route).
#
#  It is suggested that for this file, use hostnames only if
#  they are in /etc/hosts, or use numeric IP addresses.
#
#  If DNS gets used, the DNS server is implicitly trusted, which
#  could lead to compromise of this machine if the DNS server
#  has been compromised.
#
```

Global Policy Setter

You use the `ipsecconf`(1M) command to configure the IPsec policy for a host. After you configure the policy, IPsec subjects all outbound and inbound datagrams to policy checks as they exit and enter the host. If no entries are found, no policy checks are completed, and all the traffic passes through. Forwarded datagrams are not subjected to policy checks added using this command. See `ifconfig`(1M) and `tun`(7M) for information on how to protect forwarded packets. You can use the `ifconfig` command to delete a policy entry from the `/etc/inet/ipsecpolicy.conf` file, or to view the existing configuration.

You must become superuser to invoke this command. Each entry protects traffic in only one direction, that is, either outbound or inbound. Thus, to protect traffic in both directions, you need to have separate entries for each direction.

You can see the policies configured in the system when you issue the command without any arguments. The command displays each entry with an *index* followed by a number. You can use the –d option with the *index* to delete a given policy in the system. The command displays the entries in the order that they were added, which is not necessarily the order in which the traffic match takes place. To view the order in which the traffic match takes place, use the –l option.

IPsec does not preserve policy entries across reboots. Thus, you need to add the policy every time the system reboots. To configure policies early in the boot process, you can set up policies in the `/etc/inet/ipsecinit.conf` file, so that the `inetinit` startup script reads them.

Security Considerations

If, for example, the `/etc/inet/ipsecpolicy.conf` file is sent from an NFS mounted file system, an adversary can modify the data contained in the file and actually make changes to the configured policy. Consequently, you should not transmit a copy of the `/etc/inet/ipsecpolicy.conf` file over a network.

Policy is latched for TCP/UDP sockets on which a `connect`(3N) or `accept`(3N) has been issued. Adding new policy entries does not have any effect on them. This latching feature might change in the future, so you should not depend on this feature.

Make sure you set up the policies before starting any communications, because existing connections might be affected by the addition of new policy entries. Similarly, do not change policies in the middle of a communication.

If your source address is a host that can be looked up over the network, and your naming system itself is compromised, then any names used are no longer trustworthy.

Security weaknesses often lie in misapplication of tools, not the tools themselves. You should be cautious when using `ipseckey`. Use a console for the safest mode of operation, or other hard-connected TTY.

Security Associations Database

Keying information for IPsec security services is maintained in security association databases (SADBs). Security associations protect both inbound and outbound packets. A user process (or possibly multiple co-operating processes) maintains SADBs by sending messages over a special kind of socket. This is analogous to the method described in the `route`(7P) man page. Only a superuser can access an SADB.

The operating system might spontaneously emit messages in response to external events, such as a request for a new SA for an outbound datagram, or to report the expiration of an existing SA. You open the channel for passing SADB control messages by using the socket call described in the previous section. More than one key socket can be open per system.

Messages include a small base header, followed by a number of extension messages (zero or more). Some messages require additional data. The base message and all extensions must be eight-byte aligned. The GET message serves as an example. It requires the base header, the SA extension, and the ADDRESS_DST extension. See the `pf_key`(7P) man pages for details.

Manual Keying Program

You use the `ipseckey`(1M) command to manually manipulate the security association databases with the `ipsecah`(7P) and `ipsecesp`(7P) network security services. You can also use the `ipseckey` command to set up security associations between communicating parties when automated key management is not available.

While the `ipseckey` command has only a limited number of general options, it supports a rich command language. You can specify that requests should be delivered by means of a programmatic interface specific for manual keying. See the `pf_key`(7P) man page for additional information. When you invoke `ipseckey` with no arguments, it enters an interactive mode that displays a prompt enabling you to make entries. Some commands require an explicit security association (SA) type, while others permit you to specify the SA type and act on all SA types.

Security Considerations

The `ipseckey` command handles sensitive cryptographic keying information. It enables a privileged user to enter cryptographic keying information. If an adversary gains access to this information, the adversary can compromise the security of IPsec traffic. You should take the following issues into account when using the `ipseckey` command:

1. Is the TTY going over a network (interactive mode)?
   - If it is, then the security of the keying material is the security of the network path for this TTY's traffic. You should avoid using `ipseckey`(1M) over a clear-text telnet or rlogin session.
   - Even local windows might be vulnerable to attacks by a concealed program that reads window events.
2. Is the file accessed over the network or readable to the world (`–f` option)?
   - An adversary can read a network-mounted file as it is being read. You should avoid using a world-readable file with keying material in it.
   - If your source address is a host that can be looked up over the network, and your naming system is compromised, then any names used are no longer trustworthy.

Security weaknesses often lie in misapplication of tools, not the tools themselves. You should be cautious when using `ipseckey`. Use a console for the safest mode of operation, or other hard-connected TTY.

IPsec Extensions to Existing Utilities

ifconfig

To support IPsec, the following security options have been added to `ifconfig`(1M):

- `auth_algs`
- `encr_auth_algs`
- `encr_algs`

auth_algs

This option enables IPsec AH for a tunnel, with the authentication algorithm specified. It has the following format:

```
auth_algs authentication_algorithm
```

The algorithm can be either a number or an algorithm name, including the parameter *any*, to express no specific algorithm preference. You must specify all IPsec tunnel properties on the same command line. To disable tunnel security, specify the following option:

```
auth_alg none
```

encr_auth_algs

This option enables IPsec ESP for a tunnel, with the authentication algorithm specified. It has the following format:

```
encr_auth_algs authentication_algorithm
```

For the algorithm, you can specify either a number or an algorithm name, including the parameter *any*, to express no specific algorithm preference. If you specify an ESP encryption algorithm, but you do not specify the authentication algorithm, the ESP authentication algorithm value defaults to the parameter, *any*.

encr_algs

This option enables IPsec ESP for a tunnel with the encryption algorithm specified. It has the following format:

```
encr_auth_algs encryption_algorithm
```

For the algorithm, you can specify either a number or an algorithm name. You must specify all IPsec tunnel properties on the same command line. To disable tunnel security, specify the following option:

```
encr_alg none
```

If you specify an ESP authentication algorithm, but not encryption algorithm, the ESP encryption value defaults to the parameter *null*.

snoop(1M)

The snoop command can now parse AH and ESP headers. Since ESP encrypts its data, snoop cannot see encrypted headers protected by ESP. AH does not encrypt data, so traffic can still be inspected with snoop. The snoop –V option can show when AH is in use on a packet. See the snoop(1M) man page for more details.

CHAPTER 19

Implementing IPsec

This chapter provides procedures for implementing IPsec on your network.

For overview information about IPsec, see Chapter 18. The `ipsecconf`(1M), `ipseckey`(1M), and `ifconfig`(1M) man pages also describe useful procedures in their respective Examples sections.

Implementing IPsec Task Map

TABLE 19–1 Implementing IPsec Task Map

Task	Description	For Instructions, Go To ...
Secure traffic between two systems	Involves adding addresses to the `/etc/hosts` file, editing the `/etc/inet/ipsecinit.conf` file, adding security associations, and invoking the `ipsecinit.conf` file.	"How to Secure Traffic Between Two Systems" on page 386
Secure a Web server using IPsec policy	Involves enabling only secure traffic by editing the `ipsecinit.conf` file and invoking this file.	"How to Secure a Web Server Using IPsec Policy" on page 389

TABLE 19–1 Implementing IPsec Task Map *(continued)*

Task	Description	For Instructions, Go To ...
Set up a virtual private network	Involves turning off IP forwarding, turning on IP strict destination multihoming, disabling most network and Internet services, adding security associations, configuring a secure tunnel, turning on IP forwarding, configuring default route, and running the routing protocol.	"How to Set Up a Virtual Private Network" on page 390
Replace current security associations	Involves flushing current security associations and entering new ones.	"How to Replace Current Security Associations" on page 395

IPsec Tasks

This section provides procedures that enable you to secure traffic between two systems, secure a Web server using IPsec policy, and set up a virtual private network.

▼ How to Secure Traffic Between Two Systems

This procedure assumes that you are invoking AH protections using any algorithm. It also assumes you want security associations shared (that is, only one pair of SAs are needed to protect the two systems) and that each system has only one IP address.

1. **Become superuser on the system console.**

   Note - Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

2. **On each system, add the address and host name for the other system in the `/etc/hosts` file. You can do this using the following command:**

   a. **On System 1:**

```
# echo "system2_addr system2_name" >> /etc/hosts
```

b. On System 2:

```
# echo "system1_addr system1_name" >> /etc/hosts
```

This enables the boot scripts to use the system names without depending on non-existent naming services.

3. On each system, edit the `/etc/inet/ipsecinit.conf` file by adding the following lines:

a. On System 1:

```
{saddr system1_name daddr system2_name}
apply {auth_algs any sa shared}
{saddr system2_name daddr system1_name}
permit {auth_algs any}
```

b. On System 2:

```
{saddr system2_name daddr system1_name}
apply {auth_algs any sa shared}
{saddr system1_name daddr system2_name}
permit {auth_algs any}
```

4. Add Security Associations using the following substeps:

a. On each system, create a read-only (600 permissions) *keyfile*, using the file name of your choice, say `MyKeyfile`, and type the following lines in this file:

```
add ah spi random-number dst system1_name authalg algorithm_name \
     authkey random-hex-string-of-algorithm-specified-length
add ah spi random-number dst system2_name authalg algorithm_name \
     authkey random-hex-string-of-algorithm-specified-length
```

b. On each system, enable the security associations by typing the following command:

```
# ipseckey -f keyfile
```

5. **On each system, do one of the following steps:**

   a. **Invoke the** `ipsecinit.conf` **file by typing the following command:**

   ```
   # ipsecconf -a /etc/inet/ipsecinit.conf
   ```

   b. **Or reboot both systems.**

   If you reboot both systems, you must first insert the following command (used in step 4) in a boot script:

   ```
   ipseckey -f keyfile
   ```

   To do so, continue with the following steps.

   c. **Change the *keyfile* name to** `ipseckey` **by typing the following command:**

   ```
   # cp keyfile /etc/inet/ipseckey
   ```

   d. **Make the** `ipseckey` **file read-only by typing the following command:**

   ```
   # chmod 600 /etc/inet/ipseckey
   ```

   e. **Create a boot script,** `/etc/rc3.d/s99ipsec_setup`**, which contains the following code:**

```
if [ -f /etc/inet/ipseckeys -a -f /etc/inet/ipsecinit.conf ]; then
     /usr/sbin/ipseckey -f /etc/inet/ipseckeys
fi
```

On subsequent reboots the `/etc/inet/ipseckeys` file will be read before booting completes. If you change keys, make sure the file gets changed on both systems.

▼ How to Secure a Web Server Using IPsec Policy

This procedure includes bypasses for Web traffic served on the Web server and DNS client requests from this Web server. All other traffic requires ESP with 3DES and SHA-1 algorithms and uses a shared SA for outbound traffic, so as not to require too many security associations.

1. **Become superuser on the system console.**

Note - Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

---

2. **Determine which services need to bypass security policy checking.**

   For a Web server, this includes TCP ports 80 (HTTP) and 443 (Secure HTTP). If the Web server does DNS name lookups, it might also need to include port 53 for both TCP and UDP.

3. **Create a read-only file, using the file name of your choice, say** `MyIPsecInitFile`**, and type the following lines in this file:**

```
# Web traffic that Web server should bypass.
 {sport 80 ulp tcp} bypass {dir out}
 {dport 80 ulp tcp} bypass {dir in}
 {sport 443 ulp tcp} bypass {dir out}
 {dport 443 ulp tcp} bypass {dir in}

 # Outbound DNS lookups should also be bypassed.
 {dport 53} bypass {dir out}
 {sport 53} bypass {dir in}

 # Require all other traffic to use ESP with 3DES and SHA-1.
 # Use a shared SA for outbound traffic, so as not to require a
 # large supply of security associations.
 {} permit {encr_algs 3des encr_auth_algs sha}
 {} apply {encr_algs 3des encr_auth_algs sha sa shared}
```

   This enables only secure traffic to access the system, with the bypass exceptions listed in the previous step.

4. **Do either one of the following two substeps:**

   a. **Copy the file you created in the previous step into** `/etc/inet/ipsecinit.conf` **and reboot using the following commands:**

```
# cp filename /etc/inet/ipsecinit.conf
# reboot
```

b. Invoke the file you created using the following command:

```
ipsecconf -a filename
```

Note - These steps are possible because the file has no name service requirements. Also, when invoking `ipsecconf`, existing TCP connections do not fall under the IPsec policy. A warning is issued by the `ipsecconf` command to this effect.

The Web server now allows only Web-server traffic, as well as outbound DNS requests and replies. No other services will work without adding security associations using `ipseckey`(1M) and enabling IPsec on the remote system.

▼ How to Set Up a Virtual Private Network

This procedure shows you how to set up a VPN using the Internet to connect two networks within an organization, and secure the traffic between the networks with IPsec. This procedure assumes that the networks' `le1` interfaces are *inside* the VPN, and the `le0` interfaces are *outside* the VPN on the two systems that implement the VPN link.

The procedure also uses ESP with DES and MD5. The algorithms used affect the key lengths, 64 bits (56 bits+ 8 bits parity) for DES and 128 bits for MD5. You must do the following procedure on the two systems that act as the gateway through the Internet. For a descripton of VPNs, see "Enabling Virtual Private Networks" on page 378.

1. Become superuser on the system console.

Note - Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

---

2. **Turn off IP forwarding by typing the following command:**

```
# ndd -set /dev/ip ip_forwarding 0
```

Turning off IP forwarding prevents packets from being forwarded from one network to another through this system.

3. **Turn on IP strict destination multihoming by typing the following command:**

```
# ndd -set /dev/ip ip_strict_dst_multihoming 0
```

Turning on IP strict destination multihoming ensures that packets for one of the system's destination addresses arrives on the interface to which that address is assigned.

When you use the `ndd`(1M) command to turn off IP forwarding and turn on IP strict destination, multihoming shuts down the flow of packets except to the system itself, and then only if the packets arrive on the interface that corresponds to the destination IP address.

4. **Disable most (if not all) network services on the Solaris system by doing the following substeps, as needed:**

Note - The VPN router should allow very few incoming requests. You need to disable all processes that accept incoming traffic (for example, comment out lines in the `inetd.conf` file, kill SNMP, and so on), or use techniques similar to the "How to Secure a Web Server Using IPsec Policy" on page 389.

---

a. **If `inetd.conf` has been edited to remove all but essential services, type the following command:**

```
# pkill -HUP inetd
```

b. **If `inetd.conf` has not been edited to remove all but essential services, type the following command on a command line:**

```
# pkill inetd
```

c. **Disable other Internet services, such as SNMP, NFS, and so on, by typing one or more commands like the following examples, as needed:**

```
# /etc/init.d/nfs.server stop
# /etc/init.d/sendmail stop
```

Disabling network services keeps IP packets from doing any harm to the system. For example, an SNMP daemon, telnet, or rlogin could be exploited.

5. On each system, add a pair of security associations between the two systems by doing the following substeps:

a. Type the following command:

```
# ipseckey
```

This enables the `ipseckey` command mode.

b. At the `ipseckey` command mode prompt, type the following command:

```
> add esp spi random-number src system1_addr dst system2_addr \
auth_alg md5 encr_alg des \
authkey very-random-hex-string-of-32-characters \
encrkey very-random-hex-string-of-16-characters
```

c. Press the Return key.

This executes the command and redisplays the `ipseckey` command mode prompt.

d. At the `ipseckey` command mode prompt, type the following command:

```
> add esp spi random-number src system2_addr dst system1_addr \
auth_alg md5 encr_alg des \
authkey very-random-hex-string-of-32-characters \
encrkey very-random-hex-string-of-16-characters
```

Note - The keys and SPI can and *should* be different for each security association.

---

e. At the `ipseckey` **command mode prompt, type Ctrl-D or Exit to exit this mode.**

6. Configure a secure tunnel, `ip.tun0`**, by doing the following substeps:**

a. On System 1 type the following commands:

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 system1-taddr system2-taddr \
tsrc system1-addr tdst system2-addr encr_algs des encr_auth_algs md5

# ifconfig ip.tun0 up
```

b. On System 2 type the following commands:

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 system2-taddr system1-taddr \
tsrc system2-addr tdst system1-addr encr_algs des encr_auth_algs md5

# ifconfig ip.tun0 up
```

This sets up the secure tunnel and adds another physical interface from the IP perspective.

7. On each system, turn on (in this example) `le1:ip_forwarding` **and** `ip.tun0:ip_forwarding` **by typing the following commands:**

```
# ndd -set /dev/ip le1:ip_forwarding 1

# ndd -set /dev/ip ip.tun0:ip_forwarding 1
```

`ip_forwarding` means that packets arriving off an interface can be forwarded, *and* packets leaving this interface might have originated on another interface. To

successfully forward a packet, both the receiving and transmitting interfaces must have their *ip_forwarding* turned on.

Since `le1` is *inside* the Intranet, and `ip.tun0` connects the two systems through the Internet, `ip_forwarding` must be turned on for these two interfaces.

The `le0` interface still has its `ip_forwarding` turned off. This prevents someone on the *outside* (that is, the Internet) from injecting packets into the protected Intranet.

8. On each system, ensure that routing protocols do not advertise the default route within the Intranet by typing the following command:

```
# ifconfig le0 private
```

While `le0` has `ip_forwarding` turned off, any routing protocol implementation (for example, `in.routed`) might still advertise that `le0` is a valid interface for forwarding packets to its peers inside the Intranet. Setting the interface's *private* flag helps reduce these advertisements.

9. On each system, manually add a default route over `le0` by typing the following commands:

```
# pkill in.rdisc
# route add default router-on-le0-subnet
```

Even though `le0` is not part of the Intranet, it does need to reach across the Internet to its peer machine. To do this, Internet routing information is needed. The VPN system looks like a host (as opposed to a router) to the rest of the Internet, so either using a default route or running router discovery is sufficient.

10. Prevent `in.rdisc` from restarting when the system is rebooted by doing the following substeps:

a. Put the IP address of the default router on the `le0` subnet in the file `/etc/defaultrouter`.

This stops `in.rdisc` from being started at reboot.

b. Type the following command.

```
# touch /etc/notrouter
```

This keeps the vulnerability down by preventing routing to occur early in the boot sequence.

c. Edit the `/etc/hostname.ip.tun0` file and add the following lines.

```
system1-taddr system2-taddr tsrc system1-addr \
     tdst system2-addr encr_algs des encr_auth_algs md5
```

d. Edit the `/etc/rc3.d/S99ipsec_setup` file and add the following lines in the if/then statement at the end.

```
ndd -set /dev/ip le1:ip_forwarding 1
ndd -set /dev/ip ip.tun0:ip_forwarding 1
ifconfig le0 private
in.routed
```

The file should look like:

```
if [ -f /etc/inet/ipseckeys -a -f /etc/inet/ipsecinit.conf ]; then
     /usr/sbin/ipseckey -f /etc/inet/ipseckeys
     ndd -set /dev/ip le1:ip_forwarding 1
     ndd -set /dev/ip ip.tun0:ip_forwarding 1
     ifconfig le0 private
     in.routed
fi
```

11. On each system, run a routing protocol by typing the following command:

```
# in.routed
```

To prevent an adversary from having too much time to break your cryptosystem, you need to replace periodically the security associations that you invoked in step 2 with new ones. Use the following procedure to replace your current security associations.

▼ How to Replace Current Security Associations

This procedure enables you to replace current security associations. You should do this procedure periodically so that an adversary has less time to break your cryptosystem.

1. **Become superuser on the system console.**

Note - Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

---

2. **On each system, flush your current security associations by doing the following substeps:**

   a. **Type the following command:**

   ```
   # ipseckey
   ```

   This enables the `ipseckey` command mode.

   b. **At the `ipseckey` command mode prompt, enter the following command:**

   ```
   > flush
   ```

3. **Do step 5 in the "How to Set Up a Virtual Private Network" on page 390 procedure to set new security associations by changing the values of SPI and keys.**

CHAPTER **20**

Modem–Related Network Services Topics

Chapter 21	Provides overview information for PPP
Chapter 22	Provides planning information for PPP
Chapter 23	Provides step-by-step instructions for setting up and troubleshooting PPP
Chapter 24	Provides reference information for working with PPP databases and files
Chapter 25	Provides background information on UUCP
Chapter 26	Provides step-by-step instructions for setting up and troubleshooting UUCP
Chapter 27	Provides reference material on UUCP database files, UUCP configuration files, UUCP shell scripts, and UUCP troubleshooting information

CHAPTER 21

Overview of PPP

This chapter presents an overview of Solaris Point-to-Point Protocol (PPP), a data-link protocol included in the TCP/IP protocol suite. The text includes product specifications, introductions to the most typical PPP configurations, and definitions of the terms related to PPP.

Overview of Solaris PPP

PPP enables you to connect computers and networks at separate physical locations by using modems and telephone lines. With PPP, users with computers at home or in remote offices can connect to your site's network. You can also use the combination of PPP software, a modem, and telephone lines as a router connecting networks in different places. PPP offers strategies for configuring these machines and networks, which are introduced in this chapter.

Solaris PPP Specifications

Solaris PPP is an asynchronous implementation of the standard data-link level PPP included in the TCP/IP protocol suite and provided by a number of router system

vendors and terminal concentrators. It includes a standard encapsulation protocol, making datagram transmission transparent to network layer protocols.

The major characteristics of the Solaris PPP protocol are:

- Implements the Internet Point-to-Point Protocol, as defined in RFC 1331
- Provides error detection through CRC
- Supports full–duplex transmission

The major functions of the protocol are:

- Interface for IP to forward packets over asynchronous serial lines
- Connection establishment on demand
- Configurable options negotiation
- Connection termination (automatic hang-up)

Transmission Facilities Used by PPP

PPP supports interfaces to RS-232-C (V.24) facilities through the CPU serial ports included on most machines running the Solaris software. In addition, PPP runs over optional asynchronous serial ports supplied or supported by many manufacturers of machines that run the Solaris software. PPP supports the maximum data rates that your machine's serial ports can achieve. Consult the manufacturer of your computer system for more details on the speeds supported by your machine's serial hardware.

Standards Conformance

PPP, and the routing functions in the Solaris software, use industry-standard conventions for performing their tasks. These conventions support:

- Forwarding of IP datagrams
- Reception of packets for forwarding from any IP-compatible networked system
- Delivery of packets to any IP-compatible networked system on local-area network media, such as Ethernet, Token Ring, and FDDI
- Use of standard routing protocols, enabling users to exchange packets with equipment that supports the PPP protocol from many manufacturers

PPP Network Interfaces

PPP enables asynchronous devices, such as modems, to become network interfaces. Solaris PPP enables you to configure two types of *virtual network interfaces*, `ipdptp`*n* and `ipd`*n*. (The letter *n* represents the device number you assign to the interface.)

PPP network interfaces are considered virtual network interfaces because they do not involve network hardware, as does, for example, an Ethernet interface. Moreover, they are not associated with any particular serial port. The PPP network interfaces reside in the `/devices` directories along with the physical network interfaces. (For information on physical network interfaces, see “Network Interfaces” on page 56.)

The type of network interface you use depends on the PPP communications link you want to set up. The `ipdptp` interface supports point-to-point PPP links; the `ipd` interface supports point-to-multipoint links (called “multipoint links”).

Extending Your Network With PPP

This section introduces PPP-related communications concepts. It also explains the most typical PPP configurations that you are likely to set up.

Point-to-Point Communications Links

The most common use of Solaris PPP is to set up a point-to-point communications link. A generic point-to-point communications configuration consists of two endpoints connected by a communications link. In a generic configuration, an *endpoint* system could be a computer or terminal, either in an isolated location or physically connected to a network. The term *communications link* refers to the hardware and software connecting these endpoint systems. The following figure illustrates these concepts.

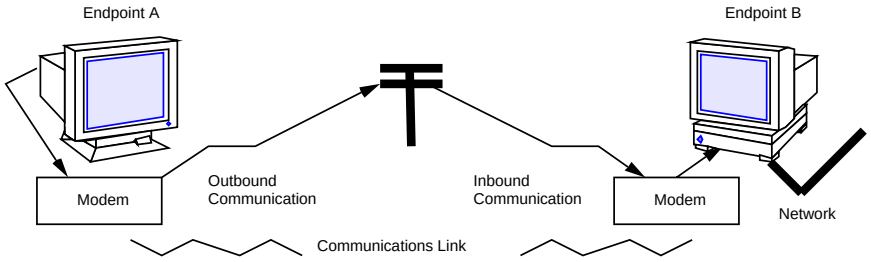


Figure 21–1 Basic Point-to-Point Link

Dial-out Operations and Outbound Communications

When an endpoint system wants to communicate with the endpoint on the other side of the communications link, it begins a *dial-out operation*. For example, to communicate with endpoint B, a user at its peer host, endpoint A, types `rlogin` *end-point-B*. This causes endpoint A to dial out over the communications link. In this instance, endpoint A functions as a *dial-out* machine. The `rlogin` command causes its modem to dial the phone number of endpoint B. The action endpoint A starts and information it passes are considered *outbound communications*.

Dial-ins and Inbound Communications

When the data travels over the link to endpoint B, this system receives incoming data and sends an acknowledgment signal to endpoint A to establish communications. In this instance, endpoint B functions as a *dial-in* machine, since it permits other systems to dial in to it. The information passed to the communications recipient and the actions the recipient takes are considered *inbound communications*.

Point-to-Point Configurations Supported by Solaris PPP

Solaris PPP supports four types of point-to-point configurations:

- Host in one location connected to a host at another physical location, as shown in Figure 21–1
- Dial-in servers with dynamic point-to-point links to remote hosts, as shown in Figure 21–2
- Network connected to another physically distant network, as shown in Figure 21–3
- Computers connected to a multipoint dial-in server physically attached to a distant network, as shown in Figure 21–4

These PPP links provide essentially the same type of connectivity provided by a local area network but without broadcast capability. The following sections summarize the

configuration types; Chapter 22 gives information for setting up each configuration type.

Two Isolated Hosts Connected by a Point-to-Point Link

PPP enables you to set up a point-to-point link to connect two standalone machines in separate locations, effectively creating a network consisting solely of these two machines. This is the simplest point-to-point configuration because it involves only the two endpoints. The generic configuration shown in Figure 21–1 also uses the host-to-host configuration.

Nomadic Machines Connected to a Dial-in Server

In the past, standard dial-up or temporary connections permitted only ASCII terminals to connect to a network. With Solaris PPP, an individual machine can become part of a physically distant network by configuring it as one endpoint of the PPP link. The advantage of this *nomadic* connection is particularly apparent if your network includes users who travel frequently or work from home.

Figure 21–2 shows nomadic computers, each with a point-to-point link to an endpoint system on the network. The endpoint on the network is a dial-in server.

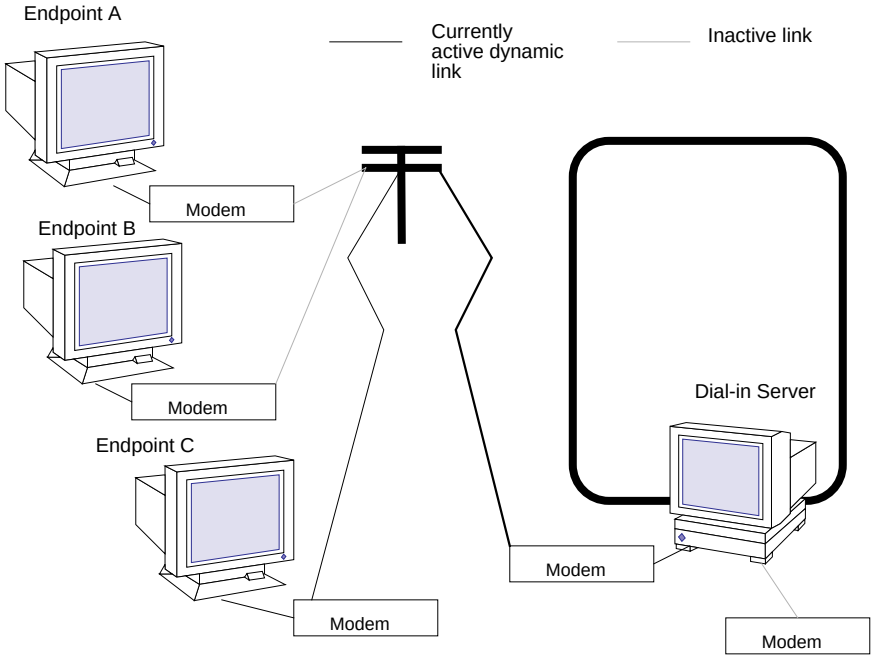


Figure 21–2 Nomadic Computers and Dynamic Link Dial-in Server

Dial-in Server With Dynamic Point-to-Point Link

The endpoint machine on the network shown in Figure 21–2 functions as a dial-in server with dynamic point-to-point links. It is called a *dial-in server* because remote machines can dial in to it to reach the network. When the server receives a request to dial in from a machine, the server allocates the PPP link to the machine on an as-needed basis.

A dial-in server can communicate with the remote hosts through a dynamic point-to-point link or through a multipoint link, as explained in “Multipoint Communications Links” on page 405. The dynamic point-to-point link has the advantages of point-to-point communications: RIP can run over the link, and broadcasting is enabled. Perhaps most importantly, more than one machine on the physical network can function as the dial-in server. This allows you to configure backup servers, thus enabling redundancy and easier administration. Although the machines in Figure 21–2 can directly communicate with the network endpoint, they cannot directly communicate with each other. They must pass information to each other through the dial-in server endpoint.

Two Networks Connected by Point-to-Point Link

You can use PPP to connect two separate networks through a point-to-point link, with one system on each network serving as an endpoint. These endpoints communicate through modems and phone lines, essentially in the same fashion as shown in Figure 21–1. But in this setup, the endpoints, modems, and PPP software become routers for their physical networks. Using this type of configuration scheme, you can create an internetwork with wide geographic reach.

The following figure shows two networks in different locations connected by a point-to-point link.

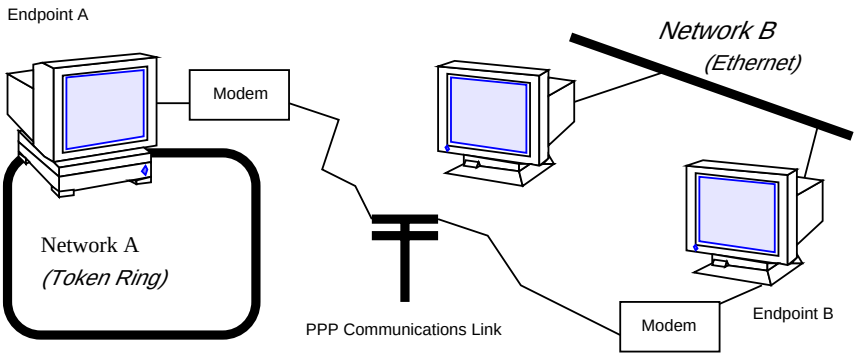


Figure 21–3 Two Networks Connected by a PPP Link

In this example, endpoints A and B, their modems, public telephone lines, and the PPP software act as a router between the networks. These networks might have other hosts serving as routers between physical networks. Sometimes, the host functioning

as the PPP router might have an additional network interface board, thus also serving as a router for a physical network.

Multipoint Communications Links

You can use Solaris PPP to set up a multipoint communications link. In this type of configuration, an individual machine functions as one endpoint on the communications link. At the other end of the link might be several endpoint machines. This differs from point-to-point configurations, with a single endpoint system at either side of the communications link.

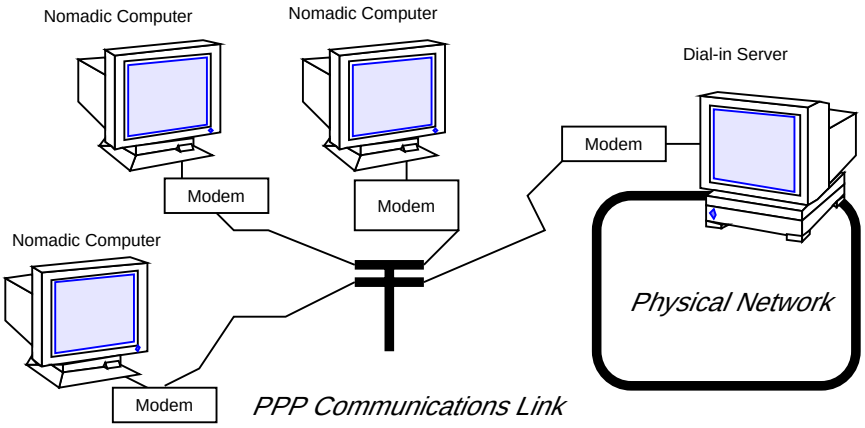


Figure 21–4 Nomadic Computers and Multipoint Dial-in Server

Multipoint Configurations Supported by PPP

Two types of multipoint links you can configure with PPP are:

- Dial-in server with multipoint connections to remote machines, as shown in Figure 21–4
- Logical, or *virtual*, network consisting of three or more nomadic computers, as shown in Figure 21–5

The following sections summarize these configurations; Chapter 22 explains how to set up the configuration.

Multipoint Dial-in Servers

Figure 21–3 shows three geographically isolated computers communicating through a point-to-point link to an endpoint machine on a network. However, the network

endpoint machine can communicate with the nomadic computers through a *multipoint* link, thus making it a multipoint dial-in server. (You can also set up a dial-in server with dynamic point-to-point connections, as explained in “Dial-in Server With Dynamic Point-to-Point Links” on page 416.)

The dial-in server can communicate with all the machines on the other end of its multipoint PPP link. Though the machines in Figure 21–4 can directly communicate with the multipoint dial-in server, they cannot communicate directly with each other. They must pass information to each other through the dial-in server.

Virtual Networks

You can use PPP to set up a *virtual network* wherein the modems, PPP software, and telephone wires become the “virtual” network media. In a physical network, such as Ethernet or Token Ring, computers are directly cabled to the network media. In a virtual network, no true network media exist.

Machines become peer hosts on the virtual network when you configure each with a multipoint communications link. Then each host can dial out through its modem over phone lines to reach another endpoint machine. Each computer also functions as a dial-in machine, permitting its peer hosts on the virtual network to dial in to it.

The following figure depicts a virtual network consisting of nomadic computers connected to each other through modems and telephone lines.

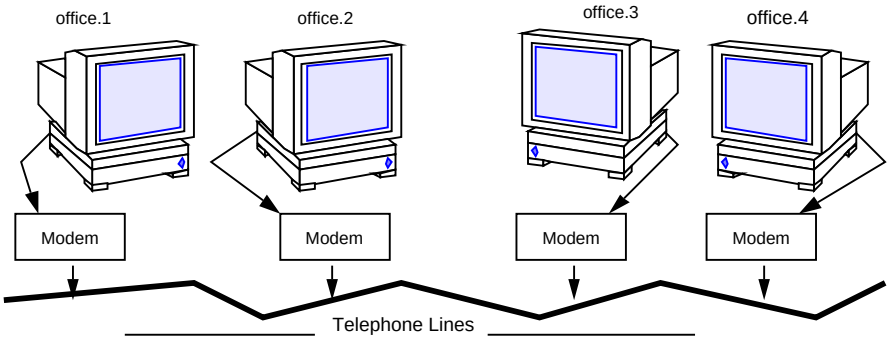


Figure 21–5 Virtual Network of Nomadic Computers

Each machine exists in a different office, perhaps in a different town from other members of the virtual network. However, each machine can establish communications with its peer hosts over its multipoint communications links.

Introducing the PPP Software

The PPP component software includes:

- Link manager (`/usr/sbin/aspppd`)
- Login service (`/usr/sbin/aspppls`)
- Configuration file (`/etc/asppp.cf`)
- Log file (`/var/adm/log/asppp.log`)
- FIFO file (`/tmp/.asppp.fifo`)

After you install the PPP software, you will find the `/etc/init.d/asppp` file, which is the run-control script for PPP. It is linked to several other files in the run-control directories.

The following figure shows the software components of PPP and how they interact.

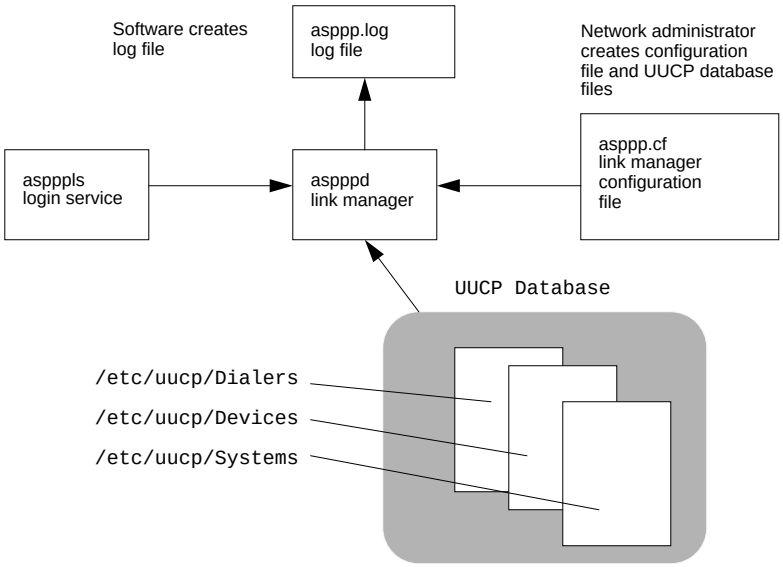


Figure 21–6 PPP Component Software

Link Manager

The `/usr/sbin/aspppd` link manager is a user-level daemon that automates the process of connecting to a remote host when PPP service is required. This automated process starts whenever any activity that generates IP traffic takes place (for example, a user logs in to a remote machine, accesses an NFS mounted file, and so on). If a

remote host tries to establish a connection, the link manager on the local host will complete the connection.

Refer to the `aspppd`(1M) man page for specific information about the link manager.

Login Service

The `/usr/sbin/aspppls` login service is invoked as a login shell that starts PPP after you dial up and log in. Its function is similar to the `/usr/lib/uucp/uucico` command described in “UUCP Software” on page 476. When configuring a machine as a dial-in server, you must specify `aspppls` as the login shell in the `/etc/passwd` file in the entries for every nomadic computer allowed to dial in to the local host.

Configuration File

The `asppp.cf` file provides the link manager with information about each remote endpoint with which the local host will communicate. You define this information in a section of the configuration file called a *path*. The path section also defines the PPP interface to be used and, optionally, other attributes determining how communications will take place, including security issues. “Parts of Basic Configuration File” on page 449 explains the sections of the `asppp.cf` file in detail. The following example shows an unmodified `asppp.cf` file.

EXAMPLE 21–1 Unmodified `asppp.cf` File

```
#ident "@(#)asppp.cf 10 93/07/07 SMI"
#
# Copyright (c) 1993 by Sun Microsystems, Inc.
#
# Sample asynchronous PPP /etc/asppp.cf file
#
#

ifconfig ipdptp0 plumb mojave gobi private up

path
   inactivity_timeout 120    # Approx. 2 minutes
   interface ipdptp0
   peer_system_name Pgobi    # The name this system logs in with when
                             # it dials this server
                             # *OR* the entry we look up in
                             # /etc/uucp/Systems when we dial out.
```

Log File

The link manager produces messages and logs them in the log file `/var/adm/log/asppp.log`. The level of detail reported into the file is controlled by the `-d` option of `aspppd` or the `debug_level` keyword in the configuration file. See “Configuration Keywords” on page 471 and the `aspppd`(1M) man page for more information.

FIFO File

The PPP FIFO file `/tmp/.asppp.fifo` is a named pipe used to communicate between `aspppd` and `asppp1s`. This file must be present in `/tmp` for the PPP login service to connect to the link manager. The `/tmp/.asppp.fifo` file is created, managed, and deleted by the link manager.

UUCP Databases

Besides its component software, Solaris PPP uses information in three UUCP files, `/etc/uucp/Systems`, `/etc/uucp/Dialers`, and `/etc/uucp/Devices`, to help it establish the communications link. You must modify these files to enable a host to dial out over the PPP link. Alternatively, you can use the file `/etc/uucp/Sysfiles` to specify different names for the `Systems`, `Devices`, and `Dialers` files.

Refer to Chapter 25 for full descriptions of the UUCP files.

How the Components Work Together

This section describes how the components of PPP function for outbound and inbound connections.

Outbound Connections Scenario

Outbound communications begin when a user on one endpoint host initiates an activity involving the peer host on the other end of the PPP link. The following activities take place when a user types an `rcp` command to copy a file from a host on the other side of the link.

1. `rcp` sends the data through the levels of the TCP/IP protocol stack.
2. A virtual network interface (`ipd`*n* or `ipdptp`*n*) receives the data in the form of IP packets.

3. The interface sends the `aspppd` link manager a connection request that initiates an outbound connection.
4. The link manager then:
   a. Verifies that the connection request corresponds to a configured path in the `/etc/asppp.cf` configuration file.
   b. Consults the UUCP database files (`/etc/uucp/Systems`, `/etc/uucp/Devices`, and `/etc/uucp/Dialers`) for specific information about the modem and destination system.
   c. Places a phone call to the destination host or attaches to the appropriate hardwired serial line.
5. The physical link to the peer host is established.
6. The link manager configures and initiates PPP.
7. The data-link layer is established, and the PPP modules on the peer host start communicating.
8. The link manager enables IP over the link.

The link manager then monitors the connection until an event, such as an idle timeout, line disconnect, or error condition, occurs. When any of these events occurs, the link manager disconnects from the peer host and returns to the idle state.

Inbound Connections Scenario

The host initiating the inbound communication logs in, which invokes the `/usr/sbin/aspppls` login service. Then the following events occur:

1. The login service connects to the link manager through the `/tmp/.asppp.fifo` file.
2. The login service provides the link manager with information such as the login name used by the endpoint at the other end of the link.
3. The link manager uses this login name to find a corresponding configured path in the configuration file.
4. The link manager then configures and initiates PPP.
5. The data-link layer is established, and the PPP modules on the peer hosts start communicating.
6. The link manager enables IP over the link.

The link manager then monitors the connection until an event occurs such as an idle time out, line disconnect, or error condition. When any of these events occur, the link manager disconnects from the peer and returns to the idle state.

PPP Security

After you have completed installing PPP on every machine involved in your configuration, you can add either one or two levels of security for the PPP link.

The first level, Password Authentication Protocol (PAP), is the least secure. A password is sent over the circuit “in the clear” until authentication is acknowledged or the connection terminated.

The second level of security, Challenge-Handshake Authentication Protocol (CHAP), periodically verifies the identity of the peer—the other end of the point-to-point link. A challenge message is sent to the peer by the authenticator—the system starting the link or challenge. The response is checked against a “secret” not sent over the link, and if the values match, authentication is acknowledged. Otherwise, the link is terminated. The process of adding PPP security is described in “Editing `asppp.cf` for PAP/CHAP Security” on page 436.

CHAPTER 22

Planning for PPP

Before configuring the PPP software, you need to prepare the hardware and software involved and gather some information that is needed during the configuration process. This chapter explains many of the tasks you need to perform prior to configuration, such as:

- “Determining Requirements for Your Configuration Type” on page 413
- “Determining IP Addressing for Your PPP Link” on page 418
- “Routing Considerations” on page 421
- “PPP Hardware Requirements” on page 421
- “Checklist for Configuring PPP” on page 421

The chapter concludes with a checklist to help you organize required information before you configure your PPP link (see Table 22–1).

Determining Requirements for Your Configuration Type

Solaris PPP supports many configuration options, including:

- Remote computer-to-network over a point-to-point link
- Remote computer-to-remote computer over a point-to-point link
- Network-to-network over a point-to-point link
- Dial-in server-to-multiple remote computers through one or more dynamic point-to-point links
- Dial-in server-to-multiple remote computers through a multipoint link

- Multiple remote computers comprising a virtual network, all communicating through multipoint links

These configurations are introduced in “Extending Your Network With PPP” on page 401.

This section describes the information you need to gather and tasks you have to perform for each configuration type before beginning the configuration process. Read the section that describes the configuration you want to set up.

Areas you need to consider are:

- Network interface
- Addressing method
- Name service used, if any
- Dial-in as well as dial-out support
- Routing requirements

Remote Computer-to-Network Configuration

The remote computer-to-network is the most common asynchronous PPP configuration. Use it to configure machines in remote offices or users’ homes that dial out over a point-to-point PPP link to a dial-in server on a network.

- **Network interface** – This point-to-point link uses the `ipdptp`*n* virtual network interface. You need to specify it in the configuration files of all remote machines that dial out to a network.
- **Addressing method** – The configuration file must include the host names or IP addresses of the machines that communicate over the link. For remote hosts, you should use existing host names and IP addresses. Refer to “Determining IP Addressing for Your PPP Link” on page 418 for complete details.
- **Name service** – NIS and NIS+ name services are not recommended for remote hosts. These services generate a great deal of network traffic, often at unexpected times. The DNS name service is more efficient for this type of configuration. You might want to set up DNS, as described in *Solaris Naming Administration Guide*, on each remote host. If you don’t use DNS, PPP accesses the `/etc/inet/hosts` file on the remote machine.
- **Dial-in and dial-out support** – Remote hosts usually implement dial-out communications only. They do not allow other machines to dial in to them directly. Therefore, you must update the UUCP files on each to support dial-out communications, as explained in “Editing UUCP Databases” on page 430.
- **Routing requirements** – Because RIP is part of the Solaris TCP/IP protocol stack, it runs by default on remote hosts. Turn off RIP to improve performance, if necessary, and instead use static routing. See “Routing Protocols” on page 143 and “Turning Off RIP” on page 433 for details.

Remote Host-to-Remote Host Configuration

Use the host-to-host configuration to establish point-to-point communications between two remote hosts in different physical locations. This configuration is useful for two standalone machines in remote offices that need to exchange information. No physical network is involved.

- **Network interface** – This basic point-to-point link uses the `ipdptp`*n* virtual network interface. You must specify the interface in the configuration files of both endpoints.
- **Addressing method** – The configuration file must include the host names or IP addresses of the machines that can communicate over the link. Use the existing host names and the IP addresses assigned to the primary network interface, if they already exist. Otherwise, create IP addresses for the endpoints. Refer to "Determining IP Addressing for Your PPP Link" on page 418 for complete details.
- **Name service** – Because only two peer hosts are involved, you don't need a true name service. The `/etc/inet/hosts` files on both peer hosts are used for address resolution.
- **Dial-in and Dial-out support** – Both machines need to perform dial-in and dial-out operations. You must modify the UUCP databases and `/etc/passwd` on both endpoints.
- **Routing requirements** – Because RIP is part of the Solaris TCP/IP protocol stack, it runs by default on remote hosts. Turn off RIP to improve performance, if necessary, and instead use static routing. See "Routing Protocols" on page 143 and "Turning Off RIP" on page 433 for details.

Network-to-Network Configuration

Use the network-to-network PPP configuration to create an internetwork joining two networks in physically separate locations. In this case, modems and PPP software function as the router connecting the networks.

- **Network interface** – The point-to-point link uses the `ipdptp`*n* virtual network interface. You must specify `ipdptp`*n* in the configuration files for both endpoint machines joining the two networks.
- **Addressing method** – The configuration file must include the host names or IP addresses of the machines that communicate over the link. Two possible addressing scenarios exist for this type of configuration; they are explained in "Determining IP Addressing for Your PPP Link" on page 418.
- **Name service** – NIS and NIS+ name services can function over this type of PPP link; however, each network should be a separate domain. If you use DNS, both networks can be part of a single domain. Refer to *Solaris Naming Administration Guide* for details. If you use local files for name service, the `/etc/inet/hosts` files on both endpoint machines are used for address resolution. They must

contain the host names and IP addresses of every host on each network that can communicate over the link.

- **Dial-in and Dial-out support** – Both network endpoint machines need to perform dial-in and dial-out operations, so you should update their UUCP and `/etc/passwd` files.
- **Routing requirements** – The endpoints in a network-to-network link usually run RIP in order to exchange routing information. Do not disable RIP for this configuration.

Dial-in Server With Dynamic Point-to-Point Links

A dynamic point-to-point link is one of two types of configurations that you can use for a dial-in server functioning as the network endpoint that remote hosts access. In this configuration scheme, the server connects to its remote hosts over a dynamically allocated point-to-point link. The dial-in server uses its dynamic links on an as-needed basis to establish communications with the remote hosts it serves.

- **Network interface** – The dynamic point-to-point link uses the `ipdptp*` virtual network interface with an asterisk wildcard character. The asterisk enables the link to be allocated dynamically. You must specify this interface in the configuration file.
- **Addressing method** – The configuration file must include the host names or IP addresses of the machines that communicate over the link. Refer to “Determining IP Addressing for Your PPP Link” on page 418 for complete details.
- **Name service** – Although NIS and NIS+ are not recommended for remote hosts, the dial-in server in a remote host-to-network configuration can be a NIS client on the network to which it is physically connected. If NIS is on the server’s physical network, make sure that the NIS maps are updated with the host names and IP addresses of the remote hosts. You can use DNS on the dial-in server and its remote hosts. For more information regarding DNS and name services in general, refer to *Solaris Naming Administration Guide*. If you use local files for name service, PPP access the `/etc/inet/hosts` file on the dial-in server for address resolution.
- **Dial-in support** – You must update the `/etc/passwd` file on the dynamic point-to-point dial-in server. The dynamic link server does not directly dial out to the remote hosts.
- **Routing requirements** – Because RIP is part of the Solaris TCP/IP protocol stack, it runs by default on remote hosts. Turn off RIP to improve performance, if necessary, and instead use static routing. See “Routing Protocols” on page 143 and “Turning Off RIP” on page 433 for details.

Multipoint Dial-in Server

A multipoint link is one of two types of configurations that you can use for a dial-in server functioning as the network endpoint that remote machines can access. In this configuration scheme, the dial-in server connects to multiple remote hosts over the same multipoint link. The remote hosts always connect to the dial-in server over a point-to-point link, as explained in “Remote Computer-to-Network Configuration” on page 414.

Use this configuration when you want to define a separate network of remote hosts and their dial-in server.

- **Network interface** – The multipoint link uses the `ipd`*n* virtual network interface. You must specify this interface in the configuration file for the dial-in server.
- **Addressing method** – The configuration file must include the host names or IP addresses of the machines that communicate over the link. Refer to “Determining IP Addressing for Your PPP Link” on page 418 for complete details. You must create a separate network for the machines on the multipoint link. See “Assigning a Network Number to the PPP Link” on page 420 for more information.
- **Name service** – Although NIS and NIS+ are not recommended for remote hosts, the dial-in server in a remote host-to-network configuration can be a NIS client on the physical network to which it is connected. If NIS is on the server’s physical network, make sure that the NIS maps are updated with the host names and IP addresses of the remote hosts. You can use DNS on the dial-in server and its remote hosts. For more information regarding DNS and name services in general, refer to *Solaris Naming Administration Guide*. If you use local files for name service, PPP uses the `/etc/inet/hosts` file on the dial-in server for address resolution.
- **Dial-in and dial-out support** – The multipoint dial-in server functions as a network router between its PPP virtual network and the physical network to which it is connected. It dials out to its remote hosts whenever it receives IP traffic from the physical network destined for its PPP network. Therefore, you must configure the multipoint dial-in server for both dial-in and dial-out support, and update its UUCP and `/etc/passwd` files.
- **Routing requirements** – The `ipd`*n* interface does not support RIP; you do not need to disable it.

Hosts on a Virtual Network

Use a virtual network configuration to connect three or more physically separated computers into a virtual network of phone lines, modems, and PPP software.

- **Network interface** – This type of configuration requires a multipoint link, which uses the `ipd`*n* virtual network interface. This interface connects each endpoint system with the other endpoints on the virtual network.

- **Addressing method** – The configuration file must include the host names or IP addresses of the machines that communicate over the link. Refer to “Determining IP Addressing for Your PPP Link” on page 418 for more information. You must assign a network number to the virtual network. Refer to “Creating a Unique IP Address and Host Name” on page 419 for complete details.
- **Name Service** – You can run NIS and NIS+ for the virtual network; however, this can affect the performance of the link. DNS is a better alternative. Refer to *Solaris Naming Administration Guide* for instructions on setting up these name services. If you use files for the name service, be sure to update `/etc/inet/hosts` on each machine with the host names and IP addresses of all machines on the virtual network.
- **Dial-in and dial-out support** – All machines in the virtual network must be configured for both dial-in and dial-out operations, so you should update their UUCP and `/etc/passwd` files.
- **Routing requirements** – The `ipd`*n* interface does not support RIP; you do not need to disable it.

Determining IP Addressing for Your PPP Link

To enable communications over the PPP link, the machine at one end of the link must know the host name and IP address of the peer host on the other end of the link. The PPP configurations often require a particular addressing scheme. This section explains the addressing schemes and where each should be used.

Specifying IP Addresses

On each endpoint machine, you specify addressing information in these places:

- `/etc/asppp.cf` configuration file
- `/etc/inet/hosts` file
- NIS+, NIS, or DNS databases, if applicable

When you edit the local machine’s `asppp.cf` file, you must provide the host names and, in certain cases, the IP addresses for each endpoint machine to be on the link. For example, you must type either the IP addresses or host names for each endpoint as arguments in the `ifconfig` section in the configuration file:

```
ifconfig ipdptp0 plumb 192.99.44.01 192.99.44.02 up
```

See “Editing the Configuration File” on page 432 for information regarding the format of /etc/asppp.cf.

Additionally, to enable communications, you must add the IP address and host name of the remote endpoints to the hosts database on the local endpoint by editing /etc/inet/hosts. This process is explained in “Configuring Network Clients” on page 103.

Types of Addressing Schemes

You have a choice of several addressing schemes for PPP, depending on your configuration type. Before you edit the asppp.cf file and hosts database, you must decide on the appropriate addressing scheme for your configuration. These schemes include:

- Using the same IP addresses for the PPP endpoints as is assigned to their primary network interface in their local /etc/inet/hosts files
- Assigning a unique IP address for each PPP endpoint
- Assigning a new network number for the network created by the PPP link

Using the Same IP Address as the Primary Network Interface

This addressing scheme is appropriate for point-to-point links only. In this scheme, you specify the addresses of the primary network interface for each endpoint. (See Chapter 2 for more information about the primary network interface.) These endpoints might be:

- Two standalone machines communicating over the PPP link (if they have existing IP addresses)
- Two network endpoints communicating over the PPP link
- Remote host connecting to a network dial-in server through a point-to-point link
- Dial-in server connecting to remote hosts through a dynamically allocated point-to-point link

When you edit the /etc/inet/hosts file on a local endpoint, supply the IP address of its primary network interface and host name and the IP address of the peer host on the other end of the link.

Creating a Unique IP Address and Host Name

In this method, you assign a unique host name and IP address to the PPP network interface. (You might want to call the interface hostname-ppp.) Use this addressing scheme for:

- Endpoint machines on a network used as a multipoint dial-in server.
- Machines on a virtual network.
- Remote host that uses a dedicated IP address for communicating with a dial-in server over a dynamically allocated PPP link. (Note that this is not a requirement for the dynamic link configuration.)
- Machine that is also configured as a router for a physical network, such as Ethernet or Token Ring.
- Machine in a standalone-to-standalone configuration that does not have an existing IP address. (The PPP interface becomes the primary network interface.)

You must specify the unique address and host name for the PPP network interface in the `asppp.cf` configuration file.

To create the new host name and IP address, add it to the `/etc/inet/hosts` file on the endpoint machines, as described in "`hosts` Database" on page 128.

Assigning a Network Number to the PPP Link

You create a new network number for the PPP configuration when it involves:

- Virtual networks of computers communicating through PPP multipoint links (required)
- A multipoint dial-in server and its remote hosts (required)
- The PPP link between two networks, particularly when one or both of the network endpoint machines are also routers for a physical network (optional)

(See Chapter 5 for information on network numbers.)

The PPP link becomes a **virtual network**, since it does not involve any physical network media. You need to type its network number in the `networks` database on all endpoint machines, along with the network numbers of the networks being linked.

The following sample shows an `/etc/inet/networks` file for an internetwork with PPP.

EXAMPLE 22–1 `/etc/inet/networks` File for an Internetwork With PPP

```
kalahari      192.9.253
negev         192.9.201
nubian-ppp    192.29.15
```

In the sample file, `kalahari` and `negev` are two local area networks, and `nubian-ppp` is the name of the PPP link.

Routing Considerations

The RIP routing protocol runs on Solaris TCP/IP networks by default. In most cases, you should leave RIP running on point-to-point links. However, if you are having performance problems with the link, you might want to disable RIP on the point-to-point link.

Note - RIP is not started on multipoint links. Therefore, you must set up static routing for the multipoint link. Refer to “Routing Protocols” on page 143 for instructions.

For instructions on how to disable RIP, see “How to Turn Off RIP” on page 433 .

PPP Hardware Requirements

The basic PPP configuration involves a computer, a modem, and RS-232 telephone lines. However, before you configure, you need to verify whether the hardware you selected can support PPP. This section describes the hardware requirements for PPP.

- **Modem requirements** – To run PPP, each endpoint machine must have a modem that supports at least 9600 bps or faster bidirectional connections. Such a modem implements the V.32 or V.32bis specification.
- **Serial port selection (for dial-in servers only)** – You can configure either serial port A or serial port B on most CPUs for PPP usage. Use the Solaris Serial Port Manager to initialize the ports on the dial-in server. *System Administration Guide, Volume 1* contains instructions for selecting the appropriate port. If you have additional serial cards installed, you can also use their serial ports for PPP connections.
- **Disk space** – You must have 300 Kbytes of free space in `/usr` to install PPP.

Note - You need an additional 300 Kbytes of free space in `/usr` to install 64-bit PPP.

Checklist for Configuring PPP

Use this checklist to prepare for configuring PPP. It lists the information you need to gather and the tasks you need to do before starting the configuration process.

TABLE 22–1 Checklist for Configuring PPP

Do you have 300 Kbytes of free space available in `/usr`?	Yes / No
If you are installing 64–bit PPP, do you have an additional 300 Kbytes of free space in `/usr`?	Yes / No
Do you have 4 Kbytes of free space available in `/` (root)?	Yes / No
Do the modems for each endpoint support V.32 or V.32bis or higher?	Yes / No
Have you used the Serial Port Manager on the dial-in server to designate the serial port for the modem?	Yes / No
Have you ensured that Solaris PPP is installed on each endpoint machine? (If PPP hasn't been installed, you can use the `pkgadd` program or `admintool` software manager to install it. Refer to *Solaris Advanced Installation Guide* for instructions.)	Yes / No
Have you ensured that no other versions of PPP are running on each endpoint? (If they are, disable them, as explained in their documentation.)	Yes / No
Have you determined which IP addresses to use for all computers involved in the PPP link?	Yes / No
List the host names and IP addresses of these machines here.	_____ _____ _____ _____
Write the name and IP address of the dial-in server (if applicable).	_____
Write the name of the network interface that you need to use.	_____

CHAPTER 23

Managing PPP

This chapter contains procedures and information for configuring PPP, procedures to set up less commonly used PPP links, as well as some troubleshooting procedures. The following topics are covered:

PPP Task Maps

This section shows the various task maps for configuring PPP, maintaining PPP once it is installed, and troubleshooting problems with PPP

TABLE 23–1 PPP Configuration Task Map

Task...	Description	For Instructions, Go To ...
Verify that PPP is installed on all machines	Use `pkginfo` to check that PPP is installed on all machines involved in the PPP link	"How to Verify Installation" on page 426
Configure the remote machine's `hosts` database	Configure the remote machine's `host` database by adding the IP address and host name to the `/etc/inet/hosts` file	"How to Configure the Remote Machine's `hosts` Database" on page 428
Configure the dial-in server's hosts database	Configure the dial-in server's `hosts` database by adding entries to `/etc/hosts` and `/etc/inet/networks` files.	"How to Configure the Dial-In Server's `hosts` Database" on page 429
Edit the `/etc/asppp.cf` file to add entries so that they are recognized at boot time.	Add information necessary to establish and maintain communications with a remote endpoint to the `/etc/asppp.cf` file.	"How to Edit the `asppp.cf` Configuration File" on page 432
Turn off RIP.	Add `norip` to the `/etc/gateways` file to turn off RIP.	"How to Turn Off RIP" on page 433
Update remote host	Add the IP address and host name to the `/etc/inet/hosts` file to update a remote host.	"How to Update a Remote Host" on page 434
Update the dial-in server	Update the `/etc/inet/hosts` file to add entries for each remote host served.	"How to Update the Dial-In Server" on page 435
Add PAP/CHAP support	Add `require_authentication` and `will_do_authentication` keywords to the `/etc/asppp` file to establish security as to whether parts of the link respond to PAP or CHAP.	"How to Install PAP/CHAP" on page 436

TABLE 23–2 PPP Maintenance Task Map

Task...	Description	For Instructions, Go To ...
Manually start PPP	Use the `/etc/init.d/asppp start` command to start PPP. Normally, PPP is started automatically and you do not need to use this command.	"How to Manually Start PPP" on page 439
Verify that PPP is running	Use the `ps` and `ping` commands to see if PPP is running.	"How to Verify That PPP Is Running" on page 439
Stop PPP	Use the `/etc/init.d/asppp stop` command to stop PPP.	"How to Stop PPP" on page 440
Check interface status	Use the `ifconfig` command to monitor the current state of the line.	"How to Check Interface Status" on page 440
Check connectivity	Use the `ping` command to verify that the connection is up.	"How to Check Connectivity" on page 441
Check interface activity	Use the `netstat` command to verify that packets are being sent and received.	"How to Check Interface Activity" on page 442
Check local routing tables	Use the `netstat` command to display local routing tables.	"How to Check the Local Routing Tables" on page 442
Add routes using `in.routed`	Use the `in.routed` command to add routes when running dynamic routing.	"How to Add Routes Using `in.routed`" on page 443

TABLE 23–3 PPP Troubleshooting Task Map

Task...	Description	For Instructions, Go To ...
Set up diagnostics	How to set up PPP diagnostics for troubleshooting.	"How to Set Diagnostics for Your Machine" on page 445

Overview of the Configuration Process

You have completed the preinstallation activities noted in Chapter 22. Now you can begin PPP configuration.

PPP requires that you:

1. Install the PPP software, if it isn't already installed.
2. Edit the `/etc/inet/hosts` files on all machines involved.
3. Edit the UUCP database files for all dial-out machines.
4. Edit the `/etc/passwd` and `/etc/shadow` files for the dial-in machine.
5. Edit the `/etc/asppp.cf` file on each machine on the link.
6. Start the link manager `aspppd` on each machine on a link.
7. Verify that PPP is running successfully.

Although you don't have to perform Tasks 1–4 in order, you must complete them before you can edit the PPP-configuration file.

The sections in this chapter explain the procedures for configuring PPP.

Installing the PPP Software

The PPP software is automatically included when you run the Solaris installation program and select the entire distribution. If you did not select the entire distribution, you need to install PPP as a separate package.

▼ How to Verify Installation

Before proceeding further, you must check that the Solaris version of PPP is installed on all machines to be involved in the PPP link.

1. **Become superuser.**

2. **On each endpoint involved in the link, type**:

```
# pkginfo | grep ppp
```

If 32–bit PPP is installed, the following package names are displayed:

```
SUNWapppr      PPP/IP Asynchronous PPP daemon configuration files
SUNWapppu      PPP/IP Asynchronous PPP daemon and PPP login service
```

(Continuation)

```
SUNWpppk       PPP/IP and IPdialup Device Drivers
```

If 64–bit PPP is installed, the following package names are displayed:

```
SUNWapppr      PPP/IP Asynchronous PPP daemon configuration files
SUNWapppu      PPP/IP Asynchronous PPP daemon and PPP login service
SUNWpppk       PPP/IP and IPdialup Device Drivers
SUNWpppkx      PPP/IP and IPdialup Device Drivers (64-bit)
```

3. **If PPP is not installed on an endpoint system, install it using either the** `pkgadd` **program or** `admintool` **software manager.**

Note - When using `pkgadd` to install PPP, you must install the packages in the order listed.

Refer to *System Administration Guide, Volume 1* for more information about `pkgadd` and `admintool` software manager.

Sample PPP Configuration

This and the following sections show you how to edit the appropriate files to support the most common PPP configuration: remote hosts and their dial-in server. Figure 23–1 illustrates the configuration used as the example for this chapter. It depicts three remote machines (`nomada`, `nomadb`, `nomadc`) and their dial-in server `nubian`, which compose the network 192.41.43. This is a separate network from the local area network 192.41.40, to which the dial-in server `nubian` is directly attached. Network 192.41.40 runs NIS as its name service.

The IP number shown for each remote host is the address of its PPP network interface. However, the dial-in server has a specially created IP address for the PPP

interface, 192.41.43.10, in addition to the IP address for its primary network interface, 192.41.40.45.

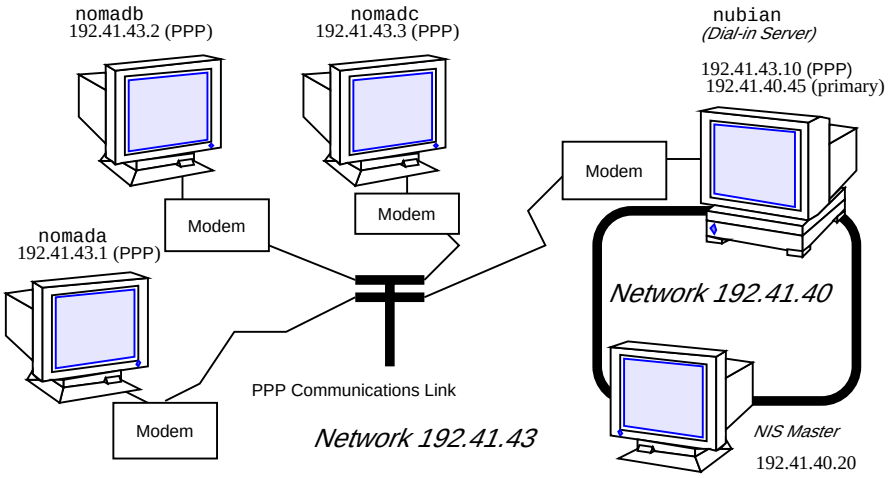


Figure 23–1 Sample Network of Remote Hosts and Multipoint Dial-In Server

Editing the `/etc/inet/hosts` File

After ensuring that PPP is installed on every machine involved in your configuration, your next task is to edit the `/etc/inet/hosts` files on each machine. You must add host information to the `hosts` database for every machine on the other end of the PPP link that the local machine needs to communicate with.

Note - You must update `/etc/inet/hosts` regardless of the name service in use on the physical network. This is necessary because PPP starts before the name service daemons during the booting process.

▼ How to Configure the Remote Machine's `hosts` Database

1. **Become superuser.**

2. **Edit the `/etc/inet/hosts` file to do the following:**
   a. **Add an entry with the IP address and host name of the PPP network interface for the dial-in server on the other end of the link.**

In Figure 23–1, nomada must have in its /etc/inet/hosts file an entry with the IP address for dial-in server nubian's PPP network interface. This is true also for the /etc/inet/hosts files for nomadb and nomadc.

b. Add entries with the IP addresses of any machines on the dial-in server's physical network that the remote host can remotely log in to.

The /etc/inet/hosts file on nomadc would look like:

```
# Internet host table
#
127.0.0.1         localhost      loghost
192.41.43.3       nomadc
192.41.43.10      nubian-ppp
192.41.40.20      nismaster
```

3. Update the databases on the name server (if the network has one) with the host names and IP addresses of the remote hosts.

Multipoint Dial-in Server hosts Database

Multipoint dial-in servers must have a unique IP address for the PPP interface, besides the local IP address for the primary network interface. When configuring the hosts database for the dial-in server, you need to perform the following procedure.

▼ How to Configure the Dial-In Server's hosts Database

1. Become superuser.

2. Add an entry with the IP address for the PPP interface to the /etc/inet/hosts file for the dial-in server.

For example, the /etc/hosts file on dial-in server nubian in Figure 23–1 would have the following entries.

```
# Internet host table
#
127.0.0.1           localhost       loghost
192.41.43.10        nubian-ppp
```

(Continuation)

```
192.41.40.45        nubian
```

3. **For configurations where the server's physical network does not use a name service:**
   a. **Add entries to the server's `/etc/inet/hosts` files for each remote host served.**
   b. **Add entries for the remote hosts to the `/etc/inet/hosts` files of every machine on the physical network permitted to communicate with the remote machines.**

4. **Add a new network number to the dial-in server's `/etc/inet/networks` file for the network that consists of the server and its remote hosts.**
   Refer to "Assigning a Network Number to the PPP Link" on page 420 for more information.

Editing UUCP Databases

Before a machine can dial out over the PPP link, you must edit these files in its UUCP database:

- `/etc/uucp/Devices`
- `/etc/uucp/Dialers`
- `/etc/uucp/Systems`

You must edit these files for remote hosts serving as PPP dial-out machines. Additionally, you must edit these files on the dial-in server if it is to dial out to the remote hosts (a requirement for multipoint dial-in servers). Chapter 25 describes these files in detail.

Modifying the /etc/passwd File

To configure a dial-in server, you must also edit the /etc/passwd and /etc/shadow files.

You must add entries to the /etc/passwd file on the dial-in server for each user on a remote host authorized to log in to the server. When a remote host calls the dial-in server, it reads its UUCP databases and passes the server a user name or user ID for the host initiating the call. The server then verifies this user information in its /etc/passwd file.

If the user's password is authenticated, the server then logs the user in to a special shell for PPP hosts, /usr/sbin/aspppls. The server gets this information from the login shell entry in its /etc/passwd file. Using the example in Figure 23–1, dial-in server nubian might have the following entries in its /etc/passwd file:

```
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
nomada:x:121:99:R. Burton:/:/usr/sbin/aspppls
nomadb:x:122:99:T. Sherpa:/:/usr/sbin/aspppls
nomadc:x:123:99:S. Scarlett:/:/usr/sbin/aspppls
```

Refer to *System Administration Guide, Volume 1* for information about the /etc/passwd file.

Note - In addition to the information in the /etc/passwd file, you update the /etc/shadow file with the passwords for the login names used by each endpoint machine permitted to dial in to the server. For more information, refer to *System Administration Guide, Volume 1*.

Editing the /etc/asppp.cf Configuration File

The /etc/asppp.cf configuration file provides the PPP link manager on one endpoint machine with information about the machine on the other end of the link—or the machines on the other end of a multipoint (or dynamic point-to-point) link. When the machine boots, the link manager uses this information to establish and maintain communication with a remote endpoint.

Editing the Configuration File

When editing asppp.cf:

- Separate keywords in the configuration file by white space (blanks, tabs, and new lines).
- Use a # sign before all character strings meant as comments. All characters placed between a # sign and the next new line are considered comments and ignored.

No other format requirements apply for the placement of the keywords in the file.

▼ How to Edit the asppp.cf Configuration File

1. **Become superuser on one endpoint machine.**
2. **Change to the /etc directory.**
3. **Edit the generic asppp.cf file to add the information defining the PPP link for this machine.**
4. **Save the file, making sure the permissions are set to 600.**
5. **Change to the /etc directories on the remaining endpoints and repeat Steps 2 and 3.**

Turning Off RIP

You can disable RIP on a point-to-point link through the file `/etc/gateways`. This file does not come with your operating system: you must create it with a text editor.

▼ How to Turn Off RIP

1. **Become superuser.**

2. **Edit `/etc/gateways` and add the following entry:**

```
norip ipdptpn
```

   where `ipdptp`*n* represents the device name of the point-to-point PPP interface used.

   For more information, refer to the `in.routed`(1M) man page.

Adding PPP Security

After you have completed installing PPP on every machine involved in your configuration, you can add either PAP or CHAP levels of security for the PPP link by modifying the `asppp.cf` file. Refer to “Editing `asppp.cf` for PAP/CHAP Security” on page 436.

Configuring Dynamically Allocated PPP Links

A dial-in server with a dynamic point-to-point link gives your site all the advantages of point-to-point communications. Chapter 21 introduces this configuration type. It consists of remote hosts communicating with at least one dial-in server that dynamically allocates point-to-point links on an as-needed basis. The following sample configuration is used throughout this section.

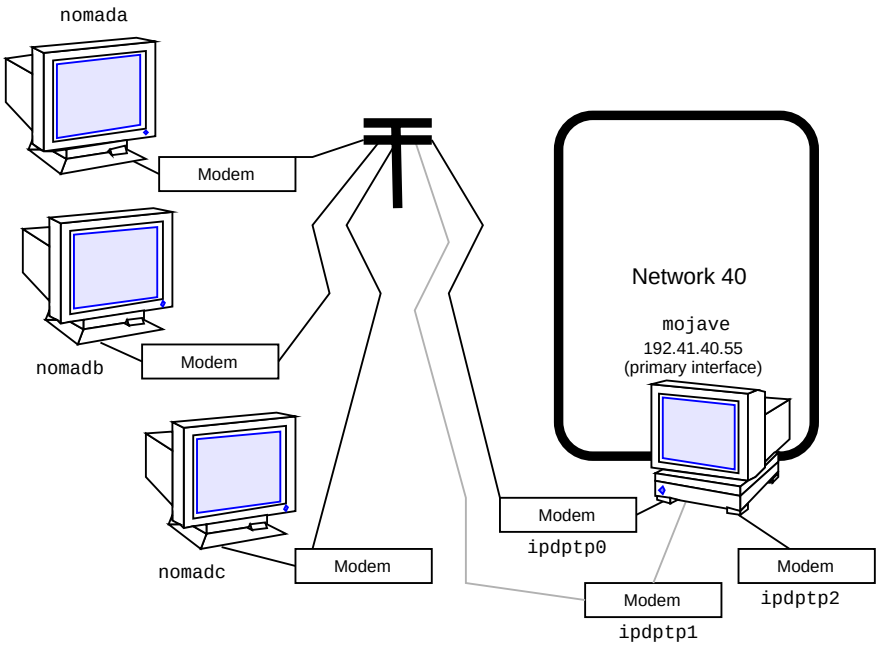


Figure 23–2 Network of Remote Hosts and Dynamic Link Dial-In Servers

Each remote host communicates with the dial-in server using a standard point-to-point link. However, unlike the multipoint dial-in server in Figure 23–1, dial-in server `mojave` connects to a calling host over a dynamic point-to-point link. The server allocates an available link whenever a remote host attempts to establish a connection.

The idea behind a dynamic link is that the server provides the client with an IP address each time a connection is established. When the connection is established, the server allocates an available IP interface to the client. The remote IP address of the interface then becomes the client's IP address for the duration of the connection. When the connection is terminated, the IP interface is returned to the pool of available interfaces, ready to be used for another connection.

You use the same generic procedures for configuring dynamic links as you do for the remote host–to–multipoint dial-in server link, as described in "Overview of the Configuration Process" on page 426. However, the dynamic point-to-point link has its own set of issues and requires slightly different modifications to the files involved in configuration.

▼ How to Update a Remote Host

When configuring the `hosts` databases on the remote machines, do the following:

1. **Become superuser.**

2. **Add to the** `/etc/inet/hosts` **file the IP address and host name of the primary network interface for each dial-in server on the other end of the link.**

   For example, in Figure 23–2, the `/etc/inet/hosts` file for `nomada`, `nomadb`, and `nomadc` should each include the IP address of the primary network interface of the dial-in server `mojave`.

3. **Add the dummy IP address.**

   This IP address is used only when PPP is started.

   The `/etc/inet/hosts` file on `nomadc` might look like:

```
# Internet host table
#
127.0.0.1          localhost        loghost
192.41.40.55       mojave
1.2.3.4            dummy
```

4. **Add to the** `/etc/inet/hosts` **file the IP addresses of all machines on the dial-in server's physical network that the remote host can remotely log in to.**

5. **Update the databases on any name server on the physical network with the host names and IP addresses of the remote hosts.**

▼ How to Update the Dial-In Server

You do not have to add any PPP-specific address to the `hosts` database for the dial-in server. The dynamically allocated link must use the server's primary network interface. Therefore, when configuring the `hosts` database for the dial-in server, do the following:

1. **Become superuser.**

2. **Add entries to the server's** `/etc/inet/hosts` **files for each remote host served.**

3. **Add to the** `/etc/inet/hosts` **files of every machine on the physical network the entries for any remote hosts they are permitted to communicate with.**

Editing `asppp.cf` for PAP/CHAP Security

You can edit the `asppp.cf` file to establish security and to specify whether parts of the link will respond to Password Authentication Protocol (PAP), or Challenge-Handshake Authentication Protocol (CHAP), as described in "PPP Security" on page 411. The `asppp.cf` file is edited by adding a series of keywords. In this section, *authenticator* is the system starting the link or challenge, and is frequently the server. *Peer* is the other end of the link, and is often the client.

The keywords to be added are `require_authentication` and `will_do_authentication`. The authenticator or server generally require authentication and the peer or client generally does authentication.

TABLE 23–4 Authenticator Keywords and Associated Strings

`require_authentication pap`	`require_authentication chap`

`pap_peer_id`
 	
`pap_peer_password`
	`chap_peer_secret`
	
	`chap_peer_name`

TABLE 23–5 Peer Keywords and Associated Strings

`will_do__authentication pap`	`will_do_authentication chap`

`pap_id`
 	
`pap_password`
	`chap_secret`
	
	`chap_name`

▼ How to Install PAP/CHAP

1. **Become superuser on the server.**
2. **Edit the `/etc/asppp.cf` file.**

3. **Add the** `require_authentication` **keyword for each machine on the link to use either CHAP or PAP security.**

   a. **For each pap keyword add an associated** `pap_peer_id` **and** `pap_peer_password` **string.**

   b. **For each chap keyword add an associated** `chap_peer_secret` **and** `chap_peer_name` **string.**

      You can state the keywords explicitly, or if you prefer, you can use the default for the path. Refer to Table 24–1 to see what each keyword specifies. Examples can be found in Example 23–1.

4. **On each remote host on the link to use either PAP or CHAP security, add an entry in the remote host's** `/etc/asppp.cf` **file with the** `will_do_authentication` **keyword.**

   a. **For each pap keyword entry add an associated** `pap_id` **and** `pap_password` **string.**

   b. **For each chap keyword entry add an associated** `chap_secret` **and** `chap_name` **string.**

PAP/CHAP Examples

The example below shows the `asppp.cf` file for the server `mojave` with PAP and CHAP authentication required. The peers are `nomada` (PAP) and `nomadb` (CHAP).

EXAMPLE 23–1 Code Example for Server `mojave`

```
ifconfig ipdptp0 plumb mojave nomada up
ifconfig ipdptp1 plumb mojave nomanb up
path
      peer_system_name tamerlane
      require_authentication pap  #tells nomada that mojave
                                  #requires pap authentication
      pap_peer_id desert
      pap_peer_password oasis
path
      peer_system_name lawrence
      require_authentication chap  #tells nomadb that mojave
                                   #requires chap authentication
      chap_peer_name another\sdesert
      chap_peer_secret secret\soasis\swith\007bell
```

The next sample shows mojave's remote host nomada offering to do both PAP and CHAP authentication.

EXAMPLE 23–2 Code Example for Remote Host nomada

```
ifconfig ipdptp0 plumb tamerlane mojave up
path
      interface ipdptp0
      peer_system_name mojave
      will_do_authentication chap pap #nomada tells mojave
                                      #that it will do chap and
                                      #pap authentication
      pap_id desert
      pap_password oasis
      chap_name desert\srain
      chap_secret %$#@7&*(+|‘P’12
```

The next example shows mojave's remote host nomadb offering to do CHAP authentication.

EXAMPLE 23–3 Code Example for Remote Host nomadb

```
ifconfig ipdptp0 plumb nomadb mojave private up
path
     interface ipdptp0
     peer_system_name mojave
     will_do_authentication chap   #nomadb tells mojave that it
                                   #will do chap authentication
     chap_name another\sdesert
     chap_secret secret\soasis\swith\007bell
```

Ideally, both CHAP and PAP are included in the configuration file, with the server requiring authentication and the remote host willing to do authentication. However this is reversible so that either side can require authentication. CHAP secrets need to be delivered by secure means. This generally involves manually releasing them.

Starting Up and Stopping Your New PPP Link

You can start PPP either automatically, at boot time, or manually from the command line.

▼ How to Manually Start PPP

You can start PPP manually, although this is not normally required.

1. **Become superuser.**

2. **Type:**

```
# /etc/init.d/asppp start
```

▼ How to Verify That PPP Is Running

1. **Become superuser.**

2. **Run the** `ps` **command:**

```
 # ps -e | grep asppp
```

   The resulting output from `grep` should list the `aspppd` daemon, indicating that PPP is running.

3. **If you do get results, verify that you can reach the remote PPP link by typing:**

```
# ping remote-host 300
```

   This version of `ping` sets a timeout value of 5 minutes (300 seconds). You should receive output similar to *remote-host* `is alive`. If you receive a different notice, such as *remote-host* `unreachable`, route configuration has failed.

4. **Check for errors in the configuration process by examining the log file.**

```
# tail /var/adm/log/asppp.log
```

   The `asppp.log` contains error messages if any errors were encountered during configuration.

See “Common Check” on page 440 for information on troubleshooting and problem solving.

▼ How to Stop PPP

To stop PPP operations on your network:

1. **Become superuser.**

2. **Type:**

```
# /etc/init.d/asppp stop
```

Common Check

The following section contains some common checks that you might need to perform to verify the operation of your PPP setup.

Note - You must become superuser to perform these checks.

Checking Hardware

Make sure that all modem and power cables are tightly seated. If you are having problems with PPP, always check the modems, cables, serial card, and phone lines first.

▼ How to Check Interface Status

After PPP is started, you can use `ifconfig` to monitor the current state of the line, using only the PPP interface name as an argument. Example 23–4 shows sample output from `ifconfig` for PPP links that are running.

Note - If a user is privileged (root), and issues an `ifconfig` command, machine addresses are displayed in the output as shown in the following example.

1. **Become superuser.**

2. **Type:**

```
ifconfig ipdptp0
```

EXAMPLE 23–4 `ifconfig` Output for Point-to-Point Link

```
nomadb# ifconfig ipdptp0

ipdptp0: flags=28d1<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST,UNNUMBERED> mtu 1500
        inet 129.144.111.26 --> 129.144.116.157 netmask ffff0000
        ether 0:0:0:0:0:0
```

You receive output similar to that in Example 23–5 for both standard and dynamic point-to-point links.

EXAMPLE 23–5 `ifconfig` Output for Multipoint Link

```
nubian# ifconfig ipd0

ipd0: flags=c1<UP,RUNNING,NOARP> mtu 1500
        inet 129.144.201.191 netmask ffffff00
        ether 0:0:0:0:0:0
```

If `ifconfig` does not display `UP` and `RUNNING`, you did not configure PPP correctly. For more information on `ifconfig`, see “`ifconfig` Command” on page 114 and the `ifconfig`(1M) man page.

▼ How to Check Connectivity

Use the `ping` command to verify that the connection is up or can be established. For example, consider the following simple round-trip test:

1. **Become superuser.**

2. **Type**:

```
# ping elvis
```

where `elvis` is the name of the PPP interface on the remote host. If the resulting display is

```
elvis is alive
```

then packets can be sent to and received from `elvis`. If not, a routing problem exists at some point between the local and remote hosts. For more information on `ping`, refer to “`ping` Command” on page 112 and the `ping`(1M) man page.

▼ How to Check Interface Activity

Use the `netstat` command as follows to check that packets are being sent and received correctly:

1. **Become superuser.**

2. **Type:**

```
# netstat -i
```

Refer to “`netstat` Command” on page 115 and the `netstat`(1M) man page.

▼ How to Check the Local Routing Tables

Use the `netstat` command to display the local routing tables:

1. **Become superuser.**

2. **Type:**

```
# netstat -r
```

The following is sample output:

```
Routing tables
Destination     Gateway    Flags  Ref    Use    Interface
------------- ------------ ----- ------ ------- ----------
sahara        deserted     UGH    0           0  ie1
karakum       labia        UGH    0           0  ie1
frodo         bilbo        UGH    1       12897  ipdptp0
route7        route7       UGH    0           0  ie0
eastgate      route71      UGH    0         158  ie0
backbone      pitstopbb    U      1       16087  ie1
dresdenpc     route1       UG     0           0  ie1
loopback      localhost    U      2      113436  lo0
swan-bb       pitstop      U      406    146044  ie0
dallas2       route7       UG     0           0  ie0
trainingpc    route62      UG     0           0  ie1
```

Make sure a routing table entry exists for each possible destination network. In particular, PPP devices, listed under `Interface`, should be matched with the

appropriate host names listed under `Gateway`. The `Gateway` entry should, in turn, be matched with the correct entry under `Destination`.

Otherwise, if you are using *static routing*, add the appropriate static routes.

How to Add Routes Using `in.routed`

If you are using *dynamic routing* with `in.routed`:

1. **Become superuser.**

2. **Verify that `in.routed` is running by typing:**

```
# ps -e | grep route
```

   If the routing tables still don't look correct, become superuser and continue with the next steps.

3. **Kill `in.routed` by typing the process ID you got from `ps -e` as the argument to `kill`. For example, if *1384* was the process ID, you would type:**

```
# kill 1384
```

4. **Flush the routing tables as follows:**

```
#  /usr/sbin/route -f
```

5. **Restart `in.routed`:**

```
# /usr/sbin/in.routed
```

Checking Permissions

If you attempt to use `rsh` and receive the message `Permission denied`, the remote system's `/etc/hosts.equiv` or `/.rhosts` file does not contain the sending system's host name or does not contain the line `+`.

Checking Packet Flow

Check the packet flow next. Use the snoop command to observe packets from the network and their contents. The example below shows some sample output from snoop.

EXAMPLE 23–6 Sample Output From snoop

```
# snoop -d ipdptp0
Using device ipdptp0 (promiscuous mode)
corey -> pacifica7    RLOGIN C port=1019
       hugo -> ponc3          RPC R XID=22456455 Success
       ponc3 -> hugo          NFS C WRITE FH=1B29 at 32768

    commlab3 -> commlab4      TELNET R port=34148
    commlab4 -> commlab3      IP  D=129.144.88.3 S=129.144.88.4 LEN=46, ID=41925
    commlab3 -> commlab4      TELNET R port=34148
    commlab4 -> commlab3      ICMP Echo request
    commlab3 -> commlab4      ICMP Echo reply
    commlab4 -> commlab3      FTP C port=34149
    commlab4 -> commlab3      FTP C port=34149
    commlab3 -> commlab4      FTP R port=34149
    commlab4 -> commlab3      FTP C port=34149
```

The ipdptp0 device name mentioned in the first line of the output Using device ipdptp0 indicates a point-to-point connection.

Note - You need to have the link up and some traffic generated in order to use snoop to check the line status.

---

snoop captures packets from the network and displays their contents. It uses both the network packet filter and streams buffer modules to provide efficient capture of packets from the network. Captured packets can be displayed as they are received or saved to a file for later viewing.

snoop can display packets in a single-line summary form or in verbose multiline forms. In summary form, only the data pertaining to the highest-level protocol is displayed. For example, an NFS packet will have only NFS information displayed. The underlying RPC, UDP, IP, and Ethernet frame information is suppressed but can be displayed if either of the verbose options is chosen.

For more information about the snoop command, refer to the snoop(1M) man page.

Using PPP Diagnostics for Troubleshooting

If you have problems with a link after successfully establishing modem connections, you can use PPP-level diagnostics for troubleshooting. PPP-level diagnostics report detailed information about the activities of a link to help you determine where it is failing.

To obtain diagnostic information, add the line `debug_level 8` to the `path` section of the `asppp.cf` file. (If you are very knowledgeable about data communications, you might want to use debug level 9, which provides very detailed information.) Here is a sample configuration file that invokes PPP diagnostics.

```
ifconfig ipdptp0 plumb nomada nubian-ppp up
path
   interface ipdptp0
   peer_system_name nubian-ppp   #The name in the /etc/uucp/Systems file
   inactivity_timeout 300        #Allow five minutes before timing out
   debug_level 8                 #Start up PPP diagnostics for this link
```

For complete details about the `aspppd.conf` file, refer to “Editing the `/etc/asppp.cf` Configuration File” on page 432.

▼ How to Set Diagnostics for Your Machine

Set diagnostics on the host you want to monitor as follows:

1. **Become superuser.**
2. **Change to the `/etc` directory.**
3. **Edit the current `asppp.cf` file and add the following to the `path` section: `debug_level 8`.**
4. **Save the file, making sure the permissions are set to `600`.**
5. **Kill the current `aspppd` daemon and restart it.**

```
# kill PID
# aspppd
```

where *PID* is the process ID for `aspppd`.

PPP reports diagnostic information in `/var/adm/log/asppp.log`.

CHAPTER 24

PPP Reference

This chapter provides reference material for working with PPP. The following topics are covered:

UUCP Databases

Before a machine can dial out over the PPP link, you must edit these files in its UUCP database:

- `/etc/uucp/Devices`
- `/etc/uucp/Dialers`
- `/etc/uucp/Systems`

Updating /etc/uucp/Devices for PPP

The /etc/uucp/Devices file must contain entries for every communications device that a particular host uses or must know about. For example, if a machine uses a US Robotics V.32bis modem as part of the PPP link, you should ensure that /etc/uucp/Devices has an entry similar to the following:

```
# Use these if you have a USrobotics V.32bis modem on Port B.
ACUEC   cua/b - 9600 usrv32bis-ec
ACUEC   cua/b - 19200 usrv32bis-ec
ACUEC   cua/b - 38400 usrv32bis-ec
```

Be sure that the Devices file on each PPP endpoint machine has an entry describing its modem. For more information about /etc/uucp/Devices, refer to “UUCP /etc/uucp/Devices File” on page 498.

Updating /etc/uucp/Dialers for PPP

The /etc/uucp/Dialers file must have an entry describing the conversation with the modem attached to your PPP endpoint machine. Here is a sample entry for a US Robotics V.32bis modem that is part of a PPP link:

```
usrv32bis-ec =,-,  "" \dA\pT&FE1V1X1Q0S2=255S12=255&A1&H1&M5&B2\r\c OK\r
\EATDT\T\r\c CONNECT\s14400/ARQ STTY=crtscts
```

The first parameter in the entry, usrv32bis, corresponds to the last parameter in the /etc/uucp/Devices file. The remainder of the entry describes the characters that the modem sends, those that it expects to receive, and so on. Table 27–5 defines the control codes used in the Dialers file.

Be sure that an entry is in the Dialers file for the modem attached to each dial-out endpoint on your link. If you are unsure of the correct conversation for a particular modem, refer to the *System Administration Guide, Volume 1* and the operating manual for the modem.

Updating /etc/uucp/Systems for PPP

The /etc/uucp/Systems file contains entries for every machine to which the local host can dial out. Information in an entry might include the remote host’s phone number, the line speed, and so on. Here is an example that host nomadb in Figure 23–1 might have for its dial-in server:

```
nubian-ppp  Any ACUEC 38400 5551212 "" P_ZERO ""
\r\n\c login:-\r\n\c-login:-\r\n\c-login:-
EOT-login: bnomad password: Secret-Password
```

The first field gives the server's host name, `nubian-ppp`, a value used by the `asppp.cf` file keyword `peer_system_name`. `ACUEC` and `38400` refer to the device and speed, and are used to select an entry from the `/etc/uucp/Devices` file. The remaining information includes the phone number of the machine that `nomadb` is to dial in to, the login name that `nomadb` is using to log in, and so on. "UUCP `/etc/uucp/Systems` File" on page 491 fully defines the parameters you need to supply to the `Systems` file.

On each remote host in your configuration, you must add an entry for its dial-in server. You can have additional entries in the `/etc/uucp/Systems` file for other machines to which the host can dial out for UUCP communications and for other PPP dial-in servers.

If the dial-in server also directly dials out to remote hosts, you must add entries to its `Systems` file describing each of these remote hosts.

`/etc/asppp.cf` Configuration File

The `/etc/asppp.cf` configuration file provides the PPP link manager on one endpoint machine with information about the machine on the other end of the link—or the machines on the other end of a multipoint (or dynamic point-to-point) link. When the machine boots, the link manager uses this information to establish and maintain communication with a remote endpoint.

Parts of Basic Configuration File

The basic `asppp.cf` configuration file must contain at least two main sections: an `ifconfig` line and at least one `path` section. It can also contain a `defaults` section, which you use when you want to set the default values for an endpoint. (Refer to "Configuration Keywords" on page 471 for a description of keywords used in the `defaults` section.)

The following example shows a basic configuration file such as you would create for a remote host to establish a point-to-point link with a dial-in server.

EXAMPLE 24–1 Basic Configuration File

```
ifconfig ipdptp0 plumb nomada nubian-ppp up
   path
      interface ipdptp0
      peer_system_name nubian-ppp      # The name in the /etc/uucp/Systems file
      inactivity_timeout 300           # Allow five minutes before timing out
```

`ifconfig` Section of the `asppp.cf` File

The `asppp.cf` file must contain an `ifconfig` section with this syntax:

`ifconfig` *interface-number* `plumb` *local-machine remote-machine* `up`

Here is a description of the fields:

- `ifconfig` – Tells the link manager to run the `ifconfig` command and begin configuring the PPP interface.
- *interface-number* – Identifies the PPP interface `ipdptp`*n* for a point-to-point link or `ipd`*n* for a multipoint link. (Replace the *n* with the number of the interface.)
- `plumb` – Option of `ifconfig` that enables IP to recognize the interface.
- *local-machine* – Gives the name of the local endpoint, which can be the local host name or IP address.
- *remote-machine* – Gives the name of the remote endpoint, which can be the remote host name or IP address.
- `up` – Option of `ifconfig` that marks the interface just described as "up."

The link manager first runs the `ifconfig` command on the local machine to configure the `ipdptp0` point-to-point interface. The zero in `ipdptp0` gives the device number of the interface. The `plumb` option performs various activities necessary for IP to recognize the `ipdptp0` interface. `nomada` is the name of the local host. `nubian-ppp` is the name of the dial-in server to which `nomada` connects through the point-to-point link. The `ifconfig` option `up` marks the `ipdptp0` interface as up.

Note - For more information about `ifconfig`, see "How to Check Interface Status" on page 440 and the `ifconfig`(1M) man page.

`path` Section of the `asppp.cf` File

The `path` section of the configuration file tells the link manager the name of the remote endpoint and the name of the interface linking the endpoint machines. At a minimum the `path` section should contain the following lines:

```
path
   interface interface-number
   peer_system_name endpoint-name
```

`interface` *Keyword*

This keyword defines the PPP interface (either `ipdptp`*n* or `ipd`*n*). In Example 24–1, the following information appears in the `path` section:

```
interface ipdptp0
peer_system_name nubian-ppp
```

The `interface` keyword identifies `ipdptp0` as the point-to-point interface that local endpoint `nomada` uses to communicate with the remote endpoint in the manner described in this `path` section. It associates the `peer_system_name` with the interface.

`peer_system_name` *Keyword*

On a dial-out machine such as a remote host, the `peer_system_name` keyword takes the host name of the remote endpoint as its argument. This is the name of the remote endpoint given in `/etc/uucp/Systems`. The name need not be the same as the host name on the corresponding `ifconfig` line.

Note - The argument to the `peer_system_name` keyword for a dial-in server has a different value. See Example 24–1 for details.

In Example 24–1, `peer_system_name` identifies dial-in server `nubian-ppp` as the remote endpoint at the other end of this link. When the link manager reads the `asppp.cf` file, it then looks for the entry for `nubian-ppp` in the `/etc/uucp/Systems` file. (Recall that the `Systems` file contains information about how to set up communications with the remote endpoint, including that machine's telephone number. Refer to "Updating `/etc/uucp/Systems` for PPP" on page 448.)

`inactivity_timeout` *Keyword*

The `inactivity_timeout` keyword is optional. It tells the link manager that the link can remain inactive for the interval designated. When that interval is passed, the link manager knows to automatically disconnect the link. The default interval is two minutes; you do not have to use `inactivity_timeout` unless you require a different inactivity interval.

Additional Keywords

You can supply other keywords in the `asppp.cf` file to define how endpoint machines should communicate. “Configuration Keywords” on page 471 has complete information about these keywords.

Configuration File for Multipoint Dial-in Server

The `asppp.cf` configuration file for a multipoint dial-in server contains the same basic sections as that for a point-to-point link: an `ifconfig` section, at least one `path` section, and, if desired, a `defaults` section.

The following example shows a configuration file for the dial-in server `nubian` introduced in Example 24–1.

EXAMPLE 24–2 Configuration File for a Multipoint Dial-in Server

```
ifconfig ipd0 plumb nubian-ppp up

path
   interface ipd0
   peer_system_name tamerlane  # The user name this remote
                               # machine logs in with when it
                               # dials this server
    peer_ip_address nomada
                               # nomada is a remote machine that
                               # dials in to this server

# nomadb is another remote machine that dials in to nubian

path
   interface ipd0
   peer_system_name lawrence
   peer_ip_address nomadb

# nomadc is another remote machine that dials in to nubian

path
   interface ipd0
   peer_system_name azziz
   peer_ip_address nomadc
```

`ifconfig` Section for Multipoint Dial-in Server

The `ifconfig` section for a multipoint dial-in server has a slightly different syntax than that for a point-to-point link. This syntax is:

`ifconfig ipd`*n* `plumb` *server-name* `up`

The major difference is that no destination endpoints are listed as arguments to `ifconfig`. Instead, the link manager obtains this information from the `path` section of the `asppp.cf` file.

In Example 24–2, the link manager first runs the `ifconfig` command on the dial-in server to configure multipoint interface `ipd0`. The zero in `ipd0` gives the device number of the interface. The option `plumb` performs various activities necessary for IP to recognize the `ipd0` interface. The `ifconfig` option `up` marks interface `ipd0` as up.

Note - You might have to supply a netmask + parameter on the `ifconfig` line if you use subnetting.

`path` Section for Multipoint Dial-in Server

The `path` section of the `asppp.cf` file tells the link manager the name of the remote endpoint and the name of the interface linking the endpoint machines. However, on a multipoint dial-in server, you can include more than one `path` section. Additionally, some of the arguments to the keywords are used differently with multipoint links.

```
path
    interface interface-number
    peer_system_name endpoint-username
    peer_ip_address endpoint-hostname
```

You need to define a `path` section for each nomadic endpoint with which the dial-in server can establish connections.

`interface` *Keyword*

For a multipoint dial-in server, the `interface` keyword defines the PPP interface `ipd`*n*. You must specify the same PPP interface in the `path` section for every endpoint that communicates with the server through this interface.

`peer_system_name` *Keyword*

The `peer_system_name` keyword takes a slightly different argument for a dial-in machine than a dial-out machine. For a dial-in server, this argument is the login name used by the remote host when it tries to establish communications with the server. This user name must already be present in the server's `/etc/passwd` file. When the login service reads this name, it verifies the user name in the `/etc/passwd` and `/etc/shadow` files enabling communications.

In this excerpt from Example 24–2:

```
path
    interface ipd0
    peer_system_name scarlett
    peer_ip_address nomadc
```

the argument to `peer_system_name` is `scarlett`, indicating that when `nomadc` logs in to `nubian-ppp`, it uses the login name `scarlett`.

`peer_ip_address` *Keyword*

The `peer_ip_address` keyword is required for multipoint links. It takes the host name or IP address of the remote endpoint as its argument. The previous example uses the host name `nomads` as the argument to keyword `peer_ip_address`.

Additional Keywords

You can supply other keywords in the `asppp.cf` file to define how endpoint machines should communicate. Refer to “Configuration Keywords” on page 471 for a complete list of keywords.

PPP Troubleshooting

If you have problems with a link after successfully establishing modem connections, you can use PPP-level diagnostics for troubleshooting. PPP-level diagnostics report detailed information about the activities of a link to help you determine where it is failing.

Analyzing Diagnostic Output

When a PPP link runs correctly, the `asppp.log` file includes diagnostic information in addition to its normal output. This section explains what the diagnostic messages mean. If your output is different, refer to RFC 1331.

Host and Modem Setup

This section contains messages that occur as the local host sends configuration information to the modem, and then as the modem tries to dial the remote host. These initial activities are actually handled by the UUCP daemon. You might think of them as the UUCP portion of asynchronous PPP communications. (Refer to Chapter 25 for complete details on UUCP.)

The following two messages should always appear at the beginning of the session. They indicate that the `aspppd` daemon has started successfully.

```
11:53:33 Link manager (1057) started 04/14/94
11:53:33 parse_config_file: Successful configuration
```

The next line indicates that a packet was routed to the `ipdptp0` interface on the local host. It helps you to determine if a dial-out is occurring correctly. For example, if you tried to `ping` the remote machine and this message isn't in `asppp.log`, the packet was lost, probably due to a routing problem.

Next, UUCP checks for an entry that matches `Ppac7` in a chat script in the `/etc/uucp/Systems` file. It then reports that it found an entry that had a device type ACUTEC. (For more information on the `Systems` file, refer to "UUCP `/etc/uucp/Systems` File" on page 491.)

```
11:53:46 process_ipd_msg: ipdptp0 needs connection
conn(Ppac7)
Trying entry from '/etc/uucp/Systems' - device type ACUTEC.
```

UUCP then finds the dialing information for an ACUTEC dialer in the `/etc/uucp/Devices` file. When it finds the information, it opens the appropriate serial port on the local host and sets it with a speed of 9600. (For more information on `/etc/uucp/Devices`, see "UUCP `/etc/uucp/Devices` File" on page 498.)

```
Device Type ACUTEC wanted
Trying device entry 'cua/a' from '/etc/uucp/Devices'.
processdev: calling setdevcfg(ppp, ACUTEC)
fd_mklock: ok
fixline(8, 9600)
gdial(tb9600-ec) calle
```

UUCP checks for the entry `tb9600` in the `/etc/uucp/Dialers` file and then sends out these messages.

```
Trying caller script 'tb9600-ec' from '/etc/uucp/Dialers'
expect: (''')
```

The host waits a couple of seconds and then sets the registers on the modem. The information shown in the following log is modem specific. It comes from the `/etc/uucp/Dialers` file.

```
got it
sendthem (DELAY)
APAUSE
APAUSE
APAUSE
T&D2E1V1X1Q0S2=255S12=255S50=6S58=2^M<NO CR>)
```

The next lines are the dialog between the modem and the host machine. `expect (OK^M)` means that the host expects the modem to send an "okay." The words `got it` at the end of the second line indicate that the host got the "okay" message from the modem.

```
expect: (OK^M)
AAAT&D2E1V1X1Q0S2=255S12=255S50=6S58=2^M^M^JOK^Mgot it
```

Next, the host sends the following string to the modem, which does the actual dialing. The phone number in the second line is retrieved from the entry for the remote host in the `/etc/uucp/Systems` file.

```
sendthem (ECHO CHECK ON
A^JATTDDTT99003300887744^M^M<NO CR>)
```

The line beginning with `expect` indicates that the local host expects to get a response from the modem at a speed of 9600 bps. The next line indicates that the modem responded.

```
expect: (CONNECT 9600)
^M^JCONNECT 9600got it
```

This line indicates that hardware flow control has started on the link. The host obtains flow control information from the `/etc/uucp/Dialers` file.

```
STTY crtscts
```

In the next series of messages, the local host waits for the remote host to send it a standard UNIX login prompt.

```
getty ret 8
expect: (‘‘‘‘)
got it
sandiest (^J^M)
expect: (login:)
```

The next messages indicate that the local host has received the login prompt from the remote. It then retrieves the appropriate login sequence from the chat script in the `/etc/uucp/Systems` entry for the remote host. This sequence is `Ppong^M`, which is required for login by the remote host.

```
^M^J^M^Jlogin:got it
sendthem (Ppong^M)
```

In these messages, the local host waits for the `ssword` prompt from the remote host. On receipt of the prompt, the local host sends the password retrieved from the chat script in the `/etc/uucp/Systems` entry for the remote host.

```
expect: (ssword:)
login: Ppong^M^JPassword:got it
```

The following messages indicate that dialing and modem connection completed successfully.

```
sendthem (ppptest1^M)
call cleanup(0)^M
```

Communications Between the Local and Remote Hosts

At this point, PPP communications start, as the link between local and remote hosts is now established.

The first lines in this part of the session constitute a *configuration request* (`Config-Req`). This is the first PPP packet sent to the remote host. The configuration request is one example of a Link Control Protocol (LCP) packet. It requests that configuration be set up and then sets up the PPP link between endpoint machines. The following example shows a sample configuration request.

EXAMPLE 24–3 Configuration Request

```
11:54:20 004298 ipdptp0 SEND PPP ASYNC 29 Octets LCP Config-Req
ID=4c LEN=24 MRU=1500 ACCM=00000000 MAG#=69f4f5b2 ProtFCOMP
AddrCCOMP
```

Here is a description of the configuration request.

- `11:54:20` – Time stamp field, indicating the time when the packet was sent
- `004298` – Number of the packet
- `ipdptp0` – Network interface used
- `SEND PPP ASYNC` – Indicates that the modem is sending asynchronous PPP
- `29 Octets` – Amount of data the host sent
- `LCP` – Packet type to send
- `ID=4c` – Identifier associated with the packet; it is actually part of the packet
- `LEN=24` – Length of the LCP part of the packet

The remaining items are a list of options to be negotiated between hosts.

- `MRU=1500` – Maximum receive unit (MRU), the largest packet size the calling host can receive from the remote host

- `ACCM=00000000` – Asynchronous Character Map (ACCM), the mask sent to the remote host that tells what control characters to escape on transmission
- `MAG#=69f4f5b2` – Magic number field; used for loopback–detection mechanism
- `ProtFCOMP AddrCCOMP` – Asks for the remote host to compress certain parts of the frame header (protocol field, address field)

The next lines are reporting invalid PPP packets. They come from the remote host, which is sending out UNIX text. This does not indicate a problem with PPP.

```
11:54:20 004299 ipdptp0 RECEIVE {Invalid ppp packet}PPP ASYNC 7
Octets [BAD FCS] {Unrecognized protocol:    1}

11:54:20 004299 ipdptp0 RECEIVE PPP ASYNC 73 Octets [BAD FCS]
{Unrecognized protocol:    880a}
```

In these packets, the local host receives the remote host's request for configuration, then sends out another configuration request. The packets are identical except for their ID fields. The ID field helps to distinguish between the two packets.

```
11:54:21 004301 ipdptp0 RECEIVE PPP ASYNC 29 Octets LCP Config-
Req  ID=35 LEN=24 MRU=1500 ACCM=00000000 MAG#=a8562e5f ProtFCOMP
AddrCCOMP

11:54:21 004302 ipdptp0 SEND PPP ASYNC 29 Octets LCP Config-Req
ID=4d LEN=24 MRU=1500 ACCM=00000000 MAG#=69f4f5b2 ProtFCOMP
AddrCCOMP
```

In this packet, the local host acknowledges the remote request by sending it a configuration acknowledgment (`Config-ACK`).

```
11:54:21 004303 ipdptp0 SEND PPP ASYNC 29 Octets LCP Config-ACK
ID=35 LEN=24 MRU=1500 ACCM=00000000 MAG#=a8562e5f ProtFCOMP
AddrCCOMP
```

The local host receives a configuration request (`Config-Req`) from the remote host.

```
11:54:21 004304 ipdptp0 RECEIVE PPP ASYNC 29 Octets LCP Config-
Req  ID=36 LEN=24 MRU=1500 ACCM=00000000 MAG#=a8562e5f ProtFCOMP
AddrCCOMP
```

In these packets, the local host acknowledges the second packet sent by the remote host and receives the remote host's acknowledgment.

```
11:54:21 004305 ipdptp0 SEND PPP ASYNC 29 Octets LCP Config-ACK
ID=36 LEN=24 MRU=1500 ACCM=00000000 MAG#=a8562e5f ProtFCOMP
AddrCCOMP

11:54:21 004306 ipdptp0 RECEIVE PPP ASYNC 29 Octets LCP Config-
ACK  ID=4d LEN=24 MRU=1500 ACCM=00000000 MAG#=69f4f5b2 ProtFCOMP
AddrCCOMP
```

Here the local host negotiates parameters about IP transmission. `LEN=16` gives the packet size. `VJCOMP` indicates Van Jacobsen header compression. `IPADDR` is followed by the calling host's IP address.

```
11:54:21 004307 ipdptp0 SEND PPP ASYNC 21 Octets IP_NCP Config-
Req  ID=4e LEN=16 VJCOMP MAXSID=15 Sid-comp-OK IPADDR=192.9.68.70
```

This packet indicates that the local host has received IP configuration from the remote host, including its IP address.

```
11:54:22 004308 ipdptp0 RECEIVE PPP ASYNC 21 Octets IP_NCP
Config-Req  ID=37 LEN=16 VJCOMP MAXSID=15 Sid-comp-OK
IPADDR=192.9.68.71
```

The local host sends this ACK to the remote host and receives an ACK from the remote host.

```
11:54:22 004309 ipdptp0 SEND PPP ASYNC 21 Octets IP_NCP Config-
ACK  ID=37 LEN=16 VJCOMP MAXSID=15 Sid-comp-OK IPADDR=192.9.68.71

11:54:22 004310 ipdptp0 RECEIVE PPP ASYNC 21 Octets IP_NCP
Config-ACK  ID=4e LEN=16 VJCOMP MAXSID=15 Sid-comp-OK
IPADDR=192.9.68.70
```

The first message that follows indicates that IP has started on the link. The next message indicates that the local host is sending IP traffic over the link.

```
11:54:22 start_ip: IP up on interface ipdptp0, timeout set for
120 seconds

11:54:24 004311 ipdptp0 SEND PPP ASYNC 89 Octets IP_PROTO
```

In the first message that follows, the local host receives IP traffic from the remote host. The subsequent messages indicate that the interface was disconnected because of an idle timeout.

```
11:54:25 004312 ipdptp0 RECEIVE PPP ASYNC 89 Octets IP_PROTO
11:56:25 process_ipd_msg: interface ipdptp0 has disconnected
11:56:25 disconnect: disconnected connection from  ipdptp0
```

The next messages begin the termination sequence. The first message indicates that the remote host has sent a packet to terminate the IP layer. The second is the local host's acknowledgment of the request to terminate.

```
11:56:25 004313 ipdptp0 RECEIVE PPP ASYNC 9 Octets IP_NCP Term-
REQ  ID=38 LEN=4

11:56:25 004314 ipdptp0 SEND PPP ASYNC 9 Octets IP_NCP Term-ACK
ID=38 LEN=4
```

The local host receives a request to terminate the LCP layer. The second message is an acknowledgment of the request, causing a graceful shutdown.

```
11:56:25 004315 ipdptp0 RECEIVE PPP ASYNC 9 Octets LCP Term-REQ
ID=39 LEN=4

11:56:25 004316 ipdptp0 SEND PPP ASYNC 9 Octets LCP Term-ACK
```

(Continuation)

```
ID=39 LEN=4
```

This message indicates that the link has closed.

```
11:56:29 004317 ipdptp0 PPP DIAG CLOSE
```

Dynamically Allocated PPP Links

A dial-in server with a dynamic point-to-point link gives your site all the advantages of point-to-point communications. Chapter 21 introduces this configuration type. It consists of remote hosts communicating with at least one dial-in server that dynamically allocates point-to-point links on an as-needed basis. The sample configuration shown next is used throughout this section.

Addressing Issues for Dynamically Allocated Links

You must add host information to the `/etc/inet/hosts` file for each machine that use the dynamically allocated PPP link. The IP addresses for the PPP endpoints should follow these conventions:

- For the dial-in server, you must use the IP address of the server's primary network interface (for example `le0` or `smc0`) as the address of the dynamic link.
- For a dynamic link, you don't need to assign an IP address to each remote host (as you would for a static link), but you do need to assign a remote IP address to each point-to-point IP interface on the server. The number of IP interfaces you can use is equal to the number of modems your server is connected to. For example, if you have three modems, you need three point-to-point IP interfaces and three IP addresses.
- You must include a dummy IP address for the `ifconfig` command to work properly on the client. This address acts as a placeholder for the local IP address assigned to the client IP interface when PPP is started.

Note - No restrictions are placed on the remote IP addresses that can be assigned to the IP interfaces, but, for clarity, it is probably best to include only IP addresses belonging to the same subnet.

Updating the `hosts` Database for Dynamic Links

You must update the `hosts` database on all machines involved in the dynamic-link configuration.

Considerations for Other Files

The next steps in the configuration process involve editing the `/etc/passwd` file and the `/etc/shadow` file. Edit these files for the dynamic–link configurations just as you would for the remote host–to–multipoint dial-in server configuration. Refer to “Modifying the `/etc/passwd` File” on page 431 for information regarding the `/etc/passwd` and `/etc/shadow` files.

Editing `asppp.cf` for Dynamic Link

The `asppp.cf` configuration file for a dynamic–link configuration must contain information about remote hosts and the interfaces to use for the PPP link. After the dial-in server boots, its link manager uses this information to establish communications whenever the server is called by a remote endpoint.

Remote Host with Dynamic Link

The `asppp.cf` configuration file for a remote host is the same as the one described in “Parts of Basic Configuration File” on page 449, except for the addition of the parameter *negotiate_address*:

```
ifconfig ipdptp0 plumb dummy mojave up
path
    interface ipdptp0
    peer_system_name mojave-ppp
    connectivity_timeout 300
    negotiate_address on
```

The *negotiate_address* parameter indicates whether local IP address assignment is obtained through negotiation and assigned dynamically. If set to “on”, the IP address supplied by the server is used as the client’s local address for the duration of the connection.

Dial-in Server With Dynamic Link

When the dial-in server receives an incoming packet, the link manager reads the `path` sections of its configuration file to identify the remote endpoint and determine the interface to use. The configuration file shown in Example 24–4 does not contain an interface keyword. Instead, the link manager uses interface information established in the `defaults` section.

The `asppp.cf` configuration file for a dial-in server with dynamically allocated links might look like the following:

EXAMPLE 24–4 Configuration File for Server With Dynamically Allocated Link

```
ifconfig ipdptp0 plumb mojave clienta down
ifconfig ipdptp1 plumb mojave clientb down
ifconfig ipdptp2 plumb mojave clientc down

# This means grab whatever interface is available (not in use)
defaults
     interface ipdptp*

# Each path specifies a machine that might dial up / log
# in to this server

path
    peer_system_name tamerlane   # nomada uses the login name
                                 # tamerlane

path
    peer_system_name lawrence    # nomadb uses the name lawrence
                                 # for login

path
    peer_system_name nomadc
```

`ifconfig` *Section for Server With Dynamic Links*

The `ifconfig` section for a dial-in server with a dynamically allocated link has the syntax:

`ifconfig ipdptp`*n* `plumb` *server-name client-address* `down`

Example 24–4 contains three `ifconfig` lines, each initializing a point-to-point interface.

```
ifconfig  ipdptp0  plumb  mojave  clienta  down
ifconfig  ipdptp1  plumb  mojave  clientb  down
ifconfig  ipdptp2  plumb  mojave  clientc  down
```

`defaults` *Section for Server With Dynamic Links*

When you configure a dynamically allocated link, you might want to include a `defaults` section in the `asppp.cf` file. This section sets the defaults for the value replacing *keyword*, wherever *keyword* subsequently appears in the `asppp.cf` file. The syntax for the `defaults` section is:

```
default
     keyword
```

Example 24–4 uses the keyword `interface` to define the interface as `ipdptp*`, indicating a dynamic link. The asterisk wildcard tells the link manager to use any available `ipdptp` interface defined in the `ifconfig` section. Thus the link manager on server `mojave` uses either `ipdptp0`, `ipdptp1`, or `ipdptp2`—whichever is the first interface configured "down" that it finds.

`path` *Section for Server With Dynamic Links*

The configuration file for the server with dynamic links must contain `path` sections for every remote host permitted to establish connections with the server. The path section has the following syntax:

```
path
    peer_system_name endpoint-username
```

No `interface` keyword has been defined in the `path` section because this value is defined in the defaults section. The `peer_system_name` keyword has the same meaning here as it does in the configuration file for the multipoint server. See "`path` Section for Multipoint Dial-in Server" on page 453 for more information.

Additional Keywords

You can supply other keywords in the `asppp.cf` file to define how endpoint machines should communicate, including the use of security keywords as explained in "Configuration Keywords" on page 471.

Configuring a Virtual Network

Virtual networks consist of a group of standalone computers, each in an isolated location, that can connect to each other through PPP multipoint links. “Virtual Networks” on page 406 introduces virtual network concepts. This section explains how to configure a virtual network.

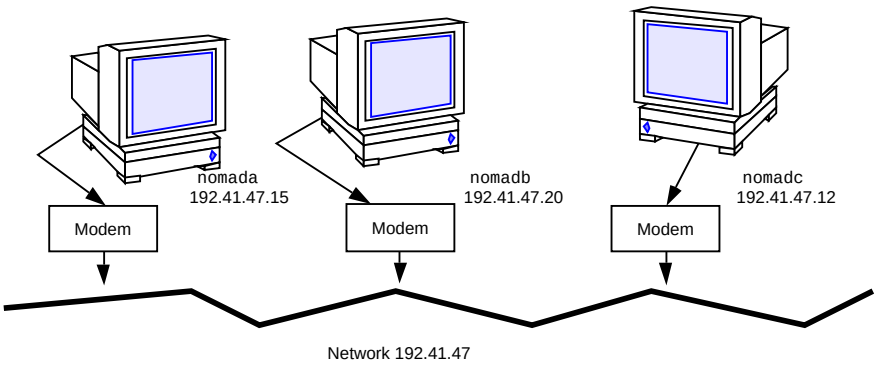


Figure 24–1 Sample Virtual Network

The network shown in the sample consists of three isolated computers. Each member of the network connects to the other members of the network through a multipoint PPP link. Therefore, to create such a network, you (and perhaps other network administrators at the remote location) have to configure a multipoint PPP link on each participating host.

You use the same generic process for configuring multipoint links as you do for configuring a multipoint dial-in server link, as described in “Overview of the Configuration Process” on page 426. However, the virtual network has its own set of issues and requires you to configure each host in the network accordingly.

Addressing Issues for Virtual Networks

You must add host information to the `/etc/hosts` file for each machine in the virtual network. When typing the IP addresses used for the PPP endpoints:

- Designate a PPP-specific IP address for its point-to-point link. Note that if the machine was not previously configured in a physical network, you must create an IP address for the PPP link. This address becomes the host’s primary network interface.
- Create a network number for the virtual network. See “Assigning a Network Number to the PPP Link” on page 420 for more information.

Updating hosts and networks Databases

The first step in the configuration process involves updating the hosts and networks databases with information about your virtual network.

/etc/inet/hosts File for the Virtual Network

The /etc/inet/hosts file on each machine must contain the addressing information for every member of the network that this host has permission to access. For example, each host in the network in Figure 24–1 would have this information:

```
# Internet host table
#
127.0.0.1           localhost loghost
192.41.47.15        nomada
192.41.47.20        nomadb
192.41.47.12        nomadc
```

/etc/inet/networks File for the Virtual Network

Because the virtual network requires a unique IP address, you must type this address in the networks database. For example, the network shown in Figure 24–1 has the number 192.41.47. Moreover, if the hosts on the network need to communicate with other networks, you should register the network with the InterNIC addressing authority. See “networks database” on page 140 for information on editing the networks database.

Each host on the virtual network must have an entry with the network’s address in the /etc/inet/networks file. For example, each host on network 192.41.47 might have the following in /etc/inet/networks:

```
# Internet networks
#
# arpanet   10          arpa
# ucb-ether 46          ucbether
#
# local networks
loopback    127
ppp         192.41.47   #remote sales offices
```

Considerations for Other Files

The next steps in the configuration process involve editing the UUCP databases, the /etc/passwd file, and the /etc/shadow file. You edit these files for the machines in the virtual network just as you would for the multipoint dial-in server configuration. Refer to “Editing UUCP Databases” on page 430 for UUCP-related

information and “Modifying the /etc/passwd File” on page 431 for information regarding the passwd file.

asppp.cf Configuration File for a Virtual Network

The configuration file for a local machine on a virtual network must contain information about all remote hosts on the network that the local host can access. Moreover, each machine on the virtual network must be configured for both dial-in and dial-out functions. After the local machine boots, its link manager reads the asppp.cf file to establish communications.

The following example shows a configuration file such as you would set up for nomada on a virtual network 192.41.47.

EXAMPLE 24–5 Configuration File for nomada

```
# /etc/asppp.cf for hosta

ifconfig ipd0 plumb nomada netmask + up
defaults
   interface ipd0
path
   peer_ip_address  nomadb
   peer_system_name lawrence     # name machine logs in with
path
   peer_ip_address nomadc
   peer_system_name azziz
```

The following example shows a configuration file such as you would set up for nomadb on virtual network 192.41.47.

EXAMPLE 24–6 Configuration File for nomadb

```
# /etc/asppp.cf for nomadb

ifconfig ipd0 plumb nomadb netmask + up
defaults
   interface ipd0
path
   peer_ip_address   nomada
   peer_system_name  tamerlane  # name the machine logs in with
path
   peer_ip_address   nomadc
```

(continued)

```
   peer_system_name  azziz
```

Rules for PAP/CHAP Keywords

- Either server or client can require authentication or offer to do authentication.
- If PAP and CHAP are both present, the authenticator first tries CHAP. If that fails, the link is terminated. The authenticator will not try PAP.
- The default value for PAP and CHAP authentication keywords is off. The syntax for keywords is:

```
require_authentication     off | pap[chap] | chap[pap]

will_do_authentication     off | pap[chap] | chap[pap]
```

- If you fail to specify `pap_id` and `pap_password` or `pap_peer_id` and `pap_peer_password` keywords and values for the associated path, the corresponding values are set to the NULL string.
- You must specify `chap_name`, `chap_secret`, `chap_peer_secret`, and `chap_peer_name` keywords and values for that path.
- **peername** – The name of the system at the other end of the point-to-point link from the authenticator. It takes the form of a string with the syntax specified below.
- **string** – A single token without embedded white space. The standard ANSI C \ escape sequence can be used to embed special characters. Use \s for the space character. Any pound sign at the beginning of the string must be escaped (\#) to avoid interpretation as a comment. A NULL (\0) truncates the string.

TABLE 24–1 PAP/CHAP Keyword Definitions

Keywords	Value Definition
`require_authentication` *keywords*	Specifies whether the peer must authenticate itself. If either `pap` or `chap` is present, the peer must participate in authentication or end the connection. The default value is off.
`pap_peer_id` *peername*	Specifies the name of the peer to be authenticated for the current path. *peername* string is one or more octets. To indicate a zero-length string, do not include the keyword.
`pap_peer_password` *string*	Specifies password for peer in one or more octets. To indicate a zero-length string, do not include the keyword.
`chap_peer_secret` *string*	Specifies the secret used with the challenge value to generate the response sent by the peer. The format is one or more octets, preferably at least 16.
`chap_peer_name` *peername*	Specifies the identity of the peer transmitting the packet. The name should not be NULL or terminated with CR/LF. The name is received from the peer in a response packet and consists of one or more octets.
`will_do_authentication` *keywords*	Specifies whether the system is willing to participate as the authenticated peer in the specified authentication process. If both `pap` and `chap` are present, the system is willing to participate in either authentication protocol. The default value is off.
`pap_id` *peername*	Specifies the name of the system to be sent to the authenticator in the response packet. To indicate a zero-length string, do not include the keyword.
`pap_password` *string*	Specifies the password for the system to be sent to the authenticator in the response packet. To indicate a zero-length string, do not include the keyword.
`chap_secret` *string*	Contains the secret that is used with the received challenge value to generate the response sent to the authenticator. The format is one or more octets, preferably at least 16.
`chap_name` *peername*	Specifies the identity of the system. The name should not end with a NULL or CR/LF. The name is sent to the authenticator in a response packet.

Configuration Keywords

This section describes the configuration keywords available for the `asppp.cf` configuration file and the values you must define for them. Most of these keywords are optional. The required ones are indicated. For further explanations of the keywords, refer to RFCs 1331, 1332, 1333, and 1334.

The following table lists required keywords that must appear in all `asppp.cf` configuration files.

TABLE 24–2 Required Keywords for `asppp.cf`

Keywords	Value Definitions
`ifconfig` *parameters*	Tells the link manager to run the `ifconfig` command with the values supplied by *parameters*. See "`ifconfig` Section of the `asppp.cf` File" on page 450, "`ifconfig` Section for Multipoint Dial-in Server" on page 452, and the `ifconfig`(1M) man page for more information.
`path`	Specifies the beginning of the token sequences that are grouped as attributes of this (current) path. The collection of attributes comprising the current path are terminated by the occurrence of a subsequent `path` keyword, `defaults` keyword, or the end–of–file character.
`interface` (`ipdptp`*n*, `ipdptp*`, or `ipd`*n*)	Specifies either an `ipdptp` (static point-to-point), `ipdptp*` (dynamic point-to-point), or `ipd` (multipoint) device for each interface in your network. For `ipdptp`*n* and `ipd`*n*, this keyword associates the specific interface defined by *n* with the current path. *n* must be a non-negative integer. It matches the interface defined in the `path` section with the interface stated in the `ifconfig` section. For the `ipdptp*` interface, the `*` indicates that the interface will match any point-to-point interface that is configured as "down."

TABLE 24–2 Required Keywords for `asppp.cf` *(continued)*

Keywords	Value Definitions
`peer_system_name` *hostname* `peer_system_name` *username*	On dial-out machines, specifies the *hostname* of the remote endpoint that the local machine wants to call. This is the same as the system name in the `/etc/uucp/Systems` file. Associates the remote system name with the current path. This name is used to look up modem- and peer-specific information for outbound connections in the `/etc/uucp/Systems` file. On dial-in machines, this keyword specifies the *username* that remote machines use when logging in to the dial-in machine. The appropriate path is determined by matching *username* with the login name that was used to obtain the connection.
`peer_ip_address` *hostname* `peer_ip_address` *ip-address*	Specifies the destination host address. It is required only for multipoint links. This address is associated with the current path. The value is ignored if the path specifies a point-to-point interface. The address format can be dotted decimal, hexadecimal, or symbolic.

The following table contains optional keywords for `asppp.cf` that you can use to further define your PPP configuration.

TABLE 24–3 Optional Keywords for `asppp.cf`

Keywords	Value Definitions
`debug_level` *0–9*	The integer between 0 and 9 defines how much debugging information should be written to the log file. The higher the number, the more output is generated.
`defaults`	Indicates that all following token sequences up to the next `path` keyword, or the end-of-file character, set default attributes that affect subsequently defined paths.
`default_route`	Tells the link manager to add the path's peer IP address to the route table as the default destination when the IP layer corresponding to the current path is fully operational. The route is removed when the IP layer is shut down.
`inactivity_timeout` *seconds*	Specifies the maximum number of seconds that the connection for the current path can remain idle before it is terminated. A zero can be specified to indicate no timeout. The default is 120 seconds.
`ipcp_async_map` *hex-number*	Specifies the asynchronous control-character map for the current path. *hex-number* indicates the natural (big-endian) form of the four octets that comprise the map. The default value is `0x FFFFFFFF`.

TABLE 24–3 Optional Keywords for `asppp.cf` *(continued)*

Keywords	Value Definitions
`ipcp_compression` (vj or off)	Specifies whether IP compression is enabled. The Van Jacobson compression algorithm (vj) is the default.
`lcp_compression` (on or off)	Specifies whether PPP address, control, and protocol field compression are enabled. The default is on.
`lcp_mru` *number*	Specifies the value of the desired maximum receive unit packet size. The number is the size in octets. The default is 1500.
`negotiate_address` (on or off)	Indicates whether local IP address assignment is obtained through negotiation and assigned dynamically or not. If enabled, the local address is obtained from the remote end of the PPP link. If so obtained, any local address other than 0.0.0.0 can be used to initially configure the interface. The default is not to negotiate (off).
`peer_ip_address` *hostname* `peer_ip_address` *ip-address*	Specifies the destination host address. This keyword is optional for point-to-point links only. *address* is associated with the current path. The address format can be dotted decimal, hexadecimal, or symbolic.
`version` *n*	Specifies that the contents of the configuration file correspond to format version *n*. If this keyword is present, it must be the first keyword in the file. If absent, the version is assumed to be 1. This book contains the definition of the version 1 format for the configuration file.

CHAPTER 25

Overview of UUCP

This chapter introduces the UNIX-to-UNIX Copy Program (UUCP) and daemons. The following topics are covered:

UUCP enables computers to transfer files and exchange mail with each other. It also enables computers to participate in large networks such as Usenet.

The Solaris environment provides the Basic Network Utilities (BNU) version of UUCP, also known as HoneyDanBer UUCP. The term *UUCP* denotes the complete range of files and utilities that make up the system, of which the program `uucp` is only a part. The UUCP utilities range from those used to copy files between computers (`uucp` and `uuto`) to those used for remote login and command execution (`cu` and `uux`).

UUCP Hardware Configurations

UUCP supports the following hardware configurations:

Direct links	You can create a direct link to another computer by running RS-232 cables between serial ports on the two machines. Direct links are useful where two computers communicate regularly and are physically close—within 50 feet of each other. You can use a limited distance–modem to increase this distance somewhat.
Telephone lines	Using an automatic call unit (ACU), such as a high-speed modem, your machine can communicate with other computers over standard phone lines. The modem dials the telephone number requested by UUCP. The recipient machine must have a modem capable of answering incoming calls.
Network	UUCP can also communicate over a network running TCP/IP or an other protocol family. After your computer has been established as a host on a network, it can contact any other host connected to the network.

This chapter assumes that your UUCP hardware has already been assembled and configured. If you need to set up a modem, refer to *System Administration Guide, Volume 1* and the manuals that came with the modem for assistance.

UUCP Software

The UUCP software is automatically included when you run the Solaris installation program and select the entire distribution. Alternatively, you can add it using `pkgadd`. The UUCP programs can be divided into three categories: daemons, administrative programs, and user programs.

UUCP Daemons

The UUCP system has four daemons: `uucico`, `uuxqt`, `uusched`, and `in.uucpd`. These daemons handle UUCP file transfers and command executions. You can also run them manually from the shell, if necessary.

`uucico`	Selects the device used for the link, establishes the link to the remote computer, performs the required login sequence and permission checks, transfers data and execute files, logs results, and notifies the user by mail of transfer completions. `uucico` acts as the "login shell" for UUCP login accounts. When the local `uucico` daemon calls a remote machine, it communicates directly with the remote `uucico` daemon during the session. After all the required files have been created,`uucp`, `uuto`, and `uux` programs execute the `uucico` daemon to contact the remote computer. `uusched` and `Uutry` all execute `uucico`. (See the `uucico`(1M) man page for details.)
`uuxqt`	Executes remote execution requests. It searches the spool directory for execute files (always named X.*file*) that have been sent from a remote computer. When an X.*file* file is found, `uuxqt` opens it to get the list of data files that are required for the execution. It then checks to see if the required data files are available and accessible. If the files are available, `uuxqt` checks the `Permissions` file to verify that it has permission to execute the requested command. The `uuxqt` daemon is executed by the `uudemon.hour` shell script, which is started by `cron`. (See the `uuxqt`(1M) man page for details.)
`uusched`	Schedules the queued work in the spool directory. `uusched` is initially run at boot time by the `uudemon.hour` shell script, which is started by `cron`. (See the `uusched`(1M) man page for details.) Before starting the `uucico` daemon, `uusched` randomizes the order in which remote computers are called.
`in.uucpd`	Supports UUCP connections over networks. The `inetd` on the remote host invokes `in.uucpd` whenever a UUCP connection is established. `uucpd` then prompts for a login name. `uucico` on the calling host must respond with a login name. `in.uucpd` then prompts for a password, unless one is not required. (See the `in.uucpd`(1M) man page for details.)

UUCP Administrative Programs

Most UUCP administrative programs are in `/usr/lib/uucp`. Most basic database files are in `/etc/uucp`. The only exception is `uulog`, which is in `/usr/bin`. The home directory of the `uucp` login ID is `/usr/lib/uucp`. When running the administrative programs through `su` or `login`, use the `uucp` user ID. It owns the programs and spooled data files.

`uulog`	Displays the contents of a specified computer's log files. Log files are created for each remote computer with which your machine communicates. The log files record each use of `uucp`, `uuto`, and `uux`. (See the `uucp`(1C) man page for details.)
`uucleanup`	Cleans up the spool directory. It is normally executed from the `uudemon.cleanup` shell script, which is started by `cron`. (See the `uucleanup`(1M) man page for details.)
`Uutry`	Tests call-processing capabilities and does moderate debugging. It invokes the `uucico` daemon to establish a communication link between your machine and the remote computer you specify. (See the `Uutry`(1M) man page for details.)
`uucheck`	Checks for the presence of UUCP directories, programs, and support files. It can also check certain parts of the `/etc/uucp/Permissions` file for obvious syntactic errors. (See the `uucheck`(1M) man page for details.)

UUCP User Programs

The UUCP user programs are in `/usr/bin`. You do not need special permission to use these programs.

`cu`	Connects your machine to a remote computer so that you can log in to both at the same time. `cu` enables you to transfer files or execute commands on either machine without dropping the initial link. (See the `cu`(1C) man page for details.)
`uucp`	Lets you copy a file from one machine to another. It creates work files and data files, queues the job for transfer, and calls the `uucico` daemon, which in turn attempts to contact the remote computer. (See the `uucp`(1C) man page for details.)
`uuto`	Copies files from the local machine to the public spool directory `/var/spool/uucppublic/receive` on the remote machine. Unlike `uucp`, which lets you copy a file to any accessible directory on the remote machine, `uuto` places the file in an appropriate spool directory and tells the remote user to pick it up with `uupick`. (See the `uuto`(1C) man page for details.)
`uupick`	Retrieves files in `/var/spool/uucppublic/receive` when files are transferred to a computer using `uuto`. (See the `uuto`(1C) man page.)

`uux`	Creates the work, data, and execute files needed to execute commands on a remote machine. (See the `uux`(1C) man page for details.)
`uustat`	Displays the status of requested transfers (`uucp`, `uuto`, or `uux`). It also provides a means of controlling queued transfers. (See the `uustat`(1C) man page for details.)

UUCP Database Files

A major part of UUCP setup is the configuration of the files making up the UUCP database. These files are in the `/etc/uucp` directory. You need to edit them to set up UUCP or PPP on your machine. The files include:

`Config`	Contains a list of variable parameters. You can manually set these parameters to configure the network.
`Devconfig`	Used to configure network communications.
`Devices`	Used to configure network communications.
`Dialcodes`	Contains dial-code abbreviations that can be used in the phone number field of `Systems` file entries. Though not required, it can be used by PPP as well as UUCP.
`Dialers`	Contains character strings required to negotiate with modems to establish connections with remote computers. It is used by PPP as well as UUCP.
`Grades`	Defines job grades, and the permissions associated with each job grade, that users can specify to queue jobs to a remote computer.
`Limits`	Defines the maximum number of simultaneous `uucico`s, `uuxqt`s, and `uusched`s permitted on your machine.
`Permissions`	Defines the level of access granted to remote hosts that attempt to transfer files or execute commands on your machine.
`Poll`	Defines machines that are to be polled by your system and when they are polled.
`Sysfiles`	Assigns different or multiple files to be used by `uucico` and `cu` as `Systems`, `Devices`, and `Dialers` files.

`Sysname`	Enables you to define a unique UUCP name for a machine in addition to its TCP/IP host name.
`Systems`	Contains information needed by the `uucico` daemon, `cu`, and PPP to establish a link to a remote computer. This information includes the name of the remote host, the name of the connecting device associated with the remote host, time when the host can be reached, telephone number, login ID, and password.

Several other files can be considered part of the supporting database but are not directly involved in establishing a link and transferring files.

Configuring UUCP Database Files

The UUCP database consists of the files shown in "`UUCP` Database Files" on page 479. However, basic UUCP configuration involves only the following critical files:

- `/etc/uucp/Systems`
- `/etc/uucp/Devices`
- `/etc/uucp/Dialers`

Because PPP uses some of the UUCP databases, you should understand at least these critical database files if you plan to configure PPP. After these databases are configured, UUCP administration is fairly straightforward. Typically, you edit the `Systems` file first, then edit the `Devices` file. You can usually use the default `/etc/uucp/Dialers` file, unless you plan to add dialers that aren't in the default file. In addition, you might also want to use the following files for basic UUCP and PPP configuration:

- `/etc/uucp/Sysfiles`
- `/etc/uucp/Dialcodes`
- `/etc/uucp/Sysname`

Because these files work closely with one another, you should understand the contents of them all before you change any one of them. A change to an entry in one file might require a change to a related entry in another file. The remaining files listed in "`UUCP` Database Files" on page 479 are not as critically intertwined.

Note - PPP uses only the files described in this section. It does not use the other UUCP database files.

CHAPTER 26

Administering UUCP

This chapter explains how to start UUCP operations after you have modified the database file relevant to your machines. The chapter contains procedures and troubleshooting information for setting up and maintaining UUCP on machines running the Solaris environment, such as:

UUCP Administration Task Map

The following table provides pointers to the procedures covered in this chapter, as well as a short description of each procedure.

TABLE 26–1 Task Map: UUCP Administration

Task...	Description	For Instructions, Go To ...
Allow remote machines to have access to your system	Edit the `/etc/paasswd` file to add entries to identify the machines permitted to access your system	"How to Add UUCP Logins" on page 482
Start UUCP	Use the supplied shell scripts to start UUCP	"How to Start UUCP" on page 484
Enable UUCP to work with TCP/IP	Edit `/etc/inetd.conf` and `/etc/uucp/Systems` files to activate UUCP for TCP/IP	"How to Activate UUCP for TCP/IP" on page 486
Troubleshoot some common UUCP problems	Diagnostic steps to use to check for faulty modems or ACUs Diagnostic steps to use for debugging transmissions	"How to Check for Faulty Modems or ACUs" on page 488 "How to Debug Transmissions" on page 488

Adding UUCP Logins

For incoming UUCP (`uucico`) requests from remote machines to be handled properly, each machine has to have a login on your system.

▼ How to Add UUCP Logins

To allow a remote machine to access your system, you need to add an entry to the `/etc/passwd` file as follows:

1. **Edit the `/etc/passwd` file and add the entry to identify the machine permitted to access your system.**

   A typical entry that you might put into the `/etc/passwd` file for a remote machine permitted to access your system with a UUCP connection would be as follows:

```
Ugobi:*:5:5:gobi:/var/spool/uucppublic:/usr/lib/uucp/uucico
```

By convention, the login name of a remote machine is the machine name preceded by the uppercase letter `U`. Note that the name should not exceed eight characters, so that in some cases you might have to truncate or abbreviate it.

The previous entry shows that a login request by `Ugobi` is answered by `/usr/lib/uucp/uucico`. The home directory is `/var/spool/uucppublic`. The password is obtained from the `/etc/shadow` file. You must coordinate the password and the login name with the UUCP administrator of the remote machine. The remote administrator must then add an appropriate entry, with login name and unencrypted password, in the remote machine's `Systems` file.

2. **Coordinate your machine name with the UUCP administrators on other systems.**

   Similarly, you must coordinate your machine's name and password with the UUCP administrators of all machines that you want to reach through UUCP.

Starting UUCP

UUCP comes with four shell scripts that poll remote machines, reschedule transmissions, and clean up old log files and unsuccessful transmissions. The scripts are:

- `uudemon.poll`
- `uudemon.hour`
- `uudemon.admin`
- `uudemon.cleanup`

These shell scripts should execute regularly to keep UUCP running smoothly. The `crontab` file to run the scripts is automatically created in `/usr/lib/uucp/uudemon.crontab` as part of the Solaris installation process, if you select the full installation. Otherwise, it is created when you install the UUCP package.

You can also run the UUCP shell scripts manually. The following is the prototype `uudemon.crontab` file that you can tailor for a particular machine:

```
#
#ident  "@(#)uudemon.crontab    1.5     97/12/09 SMI"
#
# This crontab is provided as a sample. For systems
# running UUCP edit the time schedule to suit, uncomment
# the following lines, and use crontab(1) to activate the
# new schedule.
```

(continued)

```
#
#48 8,12,16 * * * /usr/lib/uucp/uudemon.admin
#20 3 * * * /usr/lib/uucp/uudemon.cleanup
#0 * * * * /usr/lib/uucp/uudemon.poll
#11,41 * * * * /usr/lib/uucp/uudemon.hour
```

Note - By default, UUCP operations are disabled. To enable UUCP, edit the time schedule and uncomment the appropriate lines in the `uudemon.crontab` file.

▼ How to Start UUCP

To activate the `uudemon.crontab` file, do the following:

1. **Become superuser.**
2. **Edit the `/usr/lib/uucp/uudemon.crontab` file and change entries as required**
3. **Issue:**

```
crontab < /usr/lib/uucp/uudemon.crontab
```

`uudemon.poll` Shell Script

The default `uudemon.poll` shell script reads the `/etc/uucp/Poll` file once an hour. If any machines in the `Poll` file are scheduled to be polled, a work file (`C.`*sysnxxxx*) is placed in the `/var/spool/uucp/`*nodename* directory, where *nodename* represents the UUCP node name of the machine.

The shell script is scheduled to run once an hour, before `uudemon.hour`, so that the work files are there when `uudemon.hour` is called.

`uudemon.hour` Shell Script

The default `uudemon.hour` shell script:

- Calls the `uusched` program to search the spool directories for work files (`C.`) that have not been processed and schedules these files for transfer to a remote machine.
- Calls the `uuxqt` daemon to search the spool directories for execute files (`X.`) that have been transferred to your computer and were not processed at the time they were transferred.

By default, `uudemon.hour` runs twice an hour. You might want it to run more often if you expect high failure rates of calls to remote machines.

`uudemon.admin` Shell Script

The default `uudemon.admin` shell script does the following:

- Runs the `uustat` command with `p` and `q` options. The `q` reports on the status of work files (`C.`), data files (`D.`), and execute files (`X.`) that are queued. The `p` prints process information for networking processes listed in the lock files (`/var/spool/locks`).
- Sends resulting status information to the `uucp` administrative login using `mail`.

`uudemon.cleanup` Shell Script

The default `uudemon.cleanup` shell script does the following:

- Takes log files for individual machines from the `/var/uucp/.Log` directory, merges them, and places them in the `/var/uucp/.Old` directory with other old log information.
- Removes work files (`C.`) seven days old or older, data files (`D.`) seven days old or older, and execute files (`X.`) two days old or older from the spool files.
- Returns mail that cannot be delivered to the sender.
- Mails a summary of the status information gathered during the current day to the UUCP administrative login (`uucp`).

Running UUCP Over TCP/IP

To run UUCP on a TCP/IP network, you need to make a few modifications, as described in this section.

▼ How to Activate UUCP for TCP/IP

1. **Edit the `/etc/inetd.conf` file and make sure that the following entry is not preceded by a comment mark (#):**

```
uucp stream tcp nowait root /usr/sbin/in.uucpd in.uucpd
```

2. **Edit the `/etc/uucp/Systems` file to make sure that the entries have the following fields :**

   System-Name Time TCP Port networkname Standard-Login-Chat

   A typical entry would look like this:

```
rochester Any TCP - ur-seneca login: Umachine password: xxx
```

   Notice that the *networkname* field permits you to specify explicitly the TCP/IP host name. This is important for some sites. In the previous example, the site has the UUCP node name `rochester` which is different from its TCP/IP host name `ur-seneca`. Moreover, there could easily be a completely different machine running UUCP that has the TCP/IP host name of `rochester`.

   The Port field in the `Systems` file should have the entry `-`. This is equivalent to listing it as `uucp`. In almost every case, the *networkname* is the same as the system name, and the Port field is `-`, which says to use the standard uucp port from the `services` database. The `in.uucpd` daemon expects the remote machine to send its login and password for authentication, and it prompts for them much as `getty` and `login` do.

3. **Edit the `/etc/inet/services` file to set up a port for UUCP:**

```
uucp 540/tcp uucpd # uucp daemon
```

   You should not have to change the entry. However, if your machine runs NIS or NIS+ as its name service, you should change the `/etc/nsswitch.conf` entry for `/etc/services` to check `files` first, then check `nis` or `nisplus`.

UUCP Security and Maintenance

After you have set up UUCP, maintenance is straightforward. This section explains ongoing UUCP tasks with regard to security, maintenance, and troubleshooting.

Setting Up UUCP Security

The default `/etc/uucp/Permissions` file provides the maximum amount of security for your UUCP links. The default `Permissions` file contains no entries.

You can set additional parameters for each remote machine to define:

- Ways the remote machine can receive files from your machine
- Directories for which the remote machine has read and write permission
- Commands the remote machine can use for remote execution

A typical `Permissions` entry is:

```
MACHINE=datsun LOGNAME=Udatsun VALIDATE=datsun
COMMANDS=rmail REQUEST=yes SENDFILES=yes
```

This entry allows files to be sent and received (to and from the “normal” UUCP directories, not from anywhere in the system) and causes the UUCP user name to be validated at login time.

Regular UUCP Maintenance

UUCP does not require much maintenance. Apart from making sure that the `crontab` file is in place, as described in the section “How to Start UUCP” on page 484, all you have to worry about is the growth of mail files and the public directory.

Email for UUCP

All email messages generated by the UUCP programs and scripts go to the user ID `uucp`. If you do not log in frequently as that user, you might not realize that mail is accumulating (and consuming disk space). To solve this, make an alias in `/etc/mail/aliases` and redirect that email either to `root` or to yourself and others responsible for maintaining UUCP. Remember to run the `newaliases` command after modifying the `aliases` file.

UUCP Public Directory

The directory `/var/spool/uucppublic` is the one place in every system to which UUCP by default is able to copy files. Every user has permission to change to `/var/spool/uucppublic` and read and write files in it. However, its sticky bit is set, so its mode is 01777. As a result, users cannot remove files that have been copied to it and that belong to `uucp`. Only you, as UUCP administrator logged in as `root` or

uucp, can remove files from this directory. To prevent the uncontrolled accumulation of files in this directory, you should make sure to clean it up periodically.

If this is inconvenient for users, encourage them to use uuto and uupick rather than removing the sticky bit, which is set for security reasons. (See the uuto(1C) man page for instructions for using uuto and uupick.) You can also restrict the mode of the directory to only one group of people. If you do not want to run the risk of someone filling your disk, you can even deny UUCP access to it.

Troubleshooting UUCP

These procedures describe how to solve common UUCP problems.

▼ How to Check for Faulty Modems or ACUs

You can check if the modems or other ACUs are not working properly in several ways.

1. **Get counts and reasons for contact failure by running:**

```
uustat –q
```

2. **Call over a particular line and print debugging information on the attempt.**
   The line must be defined as direct in the /etc/uucp/Devices file. (You must add a telephone number to the end of the command line if the line is connected to an autodialer or the device must be set up as direct.) Type:

```
cu –d –l line
```

   where *line* is /dev/cua/a.

▼ How to Debug Transmissions

If you cannot contact a particular machine, you can check out communications to that machine with Uutry and uucp.

1. **To try to make contact by typing:**

```
/usr/lib/uucp/Uutry -r machine
```

Replace *machine* with the host name of the machine you are having problems contacting. This command:

a. **Starts the transfer daemon (**`uucico`**) with debugging. You can get more debugging information if you are** `root`**.**

b. **Directs the debugging output to** `/tmp/`***machine*****.**

c. **Prints the debugging output to your terminal by issuing:**

```
tail –f
```

Press Control-c to end output. You can copy the output from `/tmp/`*machine* if you want to save it.

2. **If** `Uutry` **doesn't isolate the problem, try to queue a job by typing:**

```
uucp -r file machine\!/dir/file
```

Replace *file* by the file you want to transfer, *machine* by the machine you want to copy to, and */dir/file* where the file will be placed on the other machine. The `r` option queues a job but does not start the transfer.

3. **Issue:**

```
Uutry
```

If you still cannot solve the problem, you might need to call your local support representative. Save the debugging output; it will help diagnose the problem.

You might also want to decrease or increase the level of debugging provided by `Uutry` through the `–x` *n* option, where *n* indicates the debug level. The default debug level for `Uutry` is 5.

Debug level 3 provides basic information as to when and how the connection is established, but not much information about the transmission itself. Debug level 9, on the other hand, provides exhaustive information about the transmission process. Be aware that debugging occurs at both ends of the transmission. If you intend to

use a level higher than 5 on a moderately large text, contact the administrator of the other site and agree on a time for doing so.

Checking the UUCP /etc/uucp/Systems File

Verify that you have up-to-date information in your Systems file if you are having trouble contacting a particular machine. Some things that might be out of date for a machine are its:

- Phone number
- Login ID
- Password

Checking UUCP Error Messages

UUCP has two types of error messages: ASSERT and STATUS.

- When a process is aborted, ASSERT error messages are recorded in /var/uucp/.Admin/errors. These messages include the file name, sccsid, line number, and text. These messages usually result from system problems.
- STATUS error messages are stored in the /var/uucp/.Status directory. The directory contains a separate file for each remote machine your computer attempts to communicate with. These files contain status information on the attempted communication and whether it was successful.

Checking Basic Information

Several commands are available for checking basic networking information:

- Use the uuname command to list those machines your machine can contact.
- Use the uulog command to display the contents of the log directories for particular hosts.
- Use the uucheck –v command to check for the presence of files and directories needed by uucp. This command also checks the Permissions file and outputs information on the permissions you have set up.

CHAPTER 27

UUCP Reference

This chapter provides reference information for working with UUCP. The following topics are covered:

UUCP `/etc/uucp/Systems` File

The `/etc/uucp/Systems` file contains the information needed by the `uucico` daemon to establish a communication link to a remote computer. It is the first file you need to edit to configure UUCP.

Each entry in the `Systems` file represents a remote computer with which your host communicates. A particular host can have more than one entry. The additional entries represent alternative communication paths that are tried in sequential order.

In addition, by default UUCP prevents any computer that does not appear in `/etc/uucp/Systems` from logging in to your host.

Using the `Sysfiles` file, you can define several files to be used as `Systems` files. See "UUCP `/etc/uucp/Sysfiles` File" on page 510 for a description of `Sysfiles`.

Each entry in the `Systems` file has the following format:

System-Name	*Time*	*Type*	*Speed*	*Phone*	*Chat-Script*

The following example shows the fields of the `Systems` file.

EXAMPLE 27–1 Fields in `/etc/uucp/Systems`

```
System-Name Time Type  Speed Phone   Chat-Script
Arabian     Any  ACUEC 38400 111222  Login: Puucp ssword:beledi
```

UUCP System-Name Field

This field contains the node name of the remote computer. On TCP/IP networks, this can be the machine's host name or a name created specifically for UUCP communications through the `/etc/uucp/Sysname` file. See "UUCP `/etc/uucp/Systems` File" on page 491. In Example 27–1, the System-Name field contains an entry for remote host `arabian`.

UUCP Time Field

This field specifies the day of week and time of day when the remote computer can be called. The format of the Time field is:

```
daytime[;retry]
```

The *day* portion can be a list containing some of the following entries:

TABLE 27–1 Day Field

`Su Mo Tu We Th Fr Sa`	For individual days.
`Wk`	For any weekday.
`Any`	For any day.
`Never`	Your host never initiates a call to the remote computer; the call must be initiated by the remote computer. Your host is then operating in *passive mode*.

Example 27–1 shows `Any` in the Time field, indicating that host `arabian` can be called at any time.

The *time* portion should be a range of times specified in 24-hour notation. (Example: `0800-1230` for 8:30 AM to 12:30 PM.) If no *time* portion is specified, any time of day is assumed to be allowed for the call.

A time range that spans 0000 is permitted. For example, `0800-0600` means all times are allowed other than times between 6 AM and 8 AM.

UUCP Retry Subfield

The *Retry* subfield enables you to specify the minimum time (in minutes) before a retry, following a failed attempt. The default wait is 60 minutes. The subfield separator is a semicolon (;). For example, `Any;9` is interpreted as call any time, but wait at least 9 minutes before retrying after a failure occurs.

If you do not specify a *retry* entry, an exponential back-off algorithm is used. What this means is that UUCP starts with a default wait time that grows larger as the number of failed attempts increases. For example, suppose the initial retry time is 5 minutes. If there is no response, the next retry is 10 minutes later. The next retry is 20 minutes later, and so on until the maximum retry time of 23 hours is reached. If *retry* is specified, that is always the retry time. Otherwise, the back-off algorithm is used.

UUCP Type Field

This field contains the device type that should be used to establish the communication link to the remote computer. The keyword used in this field is matched against the first field of `Devices` file entries.

EXAMPLE 27–2 Type Field and `/etc/uucp/Devices` File

```
File Name System-Name  Time  Type     Speed  Phone     Chap-Script

Systems   arabian      Any   ACUEC, g 38400  1112222   ogin: Puucp ssword:beledi
```

You can define the protocol used to contact the system by adding it on to the Type field. The previous example shows how to attach the protocol `g` to the device type `ACUEC`. (For information on protocols, see "UUCP Protocol Definitions in the `Devices` File" on page 503.)

UUCP Speed Field

This field (also known as the Class field) specifies the transfer speed of the device used in establishing the communication link. It can contain a letter and speed (for example, `C1200`, `D1200`) to differentiate between classes of dialers (refer to "UUCP Class Field" on page 500).

Some devices can be used at any speed, so the keyword `Any` can be used. This field must match the Class field in the associated `Devices` file entry:

EXAMPLE 27–3 Speed Field and `/etc/uucp/Devices` File

```
File Name System-Name Time  Type    Speed  Phone   Chap-Script

Systems   eagle       Any   ACU, g  D1200  NY3251  ogin: nuucp ssword: Oakgrass
```

If information is not required for this field, use a dash (`-`) as a place holder for the field.

UUCP Phone Field

This field allows you to specify the telephone number (token) of the remote computer for automatic dialers (port selectors). The telephone number consists of an optional alphabetic abbreviation and a numeric part. If an abbreviation is used, it must be one that is listed in the `Dialcodes` file:

EXAMPLE 27–4 Phone Field Correspondence

```
File Name System-Name  Time  Type  Speed  Phone     Chap-Script

Systems   nubian       Any   ACU   2400   NY5551212 ogin: Puucp ssword:Passuan
```

In the `System-Name` string, an equals sign (`=`) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash (`-`) in the string instructs the ACU to pause four seconds before dialing the next digit.

If your computer is connected to a port selector, you can access other computers connected to that selector. The `Systems` file entries for these remote machines should not have a telephone number in the Phone field. Instead, this field should contain the token to be passed on to the switch. In this way, the port selector knows the remote machine with which your host wants to communicate. (This is usually just the system name.) The associated `Devices` file entry should have a `\D` at the end of the entry to ensure that this field is not translated using the `Dialcodes` file.

UUCP Chat-Script Field

This field (also called the Login field) contains a string of characters called a *chat-script*. The chat-script contains the characters the local and remote machines must pass to each other in their initial conversation. Chat-scripts have the format:

expect send [expect send] ....

expect represents the string that the local host expects to get from the remote host to initiate conversation. *send* is the string the local host sends after it receives the *expect* string from the remote host. A chat-script can have more than one expect-send sequence.

A basic chat-script might contain:

- Login prompt that the local host expects to get from the remote machine
- Login name that the local host sends to the remote machine in order to log in
- Password prompt that the local host expects to get from the remote machine
- Password that the local host sends to the remote machine

The *expect* field can be made up of subfields of the form:

expect[-send-expect]...

where *-send* is sent if the prior *expect* is not successfully read, and *-expect* following the *send* is the next expected string.

For example, with strings `login--login`, the UUCP on the local host expects `login`. If UUCP gets `login` from the remote machine, it goes to the next field. If it

does not get `login`, it sends a carriage return, then looks for `login` again. If the local computer initially does not expect any characters, use the characters `""` (NULL string) in the *expect* field. All *send* fields are sent followed by a carriage return unless the *send* string is terminated with a `\c`.

Here is an example of a `Systems` file entry that uses an *expect-send* string:

```
System-Name  Time  Type  Speed  Phone     Chap-Script
sonora Any ACUEC 9600 2223333 "" \r \r ogin:-BREAK-ogin: Puucpx ssword: xyzzy
```

This example tells UUCP on the local host to send two carriage-returns and wait for `ogin:` (for `Login:`). If `ogin`: is not received, send a BREAK. When you do get `ogin:` send the login name `Puucpx`. When you get `ssword:` (for `Password:`), send the password `xyzzy`.

The following table lists some useful escape characters.

TABLE 27–2 Escape Characters Used in `Systems` File Chat-Script

Escape Character	Meaning
`\b`	Sends or expects a backspace character.
`\c`	If at the end of a string, suppresses the carriage return that is normally sent. Ignored otherwise.
`\d`	Delays 1–3 seconds before sending more characters.
`\E`	Starts echo checking. (From this point on, whenever a character is transmitted, it waits for the character to be received before doing anything else.)
`\e`	Echoes check-off.
`\H`	Ignores one hangup. Use this option for dialback modems.
`\K`	Sends a BREAK character.
`\M`	Turns on `CLOCAL` flag.
`\m`	Turns off `CLOCAL` flag.
`\n`	Sends or expects a newline character.
`\N`	Sends a NULL character (ASCII NUL).

TABLE 27–2 Escape Characters Used in `Systems` File Chat-Script *(continued)*

Escape Character	Meaning
`\p`	Pauses for approximately 1/4 to 1/2 second.
`\r`	Sends or expects a carriage return.
`\s`	Sends or expects a space character.
`\t`	Sends or expects a tab character.
`EOT`	Sends an EOT followed by newline twice.
`BREAK`	Sends a break character.
`\`*ddd*	Sends or expects the character represented by the octal digits (*ddd*).

Enabling Dialback Through the Chat-Script

Some companies set up dial-in servers to handle calls from remote computers. For example, your company might have a dial-in server with a dialback modem that employees can call from their home computers. After the dial-in server identifies the remote machine, it disconnects the link to the remote machine and then calls the remote machine back. The communications link is then reestablished.

You can facilitate dialback by using the `\H` option in the `Systems` file chat-script at the place where dialback should occur. Include the `\H` as part of an expect string at the place where the dial-in server is expected to hang up.

For example, suppose the chat-script that calls a dial-in server contains the following string:

```
INITIATED\Hogin:
```

The UUCP dialing facility on the local machine expects to get the characters `INITIATED` from the dial-in server. After the `INITIATED` characters have been matched, the dialing facility flushes any subsequent characters it receives until the dial-in server hangs up. The local dialing facility then waits until it receives the next part of the expect string, the characters `ogin:`, from the dial-in server. When it receives the `ogin:`, the dialing facility then continues through the chat-script.

You need not have a string of characters directly preceding or following the `\H`, as shown in the previous sample string.

UUCP Hardware Flow Control

You can also use the pseudo-send `STTY=value` string to set modem characteristics. For instance, `STTY=crtscts` enables hardware flow control. `STTY` accepts all `stty` modes. See the `stty`(1) and `termio`(7I) man pages for complete details.

The following example would enable hardware flow control in a `Systems` file entry:

```
System-Name  Time  Type  Speed  Phone     Chap-Script
unix Any ACU 2400 12015551212 "" \r login:-\r-login:-\r-login:
nuucp password: xxx "" \ STTY=crtscts
```

This pseudo-send string can also be used in entries in the `Dialers` file.

UUCP Setting Parity

In some cases, you have to reset the parity because the system that you are calling checks port parity and drops the line if it is wrong. The expect-send couplet `""` `P_ZERO` sets the high-order bit (parity bit) to 0. For example:

```
System-Name  Time  Type  Speed  Phone     Chap-Script
unix Any ACU 2400 12015551212 "" P_ZERO "" \r login:-\r-login:-\r-login:
nuucp password: xxx
```

In the same manner, `P_EVEN` sets parity to even (the default), `P_ODD` sets odd parity, and `P_ONE` sets the parity bit to 1.

The parity couplet can be inserted anywhere in the chat-script. It applies to all information in the chat-script following the `""` `P_ZERO`. It can also be used in entries in the `Dialers` file.

UUCP `/etc/uucp/Devices` File

The `/etc/uucp/Devices` file contains information for all the devices that can be used to establish a link to a remote computer. These devices include ACUs—which includes modern, high-speed modems—direct links, and network connections.

Here is an entry in `/etc/uucp/Devices` for a US Robotics V.32bis modem attached to port A and running at 38,400 bps.

```
Type   Line  Line2 Class Dialer-Token-Pairs
 ACUEC  cua/a -     38400 usrv32bis-ec
```

Each field is described in the next section.

UUCP Type Field

This field describes the type of link that the device establishes. It can contain one of the keywords described in the sections that follow.

`Direct` Keyword

The `Direct` keyword appears mainly in entries for `cu` connections. This keyword indicates that the link is a direct link to another computer or a port selector. Make a separate entry for each line that you want to reference through the `-l` option of `cu`.

`ACU` Keyword

The `ACU` keyword indicates that the link to a remote computer (whether through `cu`, UUCP, or PPP) is made through a modem. This modem can be connected either directly to your computer or indirectly through a port selector.

Port Selector

This is a variable that is replaced in the Type field by the name of a port selector. Port selectors are devices attached to a network that prompt for the name of a calling modem, then grant access. The file `/etc/uucp/Dialers` contains caller scripts only for the `micom` and `develcon` port selectors. You can add your own port selector entries to the `Dialers` file. (See “UUCP `/etc/uucp/Dialers` File” on page 504 for more information.)

Sys-Name

This variable is replaced by the name of a machine in the Type field, indicating that the link is a direct link to this particular computer. This naming scheme is used to associate the line in this `Devices` entry to an entry in `/etc/uucp/Systems` for the computer *Sys-Name*.

Type Field and /etc/uucp/Systems File

Example 27–5 shows a comparison between the fields in /etc/uucp/Devices and fields in /etc/uucp/Systems. The titles of each column apply only to fields in the Devices file.

The keyword used in the Type field of the Devices file is matched against the third field of the Systems file entries. In the Devices file, the Type field has the entry ACUEC, indicating an automatic call unit, in this case a V.32bis modem. This value is matched against the third field in the Systems file, which also contains the entry ACUEC. (See "UUCP /etc/uucp/Systems File" on page 491 for more information.)

EXAMPLE 27–5 Type Field and /etc/uucp/Systems File Equivalent

```
File Name Type   Line  Line2 Class Dialer-Token-Pairs

Devices   ACUEC  cua/a -     38400 usrv32bis-ec

System    nubian Any   ACUEC 38400 9998888 ‘‘‘‘ \d\d\r\n\c-ogin-\r\n\c-ogin.......
```

UUCP Line Field

This field contains the device name of the line (port) associated with the Devices entry. For instance, if the modem associated with a particular entry were attached to the /dev/cua/a device (serial port A), the name entered in this field would be cua/a. An optional modem control flag, M, can be used in the Line field to indicate that the device should be opened without waiting for a carrier. For example:

```
cua/a,M
```

UUCP Line2 Field

This field is a placeholder. Always use a dash (-) here. 801 type dialers, which are not supported in the Solaris environment, use the Line2 field. Non-801 dialers do not normally use this configuration, but still require a hyphen in this field.

UUCP Class Field

The Class field contains the speed of the device, if the keyword ACU or Direct is used in the Type field. However, it can contain a letter and a speed (for example, C1200, D1200) to differentiate between classes of dialers (Centrex or Dimension PBX).

This is necessary because many larger offices can have more than one type of telephone network: one network might be dedicated to serving only internal office communications while another handles the external communications. In such a case, it becomes necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications.

The keyword used in the Class field of the `Devices` file is matched against the Speed field of `Systems` file.

EXAMPLE 27–6 UUCP Class Field

```
File Name  Type   Line  Line2  Class  Dialer-Token-Pairs

Devices    ACU    cua/a -      D2400  hayes
```

Some devices can be used at any speed, so the keyword `Any` can be used in the Class field. If `Any` is used, the line matches any speed requested in the Speed field of the `Systems` file. If this field is `Any` and the `Systems` file Speed field is `Any`, the speed defaults to 2400 bps.

UUCP Dialer-Token-Pairs Field

The Dialer-Token-Pairs (DTP) field contains the name of a dialer and the token to pass it. The DTP field has this syntax:

dialer token [dialer token]

The *dialer* portion can be the name of a modem, a port monitor, or it can be `direct` or `uudirect` for a direct-link device. You can have any number of dialer-token pairs; if not present, it is taken from a related entry in the `Systems` file. The *token* portion can be supplied immediately following the dialer portion.

The last dialer token pair might not be present, depending on the associated dialer. In most cases, the last pair contains only a *dialer* portion. The *token* portion is retrieved from the Phone field of the associated `Systems` file entry.

A valid entry in the *dialer* portion can be defined in the `Dialers` file or can be one of several special dialer types. These special dialer types are compiled into the software and are therefore available without having entries in the `Dialers` file. The following table shows the special dialer types.

TABLE 27–3 Dialer-Token Pairs

`TCP`	TCP/IP network
`TLI`	Transport Level Interface Network (without STREAMS)
`TLIS`	Transport Level Interface Network (with STREAMS)

See "UUCP Protocol Definitions in the `Devices` File" on page 503 for more information.

Structure of the Dialer-Token-Pairs Field

The DTP field can be structured four different ways, depending on the device associated with the entry:

- Directly connected modem

  If a modem is connected directly to a port on your computer, the DTP field of the associated `Devices` file entry has only one pair. This pair would normally be the name of the modem. This name is used to match the particular `Devices` file entry with an entry in the `Dialers` file. Therefore, the Dialer field must match the first field of a `Dialers` file entry.

EXAMPLE 27–7 Dialers Field for Direct Connect Modem

```
Dialers   hayes =,-,  ""          \\dA\pTE1V1X1Q0S2=255S12=255\r\c
                                  \EATDT\T\r\c CONNECT
```

Notice that only the dialer portion (`hayes`) is present in the DTP field of the `Devices` file entry. This means that the *token* to be passed on to the dialer (in this case the phone number) is taken from the `Phone` field of a `Systems` file entry. (`\T` is implied, as described in Example 27–9.)

- Direct link – For a direct link to a particular computer, the DTP field of the associated entry would contain the keyword `direct`. This is true for both types of direct-link entries, `Direct` and *Sys-Name* (refer to "UUCP Type Field" on page 499).
- Computers on the same port selector – If a computer with which you want to communicate is on the same port selector switch as your computer, your computer must first access the switch. The switch then makes the connection to the other computer. This type of entry has only one pair. The *dialer* portion is used to match a `Dialers` file entry.

EXAMPLE 27–8 UUCP Dialers Field for Computers on Same Port Selector

```
Dialers    develcon ,""   ""           \pr\ps\c est:\007 \E\D\e \007
```

As shown, the *token* portion is left blank. This indicates that it is retrieved from the `Systems` file. The `Systems` file entry for this computer contains the token in the Phone field, which is normally reserved for the phone number of the computer. (Refer to "UUCP `/etc/uucp/Systems` File" on page 491.) This type of DTP contains an escape character (`\D`), which ensures that the contents of the Phone field not interpreted as a valid entry in the `Dialcodes` file.

- Modems connected to port selector – If a high-speed modem is connected to a port selector, your computer must first access the port selector switch. The switch makes the connection to the modem. This type of entry requires two dialer-token-pairs. The *dialer* portion of each pair (fifth and seventh fields of entry) is used to match entries in the `Dialers` file, as shown below.

EXAMPLE 27–9 UUCP Dialers Field for Modems Connected to Port Selector

```
Dialers    develcon ""    ""     \pr\ps\c  est:\007    \E\D\e      \007

Dialers    ventel   =&-%   t""   \r\p\r\c  $           <K\T%\r>\c  ONLINE!
```

In the first pair, `develcon` is the dialer and `vent` is the token that is passed to the Develcon switch to tell it which device (such as Ventel modem) to connect to your computer. This token is unique for each port selector, as each switch can be set up differently. After the Ventel modem has been connected, the second pair is accessed, where Ventel is the dialer and the token is retrieved from the `Systems` file.

Two escape characters can appear in a DTP field:

- `\T` – Indicates that the Phone (*token*) field should be translated using the `/etc/uucp/Dialcodes` file. This escape character is normally placed in the `/etc/uucp/Dialers` file for each caller script associated with a modem (Hayes, US Robotics, and so on). Therefore, the translation does not take place until the caller script is accessed.
- `\D` – Indicates that the Phone (*token*) field should not be translated using the `/etc/uucp/Dialcodes` file. If no escape character is specified at the end of a `Devices` entry, the `\D` is assumed (default). A `\D` is also used in the `/etc/uucp/Dialers` file with entries associated with network switches (`develcon` and `micom`).

UUCP Protocol Definitions in the `Devices` File

You can define the protocol to use with each device in `/etc/uucp/Devices`. This is usually unnecessary because you can use the default or define the protocol with the

particular system you are calling. (Refer to "UUCP `/etc/uucp/Systems` File" on page 491.) If you do specify the protocol, you must use the form:

Type,Protocol [parameters]

For example, you can use `TCP,te` to specify the TCP/IP protocol.

The following table shows the available protocols for the `Devices` file.

TABLE 27–4 Protocols Used in `/etc/uucp/Devices`

Protocol	Description
`t`	This protocol is commonly used for transmissions over TCP/IP and other reliable connections. It assumes error-free transmissions.
`g`	This is UUCP's native protocol. It is slow, reliable, and good for transmission over noisy telephone lines.
`e`	This protocol assumes transmission over error-free channels that are message oriented (as opposed to byte-stream oriented, like TCP/IP).
`f`	This protocol is used for transmission over X.25 connections. It relies on flow control of the data stream, and is meant for working over links that can (almost) be guaranteed to be error-free, specifically X.25/PAD links. A checksum is carried out over a whole file only. If a transport fails, the receiver can request retransmission(s).

Here is an example showing a protocol designation for a device entry:

```
TCP,te - - Any TCP -
```

This example indicates that, for device `TCP`, try to use the `t` protocol. If the other end refuses, use the `e` protocol.

Neither `e` nor `t` is appropriate for use over modems. Even if the modem assures error-free transmission, data can still be dropped between the modem and the CPU.

UUCP `/etc/uucp/Dialers` File

The `/etc/uucp/Dialers` file contains dialing instructions for many commonly used modems. You probably do not need to change or add entries to this file unless you plan to use a nonstandard modem or plan to customize your UUCP

environment. Nevertheless, you should understand what is in the file and how it relates to the `Systems` and `Devices` file.

The text specifies the initial conversation that must take place on a line before it can be made available for transferring data. This conversation, often referred to as a chat-script, is usually a sequence of ASCII strings that is transmitted and expected, and it is often used to dial a phone number.

As shown in the examples in "UUCP `/etc/uucp/Devices` File" on page 498, the fifth field in a `Devices` file entry is an index into the `Dialers` file or a special dialer type (`TCP`, `TLI`, or `TLIS`). The `uucico` daemon attempts to match the fifth field in the `Devices` file with the first field of each `Dialers` file entry. In addition, each odd-numbered `Devices` field, starting with the seventh position is used as an index into the `Dialers` file. If the match succeeds, the `Dialers` entry is interpreted to perform the dialer conversation.

Each entry in the `Dialers` file has the following format:

dialer	*substitutions*	*expect-send*

The following example shows the entry for a US Robotics V.32bis modem.

EXAMPLE 27–10 `/etc/uucp/Dialers` File Entry

```
Dialer        Substitution  Expaec-Send
usrv32bis-e   =,-,   ""     dA\pT&FE1V1X1Q0S2=255S12=255&A1&H1&M5&B2&W\r\c OK\r
                            \EATDT\T\r\c CONNECT\s14400/ARQ STTY=crtscts
```

The Dialer field matches the fifth and additional odd-numbered fields in the `Devices` file. The Substitutions field is a translate string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate `=` and `-` into whatever the dialer requires for "wait for dial tone" and "pause."

The remaining expect-send fields are character strings.

The following example shows some sample entries in the `Dialers` file, as distributed when you install UUCP as part of the Solaris installation program.

EXAMPLE 27–11 Excerpts From `/etc/uucp/Dialers`

```
penril =W-P "" \d > Q\c : \d- > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c OK

ventel =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!

vadic =K-K "" \005\p *-\005\p-*\005\p-* D\p BER? \E\T\e \r\c LINE
```

(continued)

```
develcon "" "" \pr\ps\c est:\007

\E\D\e \n\007 micom "" "" \s\c NAME? \D\r\c GO

hayes =,-, "" \dA\pTE1V1X1Q0S2=255S12=255\r\c OK\r \EATDT\T\r\c CONNECT

#   Telebit TrailBlazer
tb1200 =W-, "" \dA\pA\pA\pTE1V1X1Q0S2=255S12=255S50=2\r\c OK\r
\EATDT\T\r\c CONNECT\s1200
tb2400 =W-, "" \dA\pA\pA\pTE1V1X1Q0S2=255S12=255S50=3\r\c OK\r
\EATDT\T\r\c CONNECT\s2400
tbfast =W-, "" \dA\pA\pA\pTE1V1X1Q0S2=255S12=255S50=255\r\c OK\r
\EATDT\T\r\c CONNECT\sFAST

# USrobotics, Codes, and DSI modems

dsi-ec  =,-,    "" \dA\pTE1V1X5Q0S2=255S12=255*E1*F3*M1*S1\r\c OK\r \EATDT\T\r\c
CONNECT\sEC STTY=crtscts,crtsxoff

dsi-nec =,-,    "" \dA\pTE1V1X5Q0S2=255S12=255*E0*F3*M1*S1\r\c OK\r \EATDT\T\r\c CONNECT
STTY=crtscts,crtsxoff

usrv32bis-ec =,-,  "" \dA\pT&FE1V1X1Q0S2=255S12=255&A1&H1&M5&B2&W\r\c OK\r \EATDT\T\r\c
CONNECT\s14400/ARQ STTY=crtscts,crtsxoff

usrv32-nec =,-, "" \dA\pT&FE1V1X1Q0S2=255S12=255&A0&H1&M0&B0&W\r\c OK\r \EATDT\T\r\c
CONNECT STTY=crtscts,crtsxoff

codex-fast =,-, "" \dA\pT&C1&D2*MF0*AA1&R1&S1*DE15*FL3S2=255S7=40S10=40*TT5&W\r\c OK\r
\EATDT\T\r\c CONNECT\s38400 STTY=crtscts,crtsxoff

tb9600-ec =W-,  "" \dA\pA\pA\pTE1V1X1Q0S2=255S12=255S50=6\r\c OK\r
\EATDT\T\r\cCONNECT\s9600 STTY=crtscts,crtsxoff

tb9600-nec =W-, "" \dA\pA\pA\pTE1V1X1Q0S2=255S12=255S50=6S180=0\r\c OK\r \EATDT\T\r\c
CONNECT\s9600 STTY=crtscts,crtsxoff
```

The following table lists escape characters commonly used in the send strings in the `Dialers` file.

TABLE 27–5 Backslash Characters for `/etc/uucp/Dialers`

Character	Description
`\b`	Sends or expects a backspace character.
`\c`	No newline or carriage return.

TABLE 27–5 Backslash Characters for `/etc/uucp/Dialers` *(continued)*

Character	Description
`\d`	Delays (approximately 2 seconds).
`\D`	Phone number or token without `Dialcodes` translation.
`\e`	Disables echo checking.
`\E`	Enables echo checking (for slow devices).
`\K`	Insert a Break character
`\n`	Sends newline.
`\nnn`	Sends octal number. Additional escape characters that can be used are listed in the section "UUCP `/etc/uucp/Systems` File" on page 491.
`\N`	Sends or expects a NULL character (ASCII NUL)
`\p`	Pauses (approximately 12–14 seconds).
`\r`	Returns.
`\s`	Sends or expects a space character.
`\T`	Phone number or token with `Dialcodes` translation.

Here is a `penril` entry in the `Dialers` file:

```
penril =W-P "" \d > Q\c : \d- > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c OK
```

First, the substitution mechanism for the phone number argument is established, so that any `=` is replaced with a `W` (wait for dial tone) and any `-` with a `P` (pause).

The handshake given by the remainder of the line works as listed:

- `""` – Waits for nothing. (that is, proceed to the next step)
- `\d` – Delays 2 seconds, then send a carriage-return
- `>` – Waits for a `>`
- `Q\c` – Sends a `Q` without a carriage return
- `:` – Expects a `:`

- \d- – Delays 2 seconds, sends a - and a carriage-return
- > – Waits for a >
- s\p9\c – Sends an s, pauses, sends a 9 with no carriage return
-)-W\p\r\ds\p9\c-) – Waits for a). If it is not received, processes the string between the - characters as follows. Sends a W, pauses, sends a carriage return, delays, sends an s, pauses, sends a 9, without a carriage return, then waits for the).
- y\c – Sends a y with no carriage return
- : – Waits for a :
- \E\TP - Enables echo checking. (From this point on, whenever a character is transmitted, it waits for the character to be received before doing anything else.) Then, sends the phone number. The \T means take the phone number passed as an argument and applies the Dialcodes translation and the modem function translation specified by field 2 of this entry. Then sends a P and a carriage return.
- > – Waits for a >
- 9\c – Sends a 9 without a newline
- OK – Waits for the string OK

UUCP Hardware Flow Control

You can also use the pseudo-send STTY=value string to set modem characteristics. For instance, STTY=crtscts enables outbound hardware flow control; STTY=crtsxoff enables inbound hardware flow control; and STTY=crtscts,crtsxoff enables both outbound and inbound hardware flow control.

STTY accepts all the stty modes. See the stty(1) and termio(7I) man pages.

The following example would enable hardware flow control in a Dialers entry:

```
dsi =,--, "" \dA\pTE1V1X5Q0S2=255S12=255*E1*F3*M1*S1\r\c OK\r \EATDT\T\r\c
CONNECT\sEC STTY=crtscts
```

This pseudo-send string can also be used in entries in the Systems file.

UUCP Setting Parity

In some cases, you have to reset the parity because the system that you are calling checks port parity and drops the line if it is wrong. The expect-send couplet P_ZERO sets parity to zero:

```
foo =,-, "" P_ZERO "" \dA\pTE1V1X1Q0S2=255S12=255\r\c OK\r\EATDT\T\r\c CONNECT
```

In the same manner, `P_EVEN` sets it to even (the default); `P_ODD` sets it to odd; and `P_ONE` sets it to one. This pseudo-send string can also be used in entries in the `Systems` file.

Other Basic UUCP Configuration Files

The files in this section can be used in addition to the `Systems`, `Devices`, and `Dialers` file when doing basic UUCP configuration.

UUCP `/etc/uucp/Dialcodes` File

The `/etc/uucp/Dialcodes` file enables you to define dial-code abbreviations that can be used in the Phone field in the `/etc/uucp/Systems` file. You can use the `Dialcodes` files to provide additional information about a basic phone number that is used by several systems at the same site.

Each entry has the format:

abbreviation dial-sequence

where *abbreviation* represents the abbreviation used in the Phone field of the `Systems` file and *dial-sequence* represents the dial sequence passed to the dialer when that particular `Systems` file entry is accessed. The following table shows the correspondences between the two files.

TABLE 27–6 Correspondences Between `Dialcodes` and `Systems` Files

	Field Names					
Dialcodes	*Abbreviation*	Dial-Sequence				
Systems	System-Name	Time	Type	Speed	*Phone*	Chat-Script

The following table contains sample entries in a `Dialcodes` file.

TABLE 27–7 Entries in the Dialcodes File

Abbreviation	Dial-sequence
NY	1=212
jt	9+847

In the first row, NY is the abbreviation to appear in the Phone field of the Systems file. For example, the Systems file might have the entry:

```
NY5551212
```

When uucico reads NY in the Systems file, it searches the Dialcodes file for NY and obtains the dialing sequence 1=212. This is the dialing sequence needed for any phone call to New York City. It includes the number 1, an equal sign (=) meaning pause and wait for a secondary dial tone, and the area code 212. uucico sends this information to the dialer, then returns to the Systems file for the remainder of the phone number, 5551212.

The entry `jt 9=847-` would work with a Phone field in the Systems file such as jt7867. When uucico reads the entry containing jt7867 in the Systems file, it sends the sequence 9=847-7867 to the dialer, if the token in the dialer-token pair is \T.

UUCP /etc/uucp/Sysfiles File

The /etc/uucp/Sysfiles file lets you assign different files to be used by uucp and cu as Systems, Devices, and Dialers files. (For more information on cu, see the cu(1C) man page.) You might want to use Sysfiles for:

- Different Systems files, so that requests for login services can be made to different addresses than uucp services.
- Different Dialers files, so that you can assign different handshaking for cu and uucp.
- Multiple Systems, Dialers, and Devices files. The Systems file in particular can become large, making it more convenient to split it into several smaller files.

The format of the Sysfiles file is:

```
service=w systems=x:x dialers=y:y devices=z:z
```

w represents uucico, cu, or both separated by a colon. *x* represents one or more files to be used as the Systems file, with each file name separated by a colon and

read in the order presented. *y* represents one or more files to be used as the `Dialers` file. *z* is one or more files to be used as the `Devices` file.

Each file name is assumed to be relative to the `/etc/uucp` directory, unless a full path is given.

The following sample, `/etc/uucp/Sysfiles` defines a local `Systems` file (`Local_Systems`) in addition to the standard `/etc/uucp/Systems` file:

```
service=uucico:cu systems=Systems :Local_Systems
```

When this entry is in `/etc/uucp/Sysfiles`, both `uucico` and `cu` first check in the standard `/etc/uucp/Systems`. If the system they are trying to call doesn't have an entry in that file, or if the entries in the file fail, then they look in `/etc/uucp/Local_Systems`.

Given the previous entry, `cu` and `uucico` share the `Dialers` and `Devices` files.

When different `Systems` files are defined for `uucico` and `cu` services, your machine stores two different lists of `Systems`. You can print the `uucico` list using the `uuname` command or the `cu` list using the `uuname –C` command. Another example of the file, where the alternate files are consulted first and the default files are consulted in case of need is:

```
service=uucico systems=Systems.cico:Systems
  dialers=Dialers.cico:Dialers \
devices=Devices.cico:Devices
  service=cu systems=Systems.cu:Systems \
dialers=Dialers.cu:Dialers \
  devices=Devices.cu:Devices
```

UUCP `/etc/uucp/Sysname` File

Every machine that uses UUCP must have an identifying name, often referred to as the *node name*. This is the name that appears in the remote machine's `/etc/uucp/Systems` file, along with the chat-script and other identifying information. Normally, UUCP uses the same node name as is returned by the `uname -n` command, which is also used by TCP/IP.

You can specify a UUCP node name independent of the TCP/IP host name by creating the `/etc/uucp/Sysname` file. The file has a one-line entry containing the UUCP node name for your system.

UUCP /etc/uucp/Permissions File

The /etc/uucp/Permissions file specifies the permissions that remote computers have with respect to login, file access, and command execution. Some options restrict the remote computer's ability to request files and its ability to receive files queued by the local machine. Another option is available that specifies the commands that a remote machine can execute on the local computer.

UUCP Structuring Entries

Each entry is a logical line, with physical lines terminated by a backslash (\) to indicate continuation. Entries are made up of options delimited by blank space. Each option is a name-value pair in the following format:

name=value

Values can be colon-separated lists. No blank space is allowed within an option assignment.

Comment lines begin with a pound sign (#), and they occupy the entire line up to a newline character. Blank lines are ignored (even within multiple-line entries).

The types of Permissions file entries are:

- LOGNAME – Specifies the permissions that take effect when a remote computer logs in to (calls) your computer.

Note - When a remote machine calls you, its identity is questionable unless it has a unique login and verifiable password.

- MACHINE – Specifies permissions that take effect when your computer logs in to (calls) a remote computer.

LOGNAME entries contain a LOGNAME option and MACHINE entries contain a MACHINE option. One entry can contain both options.

UUCP Considerations

When using the Permissions file to restrict the level of access granted to remote computers, you should consider the following:

- All login IDs used by remote computers to log in for UUCP communications must appear in one and only one LOGNAME entry.
- Any site that is called having a name that does not appear in a MACHINE entry, has the following default permissions or restrictions:

- Local send and receive requests are executed.
- The remote computer can send files to your computer's `/var/spool/uucppublic` directory.
- The commands sent by the remote computer for execution on your computer must be one of the default commands, usually `rmail`.

UUCP `REQUEST` Option

When a remote computer calls your computer and requests to receive a file, this request can be granted or denied. The `REQUEST` option specifies whether the remote computer can request to set up file transfers from your computer. The string `REQUEST=yes` specifies that the remote computer can request to transfer files from your computer. The string `REQUEST=no` specifies that the remote computer cannot request to receive files from your computer. This is the default value; it is used if the `REQUEST` option is not specified. The `REQUEST` option can appear in either a `LOGNAME` (remote computer calls you) entry or a `MACHINE` (you call remote computer) entry.

UUCP `SENDFILES` Option

When a remote computer calls your computer and completes its work, it can attempt to take work your computer has queued for it. The `SENDFILES` option specifies whether your computer can send the work queued for the remote computer.

The string `SENDFILES=yes` specifies that your computer can send the work that is queued for the remote computer as long as itis logged in as one of the names in the `LOGNAME` option. This string is *mandatory* if you have entered `Never` in the Time field of `/etc/uucp/Systems`. This designation sets up your local machine in passive mode; it is not allowed to initiate a call to this particular remote computer. (See "UUCP `/etc/uucp/Systems` File" on page 491 for more information.)

The string `SENDFILES=call` specifies that files queued in your computer are sent only when your computer calls the remote computer. The `call` value is the default for the `SENDFILES` option. This option is only significant in `LOGNAME` entries because `MACHINE` entries apply when calls are made out to remote computers. If the option is used with a `MACHINE` entry, it is ignored.

UUCP `MYNAME` Option

This option enables you to designate a unique UUCP node name for your computer in addition to its TCP/IP host name, as returned by the `hostname` command. For instance, if you have unknowingly given your host the same name as that of some other system, you might want to set the `MYNAME` option of the `Permissions` file. Or

if you want your organization to be known as `widget` but all your modems are connected to a machine with the host name `gadget`, you can have an entry in `gadget`'s `Permissions` file that says:

```
service=uucico systems=Systems.cico:Systems
  dialers=Dialers.cico:Dialers \
  devices=Devices.cico:Devices
service=cu systems=Systems.cu:Systems \
  dialers=Dialers.cu:Dialers \
  devices=Devices.cu:Devices
```

Now the system `world` can log in to the machine `gadget` as if it were logging in to `widget`. In order for machine `world` to know you also by the aliased name `widget` when you call it, you can have an entry that says:

```
MACHINE=world MYNAME=widget
```

You can also use the `MYNAME` option for testing purposes, as it allows your machine to call itself. However, because this option could be used to mask the real identity of a machine, you should use the `VALIDATE` option, as described in "UUCP `VALIDATE` Option" on page 517.

UUCP `READ` and `WRITE` Options

These options specify the various parts of the file system that `uucico` can read from or write to. You can designate `READ` and `WRITE` options with either `MACHINE` or `LOGNAME` entries.

The default for both the `READ` and `WRITE` options is the `uucppublic` directory, as shown in the following strings:

```
READ=/var/spool/uucppublic WRITE=/var/spool/uucppublic
```

The strings `READ=/` and `WRITE=/` specify permission to access any file that can be accessed by a local user with Other permissions.

The value of these entries is a colon-separated list of path names. The `READ` option is for requesting files, and the `WRITE` option is for depositing files. One of the values must be the prefix of any full path name of a file coming in or going out. To grant permission to deposit files in `/usr/news` as well as the public directory, use the following values with the `WRITE` option:

```
WRITE=/var/spool/uucppublic:/usr/news
```

If the `READ` and `WRITE` options are used, all path names must be specified because the path names are not added to the default list. For instance, if the `/usr/news` path

name were the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

Be careful which directories you make accessible for reading and writing by remote systems. For example, the /etc directory contains many critical system files; remote users should not have permission to deposit files in this directory.

UUCP NOREAD and NOWRITE Options

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The entry:

```
READ=/ NOREAD=/etc WRITE=/var/spool/uucppublic
```

permits reading any file except those in the /etc directory (and its subdirectories—remember, these are prefixes). It permits writing only to the default /var/spool/uucppublic directory. NOWRITE works in the same manner as the NOREAD option. You can use the NOREAD and NOWRITE options in both LOGNAME and MACHINE entries.

UUCP CALLBACK Option

You can use the CALLBACK option in LOGNAME entries to specify that no transaction takes place until the calling system is called back. The two reasons to set up CALLBACKare: For security purposes; if you call back a machine, you can be sure it is the right machine. For accounting purposes; if you are doing long data transmissions, you can choose the machine that is billed for the longer call.

The string CALLBACK=yes specifies that your computer must call the remote computer back before any file transfers can take place.

The default for the CALLBACK option is CALLBACK=no. If you set CALLBACK to yes, the permissions that affect the rest of the conversation must be specified in the MACHINE entry corresponding to the caller. Do not specify these permissions in the LOGNAME, or in the LOGNAME entry that the remote machine might have set for your host.

Note - If two sites have the CALLBACK option set for each other, a conversation never gets started.

UUCP COMMANDS Option

Caution - The COMMANDS option can compromise the security of your system. Use it with extreme care.

You can use the COMMANDS option in MACHINE entries to specify the commands that a remote computer can execute on your machine. The uux program generates remote execution requests and queues them to be transferred to the remote computer. Files and commands are sent to the target computer for remote execution. This is an exception to the rule that MACHINE entries apply only when your system calls out.

Note that COMMANDS is not used in a LOGNAME entry; COMMANDS in MACHINE entries defines command permissions, whether you call the remote system or it calls you.

The string COMMANDS=rmail specifies the default commands that a remote computer can execute on your computer. If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the entry:

```
MACHINE=owl:raven:hawk:dove COMMANDS=rmail:rnews:lp
```

overrides the COMMAND default so that the computers named owl, raven, hawk, and dove can now execute rmail, rnews, and lp on your computer.

In addition to the names as just specified, there can be full path names of commands. For example:

```
COMMANDS=rmail:/usr/local/rnews:/usr/local/lp
```

specifies that command rmail uses the default search path. The default search path for UUCP is /bin and /usr/bin. When the remote computer specifies rnews or /usr/local/rnews for the command to be executed, /usr/local/rnews is executed regardless of the default path. Likewise, /usr/local/lp is the lp command that is executed.

Including the ALL value in the list means that any command from the remote computers specified in the entry will be executed. If you use this value, you give the remote computers full access to your machine.

Caution - This allows far more access than normal users have. You should use this value only when both machines are at the same site, are closely connected, and the users are trusted.

The string:

```
COMMANDS=/usr/local/rnews:ALL:/usr/local/lp
```

illustrates two points:

- The ALL value can appear anywhere in the string.

- The path names specified for `rnews` and `lp` are used (instead of the default) if the requested command does not contain the full path names for `rnews` or `lp`.

You should use the `VALIDATE` option whenever you specify potentially dangerous commands like `cat` and `uucp` with the `COMMANDS` option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (`uuxqt`).

UUCP `VALIDATE` Option

Use the `VALIDATE` option in conjunction with the `COMMANDS` option whenever you specify commands that are potentially dangerous to your machine's security. (`VALIDATE` is merely an added level of security on top of the `COMMANDS` option, though it is a more secure way to open command access than ALL.)

`VALIDATE` provides a certain degree of verification of the caller's identity by cross-checking the host name of a calling machine against the login name it uses. The string:

```
LOGNAME=Uwidget VALIDATE=widget:gadget
```

ensures that if any machine other than `widget` or `gadget` tries to log in as `Uwidget`, the connection is refused. The `VALIDATE` option requires privileged computers to have a unique login and password for UUCP transactions. An important aspect of this validation is that the login and password associated with this entry are protected. If an outsider gets that information, that particular `VALIDATE` option can no longer be considered secure.

Carefully consider which remote computers you will grant privileged logins and passwords for UUCP transactions. Giving a remote computer a special login and password with file access and remote execution capability is like giving anyone on that computer a normal login and password on your computer. Therefore, if you cannot trust someone on the remote computer, do not provide that computer with a privileged login and password.

The `LOGNAME` entry:

```
LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
```

specifies that if one of the remote computers that claims to be `eagle`, `owl`, or `hawk` logs in on your computer, it must have used the login `uucpfriend`. If an outsider gets the `uucpfriend` login and password, masquerading is easy.

But what does this have to do with the `COMMANDS` option, which appears only in `MACHINE` entries? It links the `MACHINE` entry (and `COMMANDS` option) with a `LOGNAME` entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote computer is logged in. In fact, it is an asynchronous process that does not know which computer sent the execution

request. Therefore, the real question is, how does your computer know where the execution files came from?

Each remote computer has its own spool directory on your local machine. These spool directories have write permission given only to the UUCP programs. The execution files from the remote computer are put in its spool directory after being transferred to your computer. When the `uuxqt` daemon runs, it can use the spool directory name to find the `MACHINE` entry in the `Permissions` file and get the `COMMANDS` list. Or, if the computer name does not appear in the `Permissions` file, the default list is used.

This example shows the relationship between the `MACHINE` and `LOGNAME` entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/local/rnews \
READ=/ WRITE=/
LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/ WRITE=/
```

The value in the `COMMANDS` option means that remote users can execute `rmail` and `/usr/local/rnews`.

In the first entry, you must assume that when you want to call one of the computers listed, you are really calling either `eagle`, `owl`, or `hawk`. Therefore, any files put into one of the `eagle`, `owl`, or `hawk` spool directories is put there by one of those computers. If a remote computer logs in and says that it is one of these three computers, its execution files are also put in the privileged spool directory. You therefore have to validate that the computer has the privileged login `uucpz`.

UUCP `MACHINE` Entry for `OTHER`

You might want to specify different option values for remote machines that are not mentioned in specific `MACHINE` entries. The need might arise when many computers are calling your host, and the command set changes from time to time. The name `OTHER` for the computer name is used for this entry as shown in this example:

```
MACHINE=OTHER \
COMMANDS=rmail:rnews:/usr/local/Photo:/usr/local/xp
```

All other options available for the `MACHINE` entry can also be set for the computers that are not mentioned in other `MACHINE` entries.

Combining MACHINE and LOGNAME Entries for UUCP

You can combine MACHINE and LOGNAME entries into a single entry where the common options are the same. For example, the two entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
READ=/ WRITE=/
```

and:

```
LOGNAME=uupz REQUEST=yes SENDFILES=yes \
READ=/ WRITE=/
```

share the same REQUEST, READ, and WRITE options. You can merge them, as shown:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
logname=uucpz SENDFILES-yes \
READ=/ WRITE=/
```

Combining MACHINE and LOGNAME entries makes the Permissions file more manageable and efficient.

UUCP Forwarding

When sending files through a series of machines, the intermediary machines must have the command uucp among their COMMANDS options. If you type the command:

```
% uucp sample.txt oak\!willow\!pine\!/usr/spool/uucppublic
```

the forwarding operation works only if machine willow permits machine oak to execute the program uucp, and if machine oak permits your machine to do the same. The machine pine, being the last machine designated, does not have to permit the command uucp since it is not doing any forwarding operations. Machines are not normally set up this way.

UUCP /etc/uucp/Poll File

The /etc/uucp/Poll file contains information for polling remote computers. Each entry in the Poll file contains the name of a remote computer to call, followed by a tab character or a space, and finally the hours the computer should be called. The format of entries in the Poll file are:

sys-name hour ...

For example, the entry

```
eagle 0 4 8 12 16 20
```

provides polling of computer eagle every four hours.

The uudemon.poll script processes the Poll file but does not actually perform the poll. It merely sets up a polling work file (always named C.*file*) in the spool directory. The uudemon.poll script starts the scheduler, and the scheduler examines all work files in the spool directory.

UUCP /etc/uucp/Config File

The /etc/uucp/Config file enables you to override certain parameters manually. Each entry in the Config file has this format:

parameter=value

See the Config file provided with your system for a complete list of configurable parameter names.

The following Config entry sets the default protocol ordering to Gge and changes the G protocol defaults to 7 windows and 512-byte packets.

```
Protocol=G(7,512)ge
```

UUCP/etc/uucp/Grades File

The /etc/uucp/Grades file contains the definitions for the job grades that can be used to queue jobs to a remote computer. It also contains the permissions for each job grade. Each entry in this file represents a definition of an administrator-defined job grade that lets users queue jobs.

Each entry in the `Grades` file has the following format:

User-job-grade System-job-grade Job-size Permit-type ID-list

Each entry contains fields that are separated by blank space. The last field in the entry is made up of subfields also separated by spaces. If an entry takes up more than one physical line, you can use a backslash to continue the entry onto the following line. Comment lines begin with a pound sign (#) and occupy the entire line. Blank lines are always ignored.

UUCP User-job-grade Field

This field contains an administrative-defined user job grade name of up to 64 characters.

UUCP System-job-grade Field

This field contains a one-character job grade to which *User-job-grade* is mapped. The valid list of characters is A–Z, a–z, with A having the highest priority and z the lowest.

Relationship Between User and System Job Grades

The user job grade can be bound to more than one system job grade. It is important to note that the `Grades` file is searched sequentially for occurrences of a user job grade. Therefore, any multiple occurrences of a system job grade should be listed according to the restriction on the maximum job size.

While there is no maximum number for the user job grades, the maximum number of system job grades allowed is 52. The reason is that more than one *User-job-grade* can be mapped to a *System-job-grade*, but each *User-job-grade* must be on a separate line in the file. Here is an example:

```
mail N Any User Any netnews N Any User Any
```

Given this configuration in a `Grades` file, these two *User-job-grade* will share the same *System-job-grade*. Because the permissions for a *Job-grade* are associated with a *User-job-grade* and not a *System-job-grade*, two *User-job-grade*s can share the same *System-job-grade*s and have two different sets of permissions.

Default Grade

You can define the binding of a default *User-job-grade* to a system job grade. You must use the keyword default as user job grade in the *User-job-grade* field of the

`Grades` file and the system job grade that it is bound to. The Restrictions and ID fields should be defined as `Any` so that any user and any size job can be queued to this grade. Here is an example:

```
default a Any User Any
```

If you do not define the default user job grade, the built-in default grade `Z` is used. Because the restriction field default is `Any`, multiple occurrences of the default grade are not checked.

UUCP Job-size Field

This field specifies the maximum job size that can be entered in the queue. *Job-size* is measured in bytes and can be a list of the options listed shown in the following table:

TABLE 27–8 Job-size Field

nnnn	Integer specifying the maximum job size for this job grade
n`K`	Decimal number representing the number of kilobytes (`K` is an abbreviation for kilobyte)
n`M`	Decimal number representing the number of megabytes (`M` is an abbreviation for megabyte)
`Any`	Keyword specifying that there is no maximum job size

Here are some examples:

- `5000` represents 5000 bytes
- `10K` represents 10 Kbytes
- `2M` represents 2 Mbytes

UUCP Permit-type Field

This field contains a keyword that denotes how to interpret the ID list. The following table lists the keywords and their meanings.

TABLE 27–9 Permit-type Field

Keyword	ID List Contents
`User`	Login names of users permitted to use this job grade
`Non-user`	Login names of users not permitted to use this job grade
`Group`	Group names whose members are permitted to use this group
`Non-group`	Group names whose members are not permitted to use this job grade

UUCP ID-list Field

This field contains a list of login names or group names that are to be permitted or denied queuing to this job grade. The list of names are separated by a blank space and terminated by a newline character. The keyword `Any` is used to denote that anyone is permitted to queue to this job grade.

Other UUCP Configuration Files

This section describes three less-frequently modified files that impact the use of UUCP facilities.

UUCP `/etc/uucp/Devconfig` File

The `/etc/uucp/Devconfig` file enables you to configure devices by service—`uucp` or `cu`. `Devconfig` entries define the STREAMS modules that are used for a particular device. They have the format:

`service=`*x* `device=`*y* `push=`*z*[:*z*...]

x can be `cu`, `uucico`, or both separated by a colon. *y* is the name of a network and must match an entry in the `Devices` file. *z* is replaced by the names of STREAMS modules in the order that they are to be pushed onto the Stream. Different modules and devices can be defined for `cu` and `uucp` services.

The following entries are for a STARLAN network and would most commonly be used in the file:

```
service=cu       device=STARLAN     push=ntty:tirdwr
service=uucico   device=STARLAN     push=ntty:tirdwr
```

This example pushes `ntty`, then `tirdwr`.

UUCP `/etc/uucp/Limits` File

The `/etc/uucp/Limits` file controls the maximum number of simultaneous `uucico`s, `uuxqt`s, and `uusched`s that are running in the `uucp` networking. In most cases, the default values are fine and no changes are needed. If you want to change them, however, use any text editor.

The format of the `Limits` file is:

`service=`*x* `max=`*y*:

x can be `uucico`, `uuxqt` or `uusched`, and *y* is the limit permitted for that service. The fields can be in any order and in lowercase.

The following entries should most commonly be used in the `Limits` file:

```
service=uucico max=5
service=uuxqt max=5
service=uusched max=2
```

The example allows five `uucico`s, five `uuxqt`s, and two `uusched`s running on your machine.

UUCP `remote.unknown` File

The other file that affects the use of communication facilities is the `remote.unknown` file. This file is a binary program that executes when a machine not found in any of the `Systems` files starts a conversation. It logs the conversation attempt and drops the connection.

Caution - If you change the permissions of the `remote.unknown` file so it cannot execute, your system accepts connections from any system.

This program executes when a machine that is not in any of the `Systems` starts a conversation. It logs the conversation attempt but fails to make a connection. If you change the permissions of this file so it cannot execute (`chmod 000 remote.unknown`), your system accepts any conversation requests. This is not a trivial change, and you should have good reasons for doing it.

UUCP Administrative Files

The UUCP administrative files are described next. These files are created in spool directories to lock devices, hold temporary data, or keep information about remote transfers or executions.

- *Temporary data files* (`TM`) – These data files are created by UUCP processes under the spool directory `/var/spool/uucp/`*x* when a file is received from another computer. The directory *x* has the same name as the remote computer that is sending the file. The names of the temporary data files have the format:

  `TM.`*pid.ddd*

  where *pid* is a process ID and *ddd* is a sequential three-digit number starting at 0.

  When the entire file is received, the `TM.`*pid.ddd* file is moved to the path name specified in the `C.`*sysnxxxx* file (discussed subsequently) that caused the transmission. If processing is abnormally terminated, the `TM.`*pid.ddd* file can remain in the *x* directory. These files should be automatically removed by `uucleanup`.

- *Lock files* (`LCK`) – Lock files are created in the `/var/spool/locks` directory for each device in use. Lock files prevent duplicate conversations and multiple attempts to use the same calling device. The following table shows the different types of UUCP lock files.

TABLE 27–10 UUCP Lock Files

File Name	Description
`LCK..`*sys*	*sys* represents the name of the computer using the file
`LCK.`*dev*	*dev* represents the name of a device using the file
`LCK.`*LOG*	*LOG* represents a locked UUCP log file

These files can remain in the spool directory if the communications link is unexpectedly dropped (usually on computer crashes). The lock file is ignored (removed) after the parent process is no longer active. The lock file contains the process ID of the process that created the lock.

- *Work file* (`C.`) – Work files are created in a spool directory when work (file transfers or remote command executions) has been queued for a remote computer. The names of work files have the format:

`C.`*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the ASCII character representing the grade (priority) of the work, and *xxxx* is the four-digit job sequence number assigned by UUCP. Work files contain the following information:

- Full path name of the file to be sent or requested.
- Full path name of the destination or user or file name.
- User login name.
- List of options.
- Name of associated data file in the spool directory; if the `uucp –C` or `uuto –p` option was specified, a dummy name (`D.0`) is used
- Mode bits of the source file.
- Remote user's login name to be notified on completion of the transfer.

- *Data file* (`D.`) – Data files are created when you specify on the command line to copy the source file to the spool directory. The names of data files have the following format:

  `D.`*systmxxxxyyy* – Where *systm* is the first five characters in the name of the remote computer, *xxxx* is a four-digit job sequence number assigned by `uucp`. The four–digit job sequence number can be followed by a subsequence number, *yyy* that is used when there are several `D.` files created for a work (`C.`) file.

- `X.` *(execute file)* – Execute files are created in the spool directory prior to remote command executions. The names of execute files have the following format:

  `X.`*sysnxxxx*

  sys is the name of the remote computer. *n* is the character representing the grade (priority) of the work. *xxxx* is a four-digit sequence number assigned by UUCP. Execute files contain the following information:

  - Requester's login and computer name
  - Names of files required for execution
  - Input to be used as the standard input to the command string
  - Computer and file name to receive standard output from the command execution
  - Command string
  - Option lines for return status requests

UUCP Error Messages

This section lists the error messages associated with UUCP.

UUCP ASSERT Error Messages

The following table lists ASSERT error messages.

TABLE 27–11 ASSERT Error Messages

Error Message	Description/Action
CAN'T OPEN	An `open( )` or `fopen( )` failed.
CAN'T WRITE	A `write( )`, `fwrite( )`, `fprint( )`, or similar command, failed.
CAN'T READ	A `read( )`, `fgets( )`, or similar command failed.
CAN'T CREATE	A `creat( )` call failed.
CAN'T ALLOCATE	A dynamic allocation failed.
CAN'T LOCK	An attempt to make a `LCK` (lock) file failed. In some cases, this is a fatal error.
CAN'T STAT	A `stat( )` call failed.
CAN'T CHMOD	A `chmod( )` call failed.
CAN'T LINK	A `link( )` call failed.
CAN'T CHDIR	A `chdir( )` call failed.
CAN'T UNLINK	An `unlink( )` call failed.
WRONG ROLE	This is an internal logic problem.
CAN'T MOVE TO CORRUPTDIR	An attempt to move some bad `C.` or `X.` files to the `/var/spool/uucp/.Corrupt` directory failed. The directory is probably missing or has wrong modes or owner.
CAN'T CLOSE	A `close( )` or `fclose( )` call failed.
FILE EXISTS	The creation of a `C.` or `D.` file is attempted, but the file exists. This occurs when a problem arises with the sequence file access. Usually indicates a software error.
NO uucp SERVICE NUMBER	A TCP/IP call is attempted, but no entry is in `/etc/services` for UUCP.
BAD UID	The user ID is not in the password database. Check name service configuration..
BAD LOGIN_UID	Same as previous.

TABLE 27–11 ASSERT Error Messages *(continued)*

Error Message	Description/Action
BAD LINE	A bad line is in the `Devices` file; there are not enough arguments on one or more lines.
SYSLST OVERFLOW	An internal table in `gename.c` overflowed. A single job attempted to talk to more than 30 systems.
TOO MANY SAVED C FILES	Same as previous.
RETURN FROM `fixline ioctl`	An `ioctl(2)`, which should never fail, failed. There is a system driver problem.
BAD SPEED	A bad line speed appears in the `Devices` or `Systems` file (Class or Speed field).
BAD OPTION	A bad line or option is in the `Permissions` file. It must be fixed immediately.
PKCGET READ	The remote machine probably hung up. No action need be taken.
PKXSTART	The remote machine aborted in a nonrecoverable way. This can usually be ignored.
TOO MANY LOCKS	An internal problem has occurred. Contact your system vendor.
XMV ERROR	A problem with some file or directory has occurred. It is likely the spool directory, as the modes of the destinations were supposed to be checked before this process was attempted.
CAN'T FORK	An attempt to make a `fork` and `exec` failed. The current job should not be lost but will be attempted later (`uuxqt`). No action is needed.

UUCP STATUS Error Messages

The following table is a list of the most common STATUS error messages.

TABLE 27–12 UUCP STATUS Messages

Error Message	Description/Action
OK	Status is okay.
NO DEVICES AVAILABLE	Currently no device is available for the call. Check whether a valid device is in the `Devices` file for the particular system. Check the `Systems` file for the device to be used to call the system.
WRONG TIME TO CALL	A call was placed to the system at a time other than what is specified in the `Systems` file.
TALKING	Self-explanatory.
LOGIN FAILED	The login for the given machine failed. It could be a wrong login or password, wrong number, a slow machine, or failure in getting through the Dialer-Token-Pairs script.
CONVERSATION FAILED	The conversation failed after successful startup. This usually means that one side went down, the program aborted, or the line (link) was dropped.
DIAL FAILED	The remote machine never answered. It could be a bad dialer or the wrong phone number.
BAD LOGIN/MACHINE COMBINATION	The machine called with a login/machine name that does not agree with the `Permissions` file. This could be an attempt to masquerade.
DEVICE LOCKED	The calling device to be used is currently locked and in use by another process.
ASSERT ERROR	An `ASSERT` error occurred. Check the `/var/uucp/.Admin/errors` file for the error message and refer to the section XREF.
SYSTEM NOT IN Systems FILE	The system is not in the `Systems` file.
CAN'T ACCESS DEVICE	The device tried does not exist or the modes are wrong. Check the appropriate entries in the `Systems` and `Devices` files.
DEVICE FAILED	The device could not be opened.
WRONG MACHINE NAME	The called machine is reporting a different name than expected.
CALLBACK REQUIRED	The called machine requires that it call your machine.

TABLE 27–12 UUCP STATUS Messages *(continued)*

Error Message	Description/Action
REMOTE HAS A LCK FILE FOR ME	The remote machine has a LCK file for your machine. It could be trying to call your machine. If it has an older version of UUCP, the process that was talking to your machine might have failed, leaving the LCK file. If it has the new version of UUCP and is not communicating with your machine, the process that has a LCK file is hung.
REMOTE DOES NOT KNOW ME	The remote machine does not have the node name of your machine in its Systems file.
REMOTE REJECT AFTER LOGIN	The login used by your machine to log in does not agree with what the remote machine was expecting.
REMOTE REJECT, UNKNOWN MESSAGE	The remote machine rejected the communication with your machine for an unknown reason. The remote machine might not be running a standard version of UUCP.
STARTUP FAILED	Login succeeded, but initial handshake failed.
CALLER SCRIPT FAILED	This is usually the same as DIAL FAILED. However, if it occurs often, suspect the caller script in the Dialers file. Use Uutry to check.

UUCP Numerical Error Messages

The following table lists the exit code numbers of error status messages produced by the /usr/include/sysexits.h file. Not all are currently used by uucp.

TABLE 27–13 UUCP Error Messages by Number

Message Number	Description	Meaning
64	Base Value for Error Messages	Error messages begin at this value.
64	Command–Line Usage Error	The command was used incorrectly, for example, with the wrong number of arguments, a bad flag, or a bad syntax.
65	Data Format Error	The input data was incorrect in some way. This should only be used for user's data and not system files.

TABLE 27–13 UUCP Error Messages by Number *(continued)*

Message Number	Description	Meaning
66	Cannot Open Input	An input file (not a system file) did not exist, or was not readable. This could also include errors like “No message” to a mailer.
67	Address Unknown	The user specified did not exist. This might be used for mail addresses or remote logins.
68	Host Name Unknown	The host did not exist. This is used in mail addresses or network requests.
69	Service Unavailable	A service is unavailable. This can occur if a support program or file does not exist. This message also can be a catchall message when something doesn’t work and you don’t know why.
70	Internal Software Error	An internal software error has been detected. This should be limited to non-operating system related errors if possible.
71	System Error	An operating system error has been detected. This is intended to be used for conditions like “cannot fork”, “cannot create pipe.” For instance, it includes `getuid` returning a user that does not exist in the `passwd` file.
72	Critical OS File Missing	Some system file like `/etc/passwd` or `/var/admin/utmpx` does not exist, cannot be opened, or has some error such as syntax error.
73	Can’t Create Output File	A user–specified output file cannot be created.
74	Input/Output Error	An error occurred while doing I/O on some file.
75	Temporary Failure. User is invited to retry	Temporary failure, indicating something that is not really an error. In `sendmail`, this means that a mailer, for example, could not create a connection, and the request should be reattempted later.
76	Remote Error in Protocol	The remote system returned something that was “not possible” during a protocol exchange.
77	Permission Denied	You do not have sufficient permission to perform the operation. This is not intended for file system problems, which should use `NOINPUT` or `CANTCREAT`, but rather for higher level permissions. For example, `kre` uses this to restrict students who can send mail to.
78	Configuration Error	The system detected an error in the configuration.

TABLE 27–13 UUCP Error Messages by Number *(continued)*

Message Number	Description	Meaning
79	Entry Not Found	Entry not found.
79	Maximum Listed Value	Highest value for error messages.

CHAPTER **28**

Accessing Remote File Systems Topics

Chapter 29	Provides overview information for the NFS service
Chapter 30	Provides step-by-step instructions for setting up and troubleshooting the NFS service
Chapter 31	Provides background information on the NFS service

CHAPTER 29

Solaris NFS Environment

This chapter provides an overview of the NFS environment. It includes a short introduction to networking, a description of the NFS service, and a discussion of the concepts necessary to understand the NFS environment.

NFS Servers and Clients

The terms *client* and *server* are used to describe the roles that a computer plays when sharing file systems. If a file system resides on a computer's disk and that computer makes the file system available to other computers on the network, that computer acts as a server. The computers that are accessing that file system are said to be clients. The NFS service enables any given computer to access any other computer's file systems and, at the same time, to provide access to its own file systems. A computer can play the role of client, server, or both at any given time on a network.

Clients access files on the server by mounting the server's shared file systems. When a client mounts a remote file system, it does not make a copy of the file system; rather, the mounting process uses a series of remote procedure calls that enable the client to access the file system transparently on the server's disk. The mount looks like a local mount and users type commands as if the file systems were local.

After a file system has been shared on a server through an NFS operation, it can be accessed from a client. You can mount an NFS file system automatically with autofs.

NFS File Systems

The objects that can be shared with the NFS service include any whole or partial directory tree or a file hierarchy—including a single file. A computer cannot share a file hierarchy that overlaps one that is already shared. Peripheral devices such as modems and printers cannot be shared.

In most UNIX® system environments, a file hierarchy that can be shared corresponds to a file system or to a portion of a file system; however, NFS support works across operating systems, and the concept of a file system might be meaningless in other, non-UNIX environments. Therefore, the term *file system* used throughout this guide refers to a file or file hierarchy that can be shared and mounted over the NFS environment.

About the NFS Environment

The NFS service enables computers of different architectures running different operating systems to share file systems across a network. NFS support has been implemented on many platforms ranging from the MS-DOS to the VMS operating systems.

The NFS environment can be implemented on different operating systems because it defines an abstract model of a file system, rather than an architectural specification. Each operating system applies the NFS model to its file system semantics. This means that file system operations like reading and writing function as though they are accessing a local file.

The benefits of the NFS service are that it:

- Allows multiple computers to use the same files, so everyone on the network can access the same data
- Reduces storage costs by having computers share applications instead of needing local disk space for each user application
- Provides data consistency and reliability because all users can read the same set of files
- Makes mounting of file systems transparent to users
- Makes accessing remote files transparent to users
- Supports heterogeneous environments
- Reduces system administration overhead

The NFS service makes the physical location of the file system irrelevant to the user. You can use the NFS implementation to enable users to see all the relevant files regardless of location. Instead of placing copies of commonly used files on every system, the NFS service enables you to place one copy on one computer's disk and have all other systems access it across the network. Under NFS operation, remote file systems are almost indistinguishable from local ones.

NFS Version 2

Version 2 was the first version of the NFS protocol in wide use. It continues to be available on a large variety of platforms. Solaris releases prior to Solaris 2.5 support version 2 of the NFS protocol.

NFS Version 3

An implementation of NFS version 3 protocol was a new feature of the Solaris 2.5 release. Several changes have been made to improve interoperability and performance. For optimal use, the version 3 protocol must be running on both the NFS servers and clients.

This version allows for safe asynchronous writes on the server, which improves performance by allowing the server to cache client write requests in memory. The client does not need to wait for the server to commit the changes to disk, so the response time is faster. Also, the server can batch the requests, which improves the response time on the server.

All NFS version 3 operations return the file attributes, which are stored in the local cache. Because the cache is updated more often, the need to do a separate operation to update this data arises less often. Therefore, the number of RPC calls to the server is reduced, improving performance.

The process for verifying file access permissions has been improved. In particular, version 2 would generate a message reporting a "write error" or a "read error" if users tried to copy a remote file to which they do not have permissions. In version 3, the permissions are checked before the file is opened, so the error is reported as an "open error."

The NFS version 3 implementation removes the 8-Kbyte transfer size limit. Clients and servers negotiate whatever transfer size they support, rather than be restricted by the 8-Kbyte limit that was imposed in version 2. The Solaris 2.5 implementation defaults to a 32-Kbyte transfer size.

NFS ACL Support

Access control list (ACL) support was added in the Solaris 2.5 release. ACLs provide a finer-grained mechanism to set file access permissions than is available through standard UNIX file permissions. NFS ACL support provides a method of changing and viewing ACL entries from a Solaris NFS client to a Solaris NFS server.

NFS Over TCP

The default transport protocol for the NFS protocol was changed to the Transport Control Protocol (TCP) in the Solaris 2.5 release, which helps performance on slow networks and wide area networks. TCP provides congestion control and error recovery. NFS over TCP works with version 2 and version 3. Prior to 2.5, the default NFS protocol was User Datagram Protocol (UDP).

Network Lock Manager

The Solaris 2.5 release also included an improved version of the network lock manager, which provided UNIX record locking and PC file sharing for NFS files. The locking mechanism is now more reliable for NFS files, so commands like `ksh` and `mail`, which use locking, are less likely to hang.

NFS Large File Support

The Solaris 2.6 release of the NFS version 3 protocol was changed to correctly manipulate files larger than 2 Gbytes. The NFS version 2 protocol and the Solaris 2.5 implementation of the version 3 protocol cannot handle files larger than 2 Gbytes.

NFS Client Failover

Dynamic failover of read-only file systems was added in the Solaris 2.6 release. It provides a high level of availability for read-only resources that are already replicated, such as man pages, AnswerBook™ documentation, and shared binaries. Failover can occur anytime after the file system is mounted. Manual mounts can now list multiple replicas, much like the automounter allowed in previous releases. The automounter has not changed, except that failover need not wait until the file system is remounted.

Kerberos Support for the NFS Environment

Support for Kerberos V4 clients was included in the Solaris 2.0 release. In release 2.6, the `mount` and `share` commands were altered to support NFS mounts using Kerberos V5 authentication. Also, the `share` command was changed to allow for multiple authentication flavors to different clients.

WebNFS Support

The Solaris 2.6 release also included the ability to make a file system on the Internet accessible through firewalls, using an extension to the NFS protocol. One of the advantages to using the WebNFS™ protocol for Internet access is its reliability: the service is built as an extension of the NFS version 3 and version 2 protocol. Soon, applications will be written to utilize this new file system access protocol. Also, an NFS server provides greater throughput under a heavy load than HyperText Transfer Protocol (HTTP) access to a Web server. This can decrease the amount of time required to retrieve a file. In addition, the WebNFS implementation provides the ability to share these files without the administrative overhead of an anonymous `ftp` site.

RPCSEC_GSS Security Flavor

A security flavor, called RPCSEC_GSS, is supported in the Solaris 7 release. This flavor uses the standard GSS-API interfaces to provide authentication, integrity and privacy, as well as allowing for support of multiple security mechanisms. Currently, only the client-side mechanisms to use this security flavor are integrated into the Solaris release.

Solaris 7 Extensions for NFS Mounting

Included in the Solaris 7 release are extensions to the `mount` and `automountd` command that allow for the mount request to use the public file handle instead of the MOUNT protocol. This is the same access method that the WebNFS service uses. By circumventing the MOUNT protocol, the mount can occur through a firewall. In addition, because fewer transactions need to occur between the server and client, the mount should occur faster.

The extensions also allow for NFS URLs to be used instead of the standard path name. Also, you can use the `–public` option with the `mount` command and the automounter maps to force the use of the public file handle.

Security Negotiation for the WebNFS Service

A new protocol has been added to enable a WebNFS client to negotiate a security mechanism with an NFS server. This provides the ability to use secure transactions when using the WebNFS service.

NFS Server Logging

NFS server logging allows an NFS server to provide a record of file operations performed on its file systems. The record includes information to track what is accessed, when it is accessed, and who accessed it. You can specify the location of the logs that contain this information through a set of configuration options. You can also use these options to select the operations that should be logged. This feature is particularly useful for sites that make anonymous FTP archives available to NFS and WebNFS clients.

About Autofs

File systems shared through the NFS service can be mounted using automatic mounting. Autofs, a client-side service, is a file system structure that provides automatic mounting. The autofs file system is initialized by `automount`, which is run automatically when a system is booted. The automount daemon, `automountd`, runs continuously, mounting and unmounting remote directories on an as-needed basis.

Whenever a user on a client computer running `automountd` tries to access a remote file or directory, the daemon mounts the file system to which that file or directory belongs. This remote file system remains mounted for as long as it is needed. If the remote file system is not accessed for a certain period of time, it is automatically unmounted.

Mounting need not be done at boot time, and the user no longer has to know the superuser password to mount a directory; users need not use the `mount` and `umount` commands. The autofs service mounts and unmounts file systems as required without any intervention on the part of the user.

Mounting some file hierarchies with `automountd` does not exclude the possibility of mounting others with `mount`. A diskless computer *must* mount `/` (root), `/usr`, and `/usr/kvm` through the `mount` command and the `/etc/vfstab` file.

"Autofs Administration Task Overview" on page 561 and "How Autofs Works" on page 635 give more specific information about the autofs service.

Autofs Features

Autofs works with file systems specified in the local name space. This information can be maintained in NIS, NIS+, or local files.

A fully multithreaded version of `automountd` was included in the Solaris 2.6 release. This enhancement makes autofs more reliable and allows for concurrent servicing of multiple mounts, which prevents the service from hanging if a server is unavailable.

The new `automountd` also provides better on-demand mounting. Previous releases would mount an entire set of file systems if they were hierarchically related. Now only the top file system is mounted. Other file systems related to this mount point are mounted when needed.

The autofs service supports browsability of indirect maps. This support allows a user to see what directories could be mounted, without having to actually mount each one of the file systems. A `–nobrowse` option has been added to the autofs maps, so that large file systems, such as `/net` and `/home`, are not automatically browsable. Also, you can turn off autofs browsability on each client by using the `–n` option with `automount`.

CHAPTER 30

Remote File-System Administration

This chapter provides information on how to perform such NFS administration tasks as setting up NFS services, adding new file systems to share, mounting file systems, using the Secure NFS system, or using the WebNFS functionality. The last part of the chapter includes troubleshooting procedures and a list of many of the NFS error messages and their meanings.

Your responsibilities as an NFS administrator depend on your site's requirements and the role of your computer on the network. You might be responsible for all the computers on your local network, in which case you might be responsible for determining these configuration items:

- Which computers, if any, should be dedicated servers
- Which computers should act as both servers and clients
- Which computers should be clients only

Maintaining a server after it has been set up involves the following tasks:

- Sharing and unsharing file systems as necessary
- Modifying administrative files to update the lists of file systems your computer shares or mounts automatically

- Checking the status of the network
- Diagnosing and fixing NFS related problems as they arise
- Setting up maps for autofs

Remember, a computer can be both a server and a client—sharing local file systems with remote computers and mounting remote file systems.

Automatic File-System Sharing

Servers provide access to their file systems by sharing them over the NFS environment. You specify which file systems are to be shared with the `share` command or the `/etc/dfs/dfstab` file.

Entries in the `/etc/dfs/dfstab` file are shared automatically whenever you start NFS server operation. You should set up automatic sharing if you need to share the same set of file systems on a regular basis. For example, if your computer is a server that supports home directories, you need to make the home directories available at all times. Most file-system sharing should be done automatically; the only time that manual sharing should occur is during testing or troubleshooting.

The `dfstab` file lists all the file systems that your server shares with its clients and controls which clients can mount a file system. If you want to modify `dfstab` to add or delete a file system or to modify the way sharing is done, edit the file with any supported text editor (such as `vi`). The next time the computer enters run level 3, the system reads the updated `dfstab` to determine which file systems should be shared automatically.

Each line in the `dfstab` file consists of a `share` command—the same command you type at the command-line prompt to share the file system. The `share` command is located in `/usr/sbin`.

TABLE 30–1 File-System Sharing Task Map

Task	Description	For Instructions, Go to ...
Establish automatic file-system sharing	Steps to configure a server so that file systems are automatically shared when the server is rebooted.	“How to Set Up Automatic File-System Sharing” on page 545
Enable WebNFS	Steps to configure a server so that users can access files using WebNFS	“How to Enable WebNFS Access” on page 545
Enabling NFS server logging	Steps to configure a server so that NFS logging is run on selected file systems	“How to Enable NFS Server Logging” on page 546

▼ How to Set Up Automatic File-System Sharing

1. **Become superuser.**

2. **Add entries for each file system to be shared.**

   Edit `/etc/dfs/dfstab` and add one entry to the file for each file system that you want to be automatically shared. Each entry must be on a line by itself in the file and uses this syntax:

   ```
   share [-F nfs] [-o specific-options] [-d description] pathname
   ```

   See the `share_nfs`(1M) man page for a complete list of options.

3. **Check that the NFS service is running on the server.**

   If this is the first `share` command or set of `share` commands that you have initiated, it is likely that the NFS daemons are not running. The following commands kill the daemons and restart them.

   ```
   # /etc/init.d/nfs.server stop
   # /etc/init.d/nfs.server start
   ```

   This ensures that NFS service is now running on the servers and will restart automatically when the server is at run level 3 during boot.

Where to Go From Here

The next step is to set up your autofs maps so that clients can access the file systems you have shared on the server. See “Autofs Administration Task Overview” on page 561.

▼ How to Enable WebNFS Access

Starting with the 2.6 release, by default all file systems that are available for NFS mounting are automatically available for WebNFS access. The only time that this procedure needs to be followed is on servers that do not already allow NFS mounting, if resetting the public file handle is useful to shorten NFS URLs, or if the `–index` option is required.

1. **Become superuser.**

2. **Add entries for each file system to be shared using the WebNFS service.**

   Edit `/etc/dfs/dfstab` and add one entry to the file for each file system using the `–public` option. The `–index` tag shown in the following example is optional.

```
share -F nfs -o ro,public,index=index.html /export/ftp
```

   See the `share_nfs`(1M) man page for a complete list of options.

3. **Check that the NFS service is running on the server.**

   If this is the first `share` command or set of `share` commands that you have initiated, it is likely that the NFS daemons are not running. The following commands kill and restart the daemons.

```
# /etc/init.d/nfs.server stop
# /etc/init.d/nfs.server start
```

4. **Share the file system.**

   After the entry is in `/etc/dfs/dfstab`, the file system can be shared by either rebooting the system or by using the `shareall` command. If the NFS daemons were restarted in step 2, this command does not need to be run because the script runs the command.

```
# shareall
```

5. **Verify that the information is correct.**

   Run the `share` command to check that the correct options are listed:

```
# share
-        /export/share/man   ro   ""
-        /usr/src     rw=eng   ""
-        /export/ftp    ro,public,index=index.html  ""
```

▼ How to Enable NFS Server Logging

1. **Become superuser.**

2. **Optional: Change file system configuration settings.**

   In `/etc/nfs/nfslog.conf`, you can either edit the default settings for all file systems by changing the data associated with the `global` tag or you can add a new tag for this file system. If these changes are not needed you do not need to change this file. The format of `/etc/nfs/nfslog.conf` is described in *nfslog.conf(1)*.

3. **Add entries for each file system to be shared using NFS server logging.**

   Edit `/etc/dfs/dfstab` and add one entry to the file for the file system that you want to have NFS server logging enabled on. The tag used with the `log=`*tag* option must be entered in `/etc/nfs/nfslog.conf`. This example uses the default settings in the `global` tag.

```
share -F nfs -o ro,log=global /export/ftp
```

   See the `share_nfs`(1M) man page for a complete list of options.

4. **Check that the NFS service is running on the server.**

   If this is the first `share` command or set of `share` commands that you have initiated, it is likely that the NFS daemons are not running. The following commands kill and restart the daemons.

```
# /etc/init.d/nfs.server stop
# /etc/init.d/nfs.server start
```

5. **Share the file system.**

   After the entry is in `/etc/dfs/dfstab`, the file system can be shared by either rebooting the system or by using the `shareall` command. If the NFS daemons were restarted earlier, this command does not need to be run because the script runs the command.

```
# shareall
```

6. **Verify that the information is correct.**

   Run the `share` command to check that the correct options are listed:

```
# share
-        /export/share/man   ro   ""
-        /usr/src     rw=eng   ""
-        /export/ftp    ro,log=global  ""
```

7. **Start the NFS log daemon, `nfslogd`, if it is not running already.**

   Restarting the NFS daemons using the `nfs.server` script will start the daemon if the `nfslog.conf` file exists. Otherwise the command needs to be run once by hand to create the files so that the command will automatically restart when the server is rebooted.

```
# /usr/lib/nfs/nfslogd
```

Mounting File Systems

You can mount file systems in several ways. They can be mounted automatically when the system is booted, on demand from the command line, or through the automounter. The automounter provides many advantages to mounting at boot time or mounting from the command line, but many situations require a combination of all three. In addition to these three ways of mounting a file system, there are several ways of enabling or disabling processes depending on the options you use when mounting the file system. See the following table for a complete list of the tasks associated with file system mounting.

TABLE 30–2 Mounting File Systems Task Map

Task	Description	For Instructions, Go to ...
Mount a file system at boot time	Steps so that a file system is mounted whenever a system is rebooted.	"How to Mount a File System at Boot Time" on page 549
Mount a file system using a command	Steps to mount a file system when a system is running. This procedure is useful when testing.	"How to Mount a File System From the Command Line" on page 550
Mount with the automounter	Steps to access a file system on demand without using the command line.	"Mounting With the Automounter" on page 550

TABLE 30–2 Mounting File Systems Task Map *(continued)*

Task	Description	For Instructions, Go to ...
Disallowing large files	Steps to prevent large files from being created on a file system.	"How to Disable Large Files on an NFS Server" on page 551
Using client-side failover	Steps to enable the automatic switchover to a working file system if a server fails.	"How to Use Client-Side Failover" on page 552
Disabling mount access for a client	Steps to disable the ability of one client to access a remote file system.	"How to Disable Mount Access for One Client" on page 552
Providing access to a file system through a firewall	Steps to allow access to a file system through a firewall by using the WebNFS protocol.	"How to Mount an NFS File System Through a Firewall" on page 553
Mounting a file system using a NFS URL	Steps to allow access to a file system using an NFS URL. This process allows for file-system access without using the MOUNT protocol.	"How to Mount an NFS File System Using an NFS URL" on page 553

▼ How to Mount a File System at Boot Time

If you want to mount file systems at boot time instead of using autofs maps, follow this procedure. Although you must follow this procedure for all local file systems, it is not recommended for remote file systems because it must be completed on every client.

1. **Become superuser.**

2. **Add an entry for the file system to** `/etc/vfstab`**.**

Entries in the `/etc/vfstab` file have the following syntax:

```
special  fsckdev  mountp  fstype  fsckpass  mount-at-boot  mntopts
```

See the `vfstab`(4) man page for more information.

Caution - NFS servers should not have NFS `vfstab` entries because of a potential deadlock. The NFS service is started after the entries in `/etc/vfstab` are checked, so that if two servers that are mounting file systems from each other fail at the same time, each system could hang as the systems reboot.

Example of a `vfstab` entry

You want a client computer to mount the `/var/mail` directory from the server `wasp`. You would like the file system to be mounted as `/var/mail` on the client and you want the client to have read-write access. Add the following entry to the client's `vfstab` file.

```
wasp:/var/mail - /var/mail nfs - yes rw
```

▼ How to Mount a File System From the Command Line

Mounting a file system from the command line is often done to test a new mount point or to allow for temporary access to a file system that is not available through the automounter.

1. **Become superuser.**

2. **Mount the file system.**

   Type the following command:

   ```
   # mount -F nfs -o ro bee:/export/share/local /mnt
   ```

   In this case, the `/export/share/local` file system from the server `bee` is mounted on read-only `/mnt` on the local system. Mounting from the command line allows for temporary viewing of the file system. You can unmount the file system with `umount` or by rebooting the local host.

Caution - Starting with the 2.6 release, all versions of the `mount` command will not warn about invalid options. The command silently ignores any options that cannot be interpreted. Make sure you verify all of the options that were used, to prevent unexpected behavior.

Mounting With the Automounter

"Autofs Administration Task Overview" on page 561 includes the specific instructions for establishing and supporting mounts with the automounter. Without any changes to the generic system, clients should be able to access remote file systems through the `/net` mount point. To mount the `/export/share/local` file system from the previous example, all you need to do is type:

```
% cd /net/bee/export/share/local
```

Because the automounter allows all users to mount file systems, root access is not required. It also provides for automatic unmounting of file systems, so there is no need to unmount file systems after you are finished.

▼ How to Disable Large Files on an NFS Server

For servers that are supporting clients that cannot handle a file over 2 GBytes, it is necessary to disable the ability to create large files.

Note - Previous versions of the Solaris operating environment cannot use large files. Check that clients of the NFS server are running at least the 2.6 release if the clients need to access large files.

---

1. **Become superuser.**

2. **Check that no large files exist on the file system.**

   Here is an example of a command that you can run to locate large files:

```
# cd /export/home1
# find . -xdev -size +2000000 -exec ls -l {} \;
```

   If large files are on the file system, you must remove or move them to another file system.

3. **Unmount the file system.**

```
# umount /export/home1
```

4. **Reset the file system state if the file system has been mounted using** `–largefiles`**.**

   `fsck` resets the file system state if no large files exist on the file system:

```
# fsck /export/home1
```

5. **Mount the file system using** `nolargefiles`**.**

```
# mount -F ufs -o nolargefiles /export/home1
```

You can do this from the command line, but to make the option more permanent, add an entry like the following into `/etc/vfstab`:

```
/dev/dsk/c0t3d0s1 /dev/rdsk/c0t3d0s1 /export/home1  ufs  2  yes  nolargefiles
```

▼ How to Use Client-Side Failover

1. **Become superuser.**

2. **On the NFS client, mount the file system using the `ro` option.**

   You can do this from the command line, through the automounter, or by adding an entry to `/etc/vfstab` that looks like:

   ```
   bee,wasp:/export/share/local  -  /usr/local  nfs  -  no  -o ro
   ```

   This syntax has been allowed by the automounter in earlier releases, but the failover was not available while file systems were mounted, only when a server was being selected.

Note - Servers that are running different versions of the NFS protocol cannot be mixed using a command line or in a `vfstab` entry. Mixing servers supporting NFS V2 and V3 protocols can only be done with autofs, in which case the best subset of version 2 or version 3 servers is used.

▼ How to Disable Mount Access for One Client

1. **Become superuser.**

2. **Add an entry in `/etc/dfs/dfstab`.**

   The first example allows mount access to all clients in the `eng` netgroup except the host named `rose`. The second example allows mount access to all clients in the `eng.sun.com` DNS domain except for `rose`.

```
share -F nfs -o ro=-rose:eng /export/share/man
share -F nfs -o ro=-rose:.eng.sun.com /export/share/man
```

For additional information on access lists, see “Setting Access Lists With the `share` Command” on page 610.

3. **Share the file system.**

   The NFS server does not use changes to `/etc/dfs/dfstab` until the file systems are shared again or until the server is rebooted.

```
# shareall
```

▼ How to Mount an NFS File System Through a Firewall

1. **Become superuser.**

2. **Manually mount the file system, using a command like:**

```
# mount -F nfs -o public bee:/export/share/local /mnt
```

In this example the file system `/export/share/local` is mounted on the local client using the public file handle. An NFS URL can be used instead of the standard path name. If the public file handle is not supported by the server `bee`, the mount operation will fail.

Note - This procedure requires that the file system on the NFS server be shared using the public option and any firewalls between the client and the server allow TCP connections on port `2049`. Starting with the 2.6 release, all file systems that are shared allow for public file handle access.

▼ How to Mount an NFS File System Using an NFS URL

1. **Become superuser.**

2. **Manually mount the file system, using a command such as:**

```
# mount -F nfs nfs://bee:3000/export/share/local /mnt
```

In this example, the `/export/share/local` file system is being mounted from the server `bee` using NFS port number `3000`. The port number is not required and by default uses the standard NFS port number of `2049`. You can include the public option with an NFS URL, if you want. Without the public option, the MOUNT protocol is used if the public file handle is not supported by the server. The public option will force the use of the public file handle, and the mount will fail if the public file handle is not supported.

Setting Up NFS Services

This section discusses some of the tasks necessary to initialize or use NFS services.

TABLE 30–3 NFS Services Task Map

Task	Description	For Instructions, Go To ...
Start the NFS server	Steps to start the NFS service, if it has not been started automatically.	"How to Start the NFS Services" on page 554
Stop the NFS server	Steps to stop the NFS service. Normally the service should not need to be stopped.	"How to Stop the NFS Services" on page 555
Start the automounter	Steps to start the automounter. This procedure is required when some of the automounter maps are changed.	"How to Start the Automounter" on page 555
Stop the automounter	Steps to stop the automounter. This procedure is required when some of the automounter maps are changed.	"How to Stop the Automounter" on page 555

▼ How to Start the NFS Services

1. **Become superuser.**

2. **Enable the NFS service daemons.**

Type the following command:

```
# /etc/init.d/nfs.server start
```

This starts the daemons if there is an entry in `/etc/dfs/dfstab`.

▼ How to Stop the NFS Services

1. **Become superuser.**
2. **Disable the NFS service daemons.**
   Type the following command:

```
# /etc/init.d/nfs.server stop
```

▼ How to Start the Automounter

1. **Become superuser.**
2. **Enable the autofs daemon.**
   Type the following command:

```
# /etc/init.d/autofs start
```

This starts the daemon.

▼ How to Stop the Automounter

1. **Become superuser.**
2. **Disable the autofs daemon.**
   Type the following command:

```
# /etc/init.d/autofs stop
```

Administering the Secure NFS System

To use the Secure NFS system, all the computers you are responsible for must have a domain name. A domain is an administrative entity, typically consisting of several computers, that is part of a larger network. If you are running NIS+, you should also establish the NIS+ name service for the domain. See *Solaris Naming Setup and Configuration Guide.*

You can configure the Secure NFS environment to use either Diffie-Hellman.. "Managing System Security (Overview)" in *System Administration Guide, Volume 2* discusses this authentication service.

▼ How to Set Up a Secure NFS Environment With DH Authentication

1. **Assign your domain a domain name, and make the domain name known to each computer in the domain.**

   See the *Solaris Naming Administration Guide* if you are using NIS+ as your name service.

2. **Establish public keys and secret keys for your clients' users using the** `newkey` **or** `nisaddcred` **command, and have each user establish his or her own secure RPC password using the** `chkey` **command.**

   Note - For information about these commands, see the `newkey`(1M), the `nisaddcred`(1M), and the `chkey`(1) man pages.

   When public and secret keys have been generated, the public and encrypted secret keys are stored in the `publickey` database.

3. **Verify that the name service is responding. If you are running NIS+, type the following:**

```
# nisping -u
Last updates for directory eng.acme.com. :
Master server is eng-master.acme.com.
        Last update occurred at Mon Jun  5 11:16:10 1995

Replica server is eng1-replica-replica-58.acme.com.
```

(continued)

```
        Last Update seen was Mon Jun  5 11:16:10 1995
```

If you are running NIS, verify that the `ypbind` daemon is running.

4. Verify that the `keyserv` daemon (the key server) is running.

Type the following command.

```
# ps -ef | grep keyserv
root    100     1  16    Apr 11 ?      0:00 /usr/sbin/keyserv
root   2215  2211   5  09:57:28 pts/0  0:00 grep keyserv
```

If the daemon isn't running, start the key server by typing the following:

```
# /usr/sbin/keyserv
```

5. Decrypt and store the secret key.

Usually, the login password is identical to the network password. In this case, `keylogin` is not required. If the passwords are different, the users have to log in, and then do a `keylogin`. You still need to use the `keylogin -r` command as root to store the decrypted secret key in `/etc/.rootkey`.

Note - You only need to run `keylogin -r` if the root secret key changes or `/etc/.rootkey` is lost.

---

6. Update mount options for the file system.

Edit the `/etc/dfs/dfstab` file and add the `sec=dh` option to the appropriate entries (for Diffie-Hellman authentication).

```
share -F nfs -o sec=dh /export/home
```

7. Update the automounter maps for the file system.

Edit the `auto_master` data to include `sec=dh` as a mount option in the appropriate entries (for Diffie-Hellman authentication):

```
/home auto_home -nosuid,sec=dh
```

Note - With 2.5 and earlier Solaris releases, if a client does not mount as secure a file system that is shared as secure, users have access as user `nobody`, rather than as themselves. With Version 2 on later releases, the NFS server refuses access if the security modes do not match, unless `–sec=none` is included on the `share` command line. With version 3, the mode is inherited from the NFS server, so clients do not need to specify `sec=krb4` or `sec=dh`. The users have access to the files as themselves.

When you reinstall, move, or upgrade a computer, remember to save `/etc/.rootkey` if you do not establish new keys or change them for `root`. If you do delete `/etc/.rootkey`, you can always type:

```
# keylogin -r
```

WebNFS Administration Tasks

This section provides instructions for administering the WebNFS system. This is a list of some related tasks.

TABLE 30–4 WebNFS Administration Task Map

Task	Description	For Instructions, Go To ...
Plan for WebNFS	Issues to consider before enabling the WebNFS service.	“Planning for WebNFS Access” on page 559
Enable WebNFS	Steps to enable mounting of an NFS file system using the WebNFS protocol.	“How to Enable WebNFS Access” on page 545
Enabling WebNFS through a firewall	Steps to allow access to files through a firewall by using the WebNFS protocol.	“Enabling WebNFS Access Through a Firewall” on page 560
Browsing using an NFS URL	Instructions for using an NFS URL within a web browser.	“Browsing Using an NFS URL” on page 560

TABLE 30–4 WebNFS Administration Task Map *(continued)*

Task	Description	For Instructions, Go To ...
Using a public file handle with autofs	Steps to force use of the public file handle when mounting a file system with the automounter.	"How to Use a Public File Handle With Autofs" on page 574
Using an NFS URL with autofs	Steps to add an NFS URL to the automounter maps.	"How to Use NFS URLs With Autofs" on page 574
Providing access to a file system through a firewall	Steps to allow access to a file system through a firewall using the WebNFS protocol.	"How to Mount an NFS File System Through a Firewall" on page 553
Mounting a file system using an NFS URL	Steps to allow access to a file system using an NFS URL. This process allows for file system access without using the MOUNT protocol.	"How to Mount an NFS File System Using an NFS URL" on page 553

Planning for WebNFS Access

To use the WebNFS functionality, you first need an application capable of running and loading an NFS URL (for example, `nfs://server/path`). The next step is to choose the file system that will be exported for WebNFS access. If the application is web browsing, often the document root for the web server is used. Several factors need to be considered when choosing a file system to export for WebNFS access.

1. Each server has one public file handle that by default is associated with the server's root file system. The path in an NFS URL is evaluated relative to the directory with which the public file handle is associated. If the path leads to a file or directory within an exported file system, the server provides access. You can use the `–public` option of the `share` command to associate the public file handle with a specific exported directory. Using this option allows URLs to be relative to the shared file system rather than to the servers' root file system. By default the public file handle points to the root file system, but this file handle does not allow web access unless the root file system is shared.
2. The WebNFS environment allows users who already have mount privileges to access files through a browser regardless of whether the file system is exported using the `–public` option. Because users already have access to these files through the NFS setup, this should not create any additional security risk. You only need to share a file system using the `–public` option if users who cannot mount the file system need to use WebNFS access.
3. File systems that are already open to the public make good candidates for using the `–public` option, like the top directory in an ftp archive or the main URL directory for a web site.

4. You can use the –`index` option with the `share` command to force the loading of an HTML file instead of listing the directory when an NFS URL is accessed.

   After a file system is chosen, review the files and set access permissions to restrict viewing of files or directories as needed. Establish the permissions as appropriate for any NFS file system that is being shared. For many sites, `755` permissions for directories and `644` permissions for files provides the correct level of access.

   Additional factors need to be considered if both NFS and HTTP URLs are to be used to access one eb site. These are described in "WebNFS Limitations With Web Browser Use" on page 625.

▼ Browsing Using an NFS URL

Browsers capable of supporting WebNFS access should provide access using an NFS URL that looks something like:

```
nfs://server<:port>/path
```

server	Name of the file server
port	Port number to use (the default value is `2049`)
path	Path to file, which can be relative to the public file handle or to the root file system

---

Note - In most browsers, the URL service type (for example, `nfs` or `http`) is remembered from one transaction to the next, unless a URL that includes a different service type is loaded. When using NFS URLs, if a reference to an HTTP URL is loaded, subsequent pages are loaded using the HTTP protocol instead of the NFS protocol, unless the URLs specify an NFS URL.

▼ Enabling WebNFS Access Through a Firewall

You can enable WebNFS access for clients that are not part of the local subnet by configuring the firewall to allow a TCP connection on port `2049`. Just allowing access for `httpd` does not allow NFS URLs to be used.

Autofs Administration Task Overview

This section describes some of the most common tasks you might encounter in your own environment. Recommended procedures are included for each scenario to help you configure autofs to best meet your clients' needs.

Note - Use the Solstice System Management Tools or see the *Solaris Naming Administration Guide* to perform the tasks discussed in this section.

Autofs Administration Task Map

The following table lists a description and a pointer to many of the tasks that are related to autofs.

TABLE 30–5 Autofs Administration Task Map

Task	Description	For Instructions, Go To ...
Start autofs	Start the automount service without having the reboot the system	"How to Start the Automounter" on page 555
Stop autofs	Stop the automount service without disabling other network services	"How to Stop the Automounter" on page 555
Access file systems using autofs	Access file systems using the automount service	"Mounting With the Automounter" on page 550
Modifying the autofs maps	Steps to modify the master map, which should be used to list other maps	"How to Modify the Master Map" on page 564
	Steps to modify an indirect map, which should be used for most maps	"How to Modify Indirect Maps" on page 565
	Steps to modify a direct map, which should be used when a direct association between a mount point on a client and a server is required	"How to Modify Direct Maps" on page 565
Modifying the autofs maps to access non NFS file systems	Steps to set up an autofs map with an entry for a CD-ROM application	"How to Access CD-ROM Applications With Autofs" on page 566

TABLE 30–5 Autofs Administration Task Map *(continued)*

Task	Description	For Instructions, Go To ...
	Steps to set up an autofs map with an entry for a PC-DOS diskette	"How to Access PC-DOS Data Diskettes With Autofs" on page 567
	Steps to use autofs to access a CacheFS file system	"How to Access NFS File Systems Using CacheFS" on page 567
Using /home	Example of how to set up a common `/home` map	"Setting Up a Common View of `/home`" on page 568
	Steps to set up a `/home` map that refers to multiple file systems	"How to Set Up `/home` With Multiple Home Directory File Systems" on page 569
Using a new autofs mount point	Steps to set up a project-related autofs map	"How to Consolidate Project-Related Files Under `/ws`" on page 570
	Steps to set up an autofs map that supports different client architectures	"How to Set Up Different Architectures to Access a Shared Name Space" on page 571
	Steps to set up an autofs map that supports different operating systems	"How to Support Incompatible Client Operating System Versions" on page 572
Replicating file systems with autofs	Provide access to file systems that failover	"How to Replicate Shared Files Across Several Servers" on page 573
Using security restrictions with autofs	Provide access to file systems while restricting remote `root` access to the files	"How to Apply Security Restrictions" on page 573
Using a public file handle with autofs	Force use of the public file handle when mounting a file system	"How to Use a Public File Handle With Autofs" on page 574
Using an NFS URL with autofs	Add an NFS URL so that the automounter can use it	"How to Use NFS URLs With Autofs" on page 574
Disable autofs browsability	Steps to disable browability so that autofs mount points are not automatically populated on a single client	"How to Completely Disable Autofs Browsability on a Single NFS Client" on page 575

TABLE 30–5 Autofs Administration Task Map *(continued)*

Task	Description	For Instructions, Go To ...
	Steps to disable browability so that autofs mount points are not automatically populated on all clients	"How to Disable Autofs Browsability for All Clients" on page 575
	Steps to disable browability so that a specific autofs mount point is not automatically populated on a client	"How to Disable Autofs Browsability on an NFS Client" on page 576

Administrative Tasks Involving Maps

The following tables describe several of the factors you need to be aware of when administering autofs maps. Which type of map and which name service you choose changes the mechanism which you need to use to make changes to the autofs maps.

The following table describes the types of maps and their uses.

TABLE 30–6 Types of autofs Maps and Their Uses

Type of Map	Use
Master	Associates a directory with a map
Direct	Directs autofs to specific file systems
Indirect	Directs autofs to reference-oriented file systems

The following table describes how to make changes to your autofs environment based on your name service.

TABLE 30–7 Map Maintenance

Name Service	Method
Local files	Text editor
NIS	`make` files
NIS+	`nistbladm`

TABLE 30–7 Map Maintenance *(continued)*

The next table tells you when to run the `automount` command, depending on the modification you have made to the type of map. For example, if you have made an addition or a deletion to a direct map, you need to run the `automount` command on the local system to allow the change take effect; however, if you've modified an existing entry, you do not need to run the `automount` command for the change to take effect.

TABLE 30–8 When to Run the `automount` Command

Type of Map	**Restart** `automount`**?**	
	Addition or Deletion	**Modification**
`auto_master`	`Y`	`Y`
`direct`	`Y`	`N`
`indirect`	`N`	`N`

Modifying the Maps

The following procedures require that you use NIS+ as your name service.

▼ How to Modify the Master Map

1. **Using the** `nistbladm` **command, make the changes you want to the master map.**

   See the *Solaris Naming Administration Guide*.

2. **For each client, become superuser.**

3. **For each client, run the** `automount` **command to ensure the changes you made take effect.**

4. **Notify your users of the changes.**

   Notification is required so that the users can also run the automount command as superuser on their own computers.

The `automount` command gathers information from the master map whenever it is run.

▼ How to Modify Indirect Maps

- **Using the `nistbladm` command, make the changes you want to the indirect map.**

  See the *Solaris Naming Administration Guide.*

The change takes effect the next time the map is used, which is the next time a mount is done.

▼ How to Modify Direct Maps

1. **Using the `nistbladm` command, add or delete the changes you want to the direct map.**

   See the *Solaris Naming Administration Guide.*

2. **If you added or deleted a mount-point entry in step 1, run the `automount` command.**

3. **Notify your users of the changes.**

   Notification is required so that the users can also run the `automount` command as superuser on their own computers.

Note - If you only modify or change the contents of an existing direct map entry, you do not need to run the `automount` command.

For example, suppose you modify the `auto_direct` map so that the `/usr/src` directory is now mounted from a different server. If `/usr/src` is not mounted at this time, the new entry takes effect immediately when you try to access `/usr/src`. If `/usr/src` is mounted now, you can wait until the auto-unmounting takes place, then access it.

Note - Because of the additional steps, and because they do not take up as much space in the mount table as direct maps, use indirect maps whenever possible. They are easier to construct, and less demanding on the computers' file systems.

Avoiding Mount-Point Conflicts

If you have a local disk partition mounted on `/src` and you also want to use the autofs service to mount other source directories, you might encounter a problem. If you specify the mount point `/src`, the service hides the local partition whenever you try to reach it.

You need to mount the partition somewhere else; for example, on `/export/src`. You would then need an entry in `/etc/vfstab` like:

```
/dev/dsk/d0t3d0s5 /dev/rdsk/c0t3d0s5 /export/src ufs 3 yes -
```

and this entry in `auto_src`:

```
terra  terra:/export/src
```

where `terra` is the name of the computer.

Accessing Non NFS File Systems

Autofs can also mount files other than NFS files. Autofs mounts files on removable media, such as diskettes or CD-ROM. Normally, you would mount files on removable media using the Volume Manager. The following examples show how this mounting could be done through autofs. The Volume Manager and autofs do not work together, so these entries would not be used without first deactivating the Volume Manager.

Instead of mounting a file system from a server, you put the media in the drive and reference it from the map. If you want to access non NFS file systems and you are using autofs, see the following procedures.

How to Access CD-ROM Applications With Autofs

Note - Use this procedure if you are *not* using Volume Manager.

1. **Become superuser.**

2. **Update the autofs map.**

   Add an entry for the CD-ROM file system, which should look like:

   ```
   hsfs     -fstype=hsfs,ro     :/dev/sr0
   ```

   The CD-ROM device you want to mount must appear as a name following a colon.

▼ How to Access PC-DOS Data Diskettes With Autofs

Note - Use this procedure if you are *not* using Volume Manager.

1. **Become superuser.**

2. **Update the autofs map.**

   Add an entry for the diskette file system such as:

```
pcfs     -fstype=pcfs     :/dev/diskette
```

Accessing NFS File Systems Using CacheFS

The cache file system (CacheFS) is a generic nonvolatile caching mechanism that improves the performance of certain file systems by utilizing a small, fast, local disk.

You can improve the performance of the NFS environment by using CacheFS to cache data from an NFS file system on a local disk.

▼ How to Access NFS File Systems Using CacheFS

1. **Become superuser.**

2. **Run the** `cfsadmin` **command to create a cache directory on the local disk.**

```
# cfsadmin -c /var/cache
```

3. **Add the cachefs entry to the appropriate automounter map.**

   For example, adding this entry to the master map caches all home directories:

```
/home auto_home -fstype=cachefs,cachedir=/var/cache,backfstype=nfs
```

   Adding this entry to the `auto_home` map only caches the home directory for the user named `rich`:

```
rich -fstype=cachefs,cachedir=/var/cache,backfstype=nfs dragon:/export/home1/rich
```

Note - Options that are included in maps that are searched later override options set in maps that are searched earlier. The last options found are the ones that are used. In the previous example, a specific entry added to the `auto_home` map only needs to include the options listed in the master maps if some of the options needed to be changed.

Customizing the Automounter

You can set up the automounter maps in several ways. The following tasks give detailed instructions on how to customize the automounter maps to provide an easy-to-use directory structure.

▼ Setting Up a Common View of `/home`

The ideal is for all network users to be able to locate their own, or anyone else's home directory under `/home`. This view should be common across all computers, whether client or server.

Every Solaris installation comes with a master map: `/etc/auto_master`.

```
# Master map for autofs
#
+auto_master
/net     -hosts     -nosuid,nobrowse
/home    auto_home  -nobrowse
/xfn     -xfn
```

A map for `auto_home` is also installed under `/etc`.

```
# Home directory map for autofs
#
+auto_home
```

Except for a reference to an external `auto_home` map, this map is empty. If the directories under `/home` are to be common to all computers, do not modify this `/etc/auto_home` map. All home directory entries should appear in the name service files, either NIS or NIS+.

Note - Users should not be permitted to run `setuid` executables from their home directories; without this restriction, any user could have superuser privileges on any computer.

▼ How to Set Up /home With Multiple Home Directory File Systems

1. **Become superuser.**

2. **Install home directory partitions under** `/export/home`**.**

   If there are several partitions, install them under separate directories, for example, `/export/home1`, `/export/home2`, and so on.

3. **Use the Solstice System Management Tools to create and maintain the** `auto_home` **map.**

   Whenever you create a new user account, type the location of the user's home directory in the `auto_home` map. Map entries can be simple, for example:

```
rusty        dragon:/export/home1/&
gwenda       dragon:/export/home1/&
charles      sundog:/export/home2/&
rich         dragon:/export/home3/&
```

   Notice the use of the & (ampersand) to substitute the map key. This is an abbreviation for the second occurrence of `rusty` in the following example.

```
rusty      dragon:/export/home1/rusty
```

   With the `auto_home` map in place, users can refer to any home directory (including their own) with the path `/home/`*user*, where *user* is their login name and the key in the map. This common view of all home directories is valuable when logging in to another user's computer. Autofs mounts your home directory for you. Similarly, if you run a remote windowing system client on another computer, the client program has the same view of the `/home` directory as you do on the computer providing the windowing system display.

   This common view also extends to the server. Using the previous example, if `rusty` logs in to the server `dragon`, autofs there provides direct access to the local disk by loopback-mounting `/export/home1/rusty` onto `/home/rusty`.

   Users do not need to be aware of the real location of their home directories. If `rusty` needs more disk space and needs to have his home directory relocated to another server, you need only change `rusty`'s entry in the `auto_home` map to reflect the new location. Everyone else can continue to use the `/home/rusty` path.

▼ How to Consolidate Project-Related Files Under /ws

Assume you are the administrator of a large software development project. You want to make all project-related files available under a directory called /ws. This directory is to be common across all workstations at the site.

1. **Add an entry for the /ws directory to the site auto_master map, either NIS or NIS+.**

```
/ws     auto_ws     -nosuid
```

   The auto_ws map determines the contents of the /ws directory.

2. **Add the -nosuid option as a precaution.**

   This option prevents users from running setuid programs that might exist in any workspaces.

3. **Add entries to the auto_ws map.**

   The auto_ws map is organized so that each entry describes a subproject. Your first attempt yields a map that looks like the following:

```
compiler   alpha:/export/ws/&
windows    alpha:/export/ws/&
files      bravo:/export/ws/&
drivers    alpha:/export/ws/&
man        bravo:/export/ws/&
tools      delta:/export/ws/&
```

   The ampersand (&) at the end of each entry is an abbreviation for the entry key. For instance, the first entry is equivalent to:

```
compiler  alpha:/export/ws/compiler
```

   This first attempt provides a map that looks simple, but it turns out to be inadequate. The project organizer decides that the documentation in the man entry should be provided as a subdirectory under each subproject. Also, each subproject requires subdirectories to describe several versions of the software. You must assign each of these subdirectories to an entire disk partition on the server.

   Modify the entries in the map as follows:

```
compiler \
    /vers1.0    alpha:/export/ws/&/vers1.0 \
    /vers2.0    bravo:/export/ws/&/vers2.0 \
    /man        bravo:/export/ws/&/man
windows \
    /vers1.0    alpha:/export/ws/&/vers1.0 \
    /man        bravo:/export/ws/&/man
files \
    /vers1.0    alpha:/export/ws/&/vers1.0 \
    /vers2.0    bravo:/export/ws/&/vers2.0 \
    /vers3.0    bravo:/export/ws/&/vers3.0 \
    /man        bravo:/export/ws/&/man
drivers \
    /vers1.0    alpha:/export/ws/&/vers1.0 \
    /man        bravo:/export/ws/&/man
tools \
    /           delta:/export/ws/&
```

Although the map now appears to be much larger, it still contains only the five entries. Each entry is larger because it contains multiple mounts. For instance, a reference to `/ws/compiler` requires three mounts for the `vers1.0`, `vers2.0`, and `man` directories. The backslash at the end of each line tells autofs that the entry is continued onto the next line. In effect, the entry is one long line, though line breaks and some indenting have been used to make it more readable. The `tools` directory contains software development tools for all subprojects, so it is not subject to the same subdirectory structure. The `tools` directory continues to be a single mount.

This arrangement provides the administrator with much flexibility. Software projects are notorious for consuming substantial amounts of disk space. Through the life of the project you might be required to relocate and expand various disk partitions. As long as these changes are reflected in the `auto_ws` map, the users do not need to be notified, as the directory hierarchy under `/ws` is not changed.

Because the servers `alpha` and `bravo` view the same autofs map, any users who log in to these computers can find the `/ws` name space as expected. These users are provided with direct access to local files through loopback mounts instead of NFS mounts.

▼ How to Set Up Different Architectures to Access a Shared Name Space

You need to assemble a shared name space for local executables, and applications, such as spreadsheet tools and word-processing packages. The clients of this name space use several different workstation architectures that require different executable

formats. Also, some workstations are running different releases of the operating system.

1. **Create the `auto_local` map with the `nistbladm` command.**

   See the *Solaris Naming Administration Guide*.

2. **Choose a single, site-specific name for the shared name space so that files and directories that belong to this space are easily identifiable.**

   For example, if you choose `/usr/local` as the name, the path `/usr/local/bin` is obviously a part of this name space.

3. **For ease of user community recognition, create an autofs indirect map and mount it at `/usr/local`. Set up the following entry in the NIS+ (or NIS) `auto_master` map:**

   ```
   /usr/local     auto_local     -ro
   ```

   Notice that the `–ro` mount option implies that clients will not be able to write to any files or directories.

4. **Export the appropriate directory on the server.**

5. **Include a `bin` entry in the auto_local map.**

   Your directory structure looks like this:

   ```
   bin     aa:/export/local/bin
   ```

   To satisfy the need to serve clients of different architectures, references to the `bin` directory need to be directed to different directories on the server, depending on the clients' architecture type.

6. **To serve clients of different architectures, change the entry by adding the autofs `CPU` variable.**

   ```
   bin     aa:/export/local/bin/$CPU
   ```

   - For SPARC clients – Place executables in `/export/local/bin/sparc`
   - For IA clients – Place executables in `/export/local/bin/i386`

▼ How to Support Incompatible Client Operating System Versions

1. **Combine the architecture type with a variable that determines the operating system type of the client.**

The autofs OSREL variable can be combined with the CPU variable to form a name that determines both CPU type and OS release.

2. **Create the following map entry.**

```
bin     aa:/export/local/bin/$CPU$OSREL
```

For clients running version 5.6 of the operating system, export the following file systems:

- For SPARC clients – Export /export/local/bin/sparc5.6
- For IA clients – Place executables in /export/local/bin/i3865.6

▼ How to Replicate Shared Files Across Several Servers

The best way to share replicated file systems that are read-only is to use failover. See "Client-Side Failover" on page 621 for a discussion of failover.

1. **Become superuser.**

2. **Modify the entry in the autofs maps.**

   Create the list of all replica servers as a comma-separated list, such as:

```
bin     aa,bb,cc,dd:/export/local/bin/$CPU
```

Autofs chooses the nearest server. If a server has several network interfaces, list each interface. Autofs chooses the nearest interface to the client, avoiding unnecessary routing of NFS traffic.

▼ How to Apply Security Restrictions

1. **Become superuser.**

2. **Create the following entry in the name service** auto_master **file, either NIS or NIS+:**

```
/home     auto_home     -nosuid
```

The nosuid option prevents users from creating files with the setuid or setgid bit set.

This entry overrides the entry for `/home` in a generic local `/etc/auto_master` file (see the previous example) because the `+auto_master` reference to the external name service map occurs before the `/home` entry in the file. If the entries in the `auto_home` map include mount options, the `nosuid` option is overwritten, so either no options should be used in the `auto_home` map or the `nosuid` option must be included with each entry.

Note - Do not mount the home directory disk partitions on or under `/home` on the server.

▼ How to Use a Public File Handle With Autofs

1. **Become superuser.**
2. **Create an entry in the autofs map like:**

```
/usr/local     -ro,public    bee:/export/share/local
```

The `public` option forces the public handle to be used. If the NFS server does not support a public file handle, the mount will fail.

▼ How to Use NFS URLs With Autofs

1. **Become superuser.**
2. **Create an autofs entry like:**

```
/usr/local     -ro    nfs://bee/export/share/local
```

The service tries to use the public file handle on the NFS server, but if the server does not support a public file handle, the MOUNT protocol is used.

Disabling Autofs Browsability

Starting with the Solaris 2.6 release, the default version of `/etc/auto_master` that is installed has the –`nobrowse` option added to the entries for `/home` and `/net`. In addition, the upgrade procedure adds the –`nobrowse` option to the `/home` and `/net` entries in `/etc/auto_master` if these entries have not been modified. However, it might be necessary to make these changes manually or to turn off browsability for site-specific autofs mount points after the installation.

You can turn off the browsability feature in several ways. Disable it using a command-line option to the `automountd` daemon, which completely disables autofs browsability for the client. Or disable it for each map entry on all clients using the autofs maps in either a NIS or NIS+ name space, or for each map entry on each client, using local autofs maps if no network-wide name space is being used.

▼ How to Completely Disable Autofs Browsability on a Single NFS Client

1. **Become superuser.**

2. **Add the** –n **option to the startup script.**

   As `root`, edit the `/etc/init.d/autofs` script and add the –n option to the line that starts the `automountd` daemon:

```
/usr/lib/autofs/automountd -n \
 < /dev/null > /dev/console 2>&1 # start daemon
```

3. **Restart the autofs service.**

```
# /etc/init.d/autofs stop
# /etc/init.d/autofs start
```

▼ How to Disable Autofs Browsability for All Clients

To disable browsability for all clients, you must employ a name service such as NIS or NIS+. Otherwise, you need to manually edit the automounter maps on each client. In this example, the browsability of the `/home` directory is disabled. You must follow this procedure for each indirect autofs node that needs to be disabled.

1. **Add the** –nobrowse **option to the** `/home` **entry in the name service** `auto_master` **file.**

```
/home      auto_home      -nobrowse
```

2. **On all clients: run the** `automount` **command.**

   The new behavior takes effect after running the `automount` command on the client systems or after a reboot.

```
# /usr/sbin/automount
```

▼ How to Disable Autofs Browsability on an NFS Client

In this example, browsability of the `/net` directory is disabled. The same procedure can be used for `/home` or any other autofs mount points.

1. **Check the** `automount` **entry in** `/etc/nsswitch.conf`**.**

   For local file entries to take precedence, the entry in the name service switch file should list `files` before the name service. For example:

```
automount:  files nisplus
```

   This is the default configuration in a standard Solaris installation.

2. **Check the position of the** `+auto_master` **entry in** `/etc/auto_master`**.**

   For additions to the local files to take precedence over the entries in the name space, the `+auto_master` entry must be moved below `/net`:

```
# Master map for automounter
#
/net    -hosts     -nosuid
/home   auto_home
/xfn    -xfn
+auto_master
```

   A standard configuration places the `+auto_master` entry at the top of the file. This prevents any local changes from being used.

3. **Add the** `–nobrowse` **option to the** `/net` **entry in the** `/etc/auto_master` **file.**

```
/net     -hosts     -nosuid,nobrowse
```

4. **On all clients: run the** `automount` **command.**

The new behavior takes effect after running the `automount` command on the client systems or after a reboot.

```
# /usr/sbin/automount
```

Strategies for NFS Troubleshooting

When tracking down an NFS problem, keep in mind the main points of possible failure: the server, the client, and the network. The strategy outlined in this section tries to isolate each individual component to find the one that is not working. In all cases, the `mountd` and `nfsd` daemons must be running on the server for remote mounts to succeed.

Note - The `mountd` and `nfsd` daemons start automatically at boot time only if NFS share entries are in the `/etc/dfs/dfstab` file. Therefore, `mountd` and `nfsd` must be started manually when setting up sharing for the first time.

The `–intr` option is set by default for all mounts. If a program hangs with a "server not responding" message, you can kill it with the keyboard interrupt Control-c.

When the network or server has problems, programs that access hard-mounted remote files fail differently than those that access soft-mounted remote files. Hard-mounted remote file systems cause the client's kernel to retry the requests until the server responds again. Soft-mounted remote file systems cause the client's system calls to return an error after trying for awhile. Because these errors can result in unexpected application errors and data corruption, avoid soft-mounting.

When a file system is hard mounted, a program that tries to access it hangs if the server fails to respond. In this case, the NFS system displays the following message on the console:

```
NFS server hostname not responding still trying
```

When the server finally responds, the following message appears on the console:

```
NFS server hostname ok
```

A program accessing a soft-mounted file system whose server is not responding generates the following message:

```
NFS operation failed for server hostname: error # (error_message)
```

Note - Because of possible errors, do not soft-mount file systems with read-write data or file systems from which executables are run. Writable data could be corrupted if the application ignores the errors. Mounted executables might not load properly and can fail.

NFS Troubleshooting Procedures

To determine where the NFS service has failed, you need to follow several procedures to isolate the failure. Check for the following items:

- Can the client reach the server?
- Can the client contact the NFS services on the server?
- Are the NFS services running on the server?

In the process of checking these items, it might become apparent that other portions of the network are not functioning, such as the name service or the physical network hardware. The *Solaris Naming Administration Guide* contains debugging procedures for the NIS+ name service. Also, during the process it might become obvious that the problem isn't at the client end (for instance, if you get at least one trouble call from every subnet in your work area). In this case, it is much more timely to assume that the problem is the server or the network hardware near the server, and start the debugging process at the server, not at the client.

▼ How to Check Connectivity on an NFS Client

1. **Check that the NFS server is reachable from the client. On the client, type the following command.**

```
% /usr/sbin/ping bee
bee is alive
```

   If the command reports that the server is alive, remotely check the NFS server (see "How to Check the NFS Server Remotely" on page 579).

2. **If the server is not reachable from the client, make sure that the local name service is running. For NIS+ clients type the following:**

```
% /usr/lib/nis/nisping -u
Last updates for directory eng.acme.com. :
Master server is eng-master.acme.com.
        Last update occurred at Mon Jun  5 11:16:10 1995

Replica server is eng1-replica-58.acme.com.
        Last Update seen was Mon Jun  5 11:16:10 1995
```

3. **If the name service is running, make sure that the client has received the correct host information by typing the following:**

```
% /usr/bin/getent hosts bee
129.144.83.117 bee.eng.acme.com
```

4. **If the host information is correct, but the server is not reachable from the client, run the** `ping` **command from another client.**

   If the command run from a second client fails, see "How to Verify the NFS Service on the Server" on page 581.

5. **If the server is reachable from the second client, use** `ping` **to check connectivity of the first client to other systems on the local net.**

   If this fails, check the networking software configuration on the client (`/etc/netmasks`, `/etc/nsswitch.conf`, and so forth).

6. **If the software is correct, check the networking hardware.**

   Try moving the client onto a second net drop.

▼ How to Check the NFS Server Remotely

1. **Check that the NFS services have started on the NFS server by typing the following command:**

```
% rpcinfo -s bee|egrep 'nfs|mountd'
 100003  3,2    tcp,udp                          nfs     superuser
 100005  3,2,1  ticots,ticotsord,tcp,ticlts,udp  mountd  superuser
```

If the daemons have not been started, see "How to Restart NFS Services" on page 583.

2. Check that the server's `nfsd` processes are responding. On the client, type the following command.

```
% /usr/bin/rpcinfo -u bee nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
```

If the server is running, it prints a list of program and version numbers. Using the `–t` option tests the TCP connection. If this fails, skip to "How to Verify the NFS Service on the Server" on page 581.

3. Check that the server's `mountd` is responding, by typing the following command.

```
% /usr/bin/rpcinfo -u bee mountd
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting
program 100005 version 3 ready and waiting
```

Using the `–t` option tests the TCP connection. If either attempt fails, skip to "How to Verify the NFS Service on the Server" on page 581.

4. Check the local autofs service if it is being used:

```
% cd /net/wasp
```

Choose a `/net` or `/home` mount point that you know should work properly. If this doesn't work, then as root on the client, type the following to restart the autofs service:

```
# /etc/init.d/autofs stop
# /etc/init.d/autofs start
```

5. **Verify that file system is shared as expected on the server.**

```
% /usr/sbin/showmount -e bee
/usr/src          eng
/export/share/man      (everyone)
```

Check the entry on the server and the local mount entry for errors. Also check the name space. In this instance, if the first client is not in the `eng` netgroup, that client would not be able to mount the `/usr/src` file system.

Check all entries that include mounting informtion in all of the local files. The list includes `/etc/vfstab` and all the `/etc/auto_*` files.

▼ How to Verify the NFS Service on the Server

1. **Become superuser.**

2. **Check that the server can reach the clients.**

```
# ping lilac
lilac is alive
```

3. **If the client is not reachable from the server, make sure that the local name service is running. For NIS+ clients type the following:**

```
% /usr/lib/nis/nisping -u
Last updates for directory eng.acme.com. :
Master server is eng-master.acme.com.
        Last update occurred at Mon Jun  5 11:16:10 1995

Replica server is eng1-replica-58.acme.com.
        Last Update seen was Mon Jun  5 11:16:10 1995
```

4. **If the name service is running, check the networking software configuration on the server (**`/etc/netmasks`**,** `/etc/nsswitch.conf`**, and so forth).**

5. **Type the following command to check whether the** `nfsd` **daemon is running.**

```
# rpcinfo -u localhost nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
# ps -ef | grep nfsd
root    232      1  0  Apr 07     ?     0:01 /usr/lib/nfs/nfsd -a 16
root   3127   2462  1  09:32:57  pts/3  0:00 grep nfsd
```

Also use the –`t` option with `rpcinfo` to check the TCP connection. If these commands fail, restart the NFS service (see “How to Restart NFS Services” on page 583).

6. **Type the following command to check whether the** `mountd` **daemon is running.**

```
# /usr/bin/rpcinfo -u localhost mountd
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting
program 100005 version 3 ready and waiting
# ps -ef | grep mountd
root    145      1 0 Apr 07  ?     21:57 /usr/lib/autofs/automountd
root    234      1 0 Apr 07  ?     0:04  /usr/lib/nfs/mountd
root   3084 2462 1 09:30:20 pts/3  0:00  grep mountd
```

Also use the –`t` option with `rpcinfo` to check the TCP connection. If these commands fail, restart the NFS service (see “How to Restart NFS Services” on page 583).

7. **Type the following command to check whether the** `rpcbind` **daemon is running.**

```
# /usr/bin/rpcinfo -u localhost rpcbind
program 100000 version 1 ready and waiting
program 100000 version 2 ready and waiting
program 100000 version 3 ready and waiting
```

If `rpcbind` seems to be hung, either reboot the server or follow the steps in "How to Warm-Start `rpcbind`" on page 583.

▼ How to Restart NFS Services

1. **Become superuser.**
2. **To enable daemons without rebooting, type the following commands.**

```
# /etc/init.d/nfs.server stop
# /etc/init.d/nfs.server start
```

This stops the daemons and restarts them, if there is an entry in `/etc/dfs/dfstab`.

▼ How to Warm-Start `rpcbind`

If the NFS server cannot be rebooted because of work in progress, it is possible to restart `rpcbind` without having to restart all of the services that use RPC by completing a warm start as described in this procedure.

1. **Become superuser.**
2. **Determine the PID for `rpcbind`.**

   Run `ps` to get the PID (which is the value in the second column).

```
# ps -ef |grep rpcbind
    root   115     1  0   May 31 ?        0:14 /usr/sbin/rpcbind
    root 13000  6944  0 11:11:15 pts/3    0:00 grep rpcbind
```

3. **Send a SIGTERM signal to the** `rpcbind` **process.**

   In this example, `term` is the signal that is to be sent and `115` is the PID for the program (see the `kill`(1) man page). This causes `rpcbind` to create a list of the current registered services in `/tmp/portmap.file` and `/tmp/rpcbind.file`.

```
# kill -s term 115
```

Note - If you do not kill the `rpcbind` process with the `–s term` option, you cannot complete a warm start of `rpcbind` and must reboot the server to restore service.

4. **Restart** `rpcbind`**.**

   Do a warm restart of the command so that the files created by the `kill` command are consulted, and the process resumes without requiring that all of the RPC services be restarted (see the `rpcbind`(1M) man page).

```
# /usr/sbin/rpcbind -w
```

▼ Identifying Which Host Is Providing NFS File Service

Run the `nfsstat` command with the `–m` option to gather current NFS information. The name of the current server is printed after "`currserver=`".

```
% nfsstat -m
/usr/local from bee,wasp:/export/share/local
 Flags: vers=3,proto=tcp,sec=sys,hard,intr,llock,link,synlink,
  acl,rsize=32768,wsize=32678,retrans=5
 Failover: noresponse=0, failover=0, remap=0, currserver=bee
```

▼ How to Verify Options Used With the `mount` Command

In the Solaris 2.6 release and in any versions of the `mount` command that were patched after the 2.6 release, no warning is issued for invalid options. The following procedure helps determine whether the options that were supplied either on the command line or through `/etc/vfstab` were valid.

For this example, assume that the following command has been run:

```
# mount -F nfs -o ro,vers=2 bee:/export/share/local /mnt
```

1. **Verify the options, by running the following command.**

```
% nfsstat -m
/mnt from bee:/export/share/local
Flags:  vers=2,proto=tcp,sec=sys,hard,intr,dynamic,acl,rsize=8192,wsize=8192,
        retrans=5
```

The file system from `bee` has been mounted with the protocol version set to `2`. Unfortunately, the `nfsstat` command does not display information about all of the options, but using the `nfsstat` command is the most accurate way to verify the options.

2. **Check the entry in `/etc/mnttab`.**

The `mount` command does not allow invalid options to be added to the mount table, so verifying that the options listed in the file match those listed on the command line is a way to check those options not reported by the `nfsstat` command.

```
# grep bee /etc/mnttab
bee:/export/share/local /mnt nfs ro,vers=2,dev=2b0005e 859934818
```

Troubleshooting Autofs

Occasionally, you might encounter problems with autofs. This section should make the problem-solving process easier. It is divided into two subsections.

This section presents a list of the error messages that autofs generates. The list is divided into two parts:

- Error messages generated by the verbose (`–v`) option of `automount`
- Error messages that might appear at any time

Each error message is followed by a description and probable cause of the message.

When troubleshooting, start the autofs programs with the verbose (–v) option; otherwise, you might experience problems without knowing why.

The following paragraphs are labeled with the error message you are likely to see if autofs fails, and a description of the possible problem.

Error Messages Generated by `automount -v`

`bad key` *key* `in direct map` *mapname*

While scanning a direct map, autofs has found an entry key without a prefixed `/`. Keys in direct maps must be full path names.

`bad key` *key* `in indirect map` *mapname*

While scanning an indirect map, autofs has found an entry key containing a `/`. Indirect map keys must be simple names—not path names.

`can't mount` *server*`:`*pathname: reason*

The mount daemon on the server refuses to provide a file handle for *server*:*pathname*. Check the export table on server.

`couldn't create mount point` *mountpoint*`:` *reason*

Autofs was unable to create a mount point required for a mount. This most frequently occurs when attempting to hierarchically mount all of a server's exported file systems. A required mount point can exist only in a file system that cannot be mounted (it cannot be exported) and it cannot be created because the exported parent file system is exported read-only.

`leading space in map entry` *entry* `text in` *mapname*

Autofs has discovered an entry in an automount map that contains leading spaces. This is usually an indication of an improperly continued map entry, for example:

```
fake
/blat     frobz:/usr/frotz
```

In this example, the warning is generated when autofs encounters the second line because the first line should be terminated with a backslash (\).

mapname`: Not found`

The required map cannot be located. This message is produced only when the `-v` option is used. Check the spelling and path name of the map name.

`remount` *server*`:`*pathname* `on` *mountpoint*`: server not responding`

Autofs has failed to remount a file system it previously unmounted.

`WARNING:` *mountpoint* `already mounted on`

Autofs is attempting to mount over an existing mount point. This means an internal error occurred in autofs (an anomaly).

Miscellaneous Error Messages

`dir` *mountpoint* `must start with '/'`

Automounter mount point must be given as full path name. Check the spelling and path name of the mount point.

`hierarchical mountpoints:` *pathname1* `and` *pathname2*

Autofs does not allow its mount points to have a hierarchical relationship. An autofs mount point must not be contained within another automounted file system.

`host` *server* `not responding`

Autofs attempted to contact *server*, but received no response.

hostname`: exports:` *rpc_err*

Error getting export list from *hostname*. This indicates a server or network problem.

`map` *mapname*`, key` *key*`: bad`

The map entry is malformed, and autofs cannot interpret it. Recheck the entry; perhaps the entry has characters that need escaping.

mapname`:` *nis_err*

Error in looking up an entry in a NIS map. This can indicate NIS problems.

`mount of` *server*`:`*pathname* `on` *mountpoint:reason*

Autofs failed to do a mount. This can indicate a server or network problem.

mountpoint`: Not a directory`

Autofs cannot mount itself on *mountpoint* because it is not a directory. Check the spelling and path name of the mount point.

`nfscast: cannot send packet:` *reason*

Autofs cannot send a query packet to a server in a list of replicated file system locations.

`nfscast: cannot receive reply:` *reason*

Autofs cannot receive replies from any of the servers in a list of replicated file system locations.

`nfscast: select:` *reason*

All these error messages indicate problems attempting to `ping` servers for a replicated file system. This can indicate a network problem.

`pathconf: no info for` *server*`:`*pathname*

Autofs failed to get `pathconf` information for path name (see the `fpathconf`(2) man page).

`pathconf:` *server*`: server not responding`

Autofs is unable to contact the mount daemon on *server* that provides the information to `pathconf( )`.

Other Errors With Autofs

If the `/etc/auto*` files have the execute bit set, the automounter tries to execute the maps, which creates messages like:

`/etc/auto_home: +auto_home: not found`

In this case, the `auto_home` file has incorrect permissions. Each entry in the file will generate an error message much like this one. The permissions to the file should be reset by typing the following command:

```
# chmod 644 /etc/auto_home
```

NFS Error Messages

This section shows an error message followed by a description of the conditions that should create the error and at least one way of fixing the problem.

`Bad argument specified with index option - must be a file`

You must include a file name with the –`index` option. You cannot use directory names.

`Cannot establish NFS service over /dev/`*`tcp`*`: transport setup problem`

This message is often created when the services information in the name space has not been updated. It can also be reported for UDP. To fix this problem, you must update the services data in the name space. For NIS+ the entries should be:

```
nfsd nfsd tcp 2049 NFS server daemon
nfsd nfsd ucp 2049 NFS server daemon
```

For NIS and `/etc/services`, the entries should be:

```
nfsd    2049/tcp    nfs    # NFS server daemon
nfsd    2049/ucp    nfs    # NFS server daemon
```

`Cannot use index option without public option`

Include the `public` option with the index option in the share command. You must define the public file handle for the –`index` option to work.

Note - The Solaris 2.5.1 release required that the public file handle be set using the share command. A change in the Solaris 2.6 release sets the public file handle to be `/` by default. This error message is no longer relevant.

`Could not use public filehandle in request to` *`server`*

This message is displayed if the public option is specified but the NFS server does not support the public file handle. In this case, the mount will fail. To remedy this situation, either try the mount request without using the public file handle or reconfigure the NFS server to support the public file handle.

`NOTICE: NFS3: failing over from` *`host1`* `to` *`host2`*

This message is displayed on the console when a failover occurrs. It is an advisory message only.

`filename``: File too large`

An NFS version 2 client is trying to access a file that is over 2 Gbytes.

`mount: ... server not responding:RPC_PMAP_FAILURE - RPC_TIMED_OUT`

The server sharing the file system you are trying to mount is down or unreachable, at the wrong run level, or its `rpcbind` is dead or hung.

`mount: ... server not responding: RPC_PROG_NOT_REGISTERED`

Mount registered with `rpcbind`, but the NFS mount daemon `mountd` is not registered.

`mount: ... No such file or directory`

Either the remote directory or the local directory does not exist. Check the spelling of the directory names. Run `ls` on both directories.

`mount: ...: Permission denied`

Your computer name might not be in the list of clients or netgroup allowed access to the file system you want to mount. Use `showmount -e` to verify the access list.

`nfs mount: ignoring invalid option "-`*option*`"`

The *-option* flag is not valid. Refer to the `mount_nfs`(1M) man page to verify the required syntax.

Note - This error message is not displayed when running any version of the `mount` command included in a Solaris release from 2.6 to the current release or in earlier versions that have been patched.

`nfs mount: NFS can't support "nolargefiles"`

An NFS client has attempted to mount a file system from an NFS server using the –`nolargefiles` option. This option is not supported for NFS file system types.

`nfs mount: NFS V2 can't support "largefiles"`

The NFS version 2 protocol cannot handle large files. You must use version 3 if access to large files is required.

`NFS server` *hostname* `not responding still trying`

If programs hang while doing file-related work, your NFS server might be dead. This message indicates that NFS server *hostname* is down or that a problem has occurred with the server or the network. If failover is being used, *hostname* is a list of servers. Start with "How to Check Connectivity on an NFS Client" on page 578.

`NFS fsstat failed for server` *hostname*`: RPC: Authentication error`

This error can be caused by many situations. One of the most difficult to debug is when this occurs because a user is in too many groups. Currently, a user can be in as many as 16 groups but no more if they are accessing files through NFS mounts.

If a user must have the functionality of being in more than 16 groups and if at least the Solaris 2.5 release is running on the NFS server and the NFS clients, then use ACLs to provide the needed access privileges.

`port` *number* `in nfs URL not the same as port` *number* `in port option`

The port number included in the NFS URL must match the port number included with the `–port` option to mount. If the port numbers do not match, the mount will fail. Either change the command to make the port numbers the same or do not specify the port number that is incorrect. Usually, you do not need to specify the port number both in the NFS URL and with the `–port` option.

`replicas must have the same version`

For NFS failover to function properly, the NFS servers that are replicas must support the same version of the NFS protocol. Mixing version 2 and version 3 servers is not allowed.

`replicated mounts must be read-only`

NFS failover does not work on file systems that are mounted read-write. Mounting the file system read-write increases the likelihood that a file will change. NFS failover depends on the file systems being identical.

`replicated mounts must not be soft`

Replicated mounts require that you wait for a timeout before failover occurs. The `soft` option requires that the mount fail immediately when a timeout starts, so you cannot include the `–soft` option with a replicated mount.

`share_nfs: Cannot share more than one filesystem with 'public' option`

Check that the `/etc/dfs/dfstab` file has only one file system selected to be shared with the `–public` option. Only one public file handle can be established per server, so only one file system per server can be shared with this option.

`WARNING: No network locking on` *hostname*`:`*path*`: contact admin to install server change`

An NFS client has unsuccessfully attempted to establish a connection with the network lock manager on an NFS server. Rather than fail the mount, this warning is generated to warn you that locking will not work.

CHAPTER 31

Accessing Remote File Systems Reference

This chapter provides an introduction to the NFS commands. This chapter also provides information about all of the pieces of the NFS environment and how these pieces work together.

NFS Files

You need several files to support NFS activities on any computer. Many of these files are ASCII, but some of them are data files. Table 31–1 lists these files and their functions.

TABLE 31–1 NFS Files

File Name	Function
`/etc/default/fs`	Lists the default file system type for local file systems.
`/etc/dfs/dfstab`	Lists the local resources to be shared.
`/etc/dfs/fstypes`	Lists the default file-system types for remote file systems.
`/etc/default/nfslogd`	Lists configuration information for the NFS server logging daemon, `nfslogd`.
`/etc/dfs/sharetab`	Lists the resources (local and remote) that are shared (see the `sharetab`(4) man page); do not edit this file.
`/etc/mnttab`	Lists file systems that are currently mounted, including automounted directories (see the `mnttab`(4) man page); do not edit this file.
`/etc/netconfig`	Lists the transport protocols; do not edit this file.
`/etc/nfs/nfslog.conf`	Lists general configuration information for NFS server logging.
`/etc/nfs/nfslogtab`	Lists information for log post-processing by `nfslogd`; do not edit this file.
`/etc/nfssec.conf`	Lists NFS security services; do not edit this file.
`/etc/rmtab`	Lists file systems remotely mounted by NFS clients (see the `rmtab`(4) man page); do not edit this file.
`/etc/vfstab`	Defines file systems to be mounted locally (see the `vfstab`(4) man page).

The first entry in `/etc/dfs/fstypes` is often used as the default file-system type for remote file systems. This entry defines the NFS file-system type as the default.

Only one entry is in `/etc/default/fs`: the default file-system type for local disks. You can determine the file-system types that are supported on a client or server by checking the files in `/kernel/fs`.

/etc/default/nfslogd

This file defines some of the parameters used when using NFS server logging. The following parameters can be defined.

CYCLE_FREQUENCY

Determines the number of hours that must pass before the log files are cycled. The default value is 24 hours. This option is used to prevent the log files from growing too large.

IDLE_TIME

Sets the number of seconds `nfslogd` should sleep before checking for more information in the buffer file. It also determines how often the configuration file is checked. This parameter, along with MIN_PROCESSING_SIZE, determines how often the buffer file is processed. The default value is 300 seconds. Increasing this number can improve preformance by reducing the number of checks.

MAPPING_UPDATE_INTERVAL

Specifies the number of seconds between updates of the records in the file-handle-to-path mapping tables. The default value is 86400 seconds or one day. This parameter helps keep the file-handle-to-path mapping tables up-to-date without having to continually update the tables.

MAX_LOGS_PRESERVE

Determines the number of log files to be saved. The default value is 10.

MIN_PROCESSING_SIZE

Sets the minimum number of bytes that the buffer file must reach before processing and writing to the log file. This parameter, along with IDLE_TIME, determines how often the buffer file is processed. The default value for is 524288 bytes. Increasing this number can improve preformance by reducing the number of times the buffer file is processed.

PRUNE_TIMEOUT

Selects the number of hours that must pass before a file-handle-to-path mapping record times out and can be pruned. The default value is 168 hours or 7 days.

UMASK

Specifies the permissions for the log files that are created by `nfslogd`. The default value is 0137.

/etc/nfs/nfslog.conf

This file defines the path, file names, and type of logging to be used by `nfslogd`. Each definition is associated with a *tag*. Starting NFS server logging requires that you identify the *tag* for each file system. The global tag defines the default values. The following parameters can be used with each tag as needed.

defaultdir=*path*

Specifies the default directory path for the logging files.

log=*path/filename*

Sets the path and file name for the log files.

fhtable=*path/filename*

Selects the path and file name for the file-handle-to-path database files.

buffer=*path/filename*

Determines the path and file name for the buffer files.

logformat=*basic*|*extended*

Selects the format to be used when creating user-readable log files. The basic format produces a log file similar to some `ftpd` daemons. The extended format gives a more detailed view.

For the parameters that can specify both the path and the file name, if the path is not specified, the path defined by `defaultdir` is used. Also, you can override `defaultdir` by using an absolute path.

To make identifying the files easier, place the files in separate directories. Here is an example of the changes needed.

```
% cat /etc/nfs/nfslog.conf
#ident  "@(#)nfslog.conf        1.5     99/02/21 SMI"
#
  .
  .
# NFS server log configuration file.
#

global  defaultdir=/var/nfs \
        log=nfslog fhtable=fhtable buffer=nfslog_workbuffer
```

(continued)

```
publicftp log=logs/nfslog fhtable=fh/fhtables buffer=buffers/workbuffer
```

In this example, any file system shared with `log=publicftp` would use the following values: the default directory would be `/var/nfs`, log files would be stored in `/var/nfs/logs/nfslog*`, file-handle-to-path database tables would be stored in `/var/nfs/fh/fhtables`, and buffer files would be stored in `/var/nfs/buffers/workbuffer`.

NFS Daemons

To support NFS activities, several daemons are started when a system goes into run level 3 or multiuser mode. Two of these daemons (`mountd` and `nfsd`) are run on systems that are NFS servers. The automatic startup of the server daemons depends on the existence of entries labeled with the NFS file-system type in `/etc/dfs/sharetab`.

The other two daemons (`lockd` and `statd`) are run on NFS clients to support NFS file locking. These daemons must also run on the NFS servers.

automountd

This daemon handles the mount and unmount requests from the autofs service. The syntax of the command is:

`automountd [ –Tnv ] [ -D` *name=value* `]`

where `–T` selects to display each RPC call to standard output, `–n` disables browsing on all autofs nodes, `–v` selects to log all status messages to the console, and `–D` *name=value* substitutes *value* for the automount map variable indicated by *name*. The default value for the automount map is `/etc/auto_master`. Use the `–T` option for troubleshooting.

lockd

This daemon supports record-locking operations on NFS files. It sends locking requests from the client to the NFS server. On the NFS server, it starts local locking.

The daemon is normally started without any options. You can use three options with this command (see the `lockd`(1M) man page).

The `–g` *graceperiod* option selects the number of seconds that the clients have to reclaim locks after a server reboot. During this time, the NFS server only processes reclaims of old locks. All other requests for service must wait until the grace period is over. This option affects the NFS server-side response, so it can be changed only on an NFS server. The default value for *graceperiod* is 45 seconds. Reducing this value means that NFS clients can resume operation more quickly after a server reboot, but a reduction increases the chances that a client might not be able to recover all its locks.

The `–t` *timeout* option selects the number of seconds to wait before retransmitting a lock request to the remote server. This option affects the NFS client-side service. The default value for *timeout* is 15 seconds. Decreasing the *timeout* value can improve response time for NFS clients on a noisy network, but it can cause additional server load by increasing the frequency of lock requests.

The *nthreads* option specifies the maximum number of concurrent threads that the server handles per connection. Base the value for *nthreads* on the load expected on the NFS server. The default value is 20. Because each NFS client using TCP uses a single connection with the NFS server, each TCP client is granted the ability to use up to 20 concurrent threads on the server. All NFS clients using UDP share a single connection with the NFS server. Under these conditions it might be necessary to increase the number of threads available for the UDP connection. A minimum calculation would be to allow two threads for each UDP client, but this is specific to the workload on the client, so two threads per client might not be sufficient. The disadvantage to using more threads is that when the threads are used, more memory is used on the NFS server, but if the threads are never used, increasing *nthreads* will have no effect.

`mountd`

This is a remote procedure call (RPC) server that handles file-system mount requests from remote systems and provides access control. It checks `/etc/dfs/sharetab` to determine which file systems are available for remote mounting and which systems are allowed to do the remote mounting. You can use two options with this command (see the `mountd`(1M) man page): `–v` and `–r`.

The `–v` option runs the command in verbose mode. Each time an NFS server determines the access a client should get, a message is printed on the console. The information generated can be useful when trying to determine why a client cannot access a file system.

The `–r` option rejects all future mount requests from clients. This does not affect clients that already have a file system mounted.

nfsd

This daemon handles other client file-system requests. You can use several options with this command. See the `nfsd`(1M) man page for a complete listing.

The `–l` option sets the connection queue length for the NFS/TCP over connection-oriented transports. The default value is 32 entries.

The `–c` *#_conn* option selects the maximum number of connections per connection-oriented transport. The default value for *#_conn* is unlimited.

The *nservers* option is the maximum number of concurrent requests that a server can handle. The default value for *nservers* is 1, but the startup scripts select 16.

Unlike older versions of this daemon, `nfsd` does not spawn multiple copies to handle concurrent requests. Checking the process table with `ps` only shows one copy of the daemon running.

nfslogd

This daemon provides operational logging. NFS operations against a server are logged based on the configuration options defined in `/etc/default/nfslogd`. When NFS server logging is enabled, records of all RPC operations on a selected file system are written into a buffer file by the kernel. Then `nfslogd` post-processes these requests. The name service switch is used to help map UIDs to logins and IP addresses to host names. The number is recorded if no match can be found through the identified name services.

Mapping of file handles to path names is also handled by `nfslogd`. The daemon keeps track of these mappings in a file-handle-to-path mapping table. One mapping table exists for each tag identified in `/etc/nfs/nfslogd`. After post-processing, the records are written out to ASCII log files.

statd

This daemon works with `lockd` to provide crash and recovery functions for the lock manager. It tracks the clients that hold locks on an NFS server. If a server crashes, on rebooting `statd` on the server contacts `statd` on the client. The client `statd` can then attempt to reclaim any locks on the server. The client `statd` also informs the server `statd` when a client has crashed, so that the client's locks on the server can be cleared. There are no options to select with this daemon. For more information see the `statd`(1M) man page.

In the Solaris 7 release, the way that `statd` keeps track of the clients has been improved. In all earlier Solaris releases, `statd` created files in `/var/statmon/sm` for each client using the client's unqualified host name. This caused problems if you

had two clients in different domains that shared a hos tname, or if there were clients that were not resident in the same domain as the NFS server. Because the unqualified host name only lists the host name, without any domain or IP-address information, the older version of `statd` had no way to differentiate between these types of clients. To fix this problem, the Solaris 7 `statd` creates a symbolic link in `/var/statmon/sm` to the unqualified host name using the IP address of the client. The new link will look like:

```
# ls -l /var/statmon/sm
lrwxrwxrwx   1 root          11 Apr 29 16:32 ipv4.192.9.200.1 -> myhost
--w-------   1 root          11 Apr 29 16:32 myhost
```

In this example, the client host name is `myhost` and the clients' IP address is `192.9.200.1`. If another host with the name `myhost` were mounting a file system, there would be two symbolic links to the host name.

NFS Commands

These commands must be run as root to be fully effective, but requests for information can be made by all users:

automount

This command installs autofs mount points and associates the information in the automaster files with each mount point. The syntax of the command is:

`automount [ –t` *duration* `][ –v ]`

where `–t` *duration* sets the time, in seconds, that a file system is to remain mounted, and `–v` selects the verbose mode. Running this command in the verbose mode allows for easier troubleshooting.

If not specifically set, the value for duration is set to 5 minutes. In most circumstances this is a good value; however, on systems that have many automounted file systems, you might need to increase the duration value. In particular, if a server has many users active, checking the automounted file systems every 5 minutes can be inefficient. Checking the autofs file systems every 1800 seconds (or 30 minutes) could be more optimal. By not unmounting the file systems every 5 minutes, it is possible that `/etc/mnttab`, which is checked by `df`, can become large. The output from `df` can be filtered by using the `–F` option (see the `df`(1M) man page) or by using `egrep` to help fix this problem.

Another factor to consider is that adjusting the duration also changes how quickly changes to the automounter maps will be reflected. Changes will not be seen until the file system is unmounted. Refer to “Modifying the Maps” on page 564 for instructions on how to modify automounter maps.

clear_locks

This command enables you to remove all file, record, and share locks for an NFS client. You must be `root` to run this command. From an NFS server you can clear the locks for a specific client and from an NFS client you can clear locks for that client on a specific server. The following example would clear the locks for the NFS client named `tulip` on the current system.

```
# clear_locks tulip
```

Using the `–s` option enables you to specify which NFS host to clear the locks from. It must be run from the NFS client, which created the locks. In this case, the locks from the client would be removed from the NFS server named `bee`.

```
# clear_locks -s bee
```

Caution - This command should only be run when a client crashes and cannot clear its locks. To avoid data corruption problems, do not clear locks for an active client.

mount

With this command, you can attach a named file system, either local or remote, to a specified mount point. For more information, see the mount(1M) man page. Used without arguments, mount displays a list of file systems that are currently mounted on your computer.

Many types of file systems are included in the standard Solaris installation. Each file-system type has a specific man page that lists the options to mount that are appropriate for that file-system type. The man page for NFS file systems is mount_nfs(1M); for UFS file systems it is mount_ufs(1M); and so forth.

The Solaris 7 release includes the ability to select a path name to mount from an NFS server using an NFS URL instead of the standard server:/pathname syntax. See "How to Mount an NFS File System Using an NFS URL" on page 553 for further information.

Caution - The version of the mount command included in any Solaris release from 2.6 to the current release, will not warn about options that are not valid. The command silently ignores any options that cannot be interpreted. Make sure to verify all of the options that were used to prevent unexpected behavior.

mount Options for NFS File Systems

The subsequent text lists some of the options that can follow the –o flag when mounting an NFS file system.

bg | fg

These options can be used to select the retry behavior if a mount fails. The –bg option causes the mount attempts to be run in the background. The –fg option causes the mount attempt to be run in the foreground. The default is –fg, which is the best selection for file systems that must be available. It prevents further processing until the mount is complete. –bg is a good selection for file systems that are not critical, because the client can do other processing while waiting for the mount request to complete.

forcedirectio

This option improves performance of sequential reads on large files. Data is copied directly to a user buffer and no caching is done in the kernel on the client. This option is off by default.

largefiles

This option makes it possible to access files larger than 2 Gbytes on a server running the Solaris 2.6 release. Whether a large file can be accessed can only be controlled on the server, so this option is silently ignored on NFS version 3 mounts. Starting with release 2.6, by default, all UFS file systems are mounted with –`largefiles`. For mounts using the NFS version 2 protocol, the –`largefiles` option causes the mount to fail with an error.

nolargefiles

This option for UFS mounts guarantees that there are and will be no large files on the file system (see the `mount_ufs`(1M) man page). Because the existence of large files can only be controlled on the NFS server, there is no option for –`nolargefiles` using NFS mounts. Attempts to NFS mount a file system using this option are rejected with an error.

public

This option forces the use of the public file handle when contacting the NFS server. If the public file handle is supported by the server, the mounting operation is faster because the MOUNT protocol is not used. Also, because the MOUNT protocol is not used, the public option allows mounting to occur through a firewall.

rw | ro

The –`rw` and –`ro` options indicate whether a file system is to be mounted read-write or read-only. The default is read-write, which is the appropriate option for remote home directories, mail-spooling directories, or other file systems that need to be changed by users. The read-only option is appropriate for directories that should not be changed by users; for example, shared copies of the man pages should not be writable by users.

sec=*mode*

You can use this option to specify the authentication mechanism to be used during the mount transaction. The value for *mode* can be one of the values shown in Table 31–2. The modes are also defined in `/etc/nfssec.conf`.

TABLE 31–2 NFS Security Modes

Mode	Authentication Service Selected
krb5	Kerberos Version 5
none	No authentication
dh	Diffie-Hellman (DH) authentication
sys	Standard UNIX authentication

soft | hard

An NFS file system mounted with the `soft` option returns an error if the server does not respond. The `hard` option causes the mount to continue to retry until the server responds. The default is `hard`, which should be used for most file systems. Applications frequently do not check return values from `soft`-mounted file systems, which can make the application fail or can lead to corrupted files. Even if the application does check, routing problems and other conditions can still confuse the application or lead to file corruption if the `soft` option is used. In most cases the `soft` option should not be used. If a file system is mounted using the `hard` option and becomes unavailable, an application using this file system will hang until the file system becomes available.

Using the `mount` Command

Both of these commands mount an NFS file system from the server `bee` read-only:

```
# mount -F nfs -r bee:/export/share/man /usr/man
```

```
# mount -F nfs -o ro bee:/export/share/man /usr/man
```

This command uses the `–O` option to force the man pages from the server `bee` to be mounted on the local system even if `/usr/man` has already been mounted on:

```
# mount -F nfs -O bee:/export/share/man /usr/man
```

This command uses client failover:

```
# mount -F nfs -r bee,wasp:/export/share/man /usr/man
```

Note - When used from the command line, the listed servers must support the same version of the NFS protocol. Do not mix version 2 and version 3 servers when running `mount` from the command line. You can use mixed servers with autofs, in which case the best subset of version 2 or version 3 servers is used.

Here is an example of using an NFS URL with the `mount` command:

```
# mount -F nfs nfs://bee//export/share/man /usr/man
```

Use the `mount` command with no arguments to display file systems mounted on a client.

```
% mount
/ on /dev/dsk/c0t3d0s0 read/write/setuid on Tues Jan 24 13:20:47 1995
/usr on /dev/dsk/c0t3d0s6 read/write/setuid on Tues Jan 24 13:20:47 1995
/proc on /proc read/write/setuid on Tues Jan 24 13:20:47 1995
/dev/fd on fd read/write/setuid on Tues Jan 24 13:20:47 1995
/tmp on swap read/write on Tues Jan 24 13:20:51 1995
/opt on /dev/dsk/c0t3d0s5 setuid/read/write on Tues Jan 24 13:20:51 1995
/home/kathys on bee:/export/home/bee7/kathys
  intr/noquota/nosuid/remote on Tues Jan 24 13:22:13 1995
```

umount

This command enables you to remove a remote file system that is currently mounted. The `umount` command supports the –`V` option to allow for testing. You might also use the –`a` option to umount several file systems at one time. If *mount_points* are included with the –`a` option, those file systems are unmounted. If no mount points are included, an attempt is made to unmount all file systems listed in `/etc/mnttab`, except for the “required” file systems, such as `/`, `/usr`, `/var`, `/proc`, `/dev/fd`, and `/tmp`.

Because the file system is already mounted and should have an entry in `/etc/mnttab`, you do not need to include a flag for the file-system type.

The command cannot succeed if the file system is in use. For instance, if a user has used `cd` to get access to a file system, the file system is busy until the working directory is changed. The `umount` command can hang temporarily if the NFS server is unreachable.

Using the `umount` Command

This example unmounts a file system mounted on `/usr/man`:

```
# umount /usr/man
```

This example displays the results of running `umount –a -V`:

```
# umount -a -V
umount /home/kathys
umount /opt
umount /home
umount /net
```

Notice that this command does not actually unmount the file systems.

mountall

Use this command to mount all file systems or a specific group of file systems listed in a file-system table. The command provides a way to select the file-system type to be accessed with the `–F` *FSType* option, to select all the remote file systems listed in a file-system table with the `–r` option, and to select all the local file systems with the `–l` option. Because all file systems labeled as NFS file-system type are remote file systems, some of these options are redundant. For more information, see the `mountall`(1M) man page.

Using the `mountall` Command

These two examples are equivalent:

```
# mountall -F nfs
```

```
# mountall -F nfs -r
```

umountall

Use this command to unmount a group of file systems. The `–k` option runs the `fuser –k` *mount_point* command to kill any processes associated with the *mount_point*. The `–s` option indicates that unmount is not to be performed in parallel. `–l` specifies that only local file systems are to be used, and `–r` specifies that only remote file systems are to be used. The `–h` *host* option indicates that all file systems from the named host should be unmounted. You cannot combine the `–h` option with `–l` or `–r`.

Using the `umountall` Command

This command unmounts all file systems that are mounted from remote hosts:

```
# umountall -r
```

This command unmounts all file systems currently mounted from the server bee:

```
# umountall -h bee
```

share

With this command, you can make a local file system on an NFS server available for mounting. You can also use the share command to display a list of the file systems on your system that are currently shared. The NFS server must be running for the share command to work. The NFS server software is started automatically during boot if there is an entry in /etc/dfs/dfstab. The command does not report an error if the NFS server software is not running, so you must check this yourself.

The objects that can be shared include any directory tree, but each file system hierarchy is limited by the disk slice or partition that the file system is located on. For instance, sharing the root (/) file system would not also share /usr, unless they are on the same disk partition or slice. Normal installation places root on slice 0 and /usr on slice 6. Also, sharing /usr would not share any other local disk partitions that are mounted on subdirectories of /usr.

A file system cannot be shared that is part of a larger file system already being shared. For example, if /usr and /usr/local are on one disk slice, /usr can be shared or /usr/local can be shared, but if both need to be shared with different share options, /usr/local must to be moved to a separate disk slice.

Note - You can gain access to a file system that is shared read-only through the file handle of a file system that is shared read-write if the two file systems are on the same disk slice. It is more secure to place those file systems that need to be read-write on a separate partition or disk slice than the file systems that you need to share read-only.

Non-file System Specific share Options

Some of the options that you can include with the –o flag are as follows.

rw|ro

The *pathname* file system is shared read-write or read-only to all clients.

rw=*accesslist*

The file system is shared read-write to the listed clients only. All other requests are denied. Starting with the Solaris 2.6 release, the list of clients defined in *accesslist* has been expanded. See “Setting Access Lists With the share Command” on page 610 for more information. You can use this option to override an –ro option.

NFS Specific `share` Options

The options that you can use with NFS file systems include the following.

aclok

This option enables an NFS server supporting the NFS version 2 protocol to be configured to do access control for NFS version 2 clients. Without this option all clients are given minimal access. With this option the clients have maximal access. For instance, on file systems shared with the –`aclok` option, if anyone has read permissions, everyone does. However, without this option, you can deny access to a client who should have access permissions. Whether it is preferred to permit too much access or too little, depends on the security systems already in place. See "Securing Files (Tasks)" in *System Administration Guide, Volume 2* for more information about access control lists (ACLs).

Note - To take advantage of ACLs, it is best to have clients and servers run software that supports the NFS version 3 and NFS_ACL protocols. If the software only supports the NFS version 3 protocol, clients get correct access, but cannot manipulate the ACLs. If the software supports the NFS_ACL protocol, the clients get correct access and can manipulate the ACLs. Starting with release 2.5, the Solaris system supports both protocols.

---

anon=*uid*

You use *uid* to select the user ID of unauthenticated users. If you set *uid* to `-1`, the server denies access to unauthenticated users. You can grant root access by setting `anon=0`, but this will allow unauthenticated users to have root access, so use the `root` option instead.

index=*filename*

You can use the –`index=`*filename* option to force the loading of a HyperText Markup Language (HTML) file instead of displaying a listing of the directory when a user accesses an NFS URL. This option mimics the action of current browsers if an `index.html` file is found in the directory that the HTTP URL is accessing. This is the equivalent of setting the `DirectoryIndex` option for `httpd`. For instance, if the `dfstab` file entry looks like:

```
share -F nfs -o ro,public,index=index.html /export/web
```

these URLs will display the same information:

```
nfs://<server>/<dir>
nfs://<server>/<dir>/index.html
nfs://<server>//export/web/<dir>
```

(continued)

```
nfs://<server>//export/web/<dir>/index.html
http://<server>/<dir>
http://<server>/<dir>/index.html
```

log=*tag*

This option specifies the tag in `/etc/nfs/nfslog.conf` that contains the NFS server logging configuration information for a file system. This option must be selected to enable NFS server logging.

nosuid

This option signals that all attempts to enable the `setuid` or `setgid` mode should be ignored. NFS clients cannot be able to create files with the `setuid` or `setgid` bits on.

public

The `–public` option has been added to the `share` command to enable WebNFS browsing. Only one file system on a server can be shared with this option.

root=*accesslist*

The server gives root access to the hosts in the list. By default, the server does not give root access to any remote hosts. If the selected security mode is anything other than `–sec=sys`, you can only include client host names in the *accesslist*. Starting with the Solaris 2.6 release, the list of clients defined in *accesslist* is expanded. See “Setting Access Lists With the `share` Command” on page 610 for more information.

Caution - Granting root access to other hosts has far-reaching security implications; use the `–root=` option with extreme caution.

sec=*mode*[:*mode*]

mode selects the security modes that are needed to get access to the file system. By default, the security mode is UNIX authentication. You can specify multiple modes, but use each security mode only once per command line. Each `–mode` option applies to any subsequent `–rw`, `–ro`, `–rw=`, `–ro=`, `–root=`, and `–window=` options, until another `–mode` is encountered. Using `–sec=none` maps all users to user `nobody`.

window=*value*

value selects the maximum life time in seconds of a credential on the NFS server. The default value is 30000 seconds or 8.3 hours.

Setting Access Lists With the `share` Command

In Solaris releases prior to 2.6, the *accesslist* included with either the –`ro=`, –`rw=`, or –`root=` option of the `share` command were restricted to a list of host names or netgroup names. Starting with the Solaris 2.6 release, the access list can also include a domain name, a subnet number, or an entry to deny access. These extensions should make it easier to control file access control on a single server, without having to change the name space or maintain long lists of clients.

This command provides read-only access for most systems but allows read-write access for `rose` and `lilac`:

```
# share -F nfs -o ro,rw=rose:lilac /usr/src
```

In the next example, read-only access is assigned to any host in the `eng` netgroup. The client `rose` is specifically given read-write access.

```
# share -F nfs -o ro=eng,rw=rose /usr/src
```

Note - You cannot specify both `rw` and `ro` without arguments. If no read-write option is specified, the default is read-write for all clients.

To share one file system with multiple clients, you must enter all options on the same line, because multiple invocations of the `share` command on the same object "remember" only the last command run. This command enables read-write access to three client systems, but only `rose` and `tulip` are given access to the file system as `root`.

```
# share -F nfs -o rw=rose:lilac:tulip,root=rose:tulip /usr/src
```

When sharing a file system using multiple authentication mechanisms, make sure to include the –`ro`, –`ro=`, –`rw`, –`rw=`, –`root`, and –`window` options after the correct security modes. In this example, UNIX authentication is selected for all hosts in the netgroup named `eng`. These hosts can only mount the file system in read-only mode. The hosts `tulip` and `lilac` can mount the file system read-write if they use Diffie-Hellman authentication. With these options, `tulip` and `lilac`can mount the file system read-only even if they are not using DH authentication, if the host names are listed in the `eng` netgroup.

```
# share -F nfs -o sec=dh,rw=tulip:lilac,sec=sys,ro=eng /usr/src
```

Even though UNIX authentication is the default security mode, it is not included if the –sec option is used, so it is important to include a –sec=sys option if UNIX authentication is to be used with any other authentication mechanism.

You can use a DNS domain name in the access list by preceding the actual domain name with a dot. The dot indicates that the string following it is a domain name, not a fully qualified host name. The following entry allows mount access to all hosts in the eng.sun.com domain:

```
# share -F nfs -o ro=.:.eng.sun.com /export/share/man
```

In this example, the single "." matches all hosts that are matched through the NIS or NIS+ name spaces. The results returned from these name services do not include the domain name. The ".eng.sun.com" entry matches all hosts that use DNS for name space resolution. DNS always returns a fully qualified host name, so the longer entry is required if you use a combination of DNS and the other name spaces.

You can use a subnet number in an access list by preceding the actual network number or the network name with "@". This differentiates the network name from a netgroup or a fully qualified host name. You must identify the subnet in either /etc/networks or in a NIS or NIS+ name space. The following entries have the same effect if the 129.144 subnet has been identified as the eng network:

```
# share -F nfs -o ro=@eng /export/share/man
# share -F nfs -o ro=@129.144 /export/share/man
# share -F nfs -o ro=@129.144.0.0 /export/share/man
```

The last two entries show you do not need to include the full network address.

If the network prefix is not byte aligned, as with Classless Inter-Domain Routing (CIDR), the mask length can be explicitly specified on the command line. The mask length is defined by following either the network name or the network number with a slash and the number of significant bits in the prefix of the address. For example:

```
# share -f nfs -o ro=@eng/17 /export/share/man
# share -F nfs -o ro=@129.144.132/17 /export/share/man
```

In these examples, the "/17" indicates that the first 17 bits in the address are to be used as the mask. For additional information on CIDR, look up RFC 1519.

You can also select negative access by placing a "-" before the entry. Because the entries are read from left to right, you must place the negative access entries before the entry they apply to:

```
# share -F nfs -o ro=-rose:.eng.sun.com /export/share/man
```

This example would allow access to any hosts in the `eng.sun.com` domain except the host named `rose`.

unshare

This command allows you to make a previously available file system unavailable for mounting by clients. You can use the `unshare` command to unshare any file system—whether the file system was shared explicitly with the `share` command or automatically through `/etc/dfs/dfstab`. If you use the `unshare` command to unshare a file system that you shared through the `dfstab` file, remember that it will be shared again when you exit and reenter run level 3. You must remove the entry for this file system from the `dfstab` file if the change is to continue.

When you unshare an NFS file system, access from clients with existing mounts is inhibited. The file system might still be mounted on the client, but the files will not be accessible.

Using the unshare Command

This command unshares a specific file system:

```
# unshare /usr/src
```

shareall

This command allows for multiple file systems to be shared. When used with no options, the command shares all entries in `/etc/dfs/dfstab`. You can include a file name to specify the name of a file that lists `share` command lines. If you do not include a file name, `/etc/dfs/dfstab` is checked. If you use a "`-`" to replace the file name, you can type `share` commands from standard input.

Using the shareall Command

This command shares all file systems listed in a local file:

```
# shareall /etc/dfs/special_dfstab
```

unshareall

This command makes all currently shared resources unavailable. The `–F` *FSType* option selects a list of file-system types defined in `/etc/dfs/fstypes`. This flag enables you to choose only certain types of file systems to be unshared. The default file system type is defined in `/etc/dfs/fstypes`. To choose specific file systems, use the `unshare` command.

Using the unshareall Command

This example should unshare all NFS type file systems:

```
# unshareall -F nfs
```

showmount

This command displays all clients that have remotely mounted file systems that are shared from an NFS server, or only the file systems that are mounted by clients, or the shared file systems with the client access information. The command syntax is:

showmount [–ade][*hostname*]

where –a prints a list of all the remote mounts (each entry includes the client name and the directory), –d prints a list of the directories that are remotely mounted by clients, –e prints a list of the files shared (or exported), and *hostname* selects the NFS server to gather the information from. If *hostname* is not specified the local host is queried.

Using the showmount Command

This command lists all clients and the local directories that they have mounted.

```
# showmount -a bee
lilac:/export/share/man
lilac:/usr/src
rose:/usr/src
tulip:/export/share/man
```

This command lists the directories that have been mounted.

```
# showmount -d bee
/export/share/man
/usr/src
```

This command lists file systems that have been shared.

```
# showmount -e bee
/usr/src        (everyone)
/export/share/man     eng
```

setmnt

This command creates an /etc/mnttab table. The mount and umount commands consult the table. Generally, there is no reason to run this command manually; it runs automatically when a system is booted.

Other Useful Commands

These commands can be useful when troubleshooting NFS problems.

nfsstat

You can use this command to gather statistical information about NFS and RPC connections. The syntax of the command is:

nfsstat [–cmnrsz]

where –c displays client-side information, –m displays statistics for each NFS mounted file system, –n specifies that NFS information is to be displayed (both client and server side), –r displays RPC statistics, –s displays the server-side information, and –z specifies that the statistics should be set to zero. If no options are supplied on the command line, the –cnrs options are used.

Gathering server-side statistics can be important for debugging problems when new software or hardware is added to the computing environment. Running this command at least once a week, and storing the numbers, provides a good history of previous performance.

Using the nfsstat Command

```
# nfsstat -s

Server rpc:
Connection oriented:
calls      badcalls   nullrecv   badlen      xdrcall    dupchecks  dupreqs
11420263   0          0          0           0          1428274    19
Connectionless:
calls      badcalls   nullrecv   badlen      xdrcall    dupchecks  dupreqs
14569706   0          0          0           0          953332     1601

Server nfs:
calls      badcalls
24234967   226
Version 2: (13073528 calls)
```

(continued)

(Continuation)

```
null       getattr    setattr    root       lookup     readlink   read
138612 1%  1192059 9% 45676 0%   0 0%       9300029 71% 9872 0%   1319897 10%
wrcache    write      create     remove     rename     link       symlink
0 0%       805444 6%  43417 0%   44951 0%   3831 0%    4758 0%    1490 0%
mkdir      rmdir      readdir    statfs
2235 0%    1518 0%    51897 0%   107842 0%
Version 3: (11114810 calls)
null       getattr    setattr    lookup     access     readlink   read
141059 1%  3911728 35% 181185 1% 3395029 30% 1097018 9% 4777 0%   960503 8%
write      create     mkdir      symlink    mknod      remove     rmdir
763996 6%  159257 1%  3997 0%    10532 0%   26 0%      164698 1%  2251 0%
rename     link       readdir    readdirplus fsstat    fsinfo     pathconf
53303 0%   9500 0%    62022 0%   79512 0%   3442 0%    34275 0%   3023 0%
commit
73677 0%

Server nfs_acl:
Version 2: (1579 calls)
null       getacl     setacl     getattr    access
0 0%       3 0%       0 0%       1000 63%   576 36%
Version 3: (45318 calls)
null       getacl     setacl
0 0%       45318 100% 0 0%
```

This is an example of NFS server statistics. The first five lines deal with RPC and the remaining lines report NFS activities. In both sets of statistics, knowing the average number of `badcalls` or `calls` and the number of calls per week, can help identify when something is going wrong. The `badcalls` value reports the number of bad messages from a client and can point out network hardware problems.

Some of the connections generate write activity on the disks. A sudden increase in these statistics could indicate trouble and should be investigated. For NFS version 2 statistics, the connections to note are: `setattr`, `write`, `create`, `remove`, `rename`, `link`, `symlink`, `mkdir`, and `rmdir`. For NFS version 3 statistics, the value to watch is `commit`. If the `commit` level is high in one NFS server as compared to another almost identical one, check that the NFS clients have enough memory. The number of `commit` operations on the server grow when clients do not have available resources.

pstack

This command displays a stack trace for each process. It must be run by `root`. You can use it to determine where a process is hung. The only option allowed with this command is the `PID` of the process that you want to check (see the `proc`(1) man page).

The following example is checking the `nfsd` process that is running.

```
# /usr/proc/bin/pstack 243
243:    /usr/lib/nfs/nfsd -a 16
 ef675c04 poll     (24d50, 2, ffffffff)
 000115dc ???????? (24000, 132c4, 276d8, 1329c, 276d8, 0)
 00011390 main     (3, effffff14, 0, 0, ffffffff, 400) + 3c8
 00010fb0 _start   (0, 0, 0, 0, 0, 0) + 5c
```

It shows that the process is waiting for a new connection request. This is a normal response. If the stack shows that the process is still in poll after a request is made, the process might be hung. Follow the instructions in "How to Restart NFS Services" on page 583 to fix this problem. Review the instructions in "NFS Troubleshooting Procedures" on page 578 to fully verify that your problem is a hung program.

rpcinfo

This command generates information about the RPC service running on a system. You can also use it to change the RPC service. Many options are available with this command (see the `rpcinfo`(1M) man page). This is a shortened synopsis for some of the options that you can use with the command:

`rpcinfo [ –m | –s ][` *hostname* `]`

`rpcinfo [ –t | –u ][` *hostname* `][` *progname* `]`

where `–m` displays a table of statistics of the `rpcbind` operations, `–s` displays a concise list of all registered RPC programs, `–t` displays the RPC programs that use TCP, `–u` displays the RPC programs that use UDP, *hostname* selects the host name of the server you need information from, and *progname* selects the RPC program to gather information about. If no value is given for *hostname*, the local host name is used. You can substitute the RPC program number for *progname*, but many users will remember the name and not the number. You can use the `–p` option in place of the `–s` option on those systems that do not run the NFS version 3 software.

The data generated by this command can include:

- The RPC program number
- The version number for a specific program
- The transport protocol that is being used
- The name of the RPC service
- The owner of the RPC service

This example gathers information on the RPC services running on a server. The text generated by the command is filtered by the sort command to make it more readable. Several lines listing RPC services have been deleted from the example.

```
% rpcinfo -s bee |sort -n
   program version(s) netid(s)                          service     owner
    100000  2,3,4     udp,tcp,ticlts,ticotsord,ticots  portmapper  superuser
    100001  4,3,2     ticlts,udp                       rstatd      superuser
    100002  3,2       ticots,ticotsord,tcp,ticlts,udp  rusersd     superuser
    100003  3,2       tcp,udp                          nfs         superuser
    100005  3,2,1     ticots,ticotsord,tcp,ticlts,udp  mountd      superuser
    100008  1         ticlts,udp                       walld       superuser
    100011  1         ticlts,udp                       rquotad     superuser
    100012  1         ticlts,udp                       sprayd      superuser
    100021  4,3,2,1   ticots,ticotsord,ticlts,tcp,udp  nlockmgr    superuser
    100024  1         ticots,ticotsord,ticlts,tcp,udp  status      superuser
    100026  1         ticots,ticotsord,ticlts,tcp,udp  bootparam   superuser
    100029  2,1       ticots,ticotsord,ticlts          keyserv     superuser
    100068  4,3,2     tcp,udp                          cmsd        superuser
    100078  4         ticots,ticotsord,ticlts          kerbd       superuser
    100083  1         tcp,udp                          -           superuser
    100087  11        udp                              adm_agent   superuser
    100088  1         udp,tcp                          -           superuser
    100089  1         tcp                              -           superuser
    100099  1         ticots,ticotsord,ticlts          pld         superuser
    100101  10        tcp,udp                          event       superuser
    100104  10        udp                              sync        superuser
    100105  10        udp                              diskinfo    superuser
    100107  10        udp                              hostperf    superuser
    100109  10        udp                              activity    superuser
  .
  .
    100227  3,2       tcp,udp                          -           superuser
    100301  1         ticlts                           niscachemgr superuser
    390100  3         udp                              -           superuser
1342177279  1,2       tcp                              -           14072
```

This example shows how to gather information about a particular RPC service using a particular transport on a server.

```
% rpcinfo -t bee mountd
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting
program 100005 version 3 ready and waiting
% rpcinfo -u bee nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
```

The first example checks the mountd service running over TCP. The second example checks the NFS service running over UDP.

snoop

This command is often used to watch for packets on the network. It must be run as `root`. It is a good way to make sure that the network hardware is functioning on both the client and the server. Many options are available (see the `snoop`(1M) man page). A shortened synopsis of the command follows:

`snoop [ -d` *device* `][ –o` *filename* `][ host` *hostname* `]`

where `–d` *device* specifies the local network interface, `–o` *filename* stores all the captured packets into the named file, and *hostname* indicates to display only packets going to and from a specific host.

The `–d` *device* option is useful on those servers that have multiple network interfaces. You can use many other expressions besides setting the host. A combination of command expressions with `grep` can often generate data that is specific enough to be useful.

When troubleshooting, make sure that packets are going to and from the proper host. Also, look for error messages. Saving the packets to a file can make it much easier to review the data.

truss

You can use this command to see if a process is hung. It must be run by `root`. You can use many options with this command (see the `truss`(1) man page). A shortened syntax of the command is:

`truss [ -t` *syscall* `]–p` *pid*

where `–t` *syscall* selects system calls to trace, and `–p` *pid* indicates the PID of the process to be traced. The *syscall* may be a comma-separated list of system calls to be traced. Also, starting *syscall* with a `!` selects to exclude the listed system calls from the trace.

This example shows that the process is waiting for another connection request from a new client.

```
# /usr/bin/truss -p 243
poll(0x00024D50, 2, -1)         (sleeping...)
```

This is a normal response. If the response does not change after a new connection request has been made, the process could be hung. Follow the instructions in “How to Restart NFS Services” on page 583 to fix the hung program. Review the instructions in “NFS Troubleshooting Procedures” on page 578 to fully verify that your problem is a hung program.

How It All Works Together

The following sections describe some of the complex functions of the NFS software.

Version 2 and Version 3 Negotiation

Because NFS servers might be supporting clients that are not using the NFS version 3 software, part of the initiation procedure includes negotiation of the protocol level. If both the client and the server can support version 3, that version will be used. If either the client or the server can only support version 2, that version will be selected.

You can override the values determined by the negotiation by using the `–vers` option to the `mount` command (see the `mount_nfs`(1M) man page). Under most circumstances, you should not have to specify the version level, as the best one is selected by default.

UDP and TCP Negotiation

During initiation, the transport protocol is also negotiated. By default, the first connection-oriented transport supported on both the client and the server is selected. If this does not succeed, the first available connectionless transport protocol is used. The transport protocols supported on a system are listed in `/etc/netconfig`. TCP is the connection-oriented transport protocol supported by the release. UDP is the connectionless transport protocol.

When both the NFS protocol version and the transport protocol are determined by negotiation, the NFS protocol version is given precedence over the transport protocol. The NFS version 3 protocol using UDP is given higher precedence than the NFS version 2 protocol using TCP. You can manually select both the NFS protocol version and the transport protocol with the `mount` command (see the `mount_nfs`(1M) man page). Under most conditions, allow the negotiation to select the best options.

File Transfer Size Negotiation

The file transfer size establishes the size of the buffers that are used when transferring data between the client and the server. In general, larger transfer sizes are better. The NFS version 3 protocol has an unlimited transfer size, but starting with the Solaris 2.6 release, the software bids a default buffer size of 32 Kbytes. The client can bid a smaller transfer size at mount time if needed, but under most conditions this is not necessary.

The transfer size is not negotiated with systems using the NFS version 2 protocol. Under this condition the maximum transfer size is set to 8 Kbytes.

You can use the –rsize and –wsize options to set the transfer size manually with the mount command. You might need to reduce the transfer size for some PC clients. Also, you can increase the transfer size if the NFS server is configured to use larger transfer sizes.

How File Systems Are Mounted

When a client needs to mount a file system from a server, it must obtain a file handle from the server that corresponds to the file system. This process requires that several transactions occur between the client and the server. In this example, the client is attempting to mount /home/terry from the server. A snoop trace for this transaction follows.

```
client -> server PORTMAP C GETPORT prog=100005 (MOUNT) vers=3 proto=UDP
server -> client PORTMAP R GETPORT port=33492
client -> server MOUNT3 C Null
server -> client MOUNT3 R Null
client -> server MOUNT3 C Mount /export/home9/terry
server -> client MOUNT3 R Mount OK FH=9000 Auth=unix
client -> server PORTMAP C GETPORT prog=100003 (NFS) vers=3 proto=TCP
server -> client PORTMAP R GETPORT port=2049
client -> server NFS C NULL3
server -> client NFS R NULL3
client -> server NFS C FSINFO3 FH=9000
server -> client NFS R FSINFO3 OK
client -> server NFS C GETATTR3 FH=9000
server -> client NFS R GETATTR3 OK
```

In this trace, the client first requests the mount port number from the portmap service on the NFS server. After the client received the mount port number (33492), that number is used to ping the service on the server. After the client has determined that a service is running on that port number, the client then makes a mount request. When the server responds to this request, it includes the file handle for the file system (9000) that is being mounted. The client then sends a request for the NFS port number. When the client receives the number from the server, it pings the NFS service (nfsd), and requests NFS information about the file system using the file handle.

In the following trace, the client is mounting the file system with the –public option.

```
client -> server NFS C LOOKUP3 FH=0000 /export/home9/terry
server -> client NFS R LOOKUP3 OK FH=9000
client -> server NFS C FSINFO3 FH=9000
server -> client NFS R FSINFO3 OK
client -> server NFS C GETATTR3 FH=9000
server -> client NFS R GETATTR3 OK
```

By using the default public file handle (which is `0000`), all of the transactions to get information from the portmap service and to determine the NFS port number are skipped.

Effects of the –`public` Option and NFS URLs When Mounting

Using the –`public` option can create conditions that cause a mount to fail. Adding an NFS URL can also confuse the situation. The following list describes the specifics of how a file system is mounted when using these options.

Public option with NFS URL – Forces the use of the public file handle. The mount fails if the public file handle is not supported.

Public option with regular path – Forces the use of the public file handle. The mount fails if the public file handle is not supported.

NFS URL only – Use the public file handle if enabled on the NFS server. If the mount fails using the public file handle, then try the mount using the MOUNT protocol.

Regular path only – Do not use the public file handle. The MOUNT protocol is used.

Client-Side Failover

Using client-side failover, an NFS client can switch to another server if the server supporting a replicated file system becomes unavailable. The file system can become unavailable if the server it is connected to crashes, if the server is overloaded, or if there is a network fault. The failover, under these conditions, is normally transparent to the user. After it is established, the failover can occur at any time without disrupting the processes running on the client.

Failover requires that the file system be mounted read-only. The file systems must be identical for the failover to occur successfully. See “What Is a Replicated File System?” on page 622 for a description of what makes a file system identical. A static file system or one that is not changed often is the best candidate for failover.

You cannot use file systems that are mounted using CacheFS with failover. Extra information is stored for each CacheFS file system. This information cannot be updated during failover, so only one of these two features can be used when mounting a file system.

The number of replicas that need to be established for each file system depends on many factors. In general, it is better to have a couple of servers, each supporting multiple subnets rather than have a unique server on each subnet. The process requires checking of each server in the list, so the more servers that are listed, the slower each mount will be.

Failover Terminology

To fully comprehend the process, two terms need to be understood.

- *failover* – Selecting a server from a list of servers supporting a replicated file system. Normally, the next server in the sorted list is used, unless it fails to respond.
- *remap* – Making use of a new server. Through normal use, the clients store the path name for each active file on the remote file system. During the remap, these path names are evaluated to locate the files on the new server.

What Is a Replicated File System?

For the purposes of failover, a file system can be called a *replica* when each file is the same size and has the same vnode type as the original file system. Permissions, creation dates, and other file attributes are not considered. If the file size or vnode types are different, the remap fails and the process hangs until the old server becomes available.

You can maintain a replicated file system using `rdist`, `cpio`, or another file transfer mechanism. Because updating the replicated file systems causes inconsistency, follow these suggestions for best results:

- Rename the old version of the file before installing a new one.
- Run the updates at night when client usage is low.
- Keep the updates small.
- Minimize the number of copies.

Failover and NFS Locking

Some software packages require read locks on files. To prevent these products from breaking, read locks on read-only file systems are allowed, but are visible to the client side only. The locks persist through a remap because the server doesn't

“know” about them. Because the files should not be changing, you do not need to lock the file on the server side.

Large Files

Starting with 2.6, the Solaris release supports files that are over 2 Gbytes. By default, UFS file systems are mounted with the –`largefiles` option to support the new functionality. Previous releases cannot handle files of this size. See “How to Disable Large Files on an NFS Server” on page 551 for instructions.

No changes need to occur on a Solaris 2.6 NFS client to be able to access a large file, if the file system on the server is mounted with the –`largefiles` option. However, not all 2.6 commands can handle these large files. See `largefile(5)` for a list of the commands that can handle the large files. Clients that cannot support the NFS version 3 protocol with the large file extensions cannot access any large files. Although clients running the Solaris 2.5 release can use the NFS version 3 protocol, large file support was not included in that release.

How NFS Server Logging Works

NFS server logging provides records of NFS reads and writes, as well as operations that modify the file system. This data can be used to track access to information. In addition, the records can provide a quantitative way to measure interest in the information.

When a file system with logging enabled is accessed, the kernel writes raw data into a buffer file. This data includes a timestamp, the client IP address, the UID of the requestor, the file handle of the file or directory object that is being accessed, and the type of operation that occured.

The `nfslogd` daemon converts this raw data into ASCII records that are stored in log files. During the conversion the IP addresses are modified to host names and the UIDs are modified to logins if the name service that is enabled can find matches. The file handles are also converted into path names. To accomplish this, the daemon keeps track of the file handles and stores information in a separate file handle to path table, so that the path does not have to be re-identified each time a file handle is accessed. Because there is no tracking of changes to the mappings in the file handle to path table if `nfslogd` is turned off, it is important to keep the daemon running.

How the WebNFS Service Works

The WebNFS service makes files in a directory available to clients using a public file handle. A file handle is an address generated by the kernel that identifies a file for NFS clients. The *public file handle* has a predefined value, so the server does not need

to generate a file handle for the client. The ability to use this predefined file handle reduces network traffic by eliminating the MOUNT protocol and should increase response time for the clients.

By default the public file handle on an NFS server is established on the root file system. This default provides WebNFS access to any clients that already have mount privileges on the server. You can change the public file handle to point to any file system by using the share command.

When the client has the file handle for the file system, a LOOKUP is run to determine the file handle for the file to be accessed. The NFS protocol allows the evaluation of only one path name component at a time. Each additional level of directory hierarchy requires another LOOKUP. A WebNFS server can evaluate an entire path name with a single transaction, called multicomponent lookup, when the LOOKUP is relative to the public file handle. With multicomponent lookup, the WebNFS server is able to deliver the file handle to the desired file without having to exchange the file handles for each directory level in the path name.

In addition, an NFS client can initiate concurrent downloads over a single TCP connection, which provides quick access without the additional load on the server caused by setting up multiple connections. Although Web browser applications support concurrent downloading of multiple files, each file has its own connection. By using one connection, the WebNFS software reduces the overhead on the server.

If the final component in the path name is a symbolic link to another file system, the client can access the file if the client already has access through normal NFS activities.

Normally, an NFS URL is evaluated relative to the public file handle. The evaluation can be changed to be relative to the server's root file system by adding an additional slash to the beginning of the path. In this example, these two NFS URLs are equivalent if the public file handle has been established on the /export/ftp file system.

```
nfs://server/junk
nfs://server//export/ftp/junk
```

How WebNFS Security Negotiation Works

The Solaris 8 release includes a new protocol so a WebNFS client can negotiate a selected security mechanism with a WebNFS server. The new protocol uses security negotiation multicomponent lookup, which is an extension to the multicomponent lookup used in earlier versions of the WebNFS protocol.

The WebNFS client initiates the process by making a regular multicomponent lookup request using the public file handle. Because the client has no knowledge of how the path is protected by the server, the default security mechanism is used. If the default security mechanism is not sufficient, the server replies with an AUTH_TOOWEAK error, indicating that the default mechanism is not valid and the client needs to use a stronger one.

When the client receives the AUTH_TOOWEAK error, it sends a request to the server to determine which security mechanisms are required. If the request succeeds, the server responds with an array of security mechanisms required for the specified path. Depending on the size of the array of security mechanisms, the client might have to make more requests to get the complete array. If the server does not support WebNFS security negotiation, the request fails.

After a successful request, the WebNFS client selects the first security mechanism from the array that it supports and issues a regular multicomponent lookup request using the selected security mechanism to acquire the file handle. All subsequent NFS requests are made using the selected security mechanism and the file handle.

WebNFS Limitations With Web Browser Use

Several functions that a Web site using HTTP can provide are not supported by the WebNFS software. These differences stem from the fact that the NFS server only sends the file, so any special processing must be done on the client. If you need to have one web site configured for both WebNFS and HTTP access, consider the following issues:

- NFS browsing does not run CGI scripts, so a file system with an active web site that uses many CGI scripts might not be appropriate for NFS browsing.
- The browser might start different viewers, to handle files in different file formats. Accessing these files through an NFS URL starts an external viewer as long as the file type can be determined by the file name. The browser should recognize any file name extension for a standard MIME type when an NFS URL is used. Because the WebNFS software does not check inside the file to determine the file type—unlike some Web browser applications—the only way to determine a file type is by the file name extension.
- NFS browsing cannot utilize server-side image maps (clickable images). However, it can utilize client-side image maps (clickable images) because the URLs are defined with the location. No additional response is required from the document server.

Secure NFS System

The NFS environment is a powerful and convenient way to share file systems on a network of different computer architectures and operating systems. However, the same features that make sharing file systems through NFS operation convenient also pose some security problems. Historically, most NFS implementations have used UNIX (or AUTH_SYS) authentication, but stronger authentication methods such as AUTH_DH have also been available. When using UNIX authentication, an NFS server authenticates a file request by authenticating the computer making the request, but not the user, so a client user can run su and impersonate the owner of a

file. If DH authentication is used, the NFS server authenticates the user, making this sort of impersonation much harder.

With root access and knowledge of network programming, anyone can introduce arbitrary data into the network and extract any data from the network. The most dangerous attacks are those involving the introduction of data, such as impersonating a user by generating the right packets or recording “conversations” and replaying them later. These attacks affect data integrity. Attacks involving passive eavesdropping—merely listening to network traffic without impersonating anybody—are not as dangerous, as data integrity is not compromised. Users can protect the privacy of sensitive information by encrypting data that goes over the network.

A common approach to network security problems is to leave the solution to each application. A better approach is to implement a standard authentication system at a level that covers all applications.

The Solaris operating environment includes an authentication system at the level of remote procedure call (RPC)—the mechanism on which NFS operation is built. This system, known as Secure RPC, greatly improves the security of network environments and provides additional security to services such as the NFS system. When the NFS system uses the facilities provided by Secure RPC, it is known as a Secure NFS system.

Secure RPC

Secure RPC is fundamental to the Secure NFS system. The goal of Secure RPC is to build a system at least as secure as a time-sharing system (one in which all users share a single computer). A time-sharing system authenticates a user through a login password. With data encryption standard (DES) authentication, the same is true. Users can log in on any remote computer just as they can on a local terminal, and their login passwords are their passports to network security. In a time-sharing environment, the system administrator has an ethical obligation not to change a password to impersonate someone. In Secure RPC, the network administrator is trusted not to alter entries in a database that stores *public keys.*

You need to be familiar with two terms to understand an RPC authentication system: credentials and verifiers. Using ID badges as an example, the credential is what identifies a person: a name, address, birthday, and so on. The verifier is the photo attached to the badge: you can be sure the badge has not been stolen by checking the photo on the badge against the person carrying it. In RPC, the client process sends both a credential and a verifier to the server with each RPC request. The server sends back only a verifier because the client already “knows” the server’s credentials.

RPC’s authentication is open ended, which means that a variety of authentication systems can be plugged into it. Currently, there are two systems: UNIX, and DH.

When UNIX authentication is used by a network service, the credentials contain the client's host name, UID, GID, and group-access list, but the verifier contains nothing. Because there is no verifier, a superuser could falsify appropriate credentials, using commands such as `su`. Another problem with UNIX authentication is that it assumes all computers on a network are UNIX computers. UNIX authentication breaks down when applied to other operating systems in a heterogeneous network.

To overcome the problems of UNIX authentication, Secure RPC uses either DH authentication.

DH Authentication

DH authentication uses the data encryption standard (DES) and Diffie-Hellman public-key cryptography to authenticate both users and computers in the network. DES is a standard encryption mechanism; Diffie-Hellman public-key cryptography is a cipher system that involves two keys: one public and one secret. The public and secret keys are stored in the name space. NIS stores the keys in the publickey map, and NIS+ stores the keys in the cred table. These maps contain the public key and secret key for all potential users. See the *Solaris Naming Administration Guide* for more information on how to set up the maps and tables.

The security of DH authentication is based on a sender's ability to encrypt the current time, which the receiver can then decrypt and check against its own clock. The timestamp is encrypted with DES. The requirements for this scheme to work are:

- The two agents must agree on the current time.
- The sender and receiver must be using the same encryption key.

If a network runs a time-synchronization program, the time on the client and the server is synchronized automatically. If a time-synchronization program is not available, timestamps can be computed using the server's time instead of the network time. The client asks the server for the time before starting the RPC session, then computes the time difference between its own clock and the server's. This difference is used to offset the client's clock when computing timestamps. If the client and server clocks get out of synchronization to the point where the server begins to reject the client's requests, the DH authentication system on the client resynchronizes with the server.

The client and server arrive at the same encryption key by generating a random *conversation key*, also known as the *session key*, and by using public-key cryptography to deduce a *common key*. The common key is a key that only the client and server are capable of deducing. The conversation key is used to encrypt and decrypt the client's timestamp; the common key is used to encrypt and decrypt the conversation key.

KERB Authentication

Kerberos is an authentication system developed at MIT. Encryption in Kerberos is based on DES. Kerberos support is no longer supplied as part of Secure RPC, but a client-side implementation is included with the Solaris 8 release. XREF

Using Secure RPC With NFS

Be aware of the following points if you plan to use Secure RPC:

- If a server crashes when no one is around (after a power failure for example), all the secret keys that are stored on the system are deleted. Now no process can access secure network services or mount an NFS file system. The important processes during a reboot are usually run as root, so these processes would work if root's secret key were stored away, but nobody is available to type the password that decrypts it. `keylogin -r` allows root to store the clear secret key in `/etc/.rootkey`, which `keyserv` reads.
- Some systems boot in single-user mode, with a root login shell on the console and no password prompt. Physical security is imperative in such cases.
- Diskless computer booting is not totally secure. Somebody could impersonate the boot server and boot a devious kernel that, for example, makes a record of your secret key on a remote computer. The Secure NFS system provides protection only after the kernel and the key server are running. Before that, there is no way to authenticate the replies given by the boot server. This could be a serious problem, but it requires a sophisticated attack, using kernel source code. Also, the crime would leave evidence. If you polled the network for boot servers, you would discover the devious boot server's location.
- Most setuid programs are owned by `root`; if the secret key for `root` is stored in `/etc/.rootkey`, these programs behave as they always have. If a setuid program is owned by a user, however, it might not always work. For example, if a setuid program is owned by dave and dave has not logged into the computer since it booted, the program would not be able to access secure network services.
- If you log in to a remote computer (using `login`, `rlogin`, or `telnet`) and use `keylogin` to gain access, you give access to your account. This is because your secret key gets passed to that computer's key server, which then stores it. This is only a concern if you do not trust the remote computer. If you have doubts, however, do not log in to a remote computer if it requires a password. Instead, use the NFS environment to mount file systems shared by the remote computer. As an alternative, you can use `keylogout` to delete the secret key from the key server.
- If a home directory is shared with the `–o sec=dh` option, remote logins can be a problem. If the `/etc/hosts.equiv` or `~/.rhosts` files are not set to prompt for a password, the login will succeed, but the users cannot access their home directories because no authentication has occurred locally. If the user is prompted for a password, the user will have access to his or her home directory as long as the password matches the network password.

Autofs Maps

Autofs uses three types of maps:

- Master map
- Direct maps
- Indirect maps

Master Autofs Map

The `auto_master` map associates a directory with a map. It is a master list specifying all the maps that autofs should check. The following example shows what an `auto_master` file could contain.

CODE EXAMPLE 31–1 Sample `/etc/auto_master` File

```
# Master map for automounter
#
+auto_master
/net            -hosts           -nosuid,nobrowse
/home           auto_home        -nobrowse
/xfn            -xfn
/-              auto_direct     -ro
```

This example shows the generic `auto_master` file with one addition for the `auto_direct` map. Each line in the master map `/etc/auto_master` has the following syntax:

mount-point map-name [*mount-options*]

mount-point — *mount-point* is the full (absolute) path name of a directory. If the directory does not exist, autofs creates it if possible. If the directory exists and is not empty, mounting on it hides its contents. In this case, autofs issues a warning.

The notation `/-` as a mount point indicates that the map in question is a direct map, and no particular mount point is associated with the map as a whole.

map-name — *map-name* is the map autofs uses to find directions to locations, or mount information. If the name is preceded by a slash (/), autofs interprets the name as a local file. Otherwise, autofs searches for the mount information using the search specified in the name service switch configuration file (`/etc/nsswitch.conf`). Special maps are also used for `/net` and `/xfn` (see "Mount Point `/net`" on page 631 and "Mount Point `/xfn`" on page 631).

mount-options — *mount-options* is an optional, comma-separated list of options that apply to the mounting of the entries specified in map-name, unless the entries in map-name list other options. Options for each specific type of file system are listed in the mount man page for that file system (for example, see the `mount_nfs`(1M) man page for NFS specific mount options). For NFS specific mount points, the `bg` (background) and `fg` (foreground) options do not apply.

A line beginning with # is a comment. Everything that follows until the end of the line is ignored.

To split long lines into shorter ones, put a backslash (\) at the end of the line. The maximum number of characters of an entry is 1024.

Note - If the same mount point is used in two entries, the first entry is used by the `automount` command. The second entry is ignored.

Mount Point `/home`

The mount point `/home` is the directory under which the entries listed in `/etc/auto_home` (an indirect map) are to be mounted.

Note - Autofs runs on all computers and supports `/net` and `/home` (automounted home directories) by default. These defaults can be overridden by entries in the NIS `auto.master` map or NIS+ `auto_master` table, or by local editing of the `/etc/auto_master` file.

Mount Point /net

Autofs mounts under the directory /net all the entries in the special map -hosts. This is a built-in map that uses only the hosts database. For example, if the computer gumbo is in the hosts database and it exports any of its file systems, the command:

```
%cd /net/gumbo
```

changes the current directory to the root directory of the computer gumbo. Autofs can mount only the *exported* file systems of host gumbo, that is, those on a server available to network users as opposed to those on a local disk. Therefore, all the files and directories on gumbo might not be available through /net/gumbo.

With the /net method of access, the server name is in the path and is location dependent. If you want to move an exported file system from one server to another, the path might no longer work. Instead, you should set up an entry in a map specifically for the file system you want rather than use /net.

Note - Autofs checks the server's export list only at mount time. After a server's file systems are mounted, autofs does not check with the server again until the server's file systems are automatically unmounted. Therefore, newly exported file systems are not "seen" until the file systems on the client are unmounted and then remounted.

Mount Point /xfn

This mount point provides the autofs directory structure for the resources that are shared through the FNS name space (see the *Solaris Naming Setup and Configuration Guide* for more information about FNS).

Direct Autofs Maps

A direct map is an automount point. With a direct map, there is a direct association between a mount point on the client and a directory on the server. Direct maps have a full path name and indicate the relationship explicitly. This is a typical /etc/auto_direct map:

```
/usr/local          -ro \
   /bin                   ivy:/export/local/sun4 \
   /share                 ivy:/export/local/share \
   /src                   ivy:/export/local/src
   /usr/man         -ro   oak:/usr/man \
                          rose:/usr/man \
                          willow:/usr/man
```

(continued)

(Continuation)

```
/usr/games       -ro   peach:/usr/games
/usr/spool/news  -ro   pine:/usr/spool/news \
                       willow:/var/spool/news
```

Lines in direct maps have the following syntax:

key [*mount-options*]*location*

key	*key* is the path name of the mount point in a direct map.
mount-options	*mount-options* is the options you want to apply to this particular mount. They are required only if they differ from the map default. Options for each specific type of file system are listed in the mount man page for that file system (for example, see the `mount_cachefs`(1M) man page for CacheFS specific mount options).
location	*location* is the location of the file system, specified (one or more) as *server*:*pathname* for NFS file systems or :*devicename* for High Sierra file systems (HSFS).

Note - The *pathname* should not include an automounted mount point; it should be the actual absolute path to the file system. For instance, the location of a home directory should be listed as *server*`:/export/home/`*username*, not as *server*`:/home/`*username*.

As in the master map, a line beginning with # is a comment. All the text that follows until the end of the line is ignored. Put a backslash at the end of the line to split long lines into shorter ones.

Of all the maps, the entries in a direct map most closely resemble the corresponding entries in `/etc/vfstab` (`vfstab` contains a list of all file systems to be mounted). An entry that appears in `/etc/vfstab` as:

```
dancer:/usr/local - /usr/local/tmp nfs - yes ro
```

appears in a direct map as:

```
/usr/local/tmp     -ro     dancer:/usr/local
```

Note - No concatenation of options occurs between the automounter maps. Any options added to an automounter map override all options listed in maps that are searched earlier. For instance, options included in the `auto_master` map would be overridden by corresponding entries in any other map.

See "How Autofs Selects the Nearest Read-Only Files for Clients (Multiple Locations)" on page 639 for other important features associated with this type of map.

Mount Point /–

In Code Example 31–1, the mount point `/-` tells autofs not to associate the entries in `auto_direct` with any specific mount point. Indirect maps use mount points defined in the `auto_master` file. Direct maps use mount points specified in the named map. (Remember, in a direct map the key, or mount point, is a full path name.)

An NIS or NIS+ `auto_master` file can have only one direct map entry because the mount point must be a unique value in the name space. An `auto_master` file that is a local file can have any number of direct map entries, as long as entries are not duplicated.

Indirect Autofs Maps

An indirect map uses a substitution value of a key to establish the association between a mount point on the client and a directory on the server. Indirect maps are useful for accessing specific file systems, like home directories. The `auto_home` map is an example of an indirect map.

Lines in indirect maps have the following general syntax:

key [*mount-options*]*location*

TABLE 31–3 Table Caption

key	*key* is a simple name (no slashes) in an indirect map.
mount-options	*mount-options* is the options you want to apply to this particular mount. They are required only if they differ from the map default. Options for each specific type of file system are listed in the mount man page for that file system (for example, see the `mount_nfs`(1M) man page for NFS specific mount options).
location	*location* is the location of the file system, specified (one or more) as *server*:*pathname*.

Note - The *pathname* should not include an automounted mount point; it should be the actual absolute path to the file system. For instance, the location of a directory should be listed as *server*`:/usr/local`, not as *server*`:/net/`*server*`/usr/local`.

As in the master map, a line beginning with # is a comment. All the text that follows until the end of the line is ignored. Put a backslash (\) at the end of the line to split long lines into shorter ones. Code Example 31–1 shows an `auto_master` map that contains the entry:

```
/home      auto_home        -nobrowse
```

`auto_home` is the name of the indirect map that contains the entries to be mounted under `/home`. A typical `auto_home` map might contain:

```
david                   willow:/export/home/david
rob                     cypress:/export/home/rob
gordon                  poplar:/export/home/gordon
rajan                   pine:/export/home/rajan
tammy                   apple:/export/home/tammy
jim                     ivy:/export/home/jim
linda    -rw,nosuid     peach:/export/home/linda
```

As an example, assume that the previous map is on host `oak`. If user `linda` has an entry in the password database specifying her home directory as `/home/linda`, whenever she logs in to computer `oak`, autofs mounts the directory `/export/home/linda` residing on the computer `peach`. Her home directory is mounted read-write, nosuid.

Assume the following conditions occur: User linda's home directory is listed in the password database as `/home/linda`. Anybody, including Linda, has access to this path from any computer set up with the master map referring to the map in the previous example.

Under these conditions, user `linda` can run login or rlogin on any of these computers and have her home directory mounted in place for her.

Furthermore, now Linda can also type the following command:

```
% cd ~david
```

autofs mounts David's home directory for her (if all permissions allow).

Note - No concatenation of options between the automounter maps. Any options added to an automounter map override all options listed in maps that are searched earlier. For instance, options included in the `auto_master` map are overidden by corresponding entries in any other map.

On a network without a name service, you have to change all the relevant files (such as `/etc/passwd`) on all systems on the network to accomplish this. With NIS, make the changes on the NIS master server and propagate the relevant databases to the slave servers. On a network running NIS+, propagating the relevant databases to the slave servers is done automatically after the changes are made.

How Autofs Works

Autofs is a client-side service that automatically mounts the appropriate file system. When a client attempts to access a file system that is not presently mounted, the autofs file system intercepts the request and calls `automountd` to mount the requested directory. The `automountd` daemon locates the directory, mounts it within autofs, and replies. On receiving the reply, autofs allows the waiting request to proceed. Subsequent references to the mount are redirected by the autofs—no further participation is required by `automountd` until the file system is automatically unmounted by autofs after a period of inactivity.

The components that work together to accomplish automatic mounting are:

- The `automount` command
- The `autofs` file system
- The `automountd` daemon

The `automount` command, called at system startup time, reads the master map file `auto_master` to create the initial set of autofs mounts. These autofs mounts are not automatically mounted at startup time. They are points under which file systems are mounted in the future. These points are also known as trigger nodes.

After the autofs mounts are set up, they can trigger file systems to be mounted under them. For example, when autofs receives a request to access a file system that

is not currently mounted, autofs calls `automountd`, which actually mounts the requested file system.

Starting with the Solaris 2.5 release, the `automountd` daemon is completely independent from the automount command. Because of this separation, it is possible to add, delete, or change map information without first having to stop and start the `automountd` daemon process.

After initially mounting autofs mounts, the automount command is used to update autofs mounts as necessary, by comparing the list of mounts in the `auto_master` map with the list of mounted file systems in the mount table file `/etc/mnttab` (formerly `/etc/mtab`) and making the appropriate changes. This allows system administrators to change mount information within `auto_master` and have those changes used by the autofs processes without having to stop and restart the autofs daemon. After the file system is mounted, further access does not require any action from `automountd` until the file system is automatically unmounted.

Unlike `mount`, `automount` does not read the `/etc/vfstab` file (which is specific to each computer) for a list of file systems to mount. The `automount` command is controlled within a domain and on computers through the name space or local files.

This is a simplified overview of how autofs works:

The automount daemon `automountd` starts at boot time from the `/etc/init.d/autofs` script (see Figure 31–1). This script also runs the `automount` command, which reads the master map (see "How Autofs Starts the Navigation Process (Master Map)" on page 637) and installs autofs mount points.

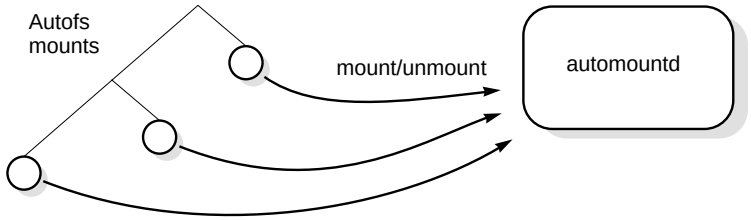


Figure 31–1 `/etc/init.d/autofs` Script Starts `automount`

Autofs is a kernel file system that supports automatic mounting and unmounting.

When a request is made to access a file system at an autofs mount point:

1. Autofs intercepts the request.
2. Autofs sends a message to the automountd for the requested file system to be mounted.
3. `automountd` locates the file system information in a map, creates the trigger nodes, and performs the mount.
4. Autofs allows the intercepted request to proceed.
5. Autofs unmounts the file system after a period of inactivity.

Note - Mounts managed through the autofs service should not be manually mounted or unmounted. Even if the operation is successful, the autofs service does not check that the object has been unmounted, resulting in possible inconsistencies. A reboot clears all of the autofs mount points.

How Autofs Navigates Through the Network (Maps)

Autofs searches a series of maps to navigate its way through the network. Maps are files that contain information such as the password entries of all users on a network or the names of all host computers on a network, that is, network-wide equivalents of UNIX administration files. Maps are available locally or through a network name service like NIS or NIS+. You create maps to meet the needs of your environment using the Solstice System Management Tools. See "Modifying How Autofs Navigates the Network (Modifying Maps)" on page 645.

How Autofs Starts the Navigation Process (Master Map)

The `automount` command reads the master map at system startup. Each entry in the master map is a direct or indirect map name, its path, and its mount options, as shown in Figure 31–2. The specific order of the entries is not important. `automount` compares entries in the master map with entries in the mount table to generate a current list.

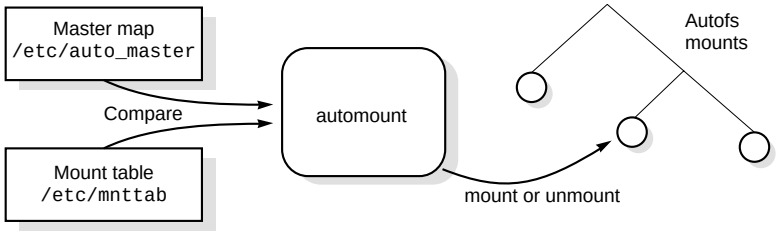


Figure 31–2 Navigation Through the Master Map

Autofs Mount Process

What the autofs service does when a mount request is triggered depends on how the automounter maps are configured. The mount process is generally the same for all

mounts, but the final result changes with the mount point specified and the complexity of the maps. Starting with the Solaris 2.6 release, the mount process has also been changed to include the creation of the trigger nodes.

Simple Autofs Mount

To help explain the autofs mount process, assume that the following files are installed.

```
$ cat /etc/auto_master
# Master map for automounter
#
+auto_master
/net        -hosts        -nosuid,nobrowse
/home       auto_home     -nobrowse
/xfn        -xfn
/share      auto_share
$ cat /etc/auto_share
# share directory map for automounter
#
ws          gumbo:/export/share/ws
```

When the `/share` directory is accessed, the autofs service creates a trigger node for `/share/ws`, which can be seen in `/etc/mnttab` as an entry that resembles the following entry:

```
-hosts  /share/ws     autofs  nosuid,nobrowse,ignore,nest,dev=###
```

When the `/share/ws` directory is accessed, the autofs service completes the process with these steps:

1. Pings the server's mount service to see if it's alive.
2. Mounts the requested file system under `/share`. Now `/etc/mnttab` file contains the following entries:

```
-hosts  /share/ws     autofs  nosuid,nobrowse,ignore,nest,dev=###
gumbo:/export/share/ws /share/ws   nfs   nosuid,dev=####    #####
```

Hierarchical Mounting

When multiple layers are defined in the automounter files, the mount process becomes more complex. If the `/etc/auto_shared` file from the previous example is expanded to contain:

```
# share directory map for automounter
#
ws       /       gumbo:/export/share/ws
         /usr    gumbo:/export/share/ws/usr
```

The mount process is basically the same as the previous example when the `/share/ws` mount point is accessed. In addition, a trigger node to the next level (`/usr`) is created in the `/share/ws` file system so that the next level can be mounted if it is accessed. In this example, `/export/share/ws/usr` must exist on the NFS server for the trigger node to be created.

Caution - Do not use the `–soft` option when specifying hierarchical layers. Refer to “Autofs Unmounting” on page 639 for an explanation of this limitation.

Autofs Unmounting

The unmounting that occurs after a certain amount of idle time is from the bottom up (reverse order of mounting). If one of the directories at a higher level in the hierarchy is busy, only file systems below that directory are unmounted. During the unmounting process, any trigger nodes are removed and then the file system is unmounted. If the file system is busy, the unmount fails and the trigger nodes are reinstalled.

Caution - Do not use the `–soft` option when specifying hierarchical layers. If the `–soft` option is used, requests to reinstall the trigger nodes can time out. The failure to reinstall the trigger notes leaves no access to the next level of mounts. The only way to clear this problem is to have the automounter unmount all of the components in the hierarchy, either by waiting for the file systems to be automatically unmounted or by rebooting the system.

How Autofs Selects the Nearest Read-Only Files for Clients (Multiple Locations)

In the example of a direct map, which was:

```
/usr/local          -ro \
   /bin                   ivy:/export/local/sun4\
   /share                 ivy:/export/local/share\
   /src                   ivy:/export/local/src
/usr/man            -ro   oak:/usr/man \
                          rose:/usr/man \
```

(Continuation)

```
                        willow:/usr/man
/usr/games          -ro   peach:/usr/games
/usr/spool/news     -ro   pine:/usr/spool/news \
                          willow:/var/spool/news
```

The mount points `/usr/man` and `/usr/spool/news` list more than one location (three for the first, two for the second). This means any of the replicated locations can provide the same service to any user. This procedure makes sense only when you mount a file system that is read-only, as you must have some control over the locations of files you write or modify. You don't want to modify files on one server on one occasion and, minutes later, modify the "same" file on another server. The benefit is that the best available server is used automatically without any effort required by the user.

If the file systems are configured as replicas (see "What Is a Replicated File System?" on page 622), the clients have the advantage of using failover. Not only is the best server automatically determined, but if that server becomes unavailable, the client automatically uses the next-best server. Failover was first implemented in the Solaris 2.6 release.

An example of a good file system to configure as a replica is man pages. In a large network, more than one server can export the current set of manual pages. Which server you mount them from does not matter, as long as the server is running and exporting its file systems. In the previous example, multiple mount locations are expressed as a list of mount locations in the map entry.

```
/usr/man -ro oak:/usr/man rose:/usr/man willow:/usr/man
```

Here you can mount the man pages from the servers `oak`, `rose`, or `willow`. Which server is best depends on a number of factors including: the number of servers supporting a particular NFS protocol level, the proximity of the server, and weighting.

During the sorting process, a count of the number of servers supporting the NFS version 2 and NFS version 3 protocols is made. Whichever protocol is supported on the most servers becomes the protocol supported by default. This provides the client with the maximum number of servers to depend on.

After the largest subset of servers with the same protocol version is found, that server list is sorted by proximity. Servers on the local subnet are given preference over servers on a remote subnet. The closest server is given preference, which reduces latency and network traffic. Figure 31–3 illustrates server proximity.

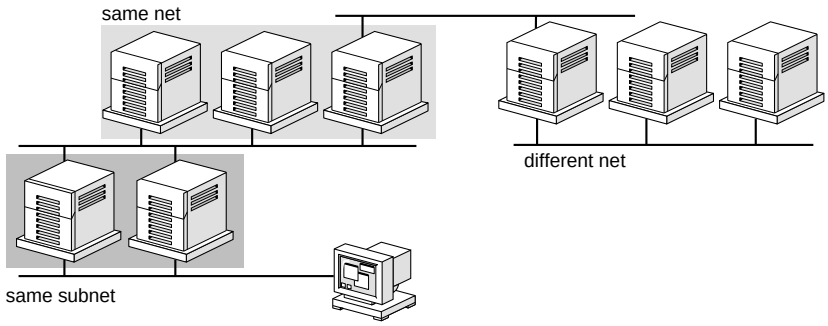


Figure 31–3 Server Proximity

If several servers supporting the same protocol are on the local subnet, the time to connect to each server is determined and the fastest is used. The sorting can also be influenced by using weighting (see “Autofs and Weighting” on page 641).

If version 3 servers are more abundant, the sorting process becomes more complex. Normally, servers on the local subnet are given preference over servers on a remote subnet. A version 2 server can complicate matters, as it might be closer than the nearest version 3 server. If there is a version 2 server on the local subnet and the closest version 3 server is on a remote subnet, the version 2 server is given preference. This preference is only checked if there are more version 3 servers than version 2 servers. If there are more version 2 servers, only a version 2 server is selected.

With failover, the sorting is checked once at mount time to select one server from which to mount, and again anytime the mounted server becomes unavailable. Multiple locations are useful in an environment where individual servers might not export their file systems temporarily.

This feature is particularly useful in a large network with many subnets. Autofs chooses the nearest server and therefore confines NFS network traffic to a local network segment. In servers with multiple network interfaces, list the host name associated with each network interface as if it were a separate server. Autofs selects the nearest interface to the client.

Autofs and Weighting

You can influence the selection of servers at the same proximity level by adding a weighting value to the autofs map. For example:

```
/usr/man -ro oak,rose(1),willow(2):/usr/man
```

The numbers in parentheses indicate a weighting. Servers without a weighting have a value of zero (most likely to be selected). The higher the weighting value, the lower the chance the server will be selected.

Note - All other server selection factors are more important than weighting. Weighting is only considered when selecting between servers with the same network proximity.

Variables in a Map Entry

You can create a client-specific variable by prefixing a dollar sign ($) to its name. This helps you to accommodate different architecture types accessing the same file system location. You can also use curly braces to delimit the name of the variable from appended letters or digits. Table 31–4 shows the predefined map variables.

TABLE 31–4 Predefined Map Variables

Variable	Meaning	Derived From	Example
ARCH	Architecture type	`uname -m`	`sun4`
CPU	Processor type	`uname -p`	`sparc`
HOST	Host name	`uname -n`	`dinky`
OSNAME	Operating system name	`uname -s`	`SunOS`
OSREL	Operating system release	`uname -r`	`5.4`
OSVERS	Operating system version (version of the release)	`uname -v`	`FCS1.0`

You can use variables anywhere in an entry line except as a key. For instance, if you have a file server exporting binaries for SPARC and IA architectures from `/usr/local/bin/sparc` and `/usr/local/bin/x86` respectively, the clients can mount through a map entry like the following:

```
/usr/local/bin    -ro server:/usr/local/bin/$CPU
```

Now the same entry for all clients applies to all architectures.

Note - Most applications written for any of the sun4 architectures can run on all sun4 platforms, so the –ARCH variable is hard-coded to `sun4` instead of `sun4m`.

Maps That Refer to Other Maps

A map entry *+mapname* used in a file map causes automount to read the specified map as if it were included in the current file. If *mapname* is not preceded by a slash, autofs treats the map name as a string of characters and uses the name service switch policy to find it. If the path name is an absolute path name, `automount` checks a local map of that name. If the map name starts with a dash (`-`), `automount` consults the appropriate built-in map, such as `xfn` or `hosts`.

This name service switch file contains an entry for autofs labeled as `automount`, which contains the order in which the name services are searched. The following file is an example of a name service switch file:

```
#
# /etc/nsswitch.nis:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it uses NIS (YP) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the /etc/netconfig
# file contains "switch.so" as a nametoaddr library for "inet" transports.
# the following two lines obviate the "+" entry in /etc/passwd and /etc/group.
passwd:         files nis
group:          files nis

# consult /etc "files" only if nis is down.
hosts:          nis [NOTFOUND=return] files
networks:       nis [NOTFOUND=return] files
protocols:      nis [NOTFOUND=return] files
rpc:            nis [NOTFOUND=return] files
ethers:         nis [NOTFOUND=return] files
netmasks:       nis [NOTFOUND=return] files
bootparams:     nis [NOTFOUND=return] files
publickey:      nis [NOTFOUND=return] files
netgroup:       nis
automount:      files nis
aliases:        files nis
# for efficient getservbyname() avoid nis
services:       files nis
```

In this example, the local maps are searched before the NIS maps, so you can have a few entries in your local `/etc/auto_home` map for the most commonly accessed home directories, and use the switch to fall back to the NIS map for other entries.

```
bill              cs.csc.edu:/export/home/bill
bonny             cs.csc.edu:/export/home/bonny
```

After consulting the included map, if no match is found, `automount` continues scanning the current map. This means you can add more entries after a `+` entry.

```
bill              cs.csc.edu:/export/home/bill
bonny             cs.csc.edu:/export/home/bonny
+auto_home
```

The map included can be a local file (remember, only local files can contain + entries) or a built-in map:

```
+auto_home_finance      # NIS+ map
+auto_home_sales        # NIS+ map
+auto_home_engineering  # NIS+ map
+/etc/auto_mystuff      # local map
+auto_home              # NIS+ map
+-hosts                 # built-in hosts map
```

Note - You cannot use + entries in NIS+ or NIS maps.

Executable Autofs Maps

You can create an autofs map that will execute some commands to generate the autofs mount points. You could benefit from using an executable autofs map if you need to be able to create the autofs structure from a database or a flat file. The disadvantage to using an executable map is that the map will need to be installed on each host. An executable map cannot be included in either the NIS or the NIS+ name service.

The executable map must have an entry in the `auto_master` file.

```
/execute    auto_execute
```

Here is an example of an executable map:

```
#!/bin/ksh
#
# executable map for autofs
#

case $1 in
          src)  echo '-nosuid,hard bee:/export1' ;;
esac
```

For this example to work, the file must be installed as `/etc/auto_execute` and must have the executable bit set (set permissions to 744). Under these circumstances running the following command:

```
% ls /execute/src
```

causes the /export1 file system from bee to be mounted.

Modifying How Autofs Navigates the Network (Modifying Maps)

You can modify, delete, or add entries to maps to meet the needs of your environment. As applications and other file systems that users require change their location, the maps must reflect those changes. You can modify autofs maps at any time. Whether your modifications take effect the next time automountd mounts a file system depends on which map you modify and what kind of modification you make.

Default Autofs Behavior With Name Services

Booting invokes autofs using the /etc/init.d/autofs script and checks for the master auto_master map (subject to the rules discussed subsequently).

Autofs uses the name service specified in the automount entry of the /etc/nsswitch.conf file. If NIS+ is specified, as opposed to local files or NIS, all map names are used as is. If NIS is selected and autofs cannot find a map that it needs, but finds a map name that contains one or more underscores, the underscores are changed to dots, which allows the old NIS file names to work. Then autofs checks the map again, as shown in Figure 31–4.

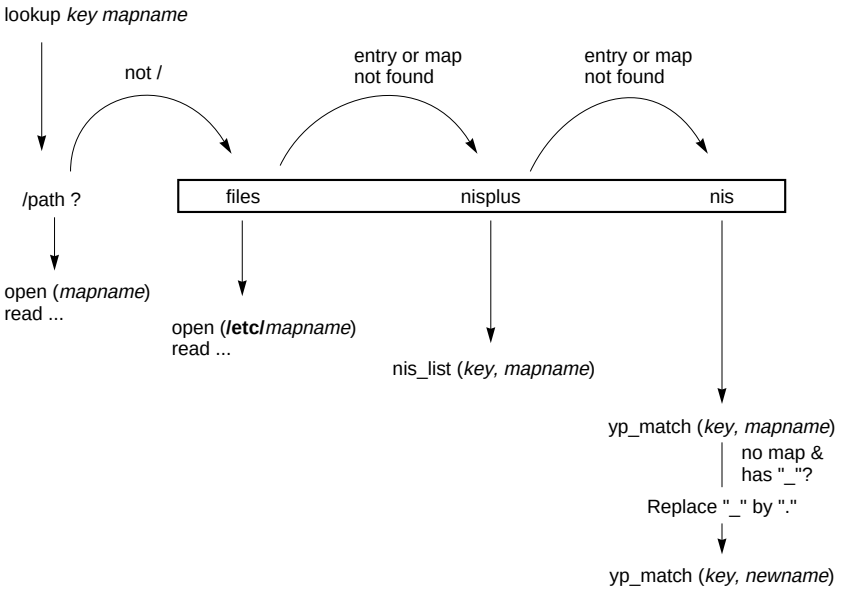


Figure 31–4 How Autofs Uses the Name Service

The screen activity for this session would look like the following example.

```
$ grep /home /etc/auto_master
/home           auto_home

$ ypmatch brent auto_home
Can't match key brent in map auto_home.  Reason: no such map in
server's domain.

$ ypmatch brent auto.home
diskus:/export/home/diskus1/&
```

If "files" is selected as the name service, all maps are assumed to be local files in the `/etc` directory. Autofs interprets a map name that begins with a slash (`/`) as local regardless of which name service it uses.

Autofs Reference

The rest of this chapter describes more advanced autofs features and topics.

Metacharacters

Autofs recognizes some characters as having a special meaning. Some are used for substitutions, some to protect other characters from the autofs map parser.

Ampersand (&)

If you have a map with many subdirectories specified, as in the following, consider using string substitutions.

```
john        willow:/home/john
mary        willow:/home/mary
joe         willow:/home/joe
able        pine:/export/able
baker       peach:/export/baker
```

You can use the ampersand character (&) to substitute the key wherever it appears. If you use the ampersand, the previous map changes to:

```
john        willow:/home/&
mary        willow:/home/&
joe         willow:/home/&
able        pine:/export/&
baker       peach:/export/&
```

You could also use key substitutions in a direct map, in situations like this:

```
/usr/man      willow,cedar,poplar:/usr/man
```

which you can also write as:

```
/usr/man      willow,cedar,poplar:&
```

Notice that the ampersand substitution uses the whole key string, so if the key in a direct map starts with a / (as it should), the slash is carried over, and you could not do, for example, the following:

```
/progs    &1,&2,&3:/export/src/progs
```

because autofs would interpret it as:

```
/progs     /progs1,/progs2,/progs3:/export/src/progs
```

Asterisk (*)

You can use the universal substitute character, the asterisk (*), to match any key. You could mount the `/export` file system from all hosts through this map entry.

```
*       &:/export
```

Each ampersand is substituted by the value of any given key. Autofs interprets the asterisk as an end-of-file character.

Special Characters

If you have a map entry that contains special characters, you might have to mount directories that have names which confuse the autofs map parser. The autofs parser is sensitive to names containing colons, commas, spaces, and so on. These names should be enclosed in double quotations, as in the following:

```
/vms    -ro    vmsserver: -  -  - "rc0:dk1 - "
/mac    -ro    gator:/ - "Mr Disk - "
```

CHAPTER **32**

Mail Services Topics

Chapter 33	Provides overview information for the mail service
Chapter 34	Provides step-by-step instructions for setting up and troubleshooting the mail service
Chapter 35	Provides background information on the mail service

CHAPTER 33

Introduction to Mail Services

Setting up and maintaining an electronic mail service are complex tasks, critical to the daily operation of your network. As network administrator, you might need to expand an existing mail service or perhaps set up mail service on a new network or subnet. To help you plan a mail service for your network, this chapter provides conceptual information about mail services and briefly outlines the tasks required for setting up typical mail configurations.

What's New With `sendmail`

Version 8.9.3 of `sendmail` has been included with the Solaris 8 release. Here is a list of the important or user-visible changes that are included in this new version:

- A new configuration file option, called `MaxHeadersLength`, limits the length of the sum of all header lines in a given message. The default value is 32768 bytes. Incoming messages with headers that exceed this value are rejected.
- A new file called `/etc/default/sendmail` can be used to store options to start sendmail with, rather than adding these options to the init script. The file makes it easier to upgrade systems, since the init scripts do not need to change.
- The `mail.local` program has been extended to use the Local Mail Transfer Protocol. The protocol allows error codes to be returned for each recipient, so that the message is resent to just the recipients that did not receive the message rather than having to re-queue the message to all of the recipients. This protocol was added to `sendmail` in the Solaris 7 release.

- A new command, named `/usr/bin/praliases`, can be used to turn the data in the alias database into plain text. If an argument is included on the command line, the command prints out a key:value pair, if the argument matches a key.
- A new program called smrsh can be used to limit the number of commands that can be run using the "`|program`" syntax of `sendmail`. Only programs included in `/var/adm/sm.bin` can be run if this feature is enabled. Adding `FEATURE('smrsh')` in the main configuration file enables this feature (see `/usr/lib/mail/README` for details.)
- New options have been added to the vacation program: `–f` can be used to select an alternate database instead of `~/.vacation.`*ext*; `–m` can be used to select an alternate message file instead of `~/.vacation.msg`; and `–s` can be used to specify the reply address instead of the UNIX From line in the incoming message.
- A change to the `mailx` program allows for the From: header to be used as the basis of the sender instead of the envelope sender. This change makes `mailx` work like `mailtool` and `dtmail`.

Additional information on the Solaris version of `sendmail` can be found at `http://www.sendmail.org/sun-specific/migration+sun.html`.

Other `sendmail` Information Sources

Here is a list of additional information sources about `sendmail`.

- `http://www.sendmail.org` – Home page for `sendmail`
- `http://www.sendmail.org/faq` – FAQ for `sendmail`
- `http://www.sendmail.org/m4/readme.html` – README for new `sendmail` configuration files
- `http://www.sendmail.org/sun-specific/migration+sun.html` – Differences between `sendmail` delivered with the 2.6 release and release 7.

Introduction to Mail Services Terminology

Many software and hardware components are required to establish a mail service. The following sections give a quick introduction to these components and some of the terminology used to describe them.

The first section defines the terminology used when discussing the software parts of the mail delivery system. The next section focuses on the functions of the hardware systems in a mail configuration.

Mail Services Software Components Overview

The following table introduces some of the software components of a mail system. See “Mail Services Software Terminology” on page 686 for a complete description of all of the software components.

Component	Description
.forward files	Files that can be set up in a user’s home directory to redirect mail or send mail to a program automatically
mailbox	A file on a mail server that is the final destination for email messages
mail addresses	Contains the name of the recipient and the system to which a mail message is delivered
mail aliases	An alternate name used in a mail address
mail queue	A collection of mail messages that needs to be processed by the mail server
postmaster	A special mail alias used to report problems and ask questions about the mail service
sendmail configuration file	A file that contains all the information necessary for mail routing

Hardware Components of a Mail Configuration

A mail configuration requires three elements, which can be combined on the same system or provided by separate systems:

- A mail host – A system configured to handle email addresses that are difficult to resolve
- At least one mail server – A system configured to hold one or more mailboxes
- Mail clients – Systems that access mail from a mail server

When you want users to communicate with networks outside your domain, you must also add a fourth element, a mail gateway.

Figure 33–1 shows a typical electronic mail configuration, using the three basic mail elements plus a mail gateway.

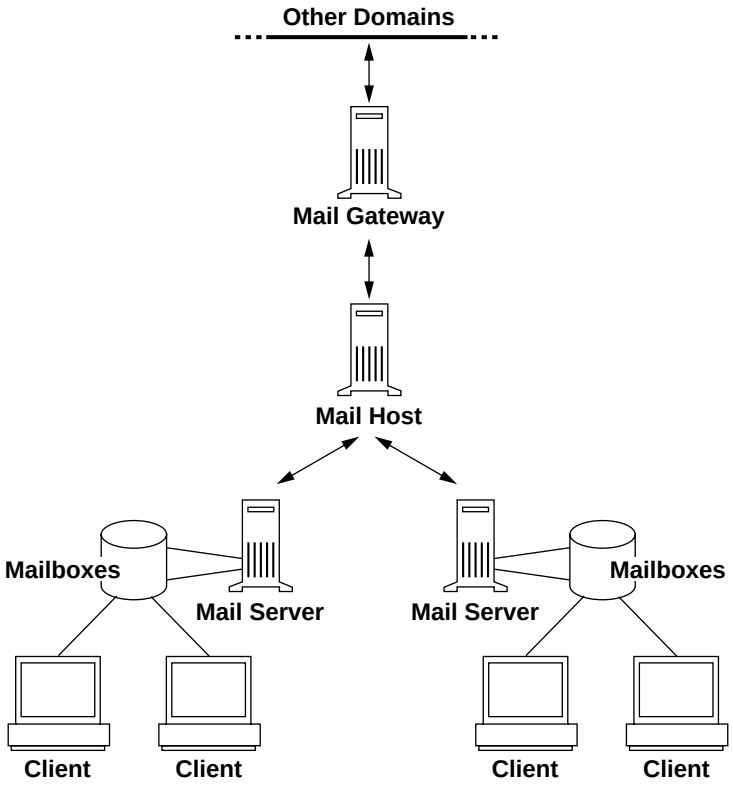


Figure 33–1 Typical Electronic Mail Configuration

Each element is described in detail in “Hardware Components of a Mail Configuration” on page 694.

CHAPTER 34

Setting Up and Administering Mail Services

This chapter describes how to set up and administer mail services. If you are not familiar with administering mail services, read Chapter 33 for an introduction to the terminology and structure of the mail services and for descriptions of several mail service configurations.

Planning Your Mail System

This section describes four basic types of mail configurations and briefly outlines the tasks required to set up each configuration. You might find this section useful if you need to set up a new mail system or if you are expanding an existing one. The configurations start with the most basic case (mail completely local, no connection to the outside world) and increase in complexity to a two-domain configuration with a mail gateway. The first decision you should make is which one of these configurations you would like to employ. The following configurations are covered:

- “Local Mail in Remote Mode” on page 657
- “Local Mail and a Remote Connection” on page 657
- “Two Domains and a Gateway” on page 658

As system administrator, you should decide on a policy for updating aliases and for forwarding mail messages. You might set up an `aliases` mailbox as a place for users to send requests for mail forwarding and for changes to their default mail alias. If your system uses NIS or NIS+, you can administer forwarding rather than forcing users to manage it themselves.

Local Mail Only

The simplest mail configuration, shown in Figure 34–1, is one mail host with two or more workstations connected to it. Mail is completely local. All the clients store mail on their local disks and act as mail servers. Mail addresses are parsed using the `/etc/mail/aliases` files.

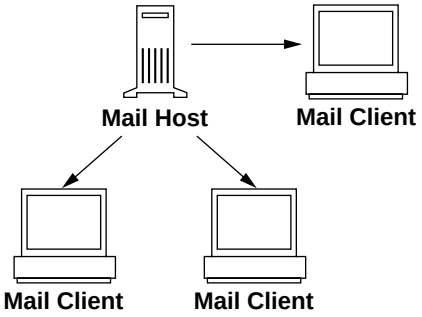


Figure 34–1 Local Mail Configuration

To set up this kind of mail configuration, you need:

- The default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)
- A server designated as the mail host (add `mailhost.`*domainname* to the `/etc/hosts` file on the mail host; then if you are not running NIS or NIS+, add the mail host IP address line to the `/etc/hosts` file of all mail clients)
- Matching `/etc/mail/aliases` files on any system that has a local mailbox (unless you are running NIS or NIS+)
- Enough space in `/var/mail` on each mail client system to hold the mailboxes

Local Mail in Remote Mode

In this configuration, each mail client mounts its mail from one mail server that provides mail spooling for client mailboxes. This server can also be the mail host. This configuration assists the backup of the mailboxes for each client.

To set up this kind of mail configuration, you need:

- The default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)
- A server designated as the mail host (add `mailhost.`*domainname* to the `/etc/hosts` file on the mail host; then if you are not running NIS or NIS+, add the mail host IP address line to the `/etc/hosts` file of all mail clients)
- Matching `/etc/mail/aliases` files on any system that has a local mailbox (unless you are running NIS or NIS+)
- Entries in each mail client's `/etc/vfstab` file or `/etc/auto_direct` (if autofs is used) to mount the `/var/mail` directory
- Enough space in `/var/mail` on the mail server to hold the client mailboxes

Local Mail and a Remote Connection

The most common mail configuration in a small network is shown in Figure 34–2. One system is the mail server, the mail host, and the mail gateway to the outside world. Mail is distributed using the `/etc/mail/aliases` files. No name service is required.

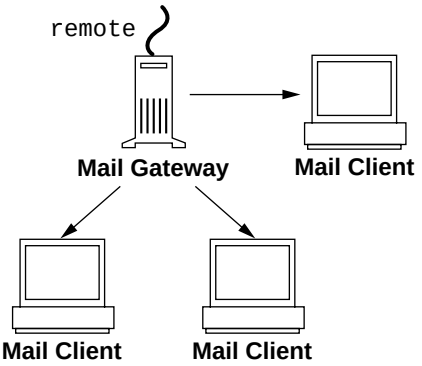


Figure 34–2 Local Mail Configuration With a UUCP Connection

To set up this kind of a mail configuration, assuming that the mail clients mount their mail files from `/var/mail` on the mail host, you need:

- The `main.cf` file on the mail gateway (no editing required if MX records are used)

- The default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)
- A server designated as the mail host (add `mailhost.`*domainname* to the `/etc/hosts` file on the mail host; if you are not running NIS or NIS+, add the mail host IP address line to the `/etc/hosts` file of all mail clients)
- Matching `/etc/mail/aliases` files on any system that has a local mailbox (unless you are running NIS or NIS+)
- Entries in each mail client's `/etc/vfstab` file or `/etc/auto_direct` (if autofs is used) to mount the `/var/mail` directory when mailboxes are located on the mail host
- Enough space in `/var/mail` on the mail server to hold the client mailboxes

Two Domains and a Gateway

The mail configuration shown in Figure 34–3 has two domains and a mail gateway. In this configuration, the mail server, the mail host, and the mail gateway (or gateways) for each domain are likely to be different systems. To assist the process of administering and distributing mail, a name service is used.

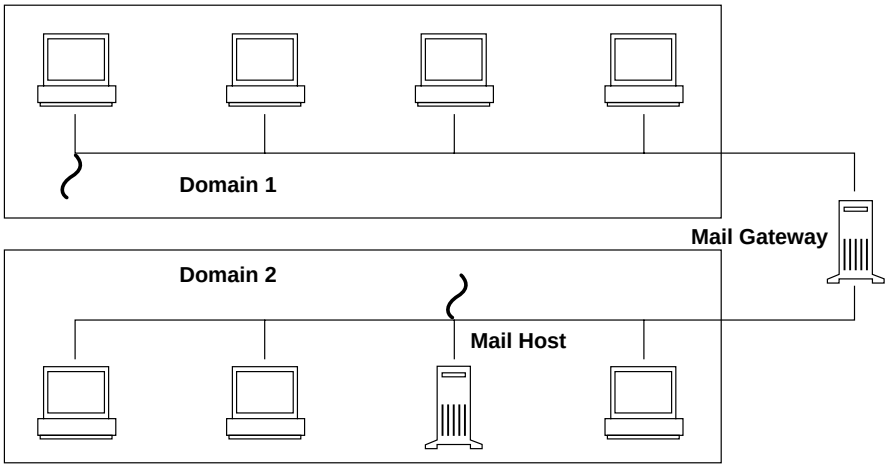


Figure 34–3 Two Domains and a Gateway

To set up this kind of a mail configuration, assuming that the mail clients mount their mail files from `/var/mail` on the mail host, you need:

- Complex gateway systems requiring a customized `sendmail.cf` file with special rules added
- The `main.cf` file on the mail gateway (no editing required if you use MX records)

- A server designated as the mail host (add `mailhost.`*domainname* to the `/etc/hosts` file on the mail host; if you are not running NIS or NIS+, add the mail host IP address line to the `/etc/hosts` file of all mail clients)
- Matching `/etc/mail/aliases` files on any system that has a local mailbox (unless you are running NIS or NIS+)
- An alias entry for each user, to point to where the mail is stored, in `mail_aliases.org_dir` for NIS+ or the aliases map for NIS
- The default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)
- Entries in each mail client's `/etc/vfstab` file or `/etc/auto_direct` (if autofs is used) to mount the `/var/mail` directory when mailboxes are located on the mail host
- Enough space in `/var/mail` on the mail server to hold the client mailboxes

Setting Up Mail Services

You can readily set up a mail service if your site does not provide connections to electronic mail (email) services outside your company or if your company is in a single domain.

Mail requires two types of configurations for local mail and two more for communication with networks outside of your domain. You can combine these configurations on the same system or provide them on separate systems. You need to set up systems on your site to perform the functions described in:

- "How to Set Up a Mail Server" on page 660
- "How to Set Up a Mail Client" on page 661
- "How to Set Up a Mail Host" on page 662
- "How to Set Up a Mail Gateway" on page 664
- "How to Use DNS With `sendmail`" on page 665

Before you begin to set up your mail service, choose the systems to act as mail servers, mail hosts, and mail gateways. You should also make a list of all the mail clients for which you are providing service and include the location of their mailboxes. This list will help you when you are ready to create mail aliases for your users. See Chapter 33 for more information about the function each of these systems provides. For your convenience, guidelines about which systems are good candidates for mail server, mail host, and mail gateways are repeated in the following sections.

To simplify the setup instructions, this chapter tells you what you need to do to set up individual mail servers, mail hosts, mail clients, and relay hosts. If a system in your mail services configuration is acting in more than one capacity, follow the

appropriate instructions for each type of system. For example, if your mail host and mail server functions are on the same system, follow the directions for setting up that system as a mail host and then follow the directions for setting up the same system as a mail server.

Note - The following procedures for setting up a mail server and mail client apply when mailboxes are NFS mounted. However, mailboxes typically are maintained in locally mounted `/var/mail` directories—in which case the following procedures are not needed.

▼ How to Set Up a Mail Server

No special steps are required to set up a mail server that is only serving mail for local users. The user must have an entry in the password file or in the name space, and the user should have a local home directory (so that `~/.forward` can be checked) for mail to be delivered. This is why home directory servers are often set up as the mail server.

The mail server can route all mail for many mail clients. The only resource requirement for this type of mail server is that it have adequate spooling space for client mailboxes. The `/var/mail` directory must be made available for remote mounting.

For this task, check that `/etc/dfs/dfstab` file shows the `/var/mail` directory is exported.

1. **Become superuser on the mail server.**

2. **Check that the `/var/mail` directory is available for remote access.**

   Type `share` and press Return. If the `/var/mail` directory is listed, you do not need to do more. If the `/var/mail` directory is not listed, continue with the next step.

3. **Make the `/var/mail` directory available for remote access.**

   Type the following command:

```
# share -F nfs /var/mail
```

4. **Make the file system permanently available for remote access.**

   Edit `/etc/dfs/dfstab` and add the command line used in step 2.

```
# cat /etc/dfs/dfstab
 . .
```

(continued)

```
share -F nfs -o rw /var/mail
```

Note - The `mail.local` program automatically creates mailboxes in the `/var/mail` directory the first time a message is delivered. You do not need to create individual mailboxes for your mail clients.

▼ How to Set Up a Mail Client

A mail client is a user of mail services, with a mailbox on a mail server, and a mail alias in the `/etc/mail/aliases` file that points to the location of the mailbox.

1. **Become superuser on the mail client's system.**

2. **Make sure a `/var/mail` mount point exists on the mail client's system.**

   Using `ls` tells you if the file system exists. The following example shows the response if the file system has not been created.

```
# ls -l /var/mail
/var/mail not found
```

   If mail files are in this directory, you should probably move them, so that they are not covered when the `/var/mail` directory is mounted from the server.

3. **Mount the `/var/mail` directory from the mail server.**

   The mail directory can be automatically mounted or mounted at boot time.

   a. **(Optional) Mount `/var/mail` automatically.**

      Edit `/etc/auto_direct` and add an entry like this one:

```
/var/mail -rw,hard,actimeo=0 server:/var/mail
```

   b. **(Optional) Mount `/var/mail` at boot time.**

      Edit the `/etc/vfstab` file and add an entry for the `/var/mail` directory on the mail server, mounting it on the local `/var/mail` directory.

```
server:/var/mail - /var/mail nfs - no rw,hard,actimeo=0
```

The client's mailbox is automatically mounted any time the system is rebooted. Type `mountall` to mount the client mailbox until the system is rebooted.

Caution - You must include the `actimeo=0` option when mounting mail from an NFS server to allow mailbox locking and access to work properly.

4. **Update `/etc/hosts`.**

   Use `admintool` to edit the `/etc/hosts` file and add an entry for the mail server. This step is not required if you are using a name service.

5. **Add an entry for the client to one of the alias files.**

   See "Administering Mail Alias Files" on page 666 for information about how to create mail aliases for different kinds of mail configurations.

   Note - The `mail.local` program automatically creates mailboxes in the `/var/mail` directory the first time a message is delivered. You do not need to create individual mailboxes for your mail clients.

6. **Restart sendmail.**

▼ How to Set Up a Mail Host

A mail host resolves email addresses and reroutes mail within your domain. A good candidate for a mail host is a system that connects your systems to the outside world or to a parent domain.

1. **Become superuser on the mail host system.**

2. **Verify the host name configuration.**

   Run the `check-hostname` script to verify if `sendmail` will be able to identify the fully qualified host name for this server:

```
% /usr/lib/mail/sh/check-hostname
hostname phoenix OK: fully qualified as phoenix.eng.acme.com
```

If this script is not successful in identifying the fully qualified host name, you need to add the fully qualified hos tname as the first alias for the host in `/etc/hosts`.

3. Update `/etc/hosts`.

Use `admintool` to edit the `/etc/hosts` file. Add the word `mailhost` and `mailhost.`*domainname* after the IP address and system name of the mail host system. The system is designated as a mail host. The *domainname* should be identical to the string given as the subdomain name in the output of the following command:

```
% /usr/lib/sendmail -bt -d0 </dev/null
Version 8.9.0+Sun
 Compiled with: MAP_REGEX LOG MATCHGECOS MIME7TO8 MIME8TO7 NAMED_BIND
                NDBM NETINET NETUNIX NEWDB NIS NISPLUS QUEUE SCANF SMTP
                USERDB XDEBUG

============ SYSTEM IDENTITY (after readcf) ============
      (short domain name) $w = phoenix
  (canonical domain name) $j = phoenix.eng.acme.com
         (subdomain name) $m = eng.acme.com
              (node name) $k = phoenix
========================================================
```

Here is an example of how the hosts file should look after these changes:

```
# cat /etc/hosts
#
# Internet host table
#
127.0.0.1       localhost
129.0.0.1       phoenix mailhost mailhost.eng.acme.com        loghost
```

4. Create an entry for the new mail host in the appropriate hosts file.

If you are using NIS or NIS+, add an entry including a host alias called `mailhost` and `mailhost.`*domainname* to the host entry for the new mail host.

If you are not using NIS or NIS+, you must create an entry in `/etc/hosts` for each system on the network. The entry should use this format:
IP_address mailhost_name `mailhost mailhost.`*domainname*

5. **Change the correct configuration file.**

   This command copies and renames the `/etc/mail/main.cf` file.

```
# cp /etc/mail/main.cf /etc/mail/sendmail.cf
```

6. **Restart `sendmail` and test your mail configuration.**

   See "How to Test the Mail Configuration" on page 677 for information.

▼ How to Set Up a Mail Gateway

A mail gateway manages communication with networks outside of your domain. The mailer on the sending mail gateway can match the mailer on the receiving system.

A good candidate for a mail gateway is a system attached to Ethernet and phone lines or a system configured as a router to the Internet. You might want to configure the mail host or another system as mail gateway. You might choose to configure more than one mail gateway for your domain. If you have UUCP connections, you should configure the system (or systems) with UUCP connections as the mail gateway.

1. **Become superuser on the mail gateway.**

2. **Change the configuration file.**

   The following command copies and renames the `main.cf` file.

```
# cp /etc/mail/main.cf /etc/mail/sendmail.cf
```

3. **Verify the host name configuration.**

   Run the `check-hostname` script to verify if `sendmail` will be able to identify the fully qualified host name for this server:

```
# /usr/lib/mail/sh/check-hostname
hostname phoenix OK: fully qualified as phoenix.eng.acme.com
```

   If this script is not successful in identifying the fully qualified hos tname, you need to add the fully qualified host name as the first alias for the host in `/etc/hosts`.

4. **Restart `sendmail` and test your mail configuration.**

   See "How to Test the Mail Configuration" on page 677 for information.

▼ How to Use DNS With sendmail

The DNS name service does not support aliases for individuals. It does support aliases for hosts or domains using *mail exchange* (MX) *records* and cname records. You can specify host names, domain names, or both in the DNS database. See the *Solaris Naming Setup and Configuration Guide* for more information about administering DNS.

1. **Become superuser.**

2. **Enable DNS host lookups (NIS+ only).**

   Edit the `/etc/nsswitch.conf` file and remove the # from the hosts definition that includes the `dns` flag. The host entry must include the `dns` flag, as shown below, for the DNS host aliases to be used.

```
# grep hosts /etc/nsswitch.conf
#hosts:      nisplus [NOTFOUND=return] files
hosts:      nisplus dns [NOTFOUND=return] files
```

3. **Check for a `mailhost` and `mailhost.`*domainname* entry.**

   Make sure an entry exists for `mailhost` and `mailhost.`*domainname* in the DNS database.

Building a sendmail Configuration File

The process to create `sendmail` configuration files has been changed. For many sites, This change should assist the administration of the configuration files. Although it is still acceptable to use an older version of `sendmail.cf` files, it would be best to move to the new system as soon as is reasonable. A complete description of the new process is described in `/usr/lib/mail/README`.

▼ How to Build a New sendmail.cf File

1. **Become superuser.**

2. **Make a copy of the configuration files that you want to change.**

```
# cd /usr/lib/mail/cf
# cp main-v7sun.mc myhost.mc
```

3. **Edit the new configuration files as needed (for example *myhost*.mc).**

4. **Build the configuration file using** m4**.**

```
# /usr/ccs/bin/make myhost.cf
```

5. **Test the new configuration file using the** –C **option to specify the new file.**

```
# /usr/lib/sendmail -C myhost.cf -v testaddr </dev/null
```

This command sends a message to `testaddr` while displaying messages as it runs. Only outgoing mail can be tested without restarting the `sendmail` service on the system. For systems that are not handling mail yet, use the full testing procedure found in "How to Test the Mail Configuration" on page 677.

6. **Install the new configuration file, after making a copy of the original.**

```
# cp /etc/mail/sendmail.cf /etc/mail/sendmail.cf.save
# cp myhost.cf /etc/mail/sendmail.cf
```

7. **Restart the sendmail service.**

```
# pkill -HUP sendmail
```

Administering Mail Alias Files

Mail aliases must be unique within the domain. This section tells you how to use command lines to search the mail aliases table for aliases, and to create mail aliases

for NIS+, NIS, or on the local system. Or you can use the AdminTool's Database Manager application to perform these tasks on the aliases database.

In addition, database files can be created for the local mail host using `makemap`. Using these database files does not provide the all of the advantages of using a name space like NIS or NIS+, but retrieving the data should be faster than using local files.

▼ How to List the Contents of an NIS+ Aliases Table

To use the `aliasadm` command, you must be either `root`, a member of the NIS+ group that owns the `mail_aliases` table, or the person who created the table.

Example—Listing All of the NIS+ `mail_aliases` Table

♦ **Type** `aliasadm -l` **and press Return.**

This lists the contents of the aliases table in alphabetical order by alias.

Note - If you have a large aliases table, listing the entire contents can take some time. If you are searching for a specific entry, pipe the output through the `grep` command (`aliasadm -l | grep` *entry*) so that you can use the `grep` search capability to find specific entries.

Example—Listing Individual Entries in the NIS+ `mail_aliases` Table

♦ **Type** `aliasadm -m` ***alias*** **and press Return.**

The alias entry is listed.

```
# aliasadm -m ignatz
ignatz: ignatz@saturn # Alias for Iggy Ignatz
```

Note - The `aliasadm -m` option matches only the complete alias name. It does not match partial strings. You cannot use metacharacters (like * and ?) with the `aliasadm -m` option. If you are interested in partial matches, type `aliasadm -l | grep` *partial-string* and press Return.

▼ How to Add Aliases to a NIS+ `mail_aliases` Table From the Command Line

1. **Compile a list of each of your mail clients, the locations of their mailboxes, and the names of the mail server systems.**

2. **Become root on any system.**

3. **Optional: If necessary, initiate a NIS+ table**

   If you are creating a completely new NIS+ `mail_aliases` table, you must first initiate the table. Type `aliasadm -I` and press Return.

4. **For each alias, type** `aliasadm -a` ***alias expanded_alias*** [***options comments***] **and press Return.**

   This adds the aliases to the NIS+ aliases table.

   ```
   # aliasadm -a iggy iggy.ignatz@saturn "Iggy Ignatz"
   ```

5. **Type** `aliasadm -m` ***alias*** **and press Return.**

   This displays the entry you created.

6. **Check that the entry is correct.**

▼ How to Add Entries by Editing a NIS+ `mail_aliases` Table

If you are adding more than two or three aliases, you might want to edit the NIS+ table directly.

1. **Compile a list of each of your mail clients, the locations of their mailboxes, and the names of the mail server systems.**

2. **Become root on any system.**

3. **Type** `aliasadm -e` **and press Return.**

   The aliases table is displayed using the editor set with the `$EDITOR` environment variable. If the variable is not set, `vi` is the default editor.

4. **Type each alias on a separate line, using these formats:**

   a. **Type the aliases in any order, at any place in the table.**

      The order is not important to the NIS+ mail_aliases table. The `aliasadm -l` command sorts the list and displays them in alphabetical order.

b. Use the format *alias*: *expanded_alias* # ["*option*"# "*comments*"]
If you leave the option column blank, type an empty pair of quotation marks ("") and then add the comments.

c. End each line by pressing Return.

5. Check that the entries are correct.

6. Save the changes.

▼ How to Change Entries in a NIS+ `mail_aliases` Table

1. Become root on any system.

2. Type `aliasadm -m` *alias* and press Return.
The information for the alias is displayed.

3. Type `aliasadm -c` *alias expanded_alias* [*options comments*] and press Return.
The alias is changed using the new information you provide.

4. Type `aliasadm -m` *alias* and press Return.
The entry you created is displayed.

5. Check that the entry is correct.

▼ How to Delete Entries From a NIS+ `mail_aliases` Table

1. Become root on any system.

2. Type `aliasadm -d` *alias* and press Return.
The alias is deleted from the NIS+ `mail_aliases` table.

▼ How to Set Up NIS `mail.aliases` Map

1. **Compile a list of each of your mail clients, the locations of their mailboxes, and the names of the mail server systems.**
2. **Become root on the NIS master server.**
3. **Edit the `/etc/mail/aliases` file, and make the following entries:**
   a. **Add an entry for each mail client.**
   b. **Change the entry `Postmaster: root` to the mail address of the person who is designated as postmaster.**
      See "How to Set Up the Postmaster Alias" on page 672 for more information.
   c. **If you have created a mailbox for administration of a mail server, create an entry for `root:`*mailbox@mailserver*.**
   d. **Save the changes.**
4. **Edit the `/etc/hosts` file on the NIS master server and create an entry for each mail server.**
5. **Type `cd /var/yp` and press Return.**
6. **Type `make` and press Return.**
   The changes in the `/etc/hosts` and `/etc/mail/aliases` files are propagated to NIS slave systems. It takes a few minutes, at most, for the aliases to take effect.

▼ How to Set Up a Local Mail Alias File

1. **Compile a list of each of your mail clients and the locations of their mailboxes.**
2. **Become root on the mail server.**
3. **Edit the `/etc/mail/aliases` file and make the following entries:**
   a. **Add an entry for each mail client.**
   b. **Change the entry `Postmaster: root` to the mail address of the person who is designated as postmaster.**
      See "How to Set Up the Postmaster Alias" on page 672 for more information.

c. **If you have created a mailbox for administration of a mail server, create an entry for** `root:` ***mailbox@mailserver*****.**

d. **Save the changes.**

4. **Type** `newaliases` **and press Return.**

   This creates an `alias` file in binary form that `sendmail` can use. The file is stored in the `/etc/mail/aliases.dir` and `/etc/mail/aliases.pag` files.

5. **Copy the** `/etc/mail/aliases`**, the** `/etc/mail/aliases.dir`**, and** `/etc/mail/aliases.pag` **files to each of the other systems.**

   When you copy all three files you do not need to run the `newaliases` command on each of the other systems.

   You can copy the files by using the `rcp` or `rdist` commands or by using a script that you create for this purpose. Remember that you must update all the `/etc/mail/aliases` files each time you add or remove a mail client.

▼ How to Create a Keyed Map File

1. **Using the editor of your choice, create the input file.**

   Entries can look like the following:

```
sandy@newdomain.com     ssmith@newdomain.com
ssmith@olddomain.com    error:nouser No such user here
@olddomain.com          %1@newdomain.com
```

   In this sample, the first entry redirects mail to a new alias; the second entry creates a message when an incorrect alias is used; and the last entry redirects all incoming mail from `olddomain` to `newdomain`.

2. **Make the database file.**

```
# /usr/sbin/makemap -o dbm newmap < newmap
```

–o	Append to instead of overwriting the file. See `makemap`(1M) for a list of the options available.
dbm	Selects the `dbm` database type. Other map types are `btree` or `hash`.
newmap	Is the name of the input file and the first part of the name of the database file. If the `dbm` database type is selected, then database files are created using a `.pag` and a `.dir` suffix. For the other two database types, the file name is followed by `.db`.

How to Set Up the Postmaster Alias

Every system should be able to send mail to a `postmaster` mailbox. You can create a NIS or NIS+ alias for `postmaster` or create one in each local `/etc/mail/aliases` file. Here is the default `/etc/mail/aliases` entry:

```
# Following alias is required by the mail protocol, RFC 822
# Set it to the address of a HUMAN who deals with this system's
# mail problems.
Postmaster: root
```

To create the `postmaster` alias, edit each system's `/etc/mail/aliases` file and change `root` to the mail address of the person who acts as postmaster.

You might want to create a separate mailbox for the postmaster to keep postmaster mail separate from personal mail. If you create a separate mailbox, use the mailbox address instead of the postmaster's mail address when you edit the `/etc/mail/aliases` files.

▼ How to Create a Separate Mailbox for `postmaster`

1. **Create a user account for the person designated as `postmaster` and put an asterisk (*) in the password field.**

2. **After mail has been delivered, type `mail -f` *postmaster* and press Return.**

   The `mail` program will be able to read and write to the mailbox name.

▼ How to Add the `postmaster` Mailbox to the Aliases

1. **Become root and edit the `/etc/mail/aliases` file on each system.**
   If your network does not run NIS or NIS+, edit the `/etc/mail/aliases` file.

2. **Change the `postmaster` alias from `root` to `Postmaster:` *postmastermailbox@postmasterhost* and save the changes.**

3. **On the postmaster's local system create an entry in the `/etc/mail/aliases` file that defines the name of the alias (`sysadmin`, for example) and includes the path to the local mailbox.**

4. **Type `newaliases` and press Return.**
   Or you could change the `Postmaster:` entry in the `aliases` file to a `Postmaster:` */usr/somewhere/somefile* entry.

Administering the Mail Queue

This section describes how to keep the mail service running smoothly.

▼ How to Display the Mail Queue

You can print the contents of the queue with `mailq`. This command is equivalent to specifying the `-bp` flag to `sendmail`.

♦ **Type `/usr/bin/mailq | more` and press Return.**

A list of the queue IDs, the size of the message, the date the message entered the queue, the message status, and the sender and recipients are displayed.

▼ How to Force Mail Queue Processing

♦ **Type `/usr/lib/sendmail -q -v` and press Return.**

This forces the processing of the queue and displays progress of the jobs as the queue is cleared.

▼ How to Run a Subset of the Mail Queue

♦ **Type `/usr/lib/sendmail -qR`*string* and press Return.**

You can run a subset of the queue at any time with the `-qR`*string* (run queue where any recipient name matches *string*) or with `-qI`*nnnnn* (run just one message with queue ID *nnnnn*). The *string* can also match host names, so any substring of *user@host.domain* will match.

This example processes everything in the queue for recipient `wnj`.

```
# /usr/lib/sendmail -qRwnj
```

▼ How to Move the Mail Queue

1. **Become root on the mail host.**

2. **Type `/etc/init.d/sendmail stop` and press Return.**

   This kills the old `sendmail` daemon to keep it from trying to process the old queue directory.

3. **Type `cd /var/spool` and press Return.**

4. **Type `mv mqueue omqueue; mkdir mqueue` and press Return.**

   This moves the directory, `mqueue`, and all its contents to the `omqueue` directory and then creates a new empty `Rmqueue` directory.

5. **Type `chmod 755 mqueue; chown daemon.daemon mqueue` and press Return.**

   These commands set the permissions of the directory to read/write/execute by owner, and read/execute by group and others; these commands also set the owner and group to `daemon`.

6. **Type `/etc/init.d/sendmail start` and press Return.**

   This starts a new `sendmail` daemon.

▼ How to Run the Old Mail Queue

1. **Type `/usr/lib/sendmail -oQ/var/spool/omqueue -q` and press Return.**

   The `-oQ` flag specifies an alternate queue directory and the `-q` flag says to run every job in the queue. Use the `-v` flag if you want to see the verbose output displayed on the screen.

2. **When the queue is finally emptied type `rmdir /var/spool/omqueue` and press Return.**

   This removes the empty directory.

Administering `.forward` Files

This section contains several procedures related to `.forward` file administration. Because these files can be edited by users, the files can cause problems.

▼ How to Disable `.forward` Files

This procedure disables `.forward` files only for a particular host.

1. **Become root.**

2. **Make a copy of `/usr/lib/mail/domain/solaris-generic.m4` or your site-specific domain m4 file:**

```
# cd /usr/lib/mail/domain
# cp solaris-generic.m4 mydomain.m4
```

3. **Add the following line to the file you just created:**

```
define('confFORWARD_PATH','')dnl
```

   If a line already exists in the domain m4 file that you are using, replace the line.

4. **Build and install a new configuration file.**

   See "How to Build a New `sendmail.cf` File" on page 665 for a complete procedure.

▼ How to Change the `.forward` File Search Path

1. **Become root.**

2. **Make a copy of** `/usr/lib/mail/domain/solaris-generic.m4` **or your site-specific domain m4 file**:

```
# cd /usr/lib/mail/domain
# cp solaris-generic.m4 mydomain.m4
```

3. **Add a line like the following to the file you just created**:

```
define('confFORWARD_PATH','~z/.forward:/var/forward/$u')dnl
```

   If a line already exists in the domain m4 file that you are using, replace the line.

4. **Build and install a new configuration file.**

   See "How to Build a New `sendmail.cf` File" on page 665 for a complete procedure.

▼ How to Create and Populate `/etc/shells`

This file is not included in the standard release, so it must be added if users are to be allowed to use `.forward` files to forward mail to a program or to a file. You can create the file by hand by using `grep` to identify all of the shells listed in your password file, then enter them manually in the file, but it is easier to use the following procedure, which employs a script that can be downloaded.

1. **Download the script from** `http://www.sendmail.org/sun-specific/gen-etc-shells.html`**.**

2. **Become root.**

3. **To generate a list of shells, run the** `gen-etc-shells` **script.**

```
# ./gen-etc-shells.sh > /tmp/shells
```

   This script uses the `getent` command to collect the names of shells included in the password file sources listed in `/etc/nsswitch.conf`.

4. **Inspect the list of shells in** `/tmp/shells`**.**

   Using the editor of your choice, remove any shells that you do not want included.

5. **Move the file to** `/etc/shells`**.**

```
# mv /tmp/shells /etc/shells
```

Troubleshooting Tips for Mail

This section provides some tips and tools that you can use for troubleshooting problems with the mail services.

▼ How to Test the Mail Configuration

1. **Restart sendmail on any system for which you have changed a configuration file.**

```
# pkill -HUP sendmail
```

2. **Send test messages from each system by typing `/usr/lib/sendmail -v` *names* `</dev/null` and press Return.**

   Specify a recipient's email address in place of the *names* variable.

   This command sends a null message to the specified recipient and displays messages while it runs.

3. **Run these tests:**

   a. **Send mail to yourself or other people on the local system by addressing the message to a regular user name.**

   b. **If you are on Ethernet, send mail to someone on another system.**

      Do this in three directions: from the main system to a client system, from a client system to the main system, and from a client system to another client system.

   c. **If you have a mail gateway, send mail to another domain from the mail host to ensure that the relay mailer and host are configured properly.**

   d. **If you have set up a UUCP connection on your phone line to another host, send mail to someone at that host and have that person send mail back or call you when the message is received.**

   e. **Ask someone to send mail to you over the UUCP connection.**

The `sendmail` program cannot tell whether the message gets through because it hands the message to UUCP for delivery.

f. Send a message to `postmaster` on different systems and make sure that it comes to your postmaster's mailbox.

▼ How to Check Mail Aliases

To verify aliases and whether mail can be delivered to a given recipient:

♦ **Type `/usr/lib/sendmail -v -bv` *recipient* and press Return.**

The command displays the aliases and identifies the final address as deliverable or not.

Here is an example of the output:

```
% /usr/lib/sendmail -v -bv sandy
sandy... aliased to   ssmith
ssmith... aliased to              sandy@phoenix
sandy@phoenix... deliverable: mailer esmtp, host phoenix, user sandy@phoenix.eng.acme.com
%
```

You should take extra care to avoid loops and inconsistent databases when both local and domain-wide aliases are used. Be especially careful when you move a user from one system to another to avoid creating alias loops.

▼ How to Test the `sendmail` Rule Sets

1. **Type `/usr/lib/sendmail -bt` and press Return.**
   Information is displayed.
2. **At the last prompt (>) type a `3,0` (zero) and the mail address you want to test.**
3. **Type Control-d to end the session.**

Here is an example of the output:

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 sandy@phoenix
rewrite: ruleset   3   input: sandy @ phoenix
rewrite: ruleset  96   input: sandy < @ phoenix>
rewrite: ruleset  96 returns: sandy < @ phoenix . eng . acme . com . >
rewrite: ruleset   3 returns: sandy < @ phoenix . eng . acme . com . >
rewrite: ruleset   0   input: sandy < @ phoenix . eng . acme . com . >
rewrite: ruleset 199   input: sandy < @ phoenix . eng . acme . com . >
rewrite: ruleset 199 returns: sandy < @ phoenix . eng . acme . com . >
rewrite: ruleset  98   input: sandy < @ phoenix . eng . acme . com . >
rewrite: ruleset  98 returns: sandy < @ phoenix . eng . acme . com . >
rewrite: ruleset 198   input: sandy < @ phoenix . eng . acme . com . >
rewrite: ruleset 198 returns: $# local $: sandy
rewrite: ruleset   0 returns: $# local $: sandy
```

▼ How to Verify Connections to Other Systems

To verify connections to other systems, you can use the `mconnect` program to open connections to other `sendmail` systems over the network. The `mconnect` program runs interactively. You can issue various diagnostic commands. See the `mconnect`(1) man page for a complete description. The following example verifies that mail to the user name `shamira` is deliverable.

```
$ mconnect phoenix
connecting to host phoenix (129.144.52.96), port 25
connection open
220 phoenix.Eng.Acme.COM Sendmail 8.9.0+Sun/8.9.0; Tue, 25 Jul 1998 10:45:28 -0700
vrfy sandy
250 Sandy Smith <sandy@phoenix.Eng.Acme.COM>
>
```

If you cannot use `mconnect` to connect to an `SMTP` port, check these conditions:

- Is the system load too high?
- Is the `sendmail` daemon running?
- Does the system have the appropriate `/etc/mail/sendmail.cf` file?
- Is port 25 (the port that `sendmail` uses) active?

System Log

The mail services log most errors using the `syslogd` program. The default is for `syslogd` to send messages to the `loghost`.

You can define a system called `loghost` in the `/etc/hosts` file to hold all logs for an entire NIS domain. The system log is supported by the `syslogd` program. You specify a `loghost` in `/etc/hosts`. If no `loghost` is specified, error messages from `syslogd` are not reported.

Code Example 34–1 shows the default `/etc/syslog.conf` file.

CODE EXAMPLE 34–1 Default `/etc/syslog.conf` File

```
#ident "@(#)syslog.conf   1.3        93/12/09 SMI"  /* SunOS 5.0 */  #
# Copyright (c) 1994 by Sun Microsystems, Inc.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (‘’) names
# that match m4 reserved words. Also, within ifdef’s, arguments
# containing commas must be quoted.
#
# Note: Have to exclude user from most lines so that user.alert
#       and user.emerg are not included, because old sendmails
#       have no 4.2BSD based systems doing network logging, you
#       can remove all the special cases for "user" logging.
# *.err;kern.debug;auth.notice;user.none         /dev/console
*.err;kern.debug;daemon,auth.notice;mail.crit;user.none /var/adm/messages
*.alert;kern.err;daemon.err;user.none            operator
*.alert;user.none                                root
*.emerg;user.none                                *
# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.notice         ifdef(‘LOGHOST’, /var/log/authlog, @loghost)
mail.debug           ifdef(‘LOGHOST’, /var/log/syslog, @loghost)
#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(‘LOGHOST’, ,
user.err             /dev/console
user.err             /var/adm/messages
user.alert           ‘root, operator’
user.emerg           *
)
```

You can change the default configuration by editing the `/etc/syslog.conf` file. You must restart the `syslog` daemon for any changes to take effect. You can add these selections to the file to gather information about mail:

- `mail.alert` – Messages about conditions that should be fixed now
- `mail.crit` – Critical messages
- `mail.warning` – Warning messages
- `mail.notice` – Messages that are not errors, but might need attention
- `mail.info` – Informational messages

- `mail.debug` – Debugging messages

The following entry sends a copy of all critical, informational, and debug messages to `/var/log/syslog`.

```
mail.crit;mail.info;mail.debug   /var/log/syslog
```

Each line in the system log contains a timestamp, the name of the system that generated it, and a message. The `syslog` file can log a large amount of information.

The log is arranged as a succession of levels. At the lowest level, only unusual occurrences are logged. At the highest level, even the most mundane and uninteresting events are recorded. As a convention, log levels under 10 are considered "useful." Log levels higher than 10 are usually used for debugging. See the "mconnect" in *System Administration Guide, Volume 2* for information about `loghost` and the `syslogd` program.

Other Mail Diagnostic Information

For other diagnostic information, check the following sources:

- Look at the `Received` lines in the header of the message. These lines trace the route the message took as it was relayed. Iin the UUCP network many sites do not update these lines, and in the Internet the lines are rearranged. To straighten them out, look at the date and time in each line. Remember to account for time–zone differences.
- Look at the messages from `MAILER-DAEMON`. These typically report delivery problems.
- Check the system log that records delivery problems for your group of systems. The `sendmail` program always records what it is doing in the system log. You might want to modify the `crontab` file to run a shell script nightly that searches the log for `SYSERR` messages and mails any that it finds to the postmaster.
- Use the `mailstats` program to test mail types and determine the number of incoming and outgoing messages.

CHAPTER 35

Mail Services Reference

The `sendmail` program is a mail transport agent that uses a configuration file to provide aliasing and forwarding, automatic routing to network gateways, and flexible configuration. The Solaris operating environment supplies standard configuration files that most sites can use. Chapter 34 explains how to set up an electronic mail system using the standard files. This chapter describes some of the differences between the generic version of `sendmail` and the Solaris version.

Solaris `sendmail` Differences

This section describes some of the changes included in the Solaris version of `sendmail` as compared to the generic Berkeley version.

Flags Used When Compiling `sendmail`

The following table lists the flags used when compiling the version of `sendmail` delivered with the Solaris 8 release. If your configuration requires other flags, you need to download the source and recompile the binary yourself. Information about this process can be found at `http://www.sendmail.org`.

Flag	Description
SOLARIS=20800	Support for the Solaris 8 operating environment
NDBM	Support for ndbm databases
NEWDB	Support for db databases
NIS	Support for nis databases
NISPLUS	Support for nisplus databases
LDAPMAP	Support for LDAP maps
USERDB	Support for the User database
MAP_REGEX	Support for regular expression maps
SUN_EXTENSIONS	Solaris flag; support for Sun extensions included in `sun_compat.o`
VENDOR_DEFAULT=VENDOR_SUN	Solaris flag; selects Sun as the default vendor
USE_VENDOR_CF_PATH	Solaris flag; allows for the configuration file to be placed in `/etc/mail`
_FFR_MAXALIASRECURSION_OPTION	Solaris flag; enables selection of MaxAliasRecursion option
_FFR_MAX_MIME_HEADER_LENGTH	Solaris flag; enables selection of MaxMimeHeaderLength option

Alternative `sendmail` Commands

The Solaris release does not include all of the command synonyms that are provided in the generic release from Berkeley. This table includes a complete list of the command aliases, whether they are included in the Solaris release, and how to generate the same behavior using `sendmail`.

TABLE 35–1 Alternate `sendmail` Commands

Alternate Name	Included in Solaris?	Options With `sendmail`
`hoststat`	no	`sendmail –bh`
`mailq`	yes	`sendmail –bp`

TABLE 35–1 Alternate `sendmail` Commands *(continued)*

Alternate Name	Included in Solaris?	Options With `sendmail`
`newaliases`	yes	`sendmail –bi`
`purgestat`	no	`sendmail –bH`
`smtpd`	no	`sendmail –bd`

Define Configuration File Version

The new version of sendmail (version 8.9.3) includes a new configuration option that defines the version of the `sendmail.cf` file. This will allow older configuration files to be used with Version 8.9.3 `sendmail`. You can set the version level to values between `0` and `8`. You can also define the vendor. Either Berkeley or Sun are valid vendor options. If the V option is not defined in the configuration file, the default setting is `V1/Sun`. If a version level is specified but no vendor is defined, `Sun` is used as the default vendor setting. The following table lists some of the valid options.

TABLE 35–2 Configuration File Version Values

Field	Description
`V1/Sun`	Use Solaris extensions of name service support. This option allows for old configuration files to be used with the new version of `sendmail`. This is the default setting if nothing is specified.
`V7/Sun`	Use for Version 8.8 of `sendmail`.
`V8/Sun`	Use for Version 8.9.3 of `sendmail`. This is the setting that is included inthe prebuilt configuration file in the Solaris 8 release.

Mail Services Terminology

In addition to the mail files and programs, many other components are required to establish a mail service. The following sections define these components and some of the terminology used to describe them.

The first section defines the terminology used when discussing the software parts of the mail delivery system. The next section focuses on the functions of the hardware systems in a mail configuration.

Mail Services Software Terminology

This section describes the software components of a mail system. Each service includes at least one of each of the following:

- Mail user agent
- Mail transfer agent
- Mail delivery agent

Other software components include domain names, mail addresses, mailboxes, and mail aliases.

Mail User Agent

The *mail user agent* is the program that acts as the interface between the user and mail transfer agent, such as the `sendmail` program. The mail user agents supplied with the Solaris operating environment are `/usr/bin/mail`, `/usr/bin/mailx`, `$OPENWINHOME/bin/mailtool`, and `/usr/dt/bin/dtmail`.

Mail Transfer Agent

The *mail transfer agent* is responsible for the routing of mail messages and resolution of mail addresses. This is also known as a mail *transport* agent. The transfer agent for the Solaris operating environment is `sendmail`. The transfer agent performs these functions:

- Accepts messages from the mail user agent
- Resolves destination addresses
- Selects a proper delivery agent to deliver the mail
- Receives incoming mail from other mail transfer agents

Mail Delivery Agent

A *mail delivery agent* is a program that implements a mail delivery protocol. The following mail delivery agents are provided with the Solaris operating environment:

- The UUCP mail delivery agent, which uses `uux` to deliver mail

- The local mail delivery agent, which is `mail.local` in the standard Solaris release

Mailers

A *mailer* is a `sendmail` specific term. You can customize a mail delivery agent. A mailer is used by `sendmail` to identify a specific instance of a customized mail delivery agent or a mail transfer agent.

You need to specify at least one mailer in the `sendmail.cf` file of all systems in your network.

The `smtp` mailer uses SMTP to transfer a message. SMTP is the standard mail protocol used on the Internet. This is an example of an SMTP mail header:

```
To: paul@phoenix.stateu.edu
From: Iggy.Ignatz@eng.acme.com
```

If mail is sent between two users in the same domain, the header looks like this:

```
To: Irving.Who@eng.acme.com
From: Iggy.Ignatz@eng.acme.com
```

Use SMTP for sending mail outside your domain, especially for mailboxes that you must reach through the Internet.

The `uucp-old` mailer uses `uux` to deliver messages, but it formats headers with a domain-style address, and the `To:` and `Cc:` lines are formatted by domain, much like the SMTP headers. The `uucp` headers look like this:

```
To: paul@phoenix.stateu.com
From: ignatz@eng.acme.com
```

Use `uucp-uudom` for UUCP mail to systems that can handle and resolve domain-style names. The sender also must be able to handle domain-style names and be able to receive replies from the Internet.

The `uucp-old` mailer uses an exclamation point address in the headers. This is one of the original mailers. The headers look like this:

```
To: edu!stateu!phoenix!paul
From: acme!ignatz
```

You can define other mail delivery agents by providing a mailer specification in the `sendmail.cf` file. Additional information about mailers can be found in `/usr/lib/mail/README`.

Domain Names

A *domain* is a directory structure for network address naming. Electronic-mail addressing also uses domains. An email address has this format:

```
user@subdomain. ... .subdomain2.subdomain1.top-level-domain
```

The part of the address to the left of the `@` sign is the local address. The local address can contain information about:

- Routing with another mail transport (for example, `bob::vmsvax@gateway` or `smallberries%mill.uucp@gateway`)
- An alias (for example, `iggy.ignatz`)

The receiving mailer is responsible for determining what the local part of the address means.

The part of the address to the right of the `@` sign shows the domain address where the local address is located. A dot separates each part of the domain address. The domain can be an organization, a physical area, or a geographic region.

Domain addresses are case insensitive. It makes no difference whether you use uppercase, lowercase, or mixed-case letters in the domain part of an address.

The order of domain information is hierarchical—the more local the address, the closer it is to the `@` sign.

The larger the number of subdomains, the more detailed the information that is provided about the destination. Just as a subdirectory in a file-system hierarchy is considered to be inside the directory above, each subdomain in the mail address is considered to be inside the location to its right.

The following table shows the top-level domains.

TABLE 35–3 Top-level Domains

Domain	Description
`Com`	Commercial sites
`Edu`	Educational sites
`Gov`	United States government installations
`Mil`	United States military installations
`Net`	Networking organizations
`Org`	Other organizations

!%@:: A Directory of Electronic Mail Addressing and Networks by Donnalyn Frey and Rick Adams (O'Reilly & Associates, Inc., 1993) contains a complete list of international top-level domain addresses; it is updated periodically.

For mail delivery, the name space domain name and the mail domain name occasionally do not match. However, the DNS domain name and the mail domain name must be identical. By default, the `sendmail` program strips the first component from the domain name to form the mail domain name. For example, if a NIS+ domain name were `bldg5.eng.acme.com`, its mail domain name would be `eng.acme.com`.

Note - Although mail domain addresses are case insensitive, the name space domain name is not. For best results use lowercase characters when setting up the mail and name space domain names.

Mail Address

The *mail address* contains the name of the recipient and the system to which the mail message is delivered.

When you administer a small mail system that does not use a name service, addressing mail is easy: login names uniquely identify users.

When, however, you are administering a mail system that has more than one system with mailboxes, one or more domains, or when you have a UUCP (or other) mail connection to the outside world, mail addressing becomes more complex. Mail addresses can be *route independent*, *route based*, or a mixture of the two. Route-based

addressing is based on old specifications and is not required or desired in most situations.

Route-Independent Addressing

Route-independent addressing requires the sender of an email message to specify the name of the recipient and the final destination address. Route-independent addresses usually indicate the use of a high-speed network like the Internet. In addition, newer UUCP connections frequently use domain-style names. Route-independent addresses can have this format:

```
user@host.domain
```

UUCP connections can use the following address format:

```
host.domain!user
```

The increased popularity of the domain hierarchical naming scheme for computers is making route-independent addresses more common. In fact, the most common route-independent address omits the host name and relies on the domain name service to properly identify the final destination of the email message:

```
user@domain
```

Route-independent addresses are read by searching for the @ sign, then reading the domain hierarchy from the right (the highest level) to the left (the most specific address to the right of the @ sign).

Route-Based Addressing

Route-based addressing requires the sender of an email message to specify the local address (typically, a user name) and its final destination, as well as the route that the message must take to reach its final destination. Route-based addresses were fairly common on UUCP networks, and have this format:

```
path!host!user
```

Whenever you see an exclamation point as part of an email address, all (or some) of the route was specified by the sender. Route-based addresses are always read from left to right.

For example, an email address that looks like this:

```
venus!acme!sierra!ignatz
```

means that mail sent to the user named `ignatz` is first sent to the system named `venus`, next to `acme`, and then to `sierra`. (This is an example and not an actual route.) If any of the mail handlers is down, the message will be delayed or returned as undeliverable.

Mail sent through the `uucp` mailer is not restricted to using route-based addressing. Some `uucp` mailers also handle route-independent addressing.

Mailbox

A *mailbox* is a file on a mail server that is the final destination for email messages. The name of the mailbox can be the user name or a place to put mail for someone with a specific function, like the postmaster. Mailboxes are in the `/var/mail/`*username* file, which can exist either on the user's local system or on a remote mail server. In either case, the mailbox is on the system to which the mail is delivered.

Mail should always be delivered to a local file system so that the user agent can pull mail from the mail spool and store it readily in the local mailbox. Do not use NFS mounted file systems as the destination for a user's mailbox. Specifically, do not direct mail to a mail client that is mounting the `/var/mail` file system from a remote server. Mail for the user, in this case, should be addressed to the mail server and not to the client host name. NFS mounted file systems can cause problems with mail delivery and handling. Clients that NFS mount `/var/mail` go into "remote mode" and should arrange to have the server send and receive mail for them.

The `/etc/mail/aliases` file and name services like NIS and NIS+ provide mechanisms for creating aliases for electronic mail addresses, so that users do not need to know the precise local name of a user's mailbox.

The following table shows some common naming conventions for special-purpose mailboxes.

TABLE 35–4 Conventions for the Format of Mailbox Names

Format	Description
username	User names are frequently the same as mailbox names.
Firstname.*Lastname* *Firstname_Lastname* *Firstinitial*.*Lastname* *Firstinitial_Lastname*	User names can be identified as full names with a dot (or an underscore) separating the first and last names, or by a first initial with a dot (or an underscore) separating the initial and the last name.

TABLE 35–4 Conventions for the Format of Mailbox Names *(continued)*

Format	Description
`postmaster`	Users can address questions and report problems with the mail system to the `postmaster` mailbox. Each site and domain should have a `postmaster` mailbox.
`MAILER-DAEMON`	`sendmail` automatically routes any mail addressed to the `MAILER-DAEMON` to the postmaster.
aliasname`-request`	Names ending in `-request` are administrative addresses for distribution lists. This address should redirect mail to the person who maintains the distribution list.
`owner-`*aliasname*	Names beginning with `owner-` are administrative addresses for distribution lists. This address should redirect mail to the person who handles mail errors.
`owner-owner`	This alias is used when there is no `owner-`*aliasname* alias for errors to be returned to. This address should redirect mail to the person who handles mail errors and should be defined on any system that maintains a large number of aliases.
local%domain	The percent sign (%) marks a local address that is expanded when the message arrives at its destination. Most mail systems interpret mailbox names with % characters as full mail addresses. The % is replaced with an @, and the mail is redirected accordingly. Although many people use the % convention, it is not a formal standard. It is referred to as the "percent hack." This feature is often used to help debug mail problems.

Starting with version 8, the envelope sender for mail sent to a group alias is changed to the address expanded from the owner alias, if an owner alias exists. This change allows for any mail errors to be sent to the alias owner rather than being returned to the sender. What users will notice is that mail they send to an alias, when delivered, will look like it came from the alias owner. The following alias format will help with some of the problems associated with this change:

```
mygroup: :include:/pathname/mygroup.list
owner-mygroup: mygroup-request
mygroup-request: sandys, ignatz
```

In this example, the `mygroup` alias is the actual mail alias for the group; the `owner-mygroup` alias receives error messages; and the `mygroup-request` alias should be used for administrative requests. This structure means that in mail sent to the `mygroup` alias, the envelope sender changes to `mygroup-request`.

Mail Aliases

An *alias* is an alternate name. For electronic mail, you can use aliases to assign a mailbox location or to define mailing lists.

For large sites, the mail alias typically defines the location of a mailbox. Providing a mail alias is like providing a mail stop as part of the address for an individual at a large corporation. If you do not provide the mail stop, the mail is delivered to a central address. Extra effort is required to determine where within the building the mail is to be delivered, and the possibility of error increases. For example, if two people named Kevin Smith are in the same building, only one of them will get mail.

Use domains and location-independent addresses as much as possible when you create mailing lists. To enhance portability and flexibility of alias files, make your alias entries in mailing lists as generic and system independent as possible. For example, if you have a user named `ignatz` on system `mars`, in domain `eng.acme.com`, create the alias `ignatz@eng` instead of `ignatz@mars`. If user `ignatz` changes the name of his system but remains within the engineering domain, you do not need to update alias files to reflect the change in system name.

When creating alias entries, type one alias per line. You should have only one entry that contains the user's system name. For example, you could create the following entries for user `ignatz`:

```
ignatz: iggy.ignatz
iggyi: iggy.ignatz
iggy.ignatz: ignatz@mars
```

You can create an alias for local names or domains. For example, an alias entry for user `fred` who has a mailbox on the system `mars` and who is in the domain `planets` could have this entry in the NIS+ aliases table:

```
fred: fred@planets
```

When creating mail lists that include users outside your domain, create the alias with the user name and the domain name. For example, if you have a user named `smallberries` on system `privet`, in domain `mgmt.acme.com`, create the alias as `smallberries@mgmt.acme.com`.

The email address of the sender is now automatically translated to a fully qualified domain name when mail goes outside the user's domain.

Uses for Aliases Files

You create mail aliases for global use in the NIS+ `mail_aliases` table, the NIS `aliases` map, or in local `/etc/mail/aliases` files. You can also create and administer mailing lists using the same alias files.

Depending on the configuration of your mail services, you can administer aliases by using the NIS or NIS+ name service to maintain a global `aliases` database or by updating all the local `/etc/mail/aliases` files to keep them synchronized.

Users can also create and use aliases. They can create aliases either in their local `~/.mailrc` file, which only they can use, or in their local `/etc/mail/aliases` file, which can be used by anyone. Users cannot normally create or administer NIS or NIS+ alias files.

Hardware Components of a Mail Configuration

A mail configuration requires three elements, which can be combined on the same system or provided by separate systems:

- A mail host
- At least one mail server
- Mail clients

When you want users to communicate with networks outside your domain, you must also add a fourth element, a mail gateway. The following sections describe each hardware component.

Mail Host

A *mail host* is the machine that you designate as the main mail machine on your network. It is the machine to which other systems at the site forward mail that they cannot deliver. You designate a system as a mail host in the `hosts` database by adding the word `mailhost` to the right of the IP address in the local `/etc/hosts` file or in the hosts file in the name service. You must also use the `main.cf` file as the mail-configuration file on the mail host system.

A good candidate for mail host is a system on the local area network that also has a modem for setting up PPP or UUCP links over telephone lines. Another good candidate is a system configured as a router from your network to the Internet global network. (See Chapter 21, Chapter 25, and “Configuring Routers” on page 105 for more information.) If none of the systems on your local network has a modem, designate one as the mail host.

Some sites use standalone machines that are not networked in a time-sharing configuration; that is, the standalone machine serves terminals attached to its serial ports. You can set up electronic mail for this configuration by treating the standalone system as the mail host of a one-system network.

Mail Server

A *mailbox* is a single file that contains email for a particular user. Mail is delivered to the system where the user's mailbox resides: the local machine or a remote server. A *mail server* is any system that maintains user mailboxes in its `/var/mail` directory.

The mail server routes all mail from a client. When a client sends mail, the mail server puts it in a queue for delivery. After the mail is in the queue, a user can reboot or turn off the client without losing those mail messages. When the recipient gets mail from a client, the path in the "From " line of the message contains the name of the mail server. If the recipient responds, the response goes to the user's mailbox. Good candidates for mail servers are systems that provide a home directory for users or that are backed up regularly.

If the mail server is not the user's local system, users in configurations using NFS software can mount the `/var/mail` directory by using the `/etc/vfstab` file (if they have root access) or by using the automounter. If NFS support is not available, the users can log in to the server to read their mail.

If users on your network send other types of mail, such as PostScript™ files, audio files, or files from desktop publishing systems, you need to allocate more space on the mail server for mailboxes.

One advantage to establishing a mail server for all mailboxes is that it makes backups easy. Having mail spread over many systems makes it hard to do backups. The disadvantage of storing many mailboxes on one server is that the server can be a single point of failure for many users, but the advantages of providing good backups usually make the risk worthwhile.

Mail Client

A *mail client* is any system that receives mail on a mail server and does not have a local `/var/mail` directory. This is known as remote mode. Remote mode is enabled by default in `/etc/mail/subsidiary.cf`.

You must check that the mail client has the appropriate entry in the `/etc/vfstab` file and a mount point to mount the mailbox from the mail server. Also make sure that the alias for the client is directed to the mail server's host name, not to the client's.

Mail Gateway

The *mail gateway* is a machine that handles connections between networks running different communications protocols or communications between different networks using the same protocol. For example, a mail gateway might connect a TCP/IP network to a network running the Systems Network Architecture (SNA) protocol suite.

The simplest mail gateway to set up is one that connects two networks that use the same protocol or mailer. This system handles mail with an address for which `sendmail` cannot find a recipient in your domain. If a mail gateway exists, `sendmail` uses it for sending and receiving mail outside your domain.

You can set up a mail gateway between two networks using unmatched mailers, as shown in the next figure. To support this, you must customize the `sendmail.cf` file on the mail gateway system, which can be a difficult and time-consuming process.

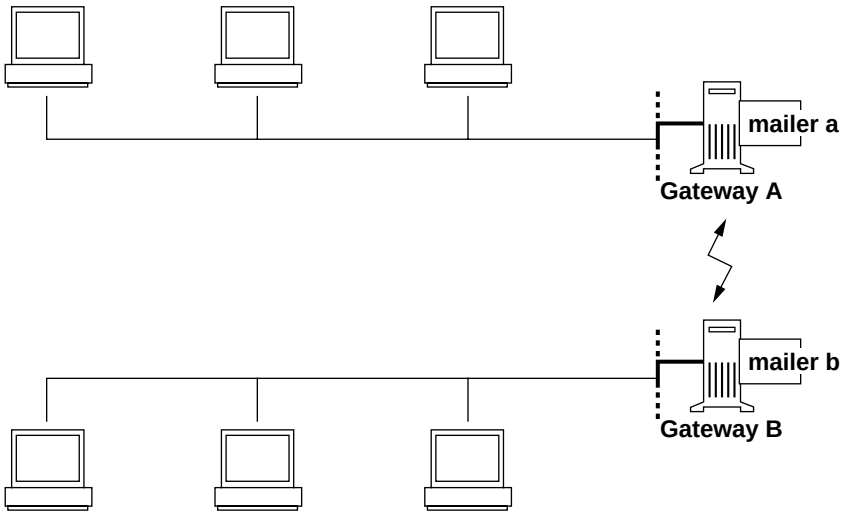


Figure 35–1 Gateway Between Different Communications Protocols

If you have to set up a mail gateway, you should find a gateway-configuration file that is close to what you need and modify it to fit your situation.

If you have a machine providing connections to the Internet, you can configure that machine as the mail gateway. Carefully consider your site's security needs before you configure a mail gateway. You might need to create a firewall gateway between your corporate network and the outside world, and set that up as the mail gateway.

Mail Service Programs and Files

Mail services include many programs and daemons that interact with each other. This section introduces the programs and the terms and concepts related to administering electronic mail. Table 35–5 shows the contents of the `/usr/bin` directory that are used for mail services.

TABLE 35–5 Contents of the /usr/bin Directory Used for Mail Services

Name	Type	Description
aliasadm	File	A program to manipulate the NIS+ aliases map
mail	File	A user agent
mailcompat	File	A filter to store mail in SunOS 4.1 mailbox format
mailq	Link	Link to /usr/lib/sendmail; used to list the mail queue
mailstats	File	A program used to read mail statistics stored in the /etc/mail/sendmail.st file (if present)
mailx	File	A user agent
mconnect	File	A program that connects to the mailer for address verification and debugging
newaliases	Link	Link to /usr/lib/sendmail; used to create the binary form of the alias database
praliases	File	A command to "uncompile" the alias database
rmail	Link	Link to /usr/bin/mail; command often used to permit only the sending of mail
vacation	File	A command to set up an automatic reply to mail

Table 35–6 shows the contents of the /etc/mail directory.

TABLE 35–6 Contents of the /etc/mail Directory

Name	Type	Description
Mail.rc	File	Default settings for the mailtool user agent
aliases	File	Mail-forwarding information
aliases.dir	File	Binary form of mail-forwarding information (created by running newaliases)
aliases.pag	File	Binary form of mail-forwarding information (created by running newaliases)

TABLE 35–6 Contents of the `/etc/mail` Directory *(continued)*

Name	Type	Description
`mailx.rc`	File	Default settings for the `mailx` user agent
`main.cf`	File	Sample configuration file for main systems
`relay-domains`	File	Contains a list of all domains for which relaying is allowed; by default, only the local domain is allowed
`sendmail.cf`	File	Configuration file for mail routing
`sendmail.cw`	File	Optional file that you can create if the number of aliases for the mail host is too long
`sendmail.hf`	File	Help file used by the SMTP `HELP` command
`sendmail.pid`	File	File that lists the PID of the listening daemon
`sendmail.st`	File	The `sendmail` statistics file; if this file is present, `sendmail` logs the amount of traffic through each mailer
`sendmailvars`	File	Stores macro and class definitions for name space lookup from `sendmail.cf`
`subsidiary.cf`	File	Sample configuration file for subsidiary systems

Table 35–7 shows the contents of the `/usr/lib` directory that are used for mail services.

TABLE 35–7 Contents of the `/usr/lib` Directory Used for Mail Services

Name	Type	Description
`mail.local`	File	Mailer that delivers mail to mailboxes
`sendmail`	File	The routing program, also known as the mail transfer agent
`smrsh`	File	Shell program to restrict programs that `sendmail` can run to those in `/var/adm/sm.bin`

Within the /usr/lib directory is a subdirectory that contains all of the files needed to build a sendmail.cf file. The contents of this directory are shown in Table 35–8.

TABLE 35–8 Contents of the /usr/lib/mail Directory Used for Mail Services

Name	Type	Description
README	File	Document describing the configuration files
cf	Directory	Site-dependent and site-independent descriptions of hosts
cf/main-v7sun.mc	File	Main configuration file
cf/makefile	File	Contains rules for building new configuration files
cf/subsidiary-v7sun.mc	File	Configuration file for hosts that NFS mount /var/mail from another host
domain	Directory	Site-dependent subdomain descriptions
domain/generic.m4	File	Generic domain file from Berkeley
domain/solaris-antispam.m4	File	Domain file with changes that make sendmail function like previous Solaris versions, except that relaying is disabled completely, sender addresses with no host name are rejected, and unresolvable domains are rejected
domain/solaris-generic.m4	File	Domain file with changes that make sendmail function like previous Solaris versions (default)
feature	Directory	Definitions of specific features for particular hosts (see README for a full description of the features)
m4	Directory	Site-independent include files
mailer	Directory	Definitions of mailers, which include local, smtp, and uucp
ostype	Directory	Definitions describing various operating system environments

TABLE 35–8 Contents of the `/usr/lib/mail` Directory Used for Mail Services *(continued)*

Name	Type	Description
`ostype/solaris2.m4`	File	Defines local mailer as `mail`
`ostype/solaris2.ml.m4`	File	Defines local mailer as `mail.local` (default)
`sh`	Directory	Shell scripts used by the `m4` build process and migration aids
`sh/check-permissions`	File	Checks permissions of `:include:` aliases and `.forward` files and their parent directory path for correct permissions
`sh/check-hostname`	File	Verifies that `sendmail` is able to determine the fully qualified host name

Several other files and directories are used by the mail services, as shown in Table 35–9.

TABLE 35–9 Other Files Used for Mail Services

Name	Type	Description
`sendmailvars.org_dir`	Table	NIS+ version of `sendmailvars` file
`/etc/default/sendmail`	File	Lists the environment variables for `sendmail`
`/etc/shells`	File	Lists the valid login shells
`/usr/sbin/in.comsat`	File	Mail-notification daemon
`/usr/sbin/makemap`	File	Builds binary forms of keyed maps
`/usr/sbin/syslogd`	File	Error message logger, used by `sendmail`
`/usr/dt/bin/dtmail`	File	CDE mail user agent

TABLE 35–9 Other Files Used for Mail Services *(continued)*

Name	Type	Description
`/var/mail/mailbox1`, `/var/mail/mailbox2`	File	Mailboxes for delivered mail
`/var/spool/mqueue`	Directory	Storage for undelivered mail
`$OPENWINHOME/bin/mailtool`	File	Window–based mail user agent

Mail services are provided by a combination of these programs, which interact as shown by the simplified diagram in Figure 35–2.

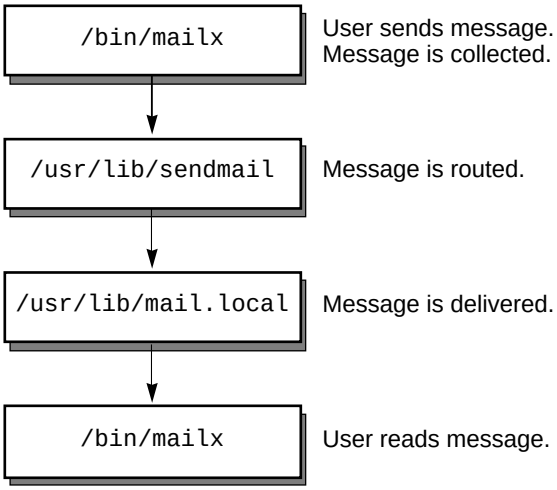


Figure 35–2 How Mail Programs Interact

Users send messages by using programs like `mailx` or `mailtool`. See the `mailx`(1) or *mailtool(1)* man pages for information about these programs.

The message is collected by the program that was used to generate it and is passed to the `sendmail` daemon. The `sendmail` daemon *parses* the addresses (divides them into identifiable segments) in the message, using information from the configuration file, `/etc/mail/sendmail.cf`, to determine network name syntax, aliases, forwarding information, and network topology. Using this information, `sendmail` determines the route a message must take to get to a recipient.

The `sendmail` daemon passes the message to the appropriate system. The `/usr/lib/mail.local` program on the local system delivers the mail to the mailbox in the `/var/mail/`*username* directory of the recipient of the message.

The recipient is notified that mail has arrived, and retrieves it using `mail`, `mailx`, `mailtool`, or a similar program.

`sendmail` Program

The `sendmail` program can use different types of communications protocols, like TCP/IP and UUCP. It also implements an SMTP server, message queueing, and mailing lists. Name interpretation is controlled by a pattern-matching system that can handle both domain-based naming and improvised conventions.

The `sendmail` program can accept domain-based naming as well as arbitrary (older) name syntaxes—resolving ambiguities by using heuristics you specify. `sendmail` can also convert messages between disparate naming schemes. The domain technique separates the issue of physical versus logical naming. See the "Domain Names" on page 86 for a complete description of Internet domain–naming conventions.

You can handle certain special cases by improvised techniques, like providing network names that appear local to hosts on other networks.

The Solaris operating environment uses the `sendmail` program as a mail router. `sendmail` is responsible for receiving and delivering electronic mail messages. It is an interface between mail–reading programs like `mail`, `mailx`, and `mailtool`, and mail–transport programs like `uucp`. The `sendmail` program controls email messages that users send, evaluates the recipients' addresses, chooses an appropriate delivery program, rewrites the addresses in a format that the delivery agent can handle, reformats the mail headers as required, and finally passes the transformed message to the mail program for delivery.

Note - Solaris releases prior to Solaris 2.4 included a binary called `sendmail.mx`. This program is now included in the sendmail program and the functionality is turned on by adding the `dns` flag to the hosts entry in `/etc/nsswitch.conf`. For more information, see "How to Use DNS With `sendmail`" on page 665.

The `sendmail` program supports three mechanisms for mail rerouting. Which mechanism you choose depends on whether this is a server or domain-wide change, or just a change for one user. In addition, by selecting a different rerouting mechanism, you can change the level of administration required.

One rerouting mechanism is aliasing, which maps names to addresses on a server-wide or a name space-wide basis, depending on the type of file that is used. Using a name space alias file allows mail rerouting changes to be administered at a single source, but there can be lagtimes created when the change is propagated. Also, name space administration is usually restricted to a select group of system administrators, so this is not a change that a normal user can make. Rerouting handled through a server alias file is managed by anyone who can become root on that server. Normally, there should be little or no lagtime associated with propagating the change, but the change only affects the local server. This limitation

might be acceptable if most of the mail is sent to one server anyway, but trying to propagate this change to many mail servers is easier using a name service. Again, this is not a change that a user can administer.

The next mechanisms, forwarding and inclusion, allow users to administer mail rerouting. Forwarding allows local users to reroute their incoming mail to either another mailbox, a different mailer, or to another mail host. This form of mail rerouting is supported through the use of `.forward` files. Further information on these files can be found in "`.forward` Files" on page 710.

The last rerouting mechanism is inclusion, which allows for alias lists to be maintained by a user instead of requiring root access. To provide this, the root user must create an appropriate entry in the alias file on the server. After this entry is created, the user can reroute mail as needed. You can find more information on inclusion in "`/etc/mail/aliases`" on page 707.

Figure 35–3 shows how `sendmail` uses aliases. Programs that read mail, like `/usr/bin/mailx`, can have aliases of their own, which are expanded before the message reaches `sendmail`. The aliases for `sendmail` can come from a number of name space sources (local files, NIS or NIS+). The order of the lookup is determined by the `nsswitch.conf` file. See the `nsswitch.conf`(4) man page.

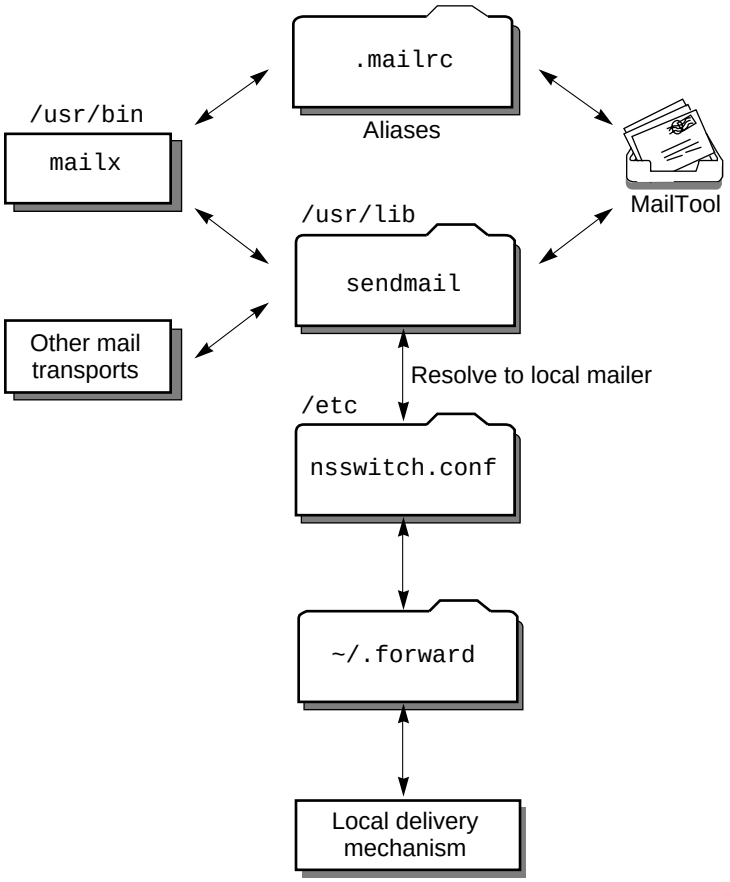


Figure 35–3 How `sendmail` Uses Aliases

`sendmail` Features

The `sendmail` program provides the following features:

- It is reliable. It is designed to correctly deliver every message. No message should ever be completely lost.
- It uses existing software for delivery whenever possible.
- It can be configured to handle complex environments, including multiple connections to a single network type (like with UUCP or Ethernet). `sendmail` checks the contents of an address as well as its syntax to determine which mailer to use.
- It uses configuration files to control mail configuration instead of requiring that configuration information be compiled into the code.

- Users can maintain their own mailing lists. In addition, individuals can specify their own forwarding without modifying the domain-wide alias file (typically located in the domain-wide aliases maintained by NIS or NIS+).
- Each user can specify a custom mailer to process incoming mail, which can provide functions like returning an “I am on vacation” message. See the `vacation`(1) man page for more information.
- It batches addresses to a single host to reduce network traffic.

Figure 35–4 shows how `sendmail` interacts with the other programs in the mail system.

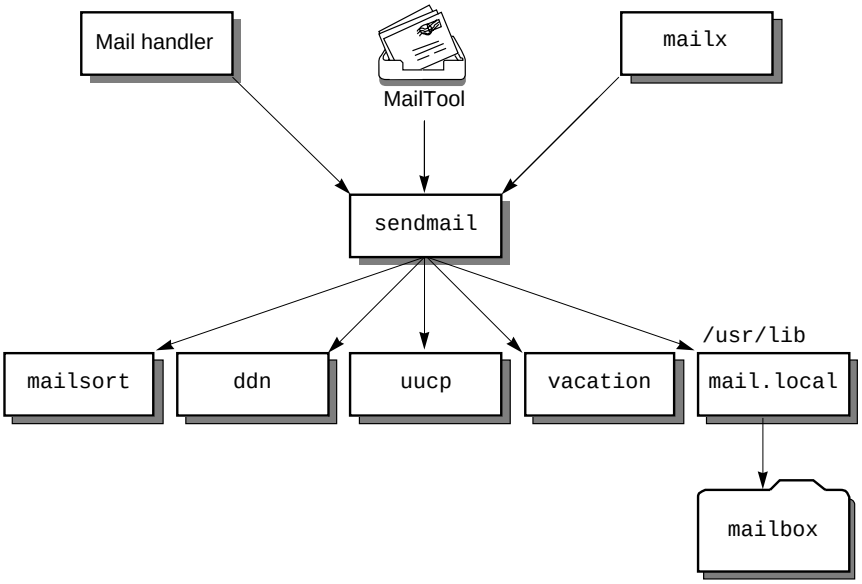


Figure 35–4 Interaction of `sendmail` With Other Mail Programs

The user interacts with a mail-generating and -sending program. When the mail is submitted, the mail-generating program calls `sendmail`, which routes the message to the correct mailers. Because some of the senders might be network servers and some of the mailers might be network clients, `sendmail` can be used as an Internet mail gateway.

`sendmail` Configuration File

A *configuration file* controls the way that `sendmail` performs its functions. The configuration file determines the choice of delivery agents, address rewriting rules, and the format of the mail header.

The `sendmail` program uses the information from the `/etc/mail/sendmail.cf` file to perform its functions. Each system has a default `sendmail.cf` file installed in

the `/etc/mail` directory. You do not need to edit or change the default configuration file for mail servers or mail clients. The only systems that require a customized configuration file are mail hosts and mail gateways.

The Solaris operating environment provides two default configuration files in the `/etc/mail` directory:

1. A configuration file named `main.cf` for the system (or systems) you designate as the mail host or a mail gateway
2. A configuration file named `subsidiary.cf` (a duplicate copy of the default `sendmail.cf` file)

The configuration file you use on a system depends on the role the system plays in your mail service.

- For mail clients or mail servers, you do not need to do anything to set up or edit the default configuration file.
- To set up a mail host or gateway, copy the `main.cf` file and rename it `sendmail.cf` (in the `/etc/mail` directory). Then reconfigure the sendmail configuration file to set the relay mailer and relay host parameters needed for your mail configuration.

The following list describes some configuration parameters you might want to change, depending on the requirements of your site:

- Time values specifies:
  - Read timeouts.
  - Length of time a message remains undelivered in the queue before it is returned to the sender.
- Delivery modes specify how quickly mail will be delivered.
- Load limiting prevents wasted time during loaded periods by not attempting to deliver large messages, messages to many recipients, and messages to sites that have been down for a long time.
- Log level specifies what kinds of problems are logged.

Mail Alias Files

You can use any of the following files to maintain aliases. Which type of file to use depends on who will be using the alias and who needs to be able to change the alias. Each type of alias file has unique format requirements. Each of these is defined in the following sections.

.mailrc Aliases

Aliases listed in a .mailrc file are accessible only by the user who owns the file. This allows users to establish an alias file they control and that is usable only by its owner. Aliases in a .mailrc file adhere to the following format:

```
alias aliasname value value value ...
```

where *aliasname* is the name the user will use when sending mail, and *value* is a valid email address.

If a user establishes a personal alias for scott that does not match the email address for scott in the name space, mail will be routed to the wrong person when other people try to reply to mail generated by that user. The only workaround is to use any of the other aliasing mechanisms.

/etc/mail/aliases

Any alias established in the /etc/mail/aliases file can be used by any user who knows the name of the alias and the host name of the system that contains the file. Distribution list formats in a local /etc/mail/aliases file adhere to the following format:

```
aliasname: value,value,value...
```

where *aliasname* is the name the user will use when sending mail to this alias and *value* is a valid email address.

If your network is not running a name service, the /etc/mail/aliases file of each system should contain entries for all mail clients. You can either edit the file on each system or edit the file on one system and copy it to each of the other systems.

The aliases in the /etc/mail/aliases file are stored in text form. When you edit the /etc/mail/aliases file, run the newaliases program to recompile the database and make the aliases available in binary form to the sendmail program. Or you can use Administration Tool's Database Manager to administer the mail aliases stored in local /etc files.

You can create aliases for only local names—a current host name or no host name. For example, an alias entry for user ignatz who has a mailbox on the system saturn would have this entry in the /etc/mail/aliases file:

```
ignatz: ignatz@saturn
```

It is a good idea to create an administrative account for each mail server. You do this by assigning root a mailbox on the mail server and adding an entry to the /etc/mail/aliases file for root. For example, if the system saturn is a mailbox server, add the entry root: sysadmin@saturn to the /etc/mail/aliases file.

Normally, the root user only can edit this file. When using the Administration Tool, then all users in group 14, which is the sysadmin group, can change the local file. Another option is to create an entry like:

```
aliasname: :include:/path/aliasfile
```

where *aliasname* is the name the user will use when sending mail and */path/aliasfile* is the full path to the file that includes the alias list. The alias file should include email entries, one entry on each line, and no other notations:

```
user1@host1
user2@host2
```

You can define additional mail files in `/etc/mail/aliases` to keep a log or a backup copy. The following entry stores all mail sent to *aliasname* in *filename.*

```
aliasname: /home/backup/filename
```

You can also route the mail to another process. The following stores a copy of the mail message in *filename* and prints a copy.

```
aliasname: "|tee -a /home/backup/filename |lp"
```

NIS Aliases Map

All users in the local domain can use entries included in the NIS aliases map. The `sendmail` program can use the NIS aliases map instead of the local `/etc/mail/aliases` files to determine mailing addresses. See the `nsswitch.conf`(4) man page for more information.

Aliases in the NIS `aliases` map adhere to the following format:

```
aliasname: value,value,value...
```

where *aliasname* is the name the user will use when sending mail and *value* is a valid email address.

The NIS aliases map should contain entries for all mail clients. In general, only the root user on the NIS master can change these entries. This type of alias might not be a good choice for aliases that are constantly changing, but can be useful if the alias points to another alias file; as in this syntax example:

```
aliasname: aliasname@host
```

where *aliasname* is the name the user will use when sending mail and *host* is the host name for the server that contains an `/etc/mail/alias` file.

NIS+ `mail_aliases` Table

The NIS+ `mail_aliases` table contains the names by which a system or person is known in the local domain. The `sendmail` program can use the NIS+ mail_aliases table instead of the local `/etc/mail/aliases` files to determine mailing addresses. See the `aliasadm`(1M) and `nsswitch.conf`(4) man pages for more information.

Aliases in the NIS+ mail_aliases table adhere to the following format:

```
alias:    expansion      [options   # "comments"]
```

Table 35–10 describes the four columns.

TABLE 35–10 Columns in the NIS+ mail_aliases Table

Column	Description
`alias`	The name of the alias
`expansion`	The value of the alias or a list of aliases as it would appear in a `sendmail /etc/mail/aliases` file
`options`	Reserved for future use
`comments`	Comments about an individual alias

The NIS+ mail_aliases table should contain entries for all mail clients. You can list, create, modify, and delete entries in the NIS+ aliases table with the `aliasadm` command. Or you can use Administration Tool's Database Manager to administer NIS+ mail aliases.

If you are creating a new NIS+ aliases table, you must initialize the table before you create the entries. If the table exists, no initialization is needed.

To use the `aliasadm` command, you must be a member of the NIS+ group that owns the aliases table or the person who created the table.

.forward Files

Users can create a .forward file in their home directories that sendmail uses to redirect mail or send mail to a custom set of programs without consulting a system administrator. When troubleshooting mail problems, particularly problems with mail not being delivered to the expected address, always check the user's home directory for a .forward file.

A common mistake users make is to put a .forward file in the home directory of host1 that forwards mail to user@host2. When the mail gets to host2, sendmail looks up user in the NIS or NIS+ aliases and sends the message back to user@host1, resulting in a loop, and more bounced mail.

Note - The root and bin accounts should never have .forward files. Creating these files will create a large security hole. If necessary, forward mail using the aliases file instead.

In order for a .forward file to be consulted during the delivery of mail, the file must be writable only by the owner of the file. This prevents other users from breaking security. In addition, the paths leading up to the home directory must be owned and writable by root only. In particular, if a .forward file is in /export/home/terry, then /export and /export/home must be owned and writable only by root. The actual home directory should be writable only by the user. Other restrictions on a .forward file are that the file cannot be a symbolic link and cannot have more than one hard link.

In addition to the standard .forward file, a .forward.*hostname* file can be created to redirect mail sent to a specific host. For example, if a user's alias has changed from a sandy@phoenix.eng.acme.com to sandy@eng.acme.com, place a .forward.phoenix file in the home directory for sandy.

```
% cat .forward.phoenix
sandy@eng.acme.com
"|/usr/bin/vacation sandy"
% cat .vacation.msg
From: sandy@eng.acme.com (via the vacation program)
Subject: my alias has changed

My alias has changed to sandy@eng.acme.com.
Please use this alias in the future.
The mail that I just received from you
has been forwarded to my new address.

Sandy
```

This allows for the mail to be forwarded to the correct place while also notifying the sender of the alias change. Because the vacation program allows only one message file, you can forward only one message at a time. However, if the message is not host specific, one vacation message file can be used by .forward files for many hosts.

Another extension to the forwarding mechanism is the `.forward+`*detail* file. The *detail* string can be any sequence of characters as long as no operator characters are used. The operator characters are `.:%&!^[]+`. Using a file like this can make it possible to determine if someone else is giving your email address away. For instance, if a user told someone to use the email address `sandy+test1@eng.acme.com`, the user would be able to identify any future mail that was delivered to this alias. By default, any mail sent to `sandy+test1@eng.acme.com` alias is checked against the alias and `.forward+`*detail* files. If there are no matches, the mail falls back to delivery to `sandy@eng.acme.com`, but the user is able to see a change in the `To:` header in their mail.

/etc/default/sendmail

This file is used to store start-up options for `sendmail` so that they are not removed when a host is upgraded. The following variables can be used:

MODE=-bd

Selects the mode to start `sendmail` with. Use the –`bd` option or leave it undefined.

QUEUEINTERVAL=#

Sets interval for the mail queues to be run. # can be a positive integer followed by either `s` for seconds, `m` for minutes, `h` for hours, `d` for days, or `w` for weeks. The syntax is checked before `sendmail` is started. If the interval is negative or if the entry does not end with an appropriate letter, the interval is ignored and `sendmail` starts with a queue interval of 15 minutes.

OPTIONS=*string*

Selects additional options to be used with the `sendmail` command. No syntax checking is done, so be careful when making changes to this variable.

How Mail Addressing Works

The path a mail message follows during delivery depends on the setup of the client system and the topology of the mail domain. Each additional level of mail hosts or mail domains can add one more round of alias resolution, but the routing process is basically the same on most hosts.

You can set up a client system to receive mail locally or select a remote to receive the mail for the client system. Receiving mail locally is known as running `sendmail` in

local mode. Local is the default mode for all mail servers and some clients. If the client is mounting `/var/mail` from a server, the client is running `sendmail` in remote mode.

Assuming that you are using the default rule set in the `sendmail.cf` file, the following examples show the route an email message takes.

On a mail client in remote mode, a mail message will go through the following routing process:

1. **Expand the mail alias, if possible, and restart the local routing process.**

   The mail address is expanded by looking up the mail alias in the name space, according to the entry in `/etc/nsswitch.conf`, and substituting the new value, if one is found. This new alias is then checked again.

2. **If the address cannot be expanded, forward it to the mail server.**

   If the mail address cannot be expanded, there could be a problem with the address or the address is not local. In both cases, the mail server needs to resolve the problem.

3. **If the expanded alias loops back to the original address, forward the mail to the mail server.**

   The process keeps a history of all of the lookups and if the original alias is generated again, the mail is forwarded to the mail server to resolve.

On the mail server or a mail client in local mode, a mail message goes through the following routing process:

1. **Expand the mail alias, if possible, and restart the local routing process.**

   The mail address is expanded by looking up the mail alias in the name space and substituting the new value, if one is found. This new alias is then checked again.

2. **If the mail is local, deliver it to `/usr/lib/mail.local`.**

   The mail will be delivered to a local mailbox.

3. **If the mail address includes a host in this mail domain, deliver the mail to that host.**

4. **If the address does not include a host in this domain, forward the mail to the mail host.**

   The mail host uses the same routing process as the mail server, but the mail host can receive mail addressed to the domain name as well as to the host name.

How sendmail Interacts With a Name Service

Mail domain is a concept used by the standard `sendmail.cf` file to determine whether mail should be delivered directly or through the mail host. Intra-domain mail is delivered through direct SMTP connection, while inter-domain mail is forwarded to a mail host.

In a secure network, only a few selected hosts are authorized to generate packets targeted to external destinations. Even if a host has the IP address of the remote host external to the mail domain, this does not guarantee that an SMTP connection can be established. The standard `sendmail.cf` assumes the following:

- The current host is not authorized to send packets directly to a host outside the mail domain.
- The mail host is capable of forwarding the mail to an authorized host that can transmit packets directly to an external host. (In fact, the mail host can itself be an authorized host.)

Given these assumptions, the mail host is responsible for delivering or forwarding inter-domain mail.

Setting Up sendmail Requirements for Name Services

`sendmail` imposes various requirements on name services. This section explains these requirements and how to satisfy them. For more information, refer to the `in.named`(1M), `nis+`(1), `nisaddent`(1M), and `nsswitch.conf`(4) man pages.

Establishing the Mail Domain Name With a Name Service

The mail domain name must be a suffix of the name service domain. For example, if the domain name of the name service is `A.B.C.D`, the mail domain name could be one of the following:

- `A.B.C.D`
- `B.C.D`
- `C.D`
- `D`

When first established, the mail domain name is often identical to the name service domain. As the network grows larger, the name service domain can be divided into

smaller pieces to make the name service more manageable. However, the mail domain often remains undivided to provide consistent aliasing.

Host Name Space Data

The host table or map in the name service must be set up to support three types of `gethostbyname( )` queries:

- `mailhost` – Some name service configurations satisfy this requirement automatically.
- Full host name (for example, `smith.admin.acme.com`) – Many name service configurations satisfy this requirement.
- Short host name (for example, `smith`) – `sendmail` must connect to the mail host to forward external mail. To determine if a mail address is within the current mail domain, `gethostbyname( )` is invoked with the full host name. If the entry is found, the address is considered internal.

  NIS, NIS+, and DNS all support `gethostbyname( )` with a short host name as an argument, so this requirement is automatically satisfied.

Two additional rules about the host name space need to be followed to establish the sendmail services within a name space properly.

1. `gethostbyname( )` with full and short host name should yield consistent results. For example, `gethostbyname(smith.admin.acme.com)` should return the same result as `gethostbyname(smith)`, as long as both functions are called from the mail domain `admin.acme.com`.
2. For all name service domains under a common mail domain, `gethostbyname( )` with a short host name should yield the same result. For example, given the mail domain `smith.admin.acme.com`, `gethostbyname(smith)` should return the same result calling from either domain `ebb.admin.acme.com` or `esg.admin.acme.com`. The mail domain name is usually shorter than the name service domain, giving this requirement special implications for various name services.

Configuration Issues With NIS and `sendmail`

This list includes all the configuration issues that you must resolve before using `sendmail`, when using NIS as your only name service.

Mail domain name – If you are setting up NIS as the primary name service, `sendmail` automatically strips the first component of the NIS domain name and uses the result as the mail domain name. For example, `ebs.admin.acme.com` becomes `admin.acme.com`.

Mail host host name – You must have a `mailhost` entry in the NIS host map.

Full host names – The normal NIS setup does not “understand” the full host name. Rather than trying to make NIS understand the full host name, turn off this

requirement from the sendmail side by editing the sendmail.cf file and replacing all occurrences of %l with %y. This turns off sendmail's inter-domain mail detection. As long as the target host can be resolved to an IP address, a direct SMTP delivery is attempted. Make sure that your NIS host map does not contain any host entry that is external to the current mail domain. Otherwise, you need to further customize the sendmail.cf file.

Matching full and short host names – Follow the previous instructions on how to turn off gethostbyname( ) for a full host name.

Multiple NIS domains in one mail domain – All NIS host maps under a common mail domain should have the same set of host entries. For example, the host map in the ebs.admin.acme.com domain should be the same as the host map in the esg.admin.acme.com. Otherwise, one address might work in one NIS domain but fail in the other NIS domain.

Configuration Issues With NIS and DNS While Using sendmail

This list includes all the configuration issues that you must resolve before using sendmail, when using NIS with DNS as your name service.

Mail domain name – If you are setting up NIS as the primary name service, sendmail automatically strips the first component of the NIS domain name and uses the result as the mail domain name, for example, ebs.admin.acme.com becomes admin.acme.com.

Mailhost host name – When the DNS forwarding feature is turned on, queries that NIS cannot resolve are forwarded to DNS, so there is no need for a mailhost entry in the NIS host map.

Full host names – Although NIS does not "understand" full host names, DNS does. This requirement is satisfied when you follow the regular procedure for setting up NIS and DNS.

Matching full and short host names – For every host entry in the NIS host table, you must have a corresponding host entry in DNS.

Multiple NIS domains in one mail domain – All NIS host maps under a common mail domain should have the same set of host entries. For example, the host map in the ebs.admin.acme.com domain should be the same as the host map in the esg.admin.acme.com. Otherwise, one address might work in one NIS domain but fail in the other NIS domain.

Configuration Issues With NIS+ and sendmail

This list includes all the configuration issues that you must resolve before using sendmail when using NIS+ as your only name service.

Mail domain name – If you are setting up NIS+ as your primary name service, sendmail can look up the mail domain from the NIS+ sendmailvars table, a two–column NIS+ table with one key column and one value column. To set up your mail domain, you must add one entry to this table. This entry should have the key column set to the literal string maildomain and the value column set to your mail domain name (for example, admin.acme.com). Although NIS+ allows any string in the sendmailvars table, the suffix rule still applies for the mail system to work correctly. You can use nistbladm to add the maildomain entry to the sendmailvars table. For example:

```
nistbladm -A key="maildomain" value=<mail domain> sendmailvars.org_dir.<NIS+ domain>
```

Notice that this mail domain is a suffix of the NIS+ domain.

Mailhost host name – You must have a mailhost entry in the NIS+ hosts table.

Full host names – NIS+ “understands” the full host name. Following the regular NIS+ setup procedure satisfies this requirement.

Matching full and short host names – To satisfy this requirement, you can duplicate the entries in the host table, or you can enter all host entries in the user name service domains into a master host table at mail domain level.

Multiple NIS domains in one mail domain – To satisfy this requirement, you can duplicate the entries in all the host tables, or you can type all host entries in the user name service domains into a master host table at mail domain level. Because you are merging (logical or physical) multiple host tables into one host table, the same host name cannot be reused in the multiple name service domain sharing a common mail domain.

Configuration Issues with NIS+ and DNS while Using sendmail

This list includes all the configuration issues that you must resolve before using sendmail when using NIS+ with DNS as your name service.

mail domain name — If you are setting up NIS+ as your primary name service, sendmail can look up the mail domain from the NIS+ sendmailvars table, a two–column NIS+ table with one key column and one value column. To set up your mail domain, you must add one entry to this table. This entry should have the key column set to the literal string maildomain and the value column set to the your mail domain name (for example, admin.acme.com). Although NIS+ allows any string in the sendmailvars table, the suffix rule still applies for the mail system to work correctly. You can use nistbladm to add the maildomail entry to the sendmailvars table. For example:

```
nistbladm -A key="maildomain" value=<mail domain> sendmailvars.org_dir.<NIS+ domain>
```

Notice that this mail domain is a suffix of the NIS+ domain.

mailhost host name — If your network uses both NIS+ and DNS as the source for the host database, you can put the `mailhost` entry in either the NIS+ or DNS host table. Make sure that your users list NIS+ and DNS as the source for the host database in the `/etc/nsswitch.conf` file.

full host names — Both NIS+ and DNS "understand" full host names. Following the regular NIS+ and DNS setup procedures satisfies this requirement.

matching full and short host names — For every host entry in the NIS+ host table, you must have a corresponding host entry in DNS.

multiple NIS domains in one mail domain — To satisfy this requirement, you can duplicate the entries in all the host tables, or you can type all host entries in the user name service domains into a master host table at the mail domain level.

CHAPTER 36

Monitoring Network Services Topics

Chapter 37	Provides step-by-step instructions for monitoring network services

CHAPTER 37

Monitoring Network Performance (Tasks)

This chapter describes the how to monitor network performance. This is a list of the step-by-step instructions in this chapter.

Monitoring Network Performance

Table 37–1 describes the commands available for monitoring network performance.

TABLE 37–1 Network Monitoring Commands

Command	Use This Command to ...
`ping`	Look at the response of hosts on the network.
`spray`	Test the reliability of your packet sizes. It can tell you whether packets are being delayed or dropped.
`snoop`	Capture packets from the network and trace the calls from each client to each server.

TABLE 37–1 Network Monitoring Commands *(continued)*

Command	Use This Command to ...
`netstat`	Display network status, including state of the interfaces used for TCP/IP traffic, the IP routing table, and the per-protocol statistics for `UDP`, `TCP`, `ICMP`, and `IGMP`.
`nfsstat`	Display a summary of server and client statistics that can be used to identify NFS problems.

▼ How to Check the Response of Hosts on the Network

Check the response of hosts on the network with the `ping` command.

```
$ ping hostname
```

If you suspect a physical problem, you can use `ping` to find the response time of several hosts on the network. If the response from one host is not what you would expect, you can investigate that host. Physical problems could be caused by:

- Loose cables or connectors
- Improper grounding
- Missing termination
- Signal reflection

For more information about this command, see `ping`(1M).

Examples—Checking the Response of Hosts on the Network

The simplest version of `ping` sends a single packet to a host on the network. If it receives the correct response, it prints the message *host* `is alive`.

```
$ ping elvis
elvis is alive
```

With the `-s` option, `ping` sends one datagram per second to a host. It then prints each response and the time it took for the round trip. For example:

```
$ ping -s pluto
64 bytes from pluto (123.456.78.90): icmp_seq=0. time=10. ms
64 bytes from pluto (123.456.78.90): icmp_seq=5. time=0. ms
64 bytes from pluto (123.456.78.90): icmp_seq=6. time=0. ms
^C
----pluto PING Statistics----
8 packets transmitted, 8 packets received, 0% packet loss

round-trip (ms) min/avg/max = 0/2/10
```

▼ How to Send Packets to Hosts on the Network

Test the reliability of your packet sizes with the `spray` command.

```
$ spray [ -c count -d interval -l packet_size] hostname
```

–`i` *count*	Number of packets to send.
–`d` *interval*	Number of microseconds to pause between sending packets. If you don't use a delay, you might run out of buffers.
–`l` *packet_size*	Is the packet size.
hostname	Is the system to send packets.

For more information about this command, see `spray`(1M).

Example—Sending Packets to Hosts on the Network

The following example sends 100 packets to a host (`-c 100`) with each packet having a size of 2048 bytes (`-l 2048`). The packets are sent with a delay time of 20 microseconds between each burst (`-d 20`).

```
$ spray -c 100 -d 20 -l 2048 pluto
sending 100 packets of length 2048 to pluto ...
no packets dropped by pluto
279 packets/sec, 573043 bytes/sec
```

▼ How to Capture Packets From the Network

To capture packets from the network and trace the calls from each client to each server, use snoop. This command provides accurate timestamps that allow some network performance problems to be isolated quickly. For more information, see snoop(1M).

```
# snoop
```

Dropped packets could be caused by insufficient buffer space, or an overloaded CPU.

▼ How to Check the Network Status

Display network status information, such as statistics about the state of network interfaces, routing tables, and various protocols, with the netstat command.

```
$ netstat [-i] [-r] [-s]
```

–i	Displays the state of the TCP/IP interfaces
–r	Displays the IP routing table
–s	Displays statistics for the UDP, TCP, ICMP, and IGMP protocols

For more information, see netstat(1M).

Examples—Checking the Network Status

The following example shows output from the netstat -i command, which displays the state of the interfaces used for TCP/IP traffic.

```
$ netstat -i
Name  Mtu  Net/Dest    Address      Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 software    localhost     1280   0     1280     0       0    0
le0   1500 loopback    venus      1628480   0   347070    16   39354    0
```

This display shows how many packets a machine has transmitted and received on each interface. A machine with active network traffic should show both Ipkts and Opkts continually increasing.

Calculate the network collisions rate by dividing the number of collision counts (Collis) by the number of out packets (Opkts). In the previous example, the

collision rate is 11 percent. A network-wide collision rate greater than 5 to 10 percent can indicate a problem.

Calculate the input packet error rate by dividing the number of input errors by the total number of input packets (`Ierrs/Ipkts`). The output packet error rate is the number of output errors divided by the total number of output packets (`Oerrs/Opkts`). If the input error rate is high (over 0.25 percent), the host might be dropping packets.

The following example shows output from the `netstat -s` command, which displays the per-protocol statistics for the `UDP`, `TCP`, `ICMP`, and `IGMP` protocols.

```
UDP
 udpInDatagrams      =196543 udpInErrors         =     0
 udpOutDatagrams     =187820

TCP
 tcpRtoAlgorithm     =     4 tcpRtoMin           =   200
 tcpRtoMax           = 60000 tcpMaxConn          =    -1
 tcpActiveOpens      = 26952 tcpPassiveOpens     =   420
 tcpAttemptFails     =  1133 tcpEstabResets      =     9
 tcpCurrEstab        =    31 tcpOutSegs          =3957636
 tcpOutDataSegs      =2731494 tcpOutDataBytes     =1865269594
 tcpRetransSegs      = 36186 tcpRetransBytes     =3762520
 tcpOutAck           =1225849 tcpOutAckDelayed    =165044
 tcpOutUrg           =     7 tcpOutWinUpdate     =   315
 tcpOutWinProbe      =     0 tcpOutControl       = 56588
 tcpOutRsts          =   803 tcpOutFastRetrans   =   741
 tcpInSegs           =4587678
 tcpInAckSegs        =2087448 tcpInAckBytes       =1865292802
 tcpInDupAck         =109461 tcpInAckUnsent      =     0
 tcpInInorderSegs    =3877639 tcpInInorderBytes   =-598404107
 tcpInUnorderSegs    = 14756 tcpInUnorderBytes   =17985602
 tcpInDupSegs        =    34 tcpInDupBytes       = 32759
 tcpInPartDupSegs    =   212 tcpInPartDupBytes   =134800
 tcpInPastWinSegs    =     0 tcpInPastWinBytes   =     0
 tcpInWinProbe       =   456 tcpInWinUpdate      =     0
 tcpInClosed         =    99 tcpRttNoUpdate      =  6862
 tcpRttUpdate        =435097 tcpTimRetrans       = 15065
 tcpTimRetransDrop   =    67 tcpTimKeepalive     =   763
 tcpTimKeepaliveProbe=     1 tcpTimKeepaliveDrop =     0

IP
 ipForwarding        =     2 ipDefaultTTL        =   255
 ipInReceives        =11757234 ipInHdrErrors       =     0
 ipInAddrErrors      =     0 ipInCksumErrs       =     0
 ipForwDatagrams     =     0 ipForwProhibits     =     0
 ipInUnknownProtos   =     0 ipInDiscards        =     0
 ipInDelivers        =4784901 ipOutRequests       =4195180
 ipOutDiscards       =     0 ipOutNoRoutes       =     0
 ipReasmTimeout      =    60 ipReasmReqds        =  8723
 ipReasmOKs          =  7565 ipReasmFails        =  1158
 ipReasmDuplicates   =     7 ipReasmPartDups     =     0
 ipFragOKs           = 19938 ipFragFails         =     0
 ipFragCreates       =116953 ipRoutingDiscards   =     0
```

(continued)

```
 tcpInErrs           =     0 udpNoPorts          =6426577
 udpInCksumErrs      =     0 udpInOverflows      =   473
 rawipInOverflows    =     0

ICMP
 icmpInMsgs          =490338 icmpInErrors        =     0
 icmpInCksumErrs     =     0 icmpInUnknowns      =     0
 icmpInDestUnreachs  =   618 icmpInTimeExcds     =   314
 icmpInParmProbs     =     0 icmpInSrcQuenchs    =     0
 icmpInRedirects     =   313 icmpInBadRedirects  =     5
 icmpInEchos         =   477 icmpInEchoReps      =    20
 icmpInTimestamps    =     0 icmpInTimestampReps =     0
 icmpInAddrMasks     =     0 icmpInAddrMaskReps  =     0
 icmpInFragNeeded    =     0 icmpOutMsgs         =   827
 icmpOutDrops        =   103 icmpOutErrors       =     0
 icmpOutDestUnreachs =    94 icmpOutTimeExcds    =   256
 icmpOutParmProbs    =     0 icmpOutSrcQuenchs   =     0
 icmpOutRedirects    =     0 icmpOutEchos        =     0
 icmpOutEchoReps     =   477 icmpOutTimestamps   =     0
 icmpOutTimestampReps=     0 icmpOutAddrMasks    =     0
 icmpOutAddrMaskReps =     0 icmpOutFragNeeded   =     0
 icmpInOverflows     =     0
IGMP:
        0 messages received
        0 messages received with too few bytes
        0 messages received with bad checksum
        0 membership queries received
        0 membership queries received with invalid field(s)
        0 membership reports received
        0 membership reports received with invalid field(s)
        0 membership reports received for groups to which we belong
        0 membership reports sent
```

The following example shows output from the `netstat -r` command, which displays the IP routing table.

```
Routing Table:
  Destination        Gateway           Flags  Ref   Use    Interface
------------------ -------------------- ----- ----- ------ ---------
localhost            localhost             UH       0   2817  lo0
earth-bb             pluto                 U        3  14293  le0
224.0.0.0            pluto                 U        3      0  le0
default              mars-gate             UG       0  14142
```

The fields in the `netstat -r` report are described in Table 37–2.

TABLE 37–2 Output From the `netstat -r` Command

Field Name		Description
`Flags`	`U`	The route is up
	`G`	The route is through a gateway
	`H`	The route is to a host
	`D`	The route was dynamically created using a redirect
`Ref`		Shows the current number of routes sharing the same link layer
`Use`		Indicates the number of packets sent out
`Interface`		Lists the network interface used for the route

▼ How to Display NFS Server and Client Statistics

The NFS distributed file service uses a remote procedure call (RPC) facility that translates local commands into requests for the remote host. The remote procedure calls are synchronous. That is, the client application is blocked or suspended until the server has completed the call and returned the results. One of the major factors affecting NFS performance is the retransmission rate.

If the file server cannot respond to a client's request, the client retransmits the request a specified number of times before it quits. Each retransmission imposes system overhead, and increases network traffic. Excessive retransmissions can cause network performance problems. If the retransmission rate is high, you could look for:

- Overloaded servers that take too long to complete requests
- An Ethernet interface dropping packets
- Network congestion, which slows the packet transmission

Table 37–3 describes the `nfsstat` options to display client and server statistics.

TABLE 37–3 Commands for Displaying Client/Server Statistics

Use ...	To Display ...
`nfsstat -c`	Client statistics
`nfsstat -s`	Server statistics
`netstat -m`	Network statistics for each file system

Use `nfsstat -c` to show client statistics, and `nfsstat -s` to show server statistics. Use `netstat -m` to display network statistics for each file system. For more information, see `nfsstat`(1M).

Examples—Displaying NFS Server and Client Statistics

The following example displays RPC and NFS data for the client `pluto`.

```
$ nfsstat -c

        Client rpc:
Connection oriented:
calls    badcalls  badxids  timeouts newcreds  badverfs   timers
1595799  1511      59       297      0         0          0
cantconn nomem     interrupts
1198      0         7
Connectionless:
calls    badcalls  retrans  badxids  timeouts  newcreds   badverfs
80785    3135      25029    193      9543      0          0
timers   nomem     cantsend
17399    0         0

Client nfs:
calls    badcalls  clgets   cltoomany
1640097  3112      1640097  0
Version 2: (46366 calls)
null     getattr   setattr  root     lookup     readlink  read
0 0%     6589 14%  2202 4%  0 0%     11506 24%  0 0%      7654 16%
wrcache  write     create   remove   rename     link      symlink
0 0%     13297 28% 1081 2%  0 0%     0 0%       0 0%      0 0%
mkdir    rmdir     readdir  statfs
24 0%    0 0%      906 1%   3107 6%
Version 3: (1585571 calls)
null    getattr    setattr  lookup     access     readlink  read
0 0%    508406 32% 10209 0% 263441 16% 400845 25% 3065 0%  117959 7%
write    create     mkdir    symlink    mknod    remove   rmdir
69201 4% 7615 0%    42 0%    16 0%      0 0%     7875 0%  51 0%
rename   link       readdir  readdir+   fsstat   fsinfo   pathconf
```

(continued)

(Continuation)

```
929 0%   597 0%     3986 0%  185145 11% 942 0%   300 0%   583 0%
commit
4364 0%

Client nfs_acl:
Version 2: (3105 calls)
null       getacl     setacl     getattr    access
0 0%       0 0%       0 0%       3105 100%  0 0%
Version 3: (5055 calls)
null       getacl     setacl
0 0%       5055 100%  0 0%
```

The output of the `nfsstat -c` command is described in Table 37–4.

TABLE 37–4 Output From the `nfsstat -c` Command

Field	Description
`calls`	The total number of calls sent.
`badcalls`	The total number of calls rejected by RPC.
`retrans`	The total number of retransmissions. For this client, the number of retransmissions is less than 1 percent (10 timeouts out of 6888 calls). These might be caused by temporary failures. Higher rates might indicate a problem.
`badxid`	The number of times that a duplicate acknowledgment was received for a single NFS request.
`timeout`	The number of calls that timed out.
`wait`	The number of times a call had to wait because no client handle was available.
`newcred`	The number of times the authentication information had to be refreshed.
`timers`	The number of times the time-out value was greater than or equal to the specified time-out value for a call.
`readlink`	The number of times a `read` was made to a symbolic link. If this number is high (over 10 percent), it could mean that there are too many symbolic links.

The following example shows output from the `nfsstat -m` command.

```
pluto$ nfsstat -m
/usr/man from pluto:/export/svr4/man
Flags: vers=2,proto=udp,auth=unix,hard,intr,dynamic,
       rsize=8192, wsize=8192,retrans=5
 Lookups: srtt=13 (32ms), dev=10 (50ms), cur=6 (120ms)
 All:     srtt=13 (32ms), dev=10 (50ms), cur=6 (120ms)
```

This output of the `nfsstat -m` command, which is displayed in milliseconds, is described in Table 37–5.

TABLE 37–5 Output From the `nfsstat -m` Command

Field	Description
`srtt`	The smoothed average of the round-trip times
`dev`	The average deviations
`cur`	The current “expected” response time

If you suspect that the hardware components of your network are creating problems, you need to look carefully at the cabling and connectors.

APPENDIX **A**

PCNFSpro Troubleshooting

The following series of troubleshooting techniques are specific to PCNFSpro running as the Windows client. The following topics are covered:

Troubleshooting

The following series of troubleshooting techniques are specific to PCNFSpro running as the Windows client. To get further information about PCNFSpro and the Windows client, consult the *SolarNet PC-Admin Admininstrator's Guide.* The following procedures are discussed.

Reboot the PC

This is a good first step if the PC is unable to connect to a server, or displays error messages as it contacts the server. Rebooting the machine resets network hardware and software. If the problem is caused by a temporary license that has expired, rebooting renews the license for 30 minutes.

For the Windows client, delete the file:

`c:\pcnfspro\dhcp\`*interface*`.bin`

Replace *interface* with the name of the actual interface in use, for example: `c:\pcnfspro\dhcp\pk0.bin`.

Running in Debug Mode

Running in DHCP debug mode reveals much of the ongoing dialog between the client and the server. This dialog provides useful clues for solving network problems.

▼ To Run a Windows Client in Debug Mode

1. **Kill the DHCP server and restart it in debug mode.**
2. **Enable the Network Event Log in the Services applet of the Configuration tool.**
3. **Close the Configuration Tool.**
4. **Start the Network Event Log directly from the program group.**
5. **Select the Display menu and highlight all priority levels.**
6. **Choose Save. After you do this,** `nfswdhcp.exe` **logs to the Network Event Log.**
7. **Exit and restart Windows.**

Client Fails to Connect With DHCP/BOOTP Server

If you have installed or added a new client, but the machine fails to connect to the server and displays an error message, check the machine's cable and adapter. If the adapter has a diagnostic program, run it to identify possible problems.

Start the Configuration application in the PCNFSpro directory. Choose Services and enable the Start Network Event Log. You can also start the Network Event Log directly from its icon. After it starts, choose Display and then Configure. Select all priority levels, down to Debug. Choose Save to save the configuration. Network Event Log entries similar to:

```
DHCP: Attempting to configure interface using DHCP
```

indicate that the machine has broadcast for a configuration. Server replies should follow.

Next, kill the `in.dhcpd` daemon on the server and run the server in diagnostic mode by typing **`in.dhcp -d`**.

After receiving output, check that there is a DHCP server or relay agent on the machine's subnet. While booting the client, run:

`snoop udp port 67` or **`udp port 68`**

on a server on the same subnet as the machine. See if a system responds. Windows client `snoop` output looks like this:

```
OLD-BROADCAST -> BROADCAST UDP D=67 S=68 LEN=311
glrr -> BROADCAST UDP D=68 S=67 LEN=490
```

The client name is not shown for the Windows client. The DHCP activity is reflected by the presence of BROADCAST.

Applications Run Out of Conventional Memory

The `CONFIG.SYS` file (by default) does not try to load the Windows client adapter drivers (packet driver, NDIS, or ODI) into upper memory. This can lead to applications reporting "not enough conventional memory" or a similar message when they start.

If you are running DOS 5.x or greater, change `CONFIG.SYS` lines from:

`DEVICE=C:\PCNFSPRO\`*filename*

to

`DEVICEHIGH=C:\PCNFSPRO\`*filename*

and `NFSWAUTO.BAT` lines that load TSRs from

tsrname

to

LH *tsrname*

Mounting Home Directories

The software default of mounting home directories on Drive H conflicts with the use of H by the MS-DOS 6.2 DoubleSpace utility. To resolve this conflict, remap the drive used by the DoubleSpace utility (`dblspace h: /host=`*new_drive*) or change your site login script to mount home directories on a different drive. The site login script is `/opt/SUNWpcnet/1.5/site/pcnfspro/login.snc`.

If you change the site login script, you must also change the HOME environment variable to a new drive.

Use of Ping

If a machine cannot mount a remote file with the `net use` command, verify that the path name to the file system you want to access has been typed correctly. Then use the `ping` command.

If a user is copying files to or from a network drive and the process stops before it is completed, use the `ping` command.

If a machine has been receiving a license and cannot receive a license when it is turned on or rebooted, and instead receives error messages, use the `ping` command.

SNC Script

The software distributes access to applications and other network services when a machine starts or when a new user logs on. This interprets the directives in the `store\login.snc` file. `%SNDRIVE%` expands to `/opt/SUNWpcnet/1.5/site/pcnfspro`, the site SNC script directory for the Windows client. This script is not accessible to users because it is protected by UNIX permissions.

The directory `/opt/SUNWpcnet/1.5/site/pcnfspro` contains the default client scripts (SNC scripts) used at boot, login and logout time to control resources on the machines. The directories are mounted temporarily and then dismounted after the scripts are run. On the Windows client, SNC commands are responsible for establishing users' unique relationships to the network.

The Windows client's Configuration program processes the boot script (by default named `boot.snc`) in the SNC script directory `/opt/SUNWpcnet/1.5/site/pcnfspro`. The names of the script and the SNC script directory can be different for each network.

The Windows client's Configuration program is a comprehensive graphical tool used to view and change a wide variety of configuration parameters. The program enables you to change and customize any Windows client's default configuration, and to save the customized configuration. Among the parameters that you control by way of the Configuration program are TCP/IP, the Local Area Network (LAN) user name, the NFS system, the printer client, NetBIOS, and SNMP. You can also standardize and control the level of configuration that the Windows client's Configuration program makes available to users at your site. See the Configuration program's online help for details about using the program.

The site SNC script directory is expanded when you add an INCLUDE directive for a new group `login.snc` script. It is also expanded when you add an INCLUDE directive in the `store\logout.snc` file.

The script directory is used to create a directory with the UNIX login name for each user who has an individualized view of applications on the network. Then it is used

to create and copy the user-specific `login.snc` and `logout.snc` scripts into each directory. The commands are:

`cd/opt/SUNWpcnet/1.5/site/pcnfspro`

`mkdir` *user1user2 user3 user4 user5user6user7*

In this example, all users except *user8*, *user9*, *user10*, and *user11* use applications that they either do not share with other users or that they share with some, but not all users.

Three default SNC scripts are provided for each type of client in the SNC script directory `/opt/SUNWpcnet/1.5/site/pcnfspro`. They are:

- `boot.snc` – The site boot script used by the Windows client's Configuration program and Login/Logout application.
- `login.snc` – The site login script used by the Windows client's Login/Logout application.
- `logout.snc` – The site logout script used by the Windows client's Login/Logout application.

The default site `logout.snc` script cleans up after the site `login.snc` script. You can copy the site `logout.snc` script, rename it, and modify it to correspond to your login scripts. You should name all logout scripts `logout.snc` in parallel with the login scripts. Place the logout scripts in user- and group-specific directories you create under the SNC script directory `/opt/SUNWpcnet/1.5/site/pcnfspro`.

DHCP Databases

Each NIS+ domain has one `dhcptab` table, which defines the configuration parameters to be returned to DHCP (or BOOTP) clients. The `/opt/SUNWpcnet/1.5/site/pcnfspro` script directory is one of the entries used in defining user views and distributing applications.

License Upgrade

Checking that license upgrade files were created requires certain procedures for the Windows clients. First run `C:\pcnfspro\upgrade`. After checking the license upgrade file and its contents, rerun the `install` program. Then:

1. **Rename** `C:\windows\pcnfswin.ini`**.**

2. **Run the** `C:\pcnfspro\binnfswpupg` **program.**

3. **Finally, restore the renamed** `pcnfswin.ini` **file to its original name of** `pcnfswin.ini`**.**

After following the other steps to be sure the upgraded license file is present and is operating correctly, delete the old DHCP configuration files and reboot the machine. Delete the file in the directory `C:\pcnfspro\dhcp` corresponding to your interface.

Loss of Host name and IP Address

Similar procedures must be followed if the machine has lost its original host name and IP address. First start the Control Panel application and choose the Network icon. View the current host name and IP address, as well as other pertinent information. Perform the procedure that creates an entry to the `dhcp_table`. Then run `C:\pcnfspro\dhcp` and follow the previous steps.

Next, copy the upgrade file to the upgrade directory in the installation directory on the server. Check that the `snaddpcs` script was executed. Then delete old DHCP configuration files and reboot the screen. Do this by deleting the file in the directory `C:\pcnfspro\dhcp` corresponding to your interface.

Distributing Applications

The software distributes access to applications and other network services when a machine starts or when a new user logs on. For the Windows client, these applications and services are provided to each user by the Login application, which starts automatically when Windows is started.

The Windows-based DHCP application initiates DHCP, receives an IP address and related network information for the machine, configures the machine's stack and services, then begins processing client (SNC) scripts that mount file systems and set search paths for shared resources, groups, individual users, and individual machines.

Logging In and Out

For the Windows client, logging in to and out of the network is done by clicking the Login/Logout application icon and filling in the resulting Windows dialog box. When a user fills in the dialog box with the user name and password, and clicks OK or presses Enter, the Windows client's site `logon.snc` script is launched by the following `INCLUDE` directive in the client's site boot script:

```
INCLUDE %SNDRIVE%\PCNFSPRO\LOGIN.SNC
```

This script mounts directories, sets certain environment variables, and provides other services for the user who is already logged in regardless of which machine the person is using to log in. It establishes the login procedure that all users in the network follow.

When the Windows client uses the same Windows-based dialog box to log out, the client's site `logout.snc` script is processed. This script undoes what the `login.snc` script did. The Logout application unmounts file systems and printers, and resets the machine's environment variables to their original, pre-login values.

The Windows client's Login/Logout application provides detailed information about the user's network settings, and enables you to change the settings. Among the settings are version and license numbers; the user name and ID; the group ID; NIS and DNS domains; the subnet mask; MAC addresses (which identify an Ethernet communications adapter); the time zone; the last drive; and names and IP addresses of available servers.

APPENDIX **B**

NFS Tuneables

You can set several parameters that can improve the functioning of the NFS service. You can define these parameters in `/etc/system`, which is read during the boot process. Each parameter can be identified by the name of the kernel module that it is in and a symbol name that identifies it.

Note - The names of the symbols, the modules that they reside in, and the default values can change between releases. Check the documentation for the version of the active SunOS release, before making changes or applying values from previous releases.

The following topics are covered in this chapter:

TABLE B–1 NFS Parameters for the nfs Module

Symbol Name	Description	Default Setting
nfs_32_time_ok	This symbol controls whether the NFS client or server allows file timestamps that are > Y2038.	Defaults to off (0). You should use this symbol if the timestamp on any file is negative, which means that the date is before 1970, and you still want to be able to access these files.
nfs_acl_cache	This symbol controls whether ACLs are cached on clients that are using the NFS_ACL protocol.	Defaults to off (0). You probably can safely enable this symbol (1), which might be in future Solaris releases.
nfs_cots_timeo	This symbol controls the default time-out value of NFS version 2 client operations over connection-oriented transports.	Defaults to 600 tenths of a second.
nfs3_cots_timeo	This symbol controls the default time-out value of NFS version 3 client operations over connection-oriented transports.	Defaults to 600 tenths of a second.
nfs_do_symlink_cache	This symbol controls whether symbolic links are cached for file systems mounted using NFS version 2 software.	Defaults to on (1). You can disable this symbol (0) if something like amd is to be used on the system. Client system performance might be reduced if this symbol is disabled.
nfs3_do_symlink_cache	This symbol controls whether symbolic links are cached for file systems mounted using NFS version 3 software.	Defaults to on (1). You can disable this symbol (0) but client system performance might be reduced.
nfs_dynamic	This symbol controls whether dynamic retransmission support is used for file systems mounted using NFS version 2 software.	Defaults to on (1). You can safely turn off this symbol (0), with possible interoperability problems with servers that are slow or cannot support full 8 KB read or write transfers.
nfs3_dynamic	This symbol controls whether dynamic retransmission support is used for file systems mounted using NFS version 3 software.	Defaults to off (0). Do not change this value.

TABLE B–1 NFS Parameters for the nfs Module *(continued)*

Symbol Name	Description	Default Setting
nfs_lookup_neg_cache	This symbol controls whether failed lookup requests are cached for file systems mounted using NFS version 2 software.	Defaults to off (0). You can probably safely enable this symbol (1) but it might negatively impact normal directory name caching.
nfs3_lookup_neg_cache	This symbol controls whether failed lookup requests are cached for file systems mounted using NFS version 3 software.	Defaults to off (0). You can probably safely enable this symbol (1) but it might negatively impact normal directory name caching.
nfs_max_threads	This symbol controls the maximum number of async threads started per file system mounted using NFS version 2 software.	Defaults to 8. Because this number affects the number of threads per file system, on a client with many file systems a large change could severely degrade performance.
nfs3_max_threads	This symbol controls the maximum number of async threads started per file system mounted using NFS version 3 software.	Defaults to 8. Because this number affects the number of threads per file system, on a client with many file systems a large change could severely degrade performance.
nfs3_max_transfer_size	This symbol controls the NFS version 3 client file block size.	Defaults to 32 KB. Strongly recommend that it not be changed.
nfs_nra	This symbol controls the number of read-ahead blocks that are read for file systems mounted using NFS version 2 software.	Defaults to 4. A higher value might not increase performance, but will cause increased memory utilization on the client.
nfs3_nra	This symbol controls the number of read-ahead blocks that are read for file systems mounted using NFS version 3 software.	Defaults to 4. A higher value might not increase performance, but will cause increased memory utilization on the client.
nrnode	This symbol controls the number of NFS rnodes that are cached.	The value assigned to this symbol is configured at boot time and scales to match the server. You can set this symbol to 1 to disable caching.

TABLE B–1 NFS Parameters for the nfs Module *(continued)*

Symbol Name	Description	Default Setting
`nfs_shrinkreaddir`	This symbol controls whether over-the-wire NFS Version 2 READDIR requests are shrunk to 1024 bytes. Some old NFS Version 2 servers could not correctly handle READDIR requests larger than 1024 bytes.	Defaults to off (0), which means do not reduce the READDIR requests. You can safely enable this symbol (1) but it might negatively impact system performance while reading directories.
`nfs_write_error_interval`	This symbol controls how often NFS ENOSPC write error messages are logged. Its units are in seconds.	Defaults to 5.
`nfs_write_error_to_cons_only`	This symbol controls whether NFS write error messages are logged to the system console or to the system console and syslog.	Defaults to off (0), which means to log all NFS write error messages to the system console and syslog. Enabling (1) this functionality means that most NFS write error messages will only be printed on the system console.

TABLE B–2 NFS Parameters for the nfssrv Module

Symbol Name	Description	Default Setting
`nfs_portmon`	This symbol controls whether the NFS server will do filtering of requests based on the IP port number. It uses the Berkeley notion of reserved port numbers.	Defaults to off (0). You can enable this symbol (1), but problems with interoperability might appear.
`nfsreadmap`	This symbol is no longer active. Map reads are no longer implemented. It is left to ease transitions.	Defaults to off (0).
`rfs_write_async`	This symbol controls whether the NFS Version 2 server will use write clustering to safely increase write throughput.	Defaults to on (1). You can disable this symbol (0), but performance might be reduced.

TABLE B–3 NFS Parameters for the rpcmod Module

Symbol Name	Description	Default Setting
svc_ordrel_timeout	This symbol lists the number of milliseconds after which the kernel forces a connection tear-down to complete. It is used in rare cases when a TCP connection that is used for kernel RPC gets hung while being torn down. The connection can get hung when an NFS server initiates a graceful close (FIN) of a connection, and a client fails to complete the close (FIN ackowledgment) handshake.	Defaults to 600000 ms (10 minutes). If the value is too small, the server does not allow clients enough time to tear down TCP connections properly. If the value is too large, a “buggy” or malicious client could tie up TCP connections on the server.

TABLE B–4 NFS Parameters for rpcsec Parameters

Symbol Name	Description	Default Setting
authdes_cachesz	This symbol controls the size of the authdes cache. It is a performance enhancement feature, that avoids verifying client credentials on every secure RPC request.	Defaults to 128.
authdes_win	This symbol controls how much clock skew is allowed between the server and clients when using AUTH_DES.	Defaults to 300 seconds.
authkerb_cachesz	This symbol controls the size of the authkerb cache.	Defaults to 128. System performance might be reduced if this value is set too high.
authkerb_win	This symbol controls how much clock skew is allowed between the server and clients when using AUTH_KERB.	Defaults to 300 seconds.
clnt_authdes_cachesz	This symbol controls the size of the cache table for sec=dh authentication handles on the client.	Defaults to 64.

How to Set the Value of a Kernel Parameter

1. **Become root.**

2. **Edit the `/etc/system` file and add a line to set the parameter.**

   Each entry should follow this form:

   set *module*:*symbol*=*value*

   where *module* is the name of the kernel module that contains the required parameter, *symbol* is the name of the parameter, and *value* is the numerical value to assign to the parameter. For example:

   ```
   set nfs:nfs_nra=4
   ```

   would change the number of read-ahead blocks that are read for file systems mounted using NFS version 2 software. See the `system`(4) man page for information about the `/etc/system` file.

3. **Reboot the system.**

Glossary

This glossary contains only definitions of new terms found in this book and are not in the Global Glossary. For definitions of other terms, see the Global Glossary at `http://docs.sun.com:80/ab2/coll.417.1/GLOBALGLOSS/@Ab2TocView`.

anycast address — An IP address that is assigned to more than one interface (typically belonging to different nodes), where a packet sent to an anycast address is routed to the *nearest* interface having that address, according to the routing protocol's measure of distance.

authentication header — An extension header that provides authentication and integrity (without confidentiality) to IPv6 datagrams.

autoconfiguration — The process of a host automatically configuring its interfaces in IPv6.

dual stack — In the context of IPv6 transition, a protocol stack that contains both IPv4 and IPv6, with the rest of the stack being identical.

encapsulating security header — An extension header that provides integrity and confidentiality to IPv6 datagrams.

encapsulation — The process of a header and payload being placed in the first packet, which is in turn placed in the second packet's payload.

firewall — Any device or software that protects an organization's private network or intranet from intrusion by external networks such as the Internet.

hop — A measure used to identify the number of routers that separate two hosts. If three routers separate a source and destination, the hosts are said to be three hops away from each other.

IPsec The security architecture (IPsec) that provides protection for IP datagrams.

IPv4 Internet Protocol, version 4. Sometimes referred to as IP. This version supports a 32–bit address space.

IPv6 Internet Protocol, version 6. This version supports a 128–bit address space.

key management The way in which you manage security associations.

link-local-use address A designation used for addressing on a single link for purposes such as automatic address configuration.

local-use address A unicast address that has only local routability scope (within the subnet or within a subscriber network), and can have a local or global uniqueness scope.

mobile IP A node able to move from one link to another without changing the its IP address.

MTU (maximum transmission unit) The size, given in octets, that can be transmitted over a link. For instance, the MTU of an Ethernet is 1500 octets.

multicast address An IP address that identifies a group of interfaces in such a way that a packet sent to a multicast address is delivered to all of the interfaces in the group.

neighbor advertisement A response to a neighbor solicitation message or the process of a node sending unsolicited neighbor advertisements to announce a link-layer address change.

neighbor discovery An IP mechanism that enables hosts to locate routers that reside on an attached link.

neighbor solicitation A solicitation sent by a node to determine the link-layer address of a neighbor, or to verify that a neighbor is still reachable by a cached link-layer address.

packet A group of information that is transmitted as a unit over communications lines. Contains a header plus payload.

redirect In a router, to inform a host of a better first-hop node to reach a particular destination.

router advertisement	The process of routers advertising their presence together with various link and Internet parameters, either periodically, or in response to a router solicitation message.
router discovery	The process of hosts locating routers that reside on an attached link.
router solicitation	The process of hosts requesting routers to generate router advertisements immediately, rather than at their next scheduled time.
SADB	(security associations database) A table that specifies cryptographic keys and algorithms used in the transmission of data.
security associations	(security associations) Associations that specify security properties from one host to another.
site-local-use address	A designation used for addressing on a single site.
SPI	(security parameters index) An integer that specifies the row in the SADB that a receiver should use to decrypt a received packet.
stateful autoconfiguration	The process of a host obtaining interface addresses and/or configuration information and parameters from a server.
stateless autoconfiguration	The process of a host generating its own addresses using a combination of locally available information and information advertised by routers.
tunneling	The mechanism by which IPv6 packets are placed inside IPv4 packets and routed through the IPv4 routers.
unicast address	An IP address that identifies a single interface.
VPN	(virtual private network) A single, secure, logical network that uses tunnels across a public network such as the Internet.

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z