NFS Services Administrator's Guide

HP-UX 11i version 3

HP Part Number: B1031-90070
Published: March 2010

Table of Contents

List of Figures

List of Tables

Preface: About This Document

The latest version of this document can be found on line at:

http://www.docs.hp.com

This document describes how to configure, and troubleshoot the NFS Services on HP-UX 11i v3.

The document printing date and part number indicate the document's current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The document part number will change when extensive changes are made.

Document updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

Intended Audience

This document is intended for system and network administrators responsible for installing, configuring, and managing the NFS Services product. Administrators are expected to have knowledge of NFS Services product.

Publishing History

Table 1 Publishing History Details

Document Manufacturing Part Number	Operating Systems Supported	Publication Date
B1031-90049	11i v2	September, 2003
B1031-90050	11.0, 11i v1, 1.5, 1.6	April, 2003
B1031-90051	11.0, 11i v1, 1.5, 1.6	August, 2003
B1031-90052	11.0, 11i v1, 1.5, 1.6	January, 2004
B1031-90053	11i v2	March, 2004
B1031-90054	11.0, 11i v1, 1.5, 1.6	March, 2004
B1031-90061	11i v3	February, 2007
B1031–90064	11i v3	January, 2008
B1031–90067	11i v3	July 2008
B1031–90068	11i v3	March 2009
B1031–90069	11i v3	September 2009
B1031-90070	11i v3	March 2010

What's in This document

This manual describes how to install, configure and troubleshoot the NFS Services product.

The manual is organized as follows:

Chapter 1	**Introduction** Describes the Network File System (NFS) services such NFS, AutoFS, and CacheFS. It also describes new features available with HP-UX 11i v3.
Chapter 2	**Configuring and Administering NFS** Describes how to configure and administer NFS services.
Chapter 3	**Configuring the Cache File System (CacheFS)** Describes the benefits of using the Cache File System and how to configure it on the HP-UX system.

Chapter 4	**Configuring and Administering AutoFS** Describes how to configure and administer AutoFS.
Chapter 5	**Troubleshooting NFS Services** Describes detailed procedures and tools for troubleshooting the NFS Services.

Typographical Conventions

This document uses the following conventions.

Italics	Identifies titles of documentation, filenames and paths.
Bold	Text that is strongly emphasized.
`monotype`	Identifies program/script, command names, parameters or display.

HP Encourages Your Comments

HP encourages your comments concerning this document. We are truly committed to providing documentation that meets your needs.

Please send comments to: feedback@fc.hp.com

Please include document title, manufacturing part number, and any comment, error found, or suggestion for improvement that you have concerning this document. Also, please include what we did right so we can incorporate it into other documents.

1 Introduction

This chapter introduces the Open Network Computing (ONC) services, such as NFS, AutoFS, and CacheFS.

This chapter addresses the following topics:

- “ONC Services Overview” (page 11)
- “Network File System (NFS)” (page 11)
- “New Features in NFS” (page 12)

ONC Services Overview

Open Network Computing (ONC) services is a technology that consists of core services which enable you to implement distributed applications in a heterogeneous, distributed computing environment. ONC also includes tools to administer clients and servers.

ONC services consists of the following components:

- Network File System (NFS) enables you to access files from any location on the network, transparently. An NFS server makes a directory available to other hosts on the network, by sharing the directory. An NFS client accesses the shared directories on the NFS server by mounting the directories. For users on the NFS client, the directories appear as a part of the local filesystem. For information on configuring and administering NFS, see Chapter 2: “Configuring and Administering NFS Services” (page 19).
- AutoFS > is a client-side service that automatically mounts and unmounts filesystems, transparently. AutoFS performs automatic mounting and unmounting by instructing the user-space daemon, `automountd`, to mount and unmount the directories it manages. For information on configuring and administering AutoFS, see “Configuring and Administering AutoFS” (page 49).
- CacheFS is a general purpose filesystem caching mechanism that can improve the NFS server performance scalability by reducing server and network load. For information on configuring and administering a cache filesystem, see “Configuring and Administering a Cache Filesystem” (page 81).
- Network Lock Manager (NLM) and Network Status Monitor (rpc.lockd and rpc.statd) provide file locking and synchronized file access to files that are shared using NFSv2 or NFSv3. File locking with NFSv2 and NFSv3 is advisory only. The `rpc.lockd` daemon starts the kernel KLM server. The `rpc.statd` daemon implements a lock recovery service used by KLM. It enables `rpc.lockd` daemon to recover locks after the NFS service restarts.

  Files can be locked using the `lockf`() or `fcntl()` system calls. For more information on daemons and system calls that enable you to lock and synchronize your files, see *lockd*(1M), *statd*(1M), *lockf*(2), and *fcntl*(2).
- Remote Procedure Call (RPC) is a mechanism that enables a client application to communicate with a server application. The NFS protocol uses RPC to communicate between NFS clients and NFS servers. You can write your own RPC applications using `rpcgen`, an RPC compiler that simplifies RPC programming. Transport-Independent RPC (TI-RPC) is supported on HP-UX 11i v3. For information on RPC, see *rpc*(3N) and *rpcgen*(1). For more information on RPC and `rpcgen`, see John Bloomer, *Power Programming with RPC*.

Network File System (NFS)

The Network File System (NFS) is a distributed filesystem that provides transparent access to files and directories that are shared by remote systems. It enables you to centralize the administration of these files and directories. NFS provides a single copy of the directory that can be shared by all the systems on the network, instead of duplicating common directories, such as `/usr/local` on each system.

How NFS works

The NFS environment consists of the following components:

- NFS Services
- NFS Shared Filesystems
- NFS Servers and Clients

NFS Services

The NFS services is a collection of daemons and kernel components, and commands that enable systems with different architectures running different operating systems to share filesystems across a network. The physical location of the filesystem does not affect the NFS services. The NFS services enable you to place a copy of the filesystem on an NFS server and allow all other systems, or a subset of systems in the network to access it.

NFS Shared Filesystems

Filesystems that are shared between an NFS server and an NFS client across a network are known as NFS filesystems. The shared filesystem can refer to an entire file hierarchy, or a single file.

NFS Servers and Clients

In the NFS context, a system that shares its filesystems over a network is known as a server, and a system that mounts and accesses these shared filesystems is known as a client. The NFS service enables a system to access a filesystem located on a remote system.

Once the filesystem is shared by a server, it can be accessed by a client. Clients access files on the server by mounting the shared filesystem. For users, these mounted filesystems appear as a part of the local filesystem.

New Features in NFS

This section discusses the new features that NFS supports on systems running HP-UX 11i v3.

NOTE: All versions of the NFS protocols are supported on HP-UX 11i v3. NFSv4 is not the default protocol. However, it can be enabled. For information on how to enable NFSv4, see "Configuring the NFSv4 Server Protocol Version " (page 22).

NFS Version 4 Protocol (NFSv4)

NFSv4 is an IETF standard protocol that provides the following features:

- Single Protocol

  In NFSv4, MOUNT, Network Lock Manager, and the ACL protocols are merged into the NFS protocol. Merging all these protocols into a single protocol enables you to configure firewalls easily. All NFSv4 requests are now sent to one port.

  In earlier versions of NFS, separate protocols were required to mount a filesystem, monitor a remote client or a server, lock or unlock a file, and unmount a filesystem. Configuring a firewall to permit access to specific ports that the protocols listened on was difficult (or impossible) if the client or the server was part of a firewalled network.

- COMPOUND Procedure

  The COMPOUND procedure decreases transport and security overhead, because of fewer over-the-wire trips between the client and the server. This feature is transparent to the user.

NFSv4 uses the COMPOUND RPC procedure to build sequence of requests into a single RPC. All RPC requests are classified as either NULL or COMPOUND. All requests that are part of the COMPOUND procedure are known as operations. An operation is a filesystem action that forms part of a COMPOUND procedure. NFSv4 currently defines 38 operations.

The server evaluates and processes operations sequentially. If an error is encountered, it is returned by the server for the entire procedure up to the first operation that causes the error.

The NFSv4 protocol design enables NFS developers to add new operations that are based on IETF specifications.

- Delegation

  In NFSv4, the server can delegate certain responsibilities to the client. Delegation enables a client to locally service operations, such as OPEN, CLOSE, LOCK, LOCKU, READ, and WRITE, without immediate interactions with the server.

  The server grants either a READ or a WRITE delegation to a client at OPEN. After the delegation is granted, the client can perform all operations locally. Delegations are disabled, by default.

  Delegations can be revoked by the server. If another client requests incompatible access to the file, the server revokes the delegation. Once the delegation is returned, the other client can access the file.

- Built-In Security

  In NFSv4, the built-in security feature enables the RPC layer to use different security mechanisms. You can specify a different security policy for each filesystem. The server and the clients can negotiate supported security flavors on a per filesystem basis.

  NFSv4 uses the RPCSEC_GSS framework for authentication, integrity, and privacy. This framework enables the RPC protocols to access the Generic Security Services Application Programming Interface (GSS-API).

  The RPCSEC_GSS/GSS-API security framework is extensible. However, the new security flavors, whenever added, must conform to the GSS-API model.

  For information on how to secure your systems, see "Secure Sharing of Directories " (page 26).

- ACLs

  An Access Control List (ACL) provides stronger file security, by enabling the owner of a file to define file permissions for the file owner, the group, and other specific users and groups. ACL support is built into the protocol. ACLs can be managed from an NFS client using either the `setacl` or the `getacl` command.

  For more information on ACLs, see *acl*(1M),*getacl*(1M), and *setacl*(1M).

NOTE: In NFSv2 and NFSv3, ACLs are manipulated using NFSACL protocol. If systems in your environment do not support the NFSACL protocol, then ACLs cannot be manipulated using this feature.

- File Handle Types

  File handles are created on the server and contain information that uniquely identify files and directories. Following are the different file handle types:

  — ROOT

    The ROOT file handle represents the conceptual root of the file system namespace on an NFS server. The NFS client starts with the ROOT file handle by using the PUTROOTFH operation. This operation instructs the server to set the current file handle to the root of the server file tree. If you use the PUTROOTFH operation, the client can traverse the entire file tree using the LOOKUP operation.

— Persistent

The persistent file handle is an assigned fixed value for the lifetime of the filesystem object that it refers to. When the server creates the file handle for a filesystem object, the server must accept the same file handle for the lifetime of the object. The persistent file handle persists across server reboots and filesystem migrations.

— Volatile

Volatile file handles can be set to expire at a certain time. For example, they can be set to expire during the filesystem migration. This file handle type is useful for servers that cannot implement persistent file handles. However, the volatile file handles do not share the same longevity characteristics of a persistent file handle, because these file handles can become invalid or expire.

NOTE: HP-UX supports only persistent file handles. The client must know how to handle persistent and volatile file handles. However, the server is not required to support both types.

- Namespace Changes

  The namespace describes the set of available files that are arranged in a hierarchy. When a server shares files, it typically shares only a portion of its namespace. In NFSv4, the shared namespace of the server forms a single hierarchy.

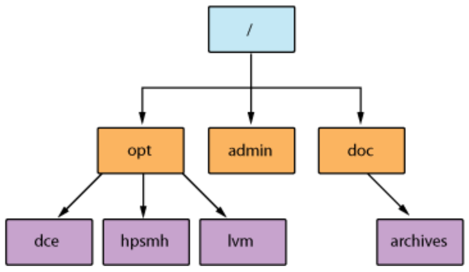
  When a server shares separate filesystems as a disconnected portion of its namespace, the server creates a pseudo filesystem to bridge the unshared portions of the namespace. This enables a client, which has been enabled to traverse remote filesystems without having to mount them, to access the shared points from a single common root.

  The NFSv4 specification does not require a client to traverse the NFS server's namespace.

  For example, a server shares `/opt/dce`, `/opt/hpsmh`, and `/doc/archives` directories. In this list, the shared list hierarchy is not connected. The `/doc/archives` directory is neither a parent nor a child directory of `/opt`.

  Figure 1-1 shows the server view of the shared directories.

  Figure 1-1 Server View of the Shared Directories

  If the administrator shares `/`, and `/opt/dce`. If the client mounts then the NFS client using NFSv2 and NFSv3 can mount `/`and `/opt/dce`. Attempts to mount `/opt` will fail.

  In NFSv4, the client can mount `/`, `/opt` and `/opt/dce`. If the client mounts `/opt` and lists the contents of the directory, only the directory dce is seen. If you change directory (`cd`) to `dce`, the contents of `dce` will be visible. HP-UX will allow the client to transparently cross this filesystem boundary if the client has installed B.11.31.03 or greater.

- String Identifiers

  NFSv4 represents users and groups in the following manner:

users@domain

Or

group@domain

Where:

users	specifies the string representation of the user
group	specifies the string representation of the group
domain	specifies a registered DNS domain or a sub-domain of a registered domain

However, UNIX systems use integers to represent users and groups in the underlying filesystems stored on the disk. As a result, using string identifiers requires mapping of string names to integers and back.

The `nfsmapid` daemon is used to map the owner and owner_group identification attributes with the local user identification (UID) and group identification (GID) numbers, which are used by both the NFSv4 server and the NFSv4 client.

On the NFS client, the `nfsmapid` daemon maps a numeric UID or a numeric GID to a string representation. On the NFS server, it maps a string representation to a numeric UID or a numeric GID.

For example, user **`casey`** with UID 121212 on an NFSv4 client whose fully qualified name is **`system.anydomain.com`** is mapped to **`casey@anydomain.com`**. The NFS client sends the string representation, **`casey@anydomain.com`**, to the NFS server. The NFS server maps the string representation, **`casey@anydomain.com`**, to the unique ID 121212.

The `nfsmapid` daemon uses the user and group entries in the `/etc/nsswitch.conf` file, to determine which name service to use while converting strings and numerical values and back.

For information on how to configure the `nfsmapid` daemon, see *nfsmapid*(1M).

- File Locking

  File locking support is integrated with the NFSv4 protocol. NFSv4 introduces leases for lock management.

  When a server grants a lock to control the access of a file for a specific period of time, it is called a lease. During the lease period, the server cannot grant conflicting access to another client.

  The lock lease interval can accept values higher than the cache consistency leases. As a result, the client requires only a smaller number of refreshes. The lease ensures that the client does not lose the locking state.

  If a client refreshes a lock, all locks held by the client with that server are validated. This reduces the number of lease refreshes by the client from one per lock to one per client for a specific lease.

For information on configuring NFSv4 for the server, see "Configuring the NFSv4 Server Protocol Version " (page 22). For information on configuring NFSv4 for the client, see "Configuring the NFSv4 Client Protocol Version" (page 35).

Sharing and Unsharing Directories

In HP-UX 11i v3, NFS replaces the `exportfs` command with the `share` command. The `share` command is used on the server to share directories and files with clients. You can use the `unshare` command to disable the sharing of directories with other systems.

In earlier versions of HP-UX, the `exportfs` command was used to export directories and files to other systems over a network. Users and programs accessed the exported files on remote systems as if they were part of the local filesystem. NFS disabled the system from exporting directories by using the `-u` option of the `exportfs` command.

For information on how to share directories with the NFS clients, see "Sharing Directories with NFS Clients" (page 23). For information on how to unshare directories, see "Unsharing (Removing) a Shared Directory" (page 33).

Following are the new `share` features that NFS supports:

- Secure sharing of directories

  Starting with HP-UX 11i v3, NFS enables you to share directories in a secure manner. If you have configured a security mechanism on your system, the `share` command enables you to specify the security mechanism under which the filesystem is shared. For example, if you have installed and configured the Kerberos Security product, you can specify *`krb5`* as the security mechanism to be used when you share a filesystem.

  For information on how to use the `share` command to share directories securely, see "Secure Sharing of Directories " (page 26).

- Sharing directories across a firewall

  Starting with HP-UX 11i v3, NFS enables you to share directories easily across a firewall. HP recommends that you share directories across a firewall in one of the following ways:

  — Using random ports (NFSv2 and NFSv3)

  For information on how to share directories across a firewall using random ports, see "Sharing directories across a firewall without fixed port numbers (NFSv2 and NFSv3)" (page 30).

  — Using the `/etc/default/nfs` file

  For information on how to share directories across a firewall using the `/etc/default/nfs` file, see "Sharing directories across a firewall using fixed port numbers in the nfs file" (page 31).

  — Using the NFSv4 protocol

  For information on how to share directories across a firewall using the NFSv4 protocol, see "Sharing directories across a firewall using the NFSv4 protocol" (page 32).

  — Using the WebNFS feature

  For information on how to share directories across a firewall using the WebNFS feature, see "Sharing directories across a firewall using the WebNFS Feature" (page 32).

Mounting and Unmounting Directories

NFS clients can mount any filesystem or a part of a filesystem that is shared by the NFS server. Filesystems can be mounted automatically when the system boots, from the command line, or through the automounter. The different ways to mount a filesystem are as follows:

- Mounting a filesystem at boot time and using the `mount` command

  For information on how to mount a filesystem at boot time, see "Mounting a Remote Directory on an NFS client" (page 37).

- Mounting a filesystem using an NFS URL

  For information on how to mount an NFS filesystem using an NFS URL, see "Examples of NFS Mounts " (page 38).

- Mounting an NFS filesystem through a firewall

  For information on how to mount an NFS filesystem through a firewall, see "Accessing Shared NFS Directories across a Firewall" (page 30).

- Mounting a filesystem securely

  For information on how to mount a filesystem in a secure manner, see "An Example for Securely Mounting a directory" (page 40).

For information on how to disable mount access for a single client, see "Unmounting (Removing) a Mounted Directory" (page 40).

Starting with HP-UX 11i v3, the `mount` command is enhanced to provide benefits such as performance improvement of large sequential data transfers and local locking for faster access. The `umount` command allows forcible unmounting of filesystems. These features can be accessed using specific options of the `mount` command. For more information on these options, see mount_nfs (1M), and *umount*(1M).

NFS clients can also unmount the directories using the `umount` command. For information on unmounting a shared directory, see "Unsharing (Removing) a Shared Directory" (page 33).

Support for WebNFS

NFS is designed as a file access protocol for LANs. WebNFS is an extension of NFS. It enables you to access files across the Internet easily. WebNFS is designed to handle unique problems associated with accessing files across the Internet.

WebNFS enables filesystems at other locations on the Internet to appear to a user as a local filesystem. WebNFS works through firewalls and implements features such as read-ahead and write-behind, to improve throughput and performance over the Internet.

For more information on WebNFS, see "Sharing directories across a firewall using the WebNFS Feature" (page 32).

Secure Sharing of Directories

In earlier versions of HP-UX, NFS used the AUTH_SYS authentication, which uses UNIX style authentication, (`uid`/`gid`), to allow access to the shared files. It is fairly simple to develop an application or server that can masquerade as a user because the `gid`/`uid` ownership of a file can be viewed.

The AUTH_DH authenticating method was introduced to address the vulnerabilities of the AUTH_SYS authentication method. The AUTH_DH security model is stronger, because it authenticates the user by using the user's private key.

Kerberos is an authentication system that provides secure transactions over networks. It offers strong user authentication, integrity and privacy. Kerberos support has been added to provide authentication and encryption capabilities. For information on how to share directories in a secure manner, see "Secure Sharing of Directories " (page 26).

Client Failover

By using client-side failover, an NFS client can specify redundant servers that are making the same data available and switch to an alternate server when the current server becomes unavailable. The filesystems on the current server can become unavailable for the following reasons:

- If the filesystem is connected to a server that crashes
- If the server is overloaded
- If a network fault occurs

A failover occurs when the filesystem is unavailable. The failover is transparent to the user. The failover can occur at any time without disrupting processes that are running on the client.

Consider the following points before enabling client-side failover:

- The filesystem must be mounted with read-only permissions.
- The filesystems must be identical on all the redundant servers for the failover to occur successfully. For information on identical filesystems, see "Replicated Filesystems" (page 18).
- A static filesystem or one that is not modified often is used for failover.

- File systems that are mounted using CacheFS are not supported for use with failover.
- If client-side failover is enabled using the command-line option, the listed servers must support the same version of the NFS protocol. For example, `onc21` and `onc23` must support the same version of NFS protocol, either NFSv2, NFSv3, or NFSv4.

For information on how to enable client-side failover, see "Enabling Client-Side Failover" (page 38).

Replicated Filesystems

A replicated filesystem contains the corresponding directory structures and identical files. A replica (identical copy) of a filesystem consists of files of the same size and same file type as the original filesystem.

HP recommends that you create these replicated filesystems using the `rdist` utility.

The `rdist` utility enables you to maintain identical copies of files on multiple hosts. It preserves the owner, group, mode, modification time of files, and updates programs that are running.

Enhanced NFS Logging

The NFS server logging enables an NFS server to provide a record of file operations that are performed on its filesystems. The record includes information about the file accessed, time of access, and the users who accessed the files. You can also specify the location of the logs that contain this information. This feature is useful for sites that make anonymous FTP archives available to NFS and WebNFS clients.

IPv6 Support

NFS supports filesystem mounting over an IPv4 or an IPv6 address where square brackets enclose the IPv6 address.

The `nsquery` feature supports ipnodes lookup request and provides support to lookup IPv6 data in the backend libraries.

2 Configuring and Administering NFS Services

This chapter describes how to configure and administer an HP-UX system as an NFS server or an NFS client, using the command-line interface.

An NFS server exports or shares its local filesystems and directories with NFS clients. An >NFS client mounts the files and directories exported or shared by the NFS servers. NFS-mounted directories and filesystems appear as a part of the NFS client's local filesystem.

This chapter addresses the following topics:

- "Prerequisites " (page 19)
- "Configuring and Administering an NFS Server" (page 21)
- "Configuring and Administering NFS Clients " (page 34)
- "Configuring and Using NFS Netgroups" (page 43)
- "Configuring RPC-based Services" (page 47)

Prerequisites

Before you configure your system as an NFS server or as an NFS client, perform the following prerequisite checks:

- Verify network connectivity
- Verify user IDs and group IDs setup
- Verify group restrictions

Verifying Network Connectivity

Before you configure NFS, you must have already installed and configured the network hardware and software on all the systems that use NFS. For information on installing and configuring the network hardware and software, see the following documents:

- *HP-UX LAN Administrator's Guide (B2355-90796)*
- Ethernet Configuration and Verification (5991-5954)

To check the network connections between the server and clients, use the `/usr/sbin/ping` command to verify network connectivity between your proposed NFS client and server systems.

Verifying User IDs and Group IDs Setup

When users request NFS access to remote files, their user IDs and group IDs are used to check file ownership and permissions.

NOTE: NFSv4 uses string identifiers that map to user IDs and group IDs in the standard integer format. For more information on string identifiers supported on NFSv4, see "New Features in NFS" (page 12).

Consider the following points when you set user IDs and group IDs:

- Each user must have the same user ID on all systems where that user has an account.
- Each group has the same group ID on all systems where that group exists.
- No two users on the network have the same user ID.
- No two groups on the network have the same group ID.

You can set user and group IDs in the following methods:

- Using the HP-UX System Files (`/etc/passwd` and `/etc/group`)
- Using NIS
- Using LDAP

Using the HP-UX System Files

If you are using the HP-UX system files, add the users and groups to the `/etc/passwd` and `/etc/group` files, respectively. Copy these files to all the systems on the network.

For more information on the `/etc/passwd` and `/etc/group` files, see *passwd*(4) and *group*(4).

Using NIS

If you are using NIS, all systems on the network request user and group information from the NIS maps on the NIS server. For more information on configuring NIS, see NIS Administrator's Guide (5991-2187).

Using LDAP

If you are using LDAP, all systems on the network request user and group information from the LDAP directory server. For more information on configuring LDAP, see the *LDAP-UX Client Services B.04.00 Administrator's Guide(J4269-90064)*.

Verifying Group Restrictions

HP-UX supports 20 group entries per user, but the maximum number of group entries supported by RPC is 16. If a user is a member of more than 16 groups, NFS truncates the number of entries to 16 when the user attempts access to files or directories. Use the `groups` command to find the group membership of a user. For more information on groups, see *groups*(1). To ensure that a user is not a member of more than 16 groups, use the following procedures depending on the user and group configuration method you use:

- Using the HP-UX System Files
- Using NIS
- Using LDAP

Using the HP-UX System Files

If you are using HP-UX system files to manage your `group` database, follow these steps:

1. To identify the number of groups that the user belongs to, enter the following command for each user on your system:

   ```
   /usr/bin/grep -x -c username /etc/group
   ```

   This command returns the number of occurrences of *`username`* in the `/etc/group` file.

2. To remove a user from one or more groups, delete the user from the group entries in the `/etc/group` file.

Using NIS

If you are using NIS to manage your `group` database, follow these steps:

1. To identify the number of groups that the user belongs to, enter the following command :

   ```
   /usr/bin/ypcat -k group | /usr/bin/grep -c username
   ```

   This command returns the number of occurrences of *`username`* in the NIS `group` map.

2. To remove the user from one or more groups in the NIS map, follow the instructions described in the NIS Administrator's Guide (5991-2187).

Using LDAP

For more information on managing user profiles using LDAP, see the *LDAP-UX Client Services B.04.00 Administrator's Guide (J4269-90064)*.

Configuring and Administering an NFS Server

Configuring an NFS server involves completing the following tasks:

1. Identify the set of directories that you want the NFS server to share. For example, consider an application `App1` running on an NFS client. Application `App1` requires access to the `abc` filesystem in an NFS server. NFS server should share the `abc` filesystem. The decision of sharing directories by an NFS server is driven by the applications running on NFS clients that require access to those directories.
2. Specify access restrictions and security modes for the shared directories. For example, you can use Kerberos (a security product) that is already configured on your system to specify access restrictions and security modes for the NFS shared directories.

NFS Configuration Files and Daemons

This section describes the NFS configuration files and daemons.

Configuration Files

Table 2-1 describes the NFS configuration files and their functions.

Table 2-1 NFS Server Configuration Files

File Name	Function
`/etc/rc.config.d/nfsconf`	Contains the variables read by the start-up scripts of the NFS subsystem.
`/etc/pcnfsd.conf`	Contains the PC NFS configuration information.
`/etc/default/nfs`	Contains the parameters to set the default behavior of various NFS commands and daemons.
`/etc/dfs/dfstab`	Contains the `share` commands that are executed when the NFS server subsystem starts.
`/etc/dfs/fstypes`	Contains the distributed filesystem types. The default filesystem is NFS.
`/etc/nfssec.conf`	Contains the valid and supported NFS security modes.
`/etc/dfs/sharetab`	Contains the system record of shared filesystems.
`/etc/rmtab`	Contains all the entries for hosts that have mounted filesystems from any system.

Daemons

Table 2-2 describes the NFS server daemons.

Table 2-2 NFS Server Daemons

Daemon Name	Function
`rpc.mountd`	Answers requests for NFS access information and filesystem mount requests. The `rpc.mountd` daemon reads the `/etc/dfs/sharetab` file to determine which filesystems are available for mounting and by which remote systems.
`nfsd`	Handles client filesystem requests. By default, `nfsd` starts over TCP, TCP6, UDP, and UDP6 transports.
`nfslogkd`	Flushes `nfslog` information from the kernel to a file.
`nfsmapid`	Maps to and from NFSv4 owner and owner group identification attributes to local UID and GID numbers used by both NFSv4 client and server.
`nfs4srvkd`	Supports server side delegation.

Table 2-2 NFS Server Daemons *(continued)*

Daemon Name	Function
`rpc.lockd`	Supports record lock and share lock operations on the NFS files.
`rpc.statd`	Maintains a list of clients that have performed the file locking operation over NFS against the server. These clients are monitored and notified in the event of a system crash.

Following are the tasks involved in configuring and administering an NFS server:

- Configuring the NFSv4 Server Protocol Version (Optional)
- "Enabling an NFS Server " (Required)
- "Sharing Directories with NFS Clients" (Required)
- "Configuring an NFS Server for use by a PC NFS client" (Optional)
- "Unsharing (Removing) a Shared Directory" (Optional)
- "Disabling the NFS Server " (Optional)

Configuring the NFSv4 Server Protocol Version

By default, the version of the NFS protocol used between the client and the server is the highest one available on both systems. On HP-UX 11i v3, the default maximum protocol version of the NFS server and the client is 3. The default minimum protocol version of the NFS server and the client is 2.

To configure the NFS server to enable clients to mount filesystems using protocol version 4 (NFSv4), follow these steps:

1. Set the value of `NFS_SERVER_VERSMAX` variable to 4 in the `/etc/default/nfs` file, as follows:

   ```
   NFS_SERVER_VERSMAX=4
   ```

The `NFS_SERVER_VERSMAX` variable specifies the maximum protocol version of the NFS protocol for communication. For more information on NFSv4, see *nfsd*(1m), *mount_nfs*(1m), and *nfsmapid*(1m).

Enabling an NFS Server

To enable an NFS server, follow these steps:

1. In the `/etc/rc.config.d/nfsconf` file, ensure that the `NFS_SERVER` and `START_MOUNTD` variables are set to 1:

   ```
   NFS_SERVER=1
   START_MOUNTD=1
   ```

2. Enter the following command to verify whether `rpcbind` daemon is running:

   ```
   ps -ae | grep rpcbind
   ```

   If the daemon is running, an output similar to the following is displayed:

   ```
   778 ? 0:04 rpcbind
   ```

   No message is displayed if the daemon is not running.

   To start the `rpcbind` daemon, enter the following command:

   ```
   /sbin/init.d/nfs.core start
   ```

3. Enter the following commands to verify whether the `lockd` and `statd` daemons are running:

   ```
   ps -ae | grep rpc.lockd
   ps -ae | grep rpc.statd
   ```

   If the daemons are running, an output similar to the following is displayed:

   ```
   1069548396 ?        0:00 rpc.lockd
   ```

```
1069640883 ?         0:00 rpc.statd
```

No message is displayed if the daemons are not running.

To start the `lockd` and `statd` daemons, enter the following command:

```
/sbin/init.d/lockmgr start
```

4. Enter the following command to run the NFS startup script:

   ```
   /sbin/init.d/nfs.server start
   ```

The NFS startup script enables the NFS server and uses the variables in the `/etc/rc.config.d/nfsconf` file to determine which processes to start.

Sharing Directories with NFS Clients

The `share` command exports or shares local directories with NFS clients. If you use the `share` command without specifying any options, it displays the list of currently shared directories. You can also use the `share` command with options to specify access restrictions and other access options for the shared directories.

NOTE: The `exportfs` command, used to export directories in versions prior to HP-UX 11i v3, is now a script that calls the `share` command. HP provides a new `exportfs` script for backward compatibility to enable you to continue using `exportfs` with the functionality supported in earlier versions of HP-UX. To use any of the new features provided in HP-UX 11i v3 you must use the `share` command. In HP-UX 11i v3, the`/etc/dfs/dfstab` file replaces the /etc/exports file and the /etc/dfs/sharetab file replaces the `/etc/xtab` file.

You can share a filesystem in the following methods:

- Automatic sharing

  Filesystems configured in the `/etc/dfs/dfstab` file are shared automatically during system reboot. To set up a directory for automatic sharing, you must configure it in the `/etc/dfs/dfstab` file.

- Manual sharing

  Filesystems can also be shared manually. However, if the system is restarted or rebooted, the filesystems will no longer be shared and must be re-shared before it can be accessed by NFS clients. To share directories manually, you must run the `share` command.

- Sharing using scripts

  Sharing of filesystems using scripts, such as Serviceguard, is a variant of manual sharing. Configure the filesystems that you share across your Serviceguard clusters through scripts that are executed under Serviceguard's control. For more information on configuring the filesystems for sharing using scripts, see Managing Serviceguard, 12th Edition, March 2006.

Consider the following points before you share a directory:

- You cannot share a directory and its ancestor or descendant if they are on the same disk or logical volume.

  For example, if you share the root directory (/), you cannot share `/opt`, unless / and `/opt` are on different disks or logical volumes. Similarly, if you share `/opt/frame`, you cannot share `/opt`, unless `/opt/frame` and `/opt` are on different disks or logical volumes.

NOTE: Use the `bdf` command to determine whether your filesystems are on different disks or logical volumes. Each entry in the `bdf` output represents a separate disk or volume that requires its own entry in the `/etc/dfs/dfstab` file, if shared. For more information on the `bdf` command, see *bdf*(1M).

- When you share a directory, the `share` options that restrict access to a shared directory are applied, in addition to the regular HP-UX permissions on that directory.

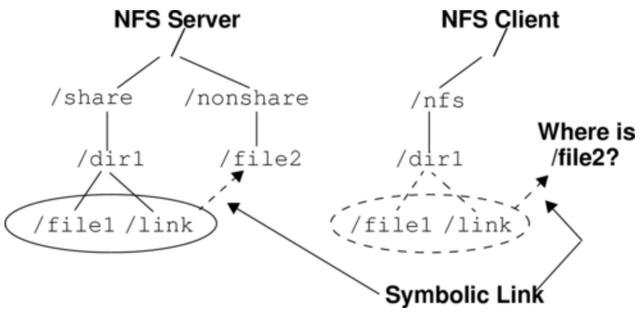
  For example, if only the owner of a file has write permission, others cannot write to the file even if it is shared with read and write permissions.
- You can also specify the access permissions on the NFS client, when a directory is mounted. If these permissions differ from the permissions for the shared directory on the NFS server, the more restrictive permissions are used.

  For example, consider an NFS client that mounts a directory with read and write permissions while the directory is shared by the NFS server with read permission. Read permissions being more restrictive, the NFS client only has read permission.
- Exercise caution when sharing a directory that contains a symbolic link, which refers to data outside the NFS mounted directory.

  Once the directory is mounted on an NFS client, the symbolic link is resolved locally on the client.

Figure 2-1 depicts symbolic links in NFS mounts. The destination of the symbolic link exists on the NFS server, but does not exist on the NFS client. This results in an error message.

Figure 2-1 Symbolic Links in NFS Mounts



Sharing a directory with NFS Clients

Before you share your filesystem or directory, determine whether you want the sharing to be automatic or manual. To share a directory with NFS clients, select one of the following methods:

- Automatic Share
- Manual Share

Automatic Share

To share your directories automatically, follow these steps:

1. Add an entry to the `/etc/dfs/dfstab` file for each directory you want to share with the NFS clients. Following is an example of an entry in the `/etc/dfs/dfstab` file for the netgroup `Developers`. All the hosts that are part of the netgroup will now have read and write permissions to the `/home` directory.

   ```
   share -F nfs -o rw="Developers" -d "home dirs"  /home
   ```

   Where:

`-F`	Specifies the filesystem type
`nfs`	Specifies that the filesystem type is NFS

`-o`	Enables you to use some of the specific options of the `share` command, such as `sec`, `async`, `public`, and others.
`-d`	Enables you to describe the filesystem being shared

When NFS is restarted or the system is rebooted, the `/etc/dfs/dfstab` file is read and all directories are shared automatically.

2. Share all the directories configured in the `/etc/dfs/dfstab` file without restarting the server by using the following command:

   ```
   shareall
   ```

   This command reads the entries in the `/etc/dfs/dfstab` file and shares all the directories.

3. Verify that your filesystem is shared by entering the following command:

   ```
   share
   ```

   An output similar to the following output is displayed:

   ```
     /home    rw=Developers, ro=    "home dirs"
   ```

   All the directories that you have shared must be present in this list.

Manual Share

To share your directories manually, follow these steps:

1. Enter the following command to add a directory to the server’s internal list of shared directories:

   ```
   share -F nfs directory_name
   ```

2. Enter the following command to verify if your filesystem is shared:

   ```
   share
   ```

   An output similar to the following output is displayed:

   ```
   /tmp rw=hpdfs001.cup.hp.com “”
   /mail rw “”
   /var rw “”
   ```

   The directory that you have shared must be present in this list.

For more information on the `share` command and a list of share options, see *share_nfs*(1M) and *share*(1M).

Examples for Sharing directories

This section discusses different examples for sharing directories.

- Sharing a directory with read-only access

  ```
  share -F nfs -o ro /tmp
  ```

  In this example, all clients are allowed read-only access to the `/tmp` directory. The `/tmp` directory needs to be configured to allow read access to users on the clients. For example, specify `-r--r--r--` permissions for the `/tmp` directory.

- Sharing a directory with varying access permissions

  ```
  share -F nfs -o ro=Jan:Feb,rw=Mar /usr/kc
  ```

  In this example, the `/usr/kc` directory is shared with clients `Jan`, `Feb`, and `Mar`. The `rw` option specifies that users on client `Mar` have read-write access to the `/usr/kc` directory. The `ro` option specifies that users on `Jan` and `Feb` have read-only access.

  In addition to the `share` options, the HP-UX permissions for the `/usr/kc` directory must be set to allow access to all users or group that includes the users on `Jan`, `Feb`, and `Mar`.

- Sharing a directory with root access for clients

  ```
  share -F nfs -o root=Red:Blue:Green /var/mail
  ```

In this example, the `/var/mail` directory is shared. Root access is allowed for clients `Red`, `Blue`, and `Green`. Superusers on all other clients are considered as unknown by the NFS server, and are given the access privileges of an anonymous user. Non-superusers on all clients are allowed read-write access to the `/var/mail` directory if the HP-UX permissions on the `/var/mail` directory allow them read-write access.

- Sharing a directory with root access for superuser and read-write access for other users

  ```
  share -F nfs -o rw=Red,root=Red /var/mail/Red
  ```

  In this example, the `/var/mail/Red` directory is shared. Only the superuser on client `Red` is granted root access to the directory. All other users on client `Red` have read-write access if they are provided read-write access by the regular HP-UX permissions. Users on other clients have read-only access if they are allowed read access through the HP-UX permissions.

- Sharing directories with anonymous users based on access rights given to the superuser

  ```
  share -F nfs -o rw=Green,root=Green,anon=65535 /vol1/grp1/Green
  ```

  In this example, superusers on host `Green` use uid 0 and are treated as root. The root users on other hosts (`Red` and `Blue`) are considered anonymous and their uids and gids are re-mapped to 65535. The superusers on host Green are allowed read-write access. All other clients get read-only access.

- Sharing directories with anonymous users based on access rights given to them

  ```
  share -F nfs -o anon=200 /export/newsletter
  ```

  In this example, the `/export/newsletter` directory is shared with all clients. Anonymous users are given the effective user ID of 200. Other users retain their own user IDs (even if they do not exist in the NFS server's `passwd` database).

  Anonymous users are users who have not been authenticated, or requests that use the `AUTH_NONE` security mode, or root users on hosts not included in the *root*=`list`. By default, anonymous users are given the effective user ID, `UID_NOBODY`. If the user ID is set to -1, access is denied.

  The `ls` command displays that a file created by a superuser is owned by user ID 200. If an anonymous user with a non-zero user ID, for example, 840, is allowed to create a file in this directory, the `ls` command displays that it is owned by user ID 840.

Secure Sharing of Directories

The `share` command enables you to specify a security mode for NFS. Use the `sec` option to specify the different security modes. Table 2-3 describes the security modes of the `share` command.

Table 2-3 Security Modes of the share command

Security Mode	Description
`sys`	Uses the default authentication method, AUTH_SYS. The `sys` mode is a simple authentication method that uses UID/GID UNIX permissions, and is used by NFS servers and NFS clients using the version 2, 3, and 4 protocol.
`dh`	Uses the Diffie-Hellman public-key system and uses the AUTH_DES authentication.
`krb5`	Uses Kerberos V5 protocol to authenticate users before granting access to the shared filesystems.
`krb5i`	Uses Kerberos V5 authentication with integrity checking to verify that the data is not tampered with, while in transit between the NFS clients and servers.
`krb5p`	Uses Kerberos V5 authentication, integrity checking, and privacy protection (encryption) on the shared filesystems.
`none`	Uses NULL authentication (AUTH_NONE). NFS clients using AUTH_NONE are mapped to the anonymous user `nobody` by NFS.

You can combine the different security modes. However, the security mode specified in the host must be supported by the client. If the modes on the client and server are different, the directory cannot be accessed.

For example, an NFS server can combine the `dh` (Diffie-Hellman) and `krb5` (Kerberos) security modes as it supports both the modes. However, if the NFS client does not support `krb5`, the shared directory cannot be accessed using `krb5` security mode.

Consider the following points before you specify or combine security modes:

- The `share` command uses the AUTH_SYS mode by default, if the `sec=`*`mode`* option is not specified.
- If your network consists of clients with differing security requirements, some using highly restrictive security modes and some using less secure modes, use multiple security modes with a single `share` command.

  For example, consider an environment where all clients do not require same level of security. This environment is usually difficult to secure and requires running various scripts. However, if you use the `share` command, you can specify different security mechanisms for each netgroup within your network.
- If one or more explicit *sec*= options are specified, you must set the `sys` security mode to continue to allow access to share directories, using the AUTH_SYS authentication method.

  For example, if you are specifying multiple security options, such as Kerberos and Diffie-Hellman, then specify the `sys` security option as well to enable users to access the shared directories using the AUTH_SYS security method.
- If `ro` and `rw` options are specified in a `sec`clause, the order of the options rule is not enforced. All hosts are granted read-only access, except those in the read-write list.

Secure NFS Setup with Kerberos

Configuring Secure NFS Server with Kerberos

Set up the NFS server as a Kerberos client before securing the NFS server.

To configure secure NFS server, follow these steps:

1. Set up the host as a Kerberos client. For more information on setting up the NFS server as a Kerberos client, see Configuration Guide for Kerberos Client Products on HP-UX (5991-7685).

NOTE: In all of this section, the following systems will be used as examples:

`Kerberos Server: onc52.ind.hp.com`

`NFS Server: onc20.ind.hp.com`

`NFS Client: onc36.ind.hp.com`

2. Synchronize the date & time of server nodes with kerberos server. To change the current date and time use date command followed by the current date and time. For example, enter `date 06101130` to set the date to June 10th and time to 11:30 AM. The time difference between the systems should not be more than 5 minutes.
3. Add a principal for all the NFS server to the Kerberos database. For example, if our NFS server is `onc20.ind.hp.com` then `nfs/onc20.ind.hp.com` principal should be added to the Kerberos database before running the NFS applications.

   To add principals use the Kerberos administration tool, *kadminl*

```
onc52# /opt/krb5/admin/kadminl
Connecting as: K/M
Connected to krb5v01 in realm ONC52.IND.HP.COM.

Command: add nfs/onc20.ind.hp.com
```

```
Enter password:
Re-enter password for verification:
Enter policy name (Press enter key to apply default policy) :
Principal added.
```

4. Copy the `/etc/krb5.conf` file from the Kerberos server to the NFS server node.

   ```
   onc52# rcp /etc/krb5.conf onc20:/etc/
   ```

5. Extract the key for the NFS service principal on the Kerberos server and store it in the /etc/krb5.keytab file on the NFS server. To extract the key, use the Kerberos administration tool kadminl.

   ```
   onc52# /opt/krb5/admin/kadminl
   Connecting as: K/M
   Connected to krb5v01 in realm ONC52.IND.HP.COM.

   Command: ext
   Name of Principal (host/onc52.ind.hp.com): nfs/onc20.ind.hp.com
   Service Key Table File Name (/opt/krb5/v5srvtab): /etc/onc20.keytab
   Principal modified.
   Key extracted.

   onc52# rcp /etc/onc20.keytab onc20:/etc/krb5.keytab
   ```

6. To verify the keys in NFS server, enter the following command in NFS server.

   ```
   onc20# klist -k
   Keytab name: FILE:/etc/krb5.keytab
   KVNO Principal
   ---- --------------------------------------------------------------------------
   1 nfs/onc20.ind.hp.com@ONC52.IND.HP.COM
   ```

7. Edit the `/etc/nfssec.conf` file and uncomment the entries for krb5, krb5i, or krb5p based on the security protocol you want to choose.

   ```
   onc20# cat /etc/nfssec.conf | grep krb5
   krb5    390003  krb5_mech     default   -                # RPCSEC_GSS
   krb5i   390004  krb5_mech     default integrity          # RPCSEC_GSS
   krb5p   390005  krb5_mech     default privacy            # RPCSEC_GSS
   ```

8. Edit the `/etc/inetd.conf` file and uncomment gssd entry.

   ```
   onc20# cat /etc/inetd.conf | grep gssd

   rpc  xti  ticotsord  swait  root  /usr/lib/netsvc/gss/gssd    100234  1    gssd
   ```

9. Re-initialize inetd on NFS servers.

   ```
   inetd –c
   ```

10. To create a credential table, enter the following command:

    ```
    onc20# gsscred -m krb5_mech -a
    ```

11. Share a directory with the Kerberos security option.

    ```
    onc20# share -F nfs -o sec=krb5,rw /share_krb5
    ```

    If you have not uncommented the entries of krb5, krb5i or krb5p, an error similar to the following error is displayed:

    ```
    # share -o sec=krb5i /aaa
    share_nfs: Invalid security mode "krb5i"
    ```

Secure NFS Client Configuration with Kerberos

To secure NFS client setup using Kerberos, follow these steps:

1. Synchronize the date & time of server nodes with kerberos server. To change the current date and time use date command followed by the current date and time. For example, enter

`date 06101130` to set the date to June 10th and time to 11:30 AM. The time difference between the systems should not be more than 5 minutes.

2. Add a principal for all the NFS client to the Kerberos database. For example, if our NFS client is `onc36.ind.hp.com` then `root` principal should be added to the Kerberos database before running the NFS applications.

   To add principals use the Kerberos administration tool, *kadminl*,

   ```
   onc52# /opt/krb5/admin/kadminl
   Connecting as: K/M
   Connected to krb5v01 in realm ONC52.IND.HP.COM.

   Command: add root
   Enter password:
   Re-enter password for verification:
   Enter policy name (Press enter key to apply default policy) :
   Principal added.
   ```

3. Copy the `/etc/krb5.conf` file from the Kerberos server to the NFS client.

   ```
   onc52# rcp /etc/krb5.conf onc36:/etc/
   ```

The following steps are to be configured in NFS client

1. To get the initial TGT to request a service from the application server, enter the following command:

   ```
   onc36# kinit root
   Password for root@ONC52.IND.HP.COM:
   ```

   The password prompt is displayed. Enter the password for the root principal that is added to the Kerberos database.

2. To verify the TGT, enter the following command:

   ```
   onc36# klist
   Ticket cache: FILE:/tmp/krb5cc_0
   Default principal: root@ONC52.IND.HP.COM

   Valid starting Expires Service principal
   02/12/09 10:46:33 02/12/09 20:46:31 krbtgt/ONC52.IND.HP.COM@ONC52.IND.HP.COM
   ```

3. Edit the /etc/nfssec.conf file and uncomment the entries for krb5, krb5i, or krb5p based on the security protocol you want to choose.

   ```
   onc36# cat /etc/nfssec.conf | krb5
   krb5    390003  krb5_mech     default   -               # RPCSEC_GSS
   krb5i   390004  krb5_mech     default integrity         # RPCSEC_GSS
   krb5p   390005  krb5_mech     default privacy           # RPCSEC_GSS
   ```

4. Edit the /etc/inetd.conf file and uncomment gssd entry.

   ```
   onc36# cat /etc/inetd.conf | grep gssd

   rpc  xti  ticotsord  swait  root  /usr/lib/netsvc/gss/gssd    100234  1    gssd
   ```

5. Re-initialize inetd on the NFS servers.

   ```
   onc36# inetd -c
   ```

6. To create a credential table, enter the following command:

   ```
   onc36# gsscred -m krb5_mech -a
   ```

7. To mount, secure NFS file system, enter the following command:

   ```
   mount -o sec=<Security flavor> <svr:/dir> </mount-point>
   ```

   Where,

   `-o`

   Enables you to use some of the specific options of the **share** command, such as `sec`, `async`, `public`, and others.

`sec`

Enables you to specify the security mode to be used. Specify `krb5`, `krb5p` or `krb5i` as the Security flavor.

`<svr:/`**`dir`**`>`

Enables you to specify the location of the directory.

`<`**`mount-point`**`>`

Enables you to specify the mount-point location where the filesystem is mounted.

An initial ticket grant is carried out when the user accesses the mounted filesystem.

Example

```
onc36# mount -F nfs -o sec=krb5 onc36:/export_krb5 /aaa
```

1. To verify that your file system is mounted, enter the following command:

```
onc36# nfsstat -m
/aaa from onc52:/export_krb51
Flags: vers=3,proto=tcp,sec=krb5,hard,intr,link,symlink,acl,devs,rsize=65536,wsize=65536,retrans=5,timeo=600
Attr cache: acregmin=3,acregmax=60,acdirmin=30,acdirmax=60
```

2. If everything is correct, 'll' command on the mount point:

```
onc36# ll /aaa
total 0
drwxrwxrwx 2 nobody sys 96 Feb 11 17:04 arul
drwxrwxrwx 2 nobody sys 96 Feb 11 17:11 congrats
```

If there is any problem,

```
onc36# ll /aaa
bad directory
onc36# cd /aaa
lsh: /aaa: Permission denied.
```

Accessing Shared NFS Directories across a Firewall

To access shared NFS directories across a firewall, you must configure the firewall based on the ports that the NFS service daemons listen on. To access NFS directories, the following daemons are required: `rpcbind`, `nfsd`, `rpc.lockd`, `rpc.statd`, and `rpc.mountd`. The `rpcbind` daemon uses a fixed port, 111, and the `nfsd` daemon uses 2049 as its default port. To configure the firewall, you must know the port numbers of the other NFS daemons, to ensure that the NFS client requests are not denied.

NOTE: This section does not document how to configure a firewall. This section documents the considerations to keep in mind while sharing a directory across a firewall.

Shared NFS directories can be accessed across a firewall in the following ways:

- Sharing directories across a firewall without fixed port numbers
- Sharing directories across a firewall using fixed port numbers in the `/etc/default/nfs` file
- Sharing directories across a firewall using the NFSv4 protocol
- Sharing directories across a firewall using the WebNFS feature

Sharing directories across a firewall without fixed port numbers (NFSv2 and NFSv3)

This is the default method of sharing directories across a firewall. In this method, the `rpc.statd and rpc.mountd` daemons do not run on fixed ports. The ports used by these daemons are assigned from the anonymous port range. By default, the anonymous port range is configured between 49152 and 65535.

The `rpc.lockd` daemon runs at port 4045 and is not configurable. To determine the port numbers currently used by `rpc.statd` and `rpc.mountd` daemons, run the `rpcinfo -p` command, and configure the firewall accordingly.

For example, to determine the port numbers, enter the following command:

```
rpcinfo -p
```

An output similar to the following output is displayed:

```
program vers proto   port  service
100024    1   udp  49157  status
100024    1   tcp  49152  status
100021    2   tcp   4045  nlockmgr
100021    3   udp   4045  nlockmgr
100005    3   udp  49417  mountd
100005    3   tcp  49259  mountd
100003    2   udp   2049  nfs
100003    3   tcp   2049  nfs
```

Each time the `rpc.statd` and `rpc.mountd` daemons are stopped and restarted they may be assigned a different port from the anonymous port range. The firewall must be reconfigured each time the NFS service is restarted.

For example, if the NFS service or the `rpc.statd` and `rpc.mountd` daemons are restarted, run the `rpcinfo -p` command to view the new port numbers.

An output similar to the following output is displayed:

```
program vers proto   port  service
100024    1   tcp  49154  status
100024    1   udp  49161  status
100021    3   tcp  49156  nlockmgr
100021    3   udp  49163  nlockmgr
100005    3   udp  49181  mountd
100005    3   tcp  49181  mountd
100003    3   udp   2049  nfs
100003    3   tcp   2049  nfs
```

Configure the firewall based on the new port numbers.

Sharing directories across a firewall using fixed port numbers in the nfs file

Using the `/etc/default/nfs` file enables you to specify fixed port numbers for the `rpc.statd` and `rpc.mountd` daemons. The `rpc.lockd` daemon runs at port 4045 and is not configurable.

To set the port numbers using the `/etc/default/nfs` file, follow these steps:

1. Assign values to the variables, *STATD_PORT* and *MOUNT_PORT*, as follows:

   ```
   STATD_PORT = port_number
   MOUNTD_PORT = port_number
   ```

   Where:

port_number	The port number on which the daemon runs. It can be set to any unique value between 1024 and 65536.
STATD_PORT	The port on which the `rpc.statd` daemon runs.
MOUNTD_PORT	The port on which the `rpc.mountd` daemon runs.

2. Activate the changes made to the`/etc/default/nfs` file by restarting the lock manager and NFS server daemons as follows:

   ```
   /sbin/init.d/nfs.server stop
   /sbin/init.d/lockmgr stop

   /sbin/init.d/lockmgr start
   /sbin/init.d/nfs.server start
   ```

3. Configure the firewall based on the port numbers configured.

Sharing directories across a firewall using the NFSv4 protocol

NFSv4 is a single protocol that handles mounting, and locking operations for NFS clients and servers. The NFSv4 protocol runs on port 2049, by default.

To override the default port number (2049) for the NFSv4 protocol, modify the port number for the `nfsd` entry in the `/etc/services` file.

Configure the firewall based on the port number set.

Sharing directories across a firewall using the WebNFS Feature

The WebNFS service makes files in a directory available to clients using a public file handle. The ability to use this predefined file handle reduces network traffic, by avoiding the MOUNT protocol.

How WebNFS works

This section compares the process of communication between an NFS client and an NFS server across LANs and WANs. Table 2-4 compares the NFS session across a LAN with a WebNFS session across a WAN.

Table 2-4 NFS Session Versus WebNFS Session

How NFS works across LANs	How WebNFS works across WANs
NFS servers must register their port assignments with the portmapper service that is registered on port 111, although the NFS server uses 2049 as the destination port.	NFS servers register on port 2049. WebNFS clients contact the WebNFS server on port 2049.
The MOUNT service is not registered on a specific port. The NFS client must use the `portmapper` service to locate the MOUNT port. Once the port is located, the client must issue a request for a file handle corresponding to the requested path.	A WebNFS client can use the PUBLIC file handle as an initial file handle, rather than using the MOUNT protocol.

Figure 2-2 shows a sample WebNFS session.

Figure 2-2 WebNFS Session

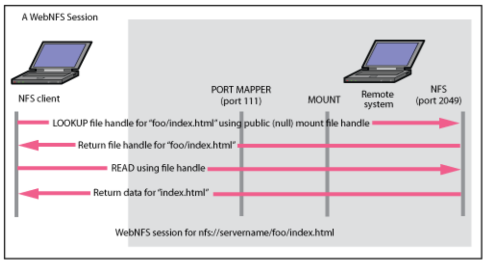

A WebNFS Session
NFS client
PORT MAPPER (port 111)
MOUNT
Remote system
NFS (port 2049)
LOOKUP file handle for "foo/index.html" using public (null) mount file handle
Return file handle for "foo/index.html"
READ using file handle
Return data for "index.html"
WebNFS session for nfs://servername/foo/index.html

Figure 2-2 depicts the following steps:

1. An NFS client uses a LOOKUP request with a PUBLIC file handle to access the `foo/index.html` file. The NFS client bypasses the portmapper service and contacts the server on port 2049 (the default port).
2. The NFS server responds with the file handle for the `foo/index.html` file.
3. The NFS client sends a READ request to the server.
4. The NFS server responds with the data.

Removing the additional overhead of the PORTMAP and MOUNT protocols reduces the binding time between the client and the server. The WebNFS protocol reduces the number of over-the-wire requests and makes traversing firewalls easier.

WebNFS offers no support for locking files across mounted filesystems. Hence, multiple clients cannot synchronize their locking calls across WebNFS mounted filesystems.

To access the shared directory across a firewall using the WebNFS feature, configure the firewall to allow connections to the port number used by the `nfsd` daemon. By default the `nfsd` daemon uses port 2049.

Configure the firewall based on the port number configured.

Configuring an NFS Server for use by a PC NFS client

PC NFS is a protocol designed to perform the following functions:

- Allow PC users who do not have UNIX style credentials to authenticate to a UNIX account
- Perform print spooling from a PC on to a UNIX server

Once a PC client has successfully authenticated itself on the NFS server, the PC uses the MOUNT and NFS protocols to mount the filesystem and to read and write to a file.

You may want to create the `/etc/pcnfsd.conf` file for the following reasons:

- If the PC NFS client software assigns user IDs smaller than 101 or greater than 60002, you can set the `uidrange` in the `/etc/pcnfsd.conf` file to allow access to a different range of user IDs, as in the following example:

  ```
  uidrange 80-60005
  ```

- You want to provide PC users a different set of default print options. For example, add an entry to the `/etc/pcnfsd.conf` file which defines `raw` as a default print option for PC users submitting jobs to the printer `lj3_2` as follows:

  ```
  printer lj3_2 lj3_2 lp -dlj3_2 -oraw
  ```

The `/etc/pcnfsd.conf` file is read when the `pcnfsd` daemon starts. If you make any changes to `/etc/pcnfsd.conf` file while `pcnfsd` is running, you must restart `pcnfsd` before the changes take effect.

A PC must have NFS client software installed in order to use the system as a PC NFS server.

The `PCNFS_SERVER` variable, configured using the `/etc/rc.config.d/nfsconf` file controls whether the `pcnfsd` daemon is started at system boot time. To configure the server to automatically start `pcnfsd` during system boot, follow these steps:

1. In the `/etc/rc.config.d/nfsconf` file, use a text editor to set the `PCNFS_SERVER` variable to 1, as follows:

   ```
   PCNFS_SERVER=1
   ```

2. To forcibly start the `pcnfsd` daemon while the server is running, run the following command:

   ```
   /sbin/init.d/nfs.server start
   ```

For more information on `pcnfsd`, see *pcnfsd*(1M).

Unsharing (Removing) a Shared Directory

NOTE: Before you `unshare` a directory, run the `showmount -a` command to verify whether any clients are accessing the shared directory. If users are accessing the shared directories, they must exit the directories before you `unshare` the directory.

A directory that is shared can be unshared. You can temporarily unshare a directory using the `unshare` command. If you want to remove a directory from being automatically shared at server restart or system reboot, remove it from the `/etc/dfs/dfstab` file.

NOTE: To unshare all the directories without restarting the server, use the `unshareall` command. This command reads the entries in the `/etc/dfs/dfstab` file and unshares all the shared directories. Use the `share` command to verify whether all the directories are unshared.

To unshare a shared directory and to prevent it from being automatically shared, follow these steps:

Automatic Unshare

1. Use a text editor to comment or remove the entries in the `/etc/dfs/dfstab` file for each directory that you want to unshare. Users on clients cannot mount the unshared directory after the server is rebooted.
2. To verify whether all the directories are unshared, enter the following command:

   ```
   share
   ```

   The directory that you have unshared should not be present in the list displayed.

Manual Unshare

1. To remove a directory from the server's internal list of shared directories, enter the following command:

   ```
   unshare directoryname
   ```

2. To verify whether all the directories are unshared, enter the following command:

   ```
   share
   ```

   The directory that you have unshared should not be present in the list displayed.

Disabling the NFS Server

To disable the NFS server, follow these steps:

1. On the NFS server, enter the following command to unshare all the shared directories:

   ```
   unshareall -F nfs
   ```

2. Enter the following command to disable NFS server capability:

   ```
   /sbin/init.d/nfs.server stop
   ```

3. On the NFS server, edit the `/etc/rc.config.d/nfsconf` file to set the `NFS_SERVER` variable to 0, as follows:

   ```
   NFS_SERVER=0
   This prevents the NFS server daemons from starting when the system reboots.
   ```

For more information about forced unmount, unmounting and unsharing, see mount_nfs (1M), *unshare*(1M), and *umount*(1M).

Configuring and Administering NFS Clients

An NFS client is a system that mounts remote directories using NFS. When a client mounts a remote filesystem, it does not make a copy of the filesystem. The mounting process uses a series of remote procedure calls that enable the client to transparently access the filesystem on the server's disk. To users, these mounted remote directories appear as if they are a part of the local filesystem. An NFS client can also be an NFS server. NFS filesystems can also be automounted using AutoFS. For information on how to automount a filesystem, see Chapter 3: "Configuring and Administering AutoFS" (page 49).

NFS Client Configuration Files and Daemons

This section describes the NFS client configuration files and daemons.

Configuration Files

Table 2-5 describes the NFS configuration files and their functions.

Table 2-5 NFS client configuration files

File Name	Function
`/etc/mnttab`	Contains the list of filesystems that are currently mounted.
`/etc/dfs/fstypes`	Contains the default distributed filesystem type.
`/etc/fstab`	Contains the list of filesystems that are automatically mounted at system boot time.

Daemons

Table 2-6 describes the NFS client daemons and their functions.

Table 2-6 NFS client daemons

Daemon Name	Function
`rpc.lockd`	Supports record lock and share lock operations on the NFS files.
`rpc.statd`	Maintains a list of clients that have performed the file locking operation over NFS against the server. These clients are monitored and notified in the event of a system crash.

Following are the tasks involved in configuring and administering an NFS client.

- Configuring the NFSv4 Client Protocol Version (Optional)
- "Deciding Between Standard-Mounted Directories and Automounted Directories" (Required)
- "Enabling an NFS Client " (Required)
- "Mounting Remote Directories " (Required)
- "Changing the Default Mount Options" (Optional)
- "Unmounting (Removing) a Mounted Directory" (Optional)
- "Disabling NFS Client Capability" (Optional)

Configuring the NFSv4 Client Protocol Version

IMPORTANT: The `nfsmapid` daemon must be running on both the NFS server and the client to use NFSv4. For more information on how to configure the NFSv4 server protocol version, see "Configuring the NFSv4 Server Protocol Version " (page 22).

By default, the version of the NFS protocol used between the client and the server is the highest one available on both systems. On HP-UX 11i v3, the default maximum protocol version of the NFS server and client is 3. The default minimum protocol version of the NFS server and client is 2.

To configure the NFS client to enable it to mount filesystems using protocol version 4 (NFSv4), follow this step:

Set the value of the *NFS_CLIENT_VERSMAX* variable to 4 in the `/etc/default/nfs` file, as follows:

```
NFS_CLIENT_VERSMAX = 4
```

The *NFS_CLIENT_VERSMAX* variable specifies the maximum version of the NFS protocol for communication.

You can also configure the client protocol version to NFSv4 by specifying `vers`=4 while mounting the directory. For example, to set the client protocol version to NFSv4 while mounting the `/usr/kc` directory, enter the following command:

```
mount -o vers=4 serv:/usr/kc /usr/kc
```

For more information on NFSv4, see *nfsd*(1m), *mount_nfs*(1m), *nfsmapid*(1m), and *nfs4cbd*(1m).

Deciding Between Standard-Mounted Directories and Automounted Directories

Before you mount any remote directories on a local system, decide whether you want each directory to be standard-mounted or automounted. You can automount directories using AutoFS. For more information on AutoFS, see Chapter 3: "Configuring and Administering AutoFS" (page 49).

Table 2-7 lists the differences between the Standard-Mounted and the Automounted directories.

Table 2-7 Standard-Mounted Versus Automounted Directories

Standard-Mounted Directory	Automounted Directory (Using AutoFS)
The directory stays mounted. You do not have to wait for it to be mounted after you issue a read or write request.	Automounted directories stay mounted until they are left idle for 10 minutes. The 10 minute time interval is the default value and is configurable.
If a directory is configured to be standard-mounted when your system boots, and the NFS server for the directory is not booted yet, system startup is delayed until the NFS server becomes available. If your system and the server are configured to mount directories from each other at boot time, standard mounts can cause both systems to hang indefinitely.	A directory automounted with AutoFS is not mounted until a user or a process requests access to it. As a result, both your system and the NFS server are able to complete the boot process before attempting to mount the directory.
You must maintain the configuration file for standard mounts (`/etc/fstab`) separately on each NFS client.	You can manage AutoFS configuration files (maps) centrally through NIS and LDAP.
If you have to configure many similar standard mounts, you must configure them individually, because you cannot use wildcard characters or environment variables when you configure standard NFS mounts.	AutoFS allows you to use wildcard characters and environment variables in configuration files (maps) as shortcuts, when you are configuring many similar automounts.
Standard NFS mounts provide no shortcut for configuring all available remote directories. You must configure each directory separately. If the NFS servers change the directories they are exporting, you must change your local NFS client configuration.	AutoFS allows you to configure a special "built-in" map known as the `-hosts` map. This built-in map automounts all the exported directories from any NFS server on the network on your system whenever anyone requests access to a directory on that server. The servers can change which directories they export, but your configuration remains valid.

Enabling an NFS Client

To enable an NFS client, follow these steps:

1. In the `/etc/rc.config.d/nfsconf` file, set the value of *NFS_CLIENT* variable to 1, as follows:

   ```
   NFS_CLIENT=1
   ```

2. Enter the following command to run the NFS client startup script:

   ```
   /sbin/init.d/nfs.client start
   ```

The NFS client startup script starts the necessary NFS client daemons, and mounts the remote directories configured in the `/etc/fstab` file.

Mounting Remote Directories

The `mount` command mounts a shared NFS directory from a remote system (NFS server).

You can mount a filesystem using the following methods:

- Automatic Mounting at System Boot time

  To set up a filesystem for automatic mounting at system boot time, you must configure it in the `/etc/fstab` file. All filesystems specified in the `/etc/fstab` file are mounted during system reboot.

- Manual Mounting

  When you manually mount a filesystem, it is not persistent across reboots or when NFS client restarts. If the NFS client is restarted or the system is rebooted, the filesystem must be mounted again after the reboot. To mount filesystems manually, you must run the `mount` command. For a list of mount options, see *mount_nfs*(1M).

Consider the following points before you mount a directory:

- Before you mount a remote directory on your system, you must configure the remote system as an NFS server and share the directory.
- You must configure a local directory as the mount-point for the NFS filesystem. HP recommends that the mount-point not contain files and directories. However, if the local directory contains files and directories, they will be hidden and inaccessible while the remote directory is mounted.

Mounting a Remote Directory on an NFS client

To mount a directory on an NFS client, select one of the following methods:

Automatic Mount

To configure a remote directory to be automatically mounted at system boot, follow these steps:

1. Add an entry to the `/etc/fstab` file, for each remote directory you want to mount on your system. Following is the syntax for the entry in the `/etc/fstab` file:

   ```
   server:remote_directory local_directory nfs defaults 0 0
   ```

   or

   ```
   server:remote_directory local_directory nfs option[,option...] 0 0
   ```

2. Mount all the NFS file systems specified in the `/etc/fstab` file by entering the following command:

   ```
   /usr/sbin/mount -a -F nfs
   ```

3. Verify that your filesystem is mounted by entering the following command:

   ```
   nfsstat -m
   ```

   An output similar to the following output is displayed:

   ```
   /mnt/nfs149 from nfs149:/
   Flags:
   vers=4,proto=tcp,sec=sys,hard,intr,link,symlink,devs,rsize=
   32768,wsize=32768,retrans=5,timeo=600
   Attr cache:acregmin=3,acregmax=60,acdirmin=30,acdirmax=60
   ```

   The directory that you have mounted must be present in this list.

Manual Mount

To mount your directories manually, follow these steps:

1. To mount a remote directory manually, enter the following command:

   ```
   mount serv:directory_name directory-name
   ```

2. Verify if your filesystem is mounted by entering the following command:

   ```
   nfsstat -m
   ```

An output similar to the following output is displayed:

```
/opt/nfstest from hpnfsweb:/home/tester
Flags:
vers=3,proto=udp,sec=sys,hard,intr,link,symlink,acl,devs,
rsize=32768,wsize=32768,retrans=5,timeo=11
Attr cache:
acregmin=3,acregmax=60,acdirmin=30,acdirmax=60
Lookups:     srtt=33 (82ms), dev=33 (165ms), cur=20 (400ms)
```

The directory that you have mounted must be present in this list.

For a list of mount options, see *mount_nfs*(1M).

Enabling Client-Side Failover

Using client-side failover, an NFS client can switch to another server if the server supporting a replicated filesystem becomes unavailable.

The failover is usually transparent to the user. Failover can occur at any time without disrupting the processes running on the client. Failover is only supported on read-only filesystems.

To mount a directory that has been replicated on multiple servers, enter the following command:

```
mount -o ro svr:dir_name,srv:dir_name dir_name
```

If the first NFS server is down, the client accesses the second NFS server.

For example, to mount the `Casey` directory to replicated servers, enter the following command:

```
mount -o ro onc21:/home/Casey, onc23:/home/Casey /Clay
```

If the NFS server `onc21` is down, the client accesses NFS server `onc23`.

To verify if the failover is in effect, enter the following command:

```
nfsstat -m
```

An output similar to the following output is displayed:

```
/Clay from onc21:/home/Casey,onc23:/home/Casey
Flags:
vers=3,proto=tcp,sec=sys,hard,intr,llock,link,symlink,acl,
devs,rsize=32768,wsize=32768,retrans=5,timeo=600
Attr cache: acregmin=3,acregmax=60,acdirmin=30,acdirmax=60
Failover: noresponse=0,failover=0,remap=0,currserver=onc23
```

The Failover line in the above output indicates that the failover is working.

Examples of NFS Mounts

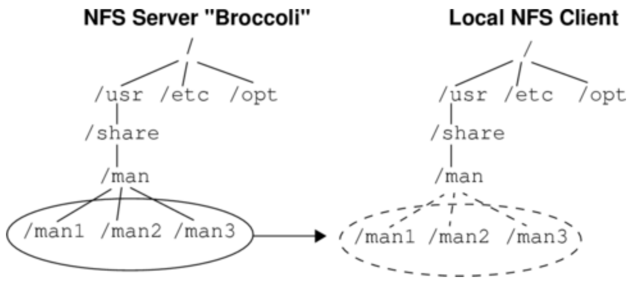
- Mounting a directory as read-only with no set userid privileges

  ```
  mount -r -o nosuid broccoli:/usr/share/man /usr/share/man
  ```

  In this example, the NFS clients mount the `/usr/share/man` directory from the NFS server `broccoli`. The local mount-point is also `/usr/share/man`. The directory is mounted as read-only.

  Figure 2-3 illustrates this example.

Figure 2-3 NFS Mount of manpages



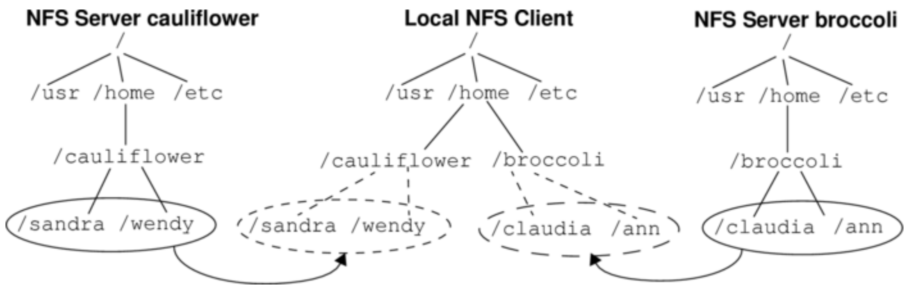
- Mounting a Home directory

```
mount -r -o nosuid broccoli:/home/broccoli /home/broccoli
mount -r -o nosuid cauliflower:/home/cauliflower /home/cauliflower
```

  In this example, the NFS client mounts the home directories from NFS servers `broccoli` and `cauliflower` . The `nosuid` option prevents programs with setuid permission from executing on the local client.

  Figure 2-4 illustrates this example.

Figure 2-4 NFS Mount of Home Directories



- Mounting an NFS Version 2 filesystem using the UDP Transport

```
mount -o vers=2,proto=udp onc21:/var/mail /var/mail
```

  In this example, the NFS client mounts the `/var/mail` directory from the NFS server, `onc21`, using NFSv2 and the UDP protocol.

- Mounting an NFS filesystem using an NFS URL

```
mount nfs://onc31/Casey/mail /Casey/mail
```

  In this example, the NFS client mounts the `/Casey/mail` directory from NFS server, `onc31`, using the WebNFS protocol.

- Mounting an NFS filesystem by using a public file handle

```
mount -o public nfs://onc31/usr/%A0abc /Casey/Clay
```

  If the `public` option or a URL is specified, the mount command attempts to connect to the server using the public file handle. The daemons `rpcbind` and `mountd` are not contacted.

  In this example, the NFS client mounts `/Casey/Clay` directory by using a public file handle, and an NFS URL that has a non 7-bit ASCII escape sequence from the NFS server, `onc31`.

- Mounting an NFS filesystem using native path

```
mount -o public onc31:C:Casey:test1 /Casey/test
```

  A native path is a pathname that is interpreted according to conventions used on the native operating system of the NFS server.

  In this example, the NFS client mounts the `/Casey/test` directory using a native path and a public file handle.

- Mounting a replicated set of NFS filesystems with same pathnames

```
mount -r onc21,onc23,onc25:/Casey/Clay /Casey/Clay
```

In this example, the NFS client mounts a single filesystem, `/Casey/Clay` that has been replicated to a number of servers with the same pathnames. This enables the NFS client to failover to either server `onc21,` `onc23,` or `onc25` if the current server has become unavailable.

- Mounting replicated set of NFS file systems with different pathnames

  ```
  mount -r onc21:/Casey/Clay,onc23:/Var/Clay,nfs://srv-z/Clay /Casey/Clay
  ```

  In this example, the NFS client mounts a replicated set of NFS file systems with different pathnames.

Secure Mounting of Directories

The `mount` command enables you to specify the security mode for each NFS mount-point. This allows the NFS client to request a specific security mode. However, if the specific mode does not exist on the server, then the mount fails. Use the `sec` option to specify the security mode.

If `sec` is not specified in NFSv2, then the AUTH_SYS mode is used by default. If `sec` is not specified in NFSv3, then the default action is for the NFS client to query the server about the security mode to be used. The resulting security mode is negotiated between the NFS client and server. For more information on the available security modes, see *nfssec*(5).

An Example for Securely Mounting a directory

In the following example, the NFS client forces the use of `krb5` as the secure mode. The `krb5` secure mode uses the Kerberos V5 protocol to authenticate users before granting access to the shared filesystems. In this example, the NFS server, `onc21`, also supports the same secure mode.

```
mount -F nfs -o sec=krb5 onc21:/casey/nfs /casey/nfs
```

NOTE: For specific configuration information, see "Secure NFS Setup with Kerberos" (page 27).

Changing the Default Mount Options

To change the default mount options, follow these steps:

1. Modify the NFS mount options in the `/etc/fstab` file, or the AutoFS map, as needed. For more information on the different mount options that NFS supports, see *mount_nfs*(1M).

   If you changed the mount options for a directory that is currently mounted, you must unmount and remount it before the changes take effect. To unmount and remount the directory, enter the following commands:

   ```
   /usr/sbin/umount local_directory
   /usr/sbin/mount local_directory
   ```

2. If you change the mount options in the AutoFS master map, you must restart AutoFS for the changes to take effect. For information on restarting AutoFS, see "Restarting AutoFS" (page 76).

For more information on the different caching mount options, see *mount_nfs*(1M).

Unmounting (Removing) a Mounted Directory

You can temporarily unmount a directory using the `umount` command. If you want to stop a directory from being automatically mounted upon server restart or system reboot, remove it from the `/etc/fstab` file.

NOTE: Before you unmount a directory, run the `fuser -cu` command to determine whether the directory is currently in use. The `fuser` command lists the process IDs and user names of all the processes that are using the mounted directory. If users are accessing the mounted directories, they must exit the directories before you unmount the directory.

To unmount a mounted directory and prevent it from being automatically mounted, follow these steps:

Automatic Unmount

1. To check for any processes accessing the mounted directory, enter the following command:
   ```
   fuser -cu mounted_directory
   ```
2. To kill the processes, enter the following command:
   ```
   fuser -ck local_mount_point
   ```
3. Unmount the filesystem as follows:
   ```
   /usr/sbin/umount mounted_directory
   ```
4. If the unmount fails, use the forcible option to unmount the filesystem:
   ```
   /usr/sbin/umount -f mounted_directory
   ```
5. Verify whether all the directories are unmounted by entering the following command:
   ```
   mount
   ```
   The directories that you have unmounted must not be present in the list displayed.
6. If you do not want the directory to be automatically mounted when the system is rebooted, remove the directory entry from the `/etc/fstab` file.

Disabling NFS Client Capability

To disable the NFS client, follow these steps:

1. On the NFS client, enter the following command to get a list of all the mounted NFS filesystems on the client
   ```
   /usr/sbin/nfsstat -m
   ```
2. For every NFS mounted directory listed by the `nfsstat` command, enter the following command to determine whether the directory is currently in use:
   ```
   /usr/sbin/fuser -cu local_mount_point
   ```
   This command lists the process IDs and user names of all processes currently using the mounted directory.
3. To kill all processes that are using the mounted directory, enter the following command:
   ```
   /usr/sbin/fuser -ck local_mount_point
   ```
4. To unmount all NFS mounted directories, enter the following command:
   ```
   /usr/sbin/umount -aF nfs
   ```
5. To disable the NFS client and AutoFS, edit the `/etc/rc.config.d/nfsconf` file on the client to set the `NFS_CLIENT` and `AUTOFS` variables to 0, as follows:
   ```
   NFS_CLIENT=0
   AUTOFS=0
   ```
   This prevents the client processes from starting up again when you reboot the client.
6. Enter the following command to disable NFS client capability:
   ```
   /sbin/init.d/nfs.client stop
   ```
   For more information, see *umount (1M)*, *mount*(1M), and *fuser*(1M).

NFS Client and Server Transport Connections

NFS runs over both UDP and TCP transport protocols. The default transport protocol is TCP. Using the TCP protocol increases the reliability of NFS filesystems working across WANs and ensures that the packets are successfully delivered. TCP provides congestion control and error recovery. NFS over TCP and UDP works with NFS Version 2 and Version 3.

NOTE: TCP is the only transport protocol supported by NFS Version 4.

Supporting 1MB Transfer for TCP mounts

By default, NFS supports 32K transfer sizes across both TCP and UDP transports.

To enable support of 1MB transfers for TCP mounts, you must first modify the following tunables:

- `nfs3_bsize` (for NFS version 3)

  This tunable controls the logical block size used by NFSv3 clients. The block size represents the amount of data the client attempts to read from or write to the server during an I/O operation.
- `nfs4_bsize` (for NFS version 4)

  This tunable controls the logical block size used by NFSv4 clients. The block size represents the amount of data the client attempts to read from or write to the server during an I/O operation.
- `nfs3_max_transfer_size` (for NFS version 3)

  This tunable specifies the maximum size of the data portion of NFSv3 READ, WRITE, READDIR, and READDIRPLUS requests. This parameter controls both the maximum size of the data that the server returns and the maximum size of the request the client generates.
- `nfs4_max_transfer_size` (for NFS version 4)

  This tunable specifies the maximum size of the data portion of NFSv4 READ, WRITE, READDIR, and READDIRPLUS requests. This parameter controls both the maximum size of the data that the server returns and the maximum size of the request the client generates.
- `nfs3_max_transfer_size_cots` (for NFS version 3)

  This tunable specifies the maximum size of the data portion of NFSv3 READ, WRITE, READDIR, and READDIRPLUS requests. This parameter controls both the maximum size of the data the server returns as well as the maximum size of the request the client generates over a connection-oriented transport, such as TCP.
- `nfs4_max_transfer_size_cots` (for NFS version 4)

  This tunable specifies the maximum size of the data portion of NFSv4 READ, WRITE, READDIR, and READDIRPLUS requests. This parameter controls both the maximum size of the data the server returns as well as the maximum size of the request the client generates over a connection-oriented transport, such as TCP.

For information on the `nfs3_bsize` and `nfs4_bsize` tunables, see *nfs3_bsize*(1M) and *nfs4_bsize*(1M).

After the tunables have been modified, set the mount option for read and write size to 1MB, as follows:

```
mount -F nfs -o rsize=1048576, wsize=1048576
```

Changes to the NFS Server Daemon

The NFS server daemon (`nfsd`) handles client filesystem requests. By default, `nfsd` starts over TCP and UDP for NFSv2 and NFSv3. If NFSv4 is enabled, the `nfsd` daemon is started to service all TCP and UDP requests.

If you want to change startup parameters for `nfsd`, you must login as superuser (`root`) and make changes to the `/etc/default/nfs` file or use the `setoncenv` command.

The `/etc/default/nfs` file provides startup parameters for the `nfsd` daemon and `rpc.lockd` daemon. For more information on the `/etc/default/nfs` file, see *nfs*(1M).

The `setoncenv` command initializes, displays, and removes the value of NFS configuration variables found in either of the following files:

- `/etc/rc.config.d/nfsconf`
- `/etc/rc.config.d/namesvrs`
- `/etc/default/autofs`
- `/etc/default/keyserv`
- `/etc/default/nfs`
- `/etc/default/nfslogd`
- `/etc/pcnfsd.conf`

For more information on the `setoncenv` command, see *setoncenv*(1M).

Configuring and Using NFS Netgroups

This section describes how to create and use NFS netgroups to restrict NFS access to your system. It describes the following tasks:

- "Creating Netgroups in the `/etc/netgroup` File"
- "Using Netgroups in Configuration Files"

Creating Netgroups in the `/etc/netgroup` File

To create netgroups in the `/etc/netgroup` file, follow these steps:

1. If you are using the local `/etc/netgroup` file or the NIS `netgroup` map for netgroups, add entries with the following syntax to the `/etc/netgroup` file.

   ```
   netgroup_name (host, user, NIS_domain) (host, user, NIS_domain) ...

   If you are using NIS, edit the /etc/netgroup file only on the NIS master server.
   ```

2. If you are using NIS to manage your `netgroups` database, enter the following commands on the NIS master server to generate the `netgroup`, `netgroup.byhost`, and `netgroup.byuser` maps from the `/etc/netgroup` file, and push the generated maps to the NIS slave servers:

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

A netgroup can be used in most NFS and NIS configuration files, instead of a host name or a user name. A netgroup does not create a relationship between users and hosts. When a netgroup is used in a configuration file, it represents either a group of hosts or a group of users, but never both.

If you are using BIND (DNS) for hostname resolution, hosts must be specified as fully qualified domain names, for example: `turtle.bio.nmt.edu`.

If *`host`*, *`user`*, or *`NIS_domain`* is left blank in a netgroup, that field can take any value. If a dash (-) is specified in any field of a netgroup, that field can take no value.

The *`NIS_domain`* field specifies the NIS domain in which the triple (*`host`*, *`user`*, *`NIS_domain`*) is valid. For example, if the `netgroup` database contains the following netgroup:

```
myfriends (sage,-,bldg1) (cauliflower,-,bldg2) (pear,-,bldg3)
```

and an NFS server running NIS in the domain `bldg1` shares a directory only to the netgroup `myfriends`, only the host `sage` can mount that directory. The other two triples are ignored, because they are not valid in the `bldg1` domain.

If an HP-UX host not running NIS exports or shares a directory to the netgroup `myfriends`, the *`NIS_domain`* field is ignored, and all three hosts (`sage`, `cauliflower`, and `pear`) can mount the directory.

If the `netgroup` database contains the following netgroup,

```
mydomain (,,bldg1)
```

and a host in the NIS domain `bldg1` shares a directory to the netgroup `mydomain`, any host *in the bldg1 domain* may mount the directory, because the *`host`* field is blank.

If an HP-UX host not running NIS shares a directory to the netgroup `mydomain`, in this case, the *`NIS_domain`* field is ignored but the *`host`* field is used. As a result, any host *in any domain* can mount the directory.

If a host in the NIS domain `bldg2` shares a directory to the netgroup `mydomain`, no host in any domain can mount the directory, because the triple is not valid in the `bldg2` domain. As a result, it is ignored.

Netgroup Examples

The following netgroup specifies a group of hosts:

```
trusted_hosts (sage, , ) (basil, , ) (thyme, , )
```

The `trusted_hosts` netgroup can be used in the `access_list` argument of an entry in the `/etc/dfs/dfstab` file, as follows:

```
/usr [access_list]=trusted_hosts
```

The following netgroup specifies a group of users:

```
administrators ( ,jane, ) ( ,art, ) ( ,mel, )
```

If this netgroup is accidentally included in a list of hosts rather than users, the blank space is interpreted as a wildcard, meaning any host. For example, if someone used this netgroup in an `[access_list]` argument in the `/etc/dfs/dfstab` file, any host can access the shared directory. If a netgroup is used strictly as a list of users, it is better to put a dash in the host field, as follows:

```
administrators (-,jane, ) (-,art, ) (-,mel, )
```

The dash indicates that no hosts are included in the netgroup.

The `trusted_hosts` and `administrators` netgroups can be used together in the `/etc/hosts.equiv` file, as follows:

```
+@trusted_hosts  +@administrators
```

The first netgroup is read for host names, and the second is read for user names. Users in the `administrators` netgroup can log in to the local host from any host in the `trusted_hosts` netgroup without supplying a password.

The two netgroups can be combined into one, as follows:

```
goodguys (sage,jane, ) (basil,art, ) (thyme,mel, )
```

If the two netgroups are combined this way, the same netgroup can be used as both the host name and the user name in the `/etc/hosts.equiv` file, as follows:

```
+@goodguys   +@goodguys
```

The first occurrence of it is read for the host name, and the second occurrence is read for the user name. No relationship exists between the host and user in any of the triples. For example, user `jane` may not even have an account on host `sage`.

A netgroup can contain other netgroups, as in the following example:

```
root-users (dill,-,  ) (sage,-, ) (thyme,- , ) (basil,-, )
mail-users (rosemary, , ) (oregano, , ) root-users
```

The root-users netgroup is a group of four systems. The mail-users netgroup uses the root-users netgroup as part of a larger group of systems. The blank space in the third field of each triple indicates that the netgroup is valid in any NIS domain.

Using Netgroups in Configuration Files

Netgroups may be used in the following files:

- /etc/dfs/dfstab, in the [access_list], -rw, -ro, and root list
- /etc/hosts.equiv or $HOME/.rhosts, in place of a host name or user name
- /etc/passwd, to instruct processes whether to look in the NIS password database, for information about the users in the netgroup
- /etc/group, to instruct processes whether to look in the NIS group database, for information about the users in the netgroup

The following sections explain how to use netgroups in configuration files.

Using Netgroups in the /etc/dfs/dfstab File

In the /etc/dfs/dfstab file, netgroups can be used in the list of NFS clients following the [access_list], -rw, -ro, or root option, as in the following example:

```
[access_list]=mail_clients
```

The mail_clients netgroup is defined, as follows:

```
mail_clients (cauliflower, , ) (broccoli, , ) (cabbage, , )
```

Only the host names from the netgroup are used. If the netgroup also contains user names, these are ignored. This netgroup is valid in any NIS domain, because the third field in each triple is left blank.

Using Netgroups in the /etc/hosts.equiv or $HOME/.rhosts File

In the /etc/hosts.equiv file, or in a .rhosts file in a user's home directory, netgroups can be used in either the host name field or the user name field, as in the following example:

```
+@our_friends   +@our_friends
```

The netgroup our_friends can be used both as host name and user name, because it includes both host names and user names. This can be illustrated in the following example:

```
our_friends (sage,sara, ) (sage,eric, ) (dill,-, )
( ,monica, )
```

The blank host name field in the fourth triple serves as a wildcard, allowing users from any host on the network to log in without supplying a password. However, only the users listed in the netgroup are given this privileged access, because each user name field contains either a user name or a dash.

Netgroups can also be used to deny privileged access to certain hosts or users in the /etc/hosts.equiv or $HOME/.rhosts file, as in the following example:

```
+   -@vandals
```

The plus (+) sign is a wildcard in the /etc/hosts.equiv or $HOME/.rhosts file syntax, allowing privileged access from any host in the network. The netgroup vandals is defined as follows:

```
vandals ( ,pat, ) ( ,harriet, ) ( ,reed, )
```

All users *except* those listed in the `vandals` netgroup can log in to the local system without supplying a password from any system in the network.

CAUTION: Users who are denied privileged access in the `/etc/hosts.equiv` file can be granted privileged access in a user's `$HOME/.rhosts` file. The `$HOME/.rhosts` file is read after the `/etc/hosts.equiv` file and overrides it.

For more information, see *hosts.equiv*(4).

Using Netgroups in the `/etc/passwd` File

In the `/etc/passwd` file, netgroups can be used to indicate whether user information must be looked up in the NIS `passwd` database.

The following sample entry from the `/etc/passwd` file indicates that users in the netgroup `animals` must be looked up in the NIS `passwd` database:

```
+@animals
```

The `animals` netgroup is defined in the `/etc/netgroup` file, as follows:

```
animals (-,mickey, ) (-,daffy, ) (-,porky, ) (-,bugs, )
```

The `/etc/passwd` file is searched sequentially. As a result, user `mickey, daffy, porky`, or `bugs` appear before the `animals` netgroup in the `/etc/passwd` file. The NIS database is not consulted for information on that user.

Netgroups can also be used to prevent lookups of certain users in the NIS `passwd` database. The following sample entries from the `/etc/passwd` file indicate that if the NIS `passwd` database contains entries for users in the `bears` netgroup, these entries cannot be used on the local system. Any other user can be looked up in the NIS database.

```
-@bears
```

For more information on NIS, see NIS Administrator's Guide (5991-2187).

For information on the `/etc/passwd` file, see *passwd*(4).

Using Netgroups in the `/etc/group` File

In the `/etc/group` file, netgroups can be used to indicate whether group information about certain users must be looked up in the NIS `group` database.

The following sample entry from the `/etc/group` file indicates that group information for users in the netgroup `animals` can be found in the NIS `group` database:

```
+@animals
```

The `animals` netgroup is defined in the `/etc/netgroup` file, as follows:

```
animals (-,mickey, ) (-,daffy, ) (-,porky, ) (-,bugs, )
```

Members of the `animals` netgroup can belong to groups listed in the local `/etc/group` file as well as in the NIS `group` database. The following entries in the `/etc/group` file give users `bugs` and `daffy` membership in the group `wiseguys` and in any group in the NIS database that includes them as members:

```
wiseguys::22:bugs,daffy
+@animals
```

Netgroups can also be used in the `/etc/group` file to prevent lookups for certain users. The `bears` netgroup is defined in the `/etc/netgroup` file, as follows:

```
bears (-,yogi, ) (-,smokey, ) (-,pooh, )
```

The following entries in the `/etc/group` file allow user `pooh` membership in the group `teddybears`, but not in any other group listed in the NIS database or after the `-@bears` entry in the `/etc/group` file:

```
teddybears::23:pooh,paddington
-@bears
```

For more information on NIS, see NIS Administrator's Guide (5991-2187).

For information on the `/etc/group` file, see *group*(4).

Configuring RPC-based Services

This section describes the following tasks:

- "Enabling Other RPC Services"
- "Restricting Access to RPC-based Services"

Enabling Other RPC Services

1. In the `/etc/inetd.conf` file, use a text editor to uncomment the entries that begin with `"rpc"`.

   Following is the list of entries in an `/etc/inetd.conf` file:

```
#rpc xit tcp nowait root /usr/sbin/rpc.rexd 100017 1 rpc.rexd

#rpc dgram udp wait root /usr/lib/netsvc/rstat/rpc.rstatd 100001
2-4 rpc.rstatd

#rpc xit datagram_v,circuit_v wait root /usr/lib/netsvc/rusers/rpc.rusersd 100002
 1-3 rpc.rusersd

#rpc xit datagram_v wait root /usr/lib/netsvc/rwall/rpc.rwalld 100008
1 rpc.rwalld

#rpc dgram udp wait root /usr/sbin/rpc.quotad 100011 1 rpc.rquotad

#rpc dgram udp wait root /usr/lib/netsvc/spray/rpc.sprayd 100012
1 rpc.sprayd

#rpc xti ticotsord swait root /usr/lib/netsvc/gss/gssd 100234 1 gssd
```

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

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

CAUTION: Do not issue the `/usr/sbin/inetd` command if NFS is not yet running on your system. The `nfs.core` startup script starts the rpcbind process, which *must* be running before you configure `inetd` to manage RPC-based services.

Table 2-8 lists the RPC daemons and services that can be started by the `inetd` daemon. It briefly describes each one and specifies the manpage you can refer to for more information.

Table 2-8 RPC Services managed by inetd

RPC Service	Description
`rexd`	The `rpc.rexd` program is the server for the `on` command, which starts the Remote Execution Facility (REX). The `on` command sends a command to be executed on a remote system. The `rpc.rexd` program on the remote system executes the command, simulating the environment of the user who issued the `on` command. For more information, see rexd (1M) and on (1).
`rstatd`	The `rpc.rstatd` program answers requests from the `rup` command, which collects and displays status information about the systems on the local network. For more information, see rstatd (1M) and rup (1).
`rusersd`	The `rpc.rusersd` program responds to requests from the `rusers` command, which collects and displays information about all users who are logged in to the systems on the local network. For more information, see rusersd (1M) and rusers (1).
`rwalld`	The `rpc.rwalld` program handles requests from the `rwall` program. The `rwall` program sends a message to a specified system where the `rpc.rwalld` program is running, and the message is written to all users logged in to the system. For more information, see rwalld (1M) and rwall (1M).
`sprayd`	The `rpc.sprayd` program is the server for the `spray` command, which sends a stream of packets to a specified host and then reports how many were received and how fast. For more information, see sprayd (1M) and spray (1M).

Table 2-8 RPC Services managed by inetd *(continued)*

RPC Service	Description
`rquotad`	The `rpc.rquotad` program responds to requests from the `quota` command, which displays information about a user's disk usage and limits. For more information, see rquotad (1M) and quota (1).
`gssd`	The `gssd` program operates between the Kernel RPC and the Generic Security Services Application Program Interface (GSS-API) to generate and validate the GSS-API tokens. For more information, see *gssd*(1M).

Restricting Access to RPC-based Services

To restrict access to RPC-based services, create an entry with the following syntax in the `/var/adm/inetd.sec` file for each service to which you want to restrict access:

`service` {allow} *`host_or_network`* [*`host_or_network`*...] {deny}

If the `/var/adm/inetd.sec` file does not exist, you may have to create it.

The *`service`* must match one of the service names in the `/etc/rpc` file.

Specify either `allow` or `deny`, but not both. Enter only one entry per service.

`The host_or_network` can be either an official host name, a network name, or an IP address. Any of the four numbers in an IP address can be specified as a range (for example, 1-28) or as a wildcard character (*).

The `inetd.sec` file is checked only when the service starts. If a service remains active and accepts more requests without being restarted, the `inetd.sec` file is not checked again.

You can use HP SMH to modify the `/var/adm/inetd.sec` file.

For more information, see inetd.conf (4) and inetd.sec (4).

Examples from `/var/adm/inetd.sec`

In the following example, only hosts on subnets 15.13.2.0 through 15.13.12.0 are allowed to use the `spray` command:

```
sprayd allow 15.13.2-12.0
```

In the following example, the host `cauliflower` is prevented from using the `rwall` command:

```
rwalld deny cauliflower
```

3 Configuring and Administering AutoFS

This chapter provides an overview of AutoFS and the AutoFS environment. It also describes how to configure and administer AutoFS on a system running HP-UX 11i v3.

This chapter addresses the following topics:

Overview

AutoFS is a client-side service that automatically mounts remote filesystems. During system startup, AutoFS is initialized by the `automount` command. The automount daemon, `automountd`, runs continuously and mounts and unmounts remote directories as required.

When a client running `automountd` attempts to access a remote file or a remote directory, `automountd` mounts it, if it has been configured. If the mounted remote filesystem is not accessed for a certain period of time, it is automatically unmounted.

AutoFS uses maps to navigate the network. Maps define the mount-points that AutoFS will mount. AutoFS can mount both directories and files. For more information on AutoFS maps, see "Maps Overview" (page 51).

Following sections describe the different components of AutoFS that work together to automatically mount and unmount filesystems, in detail.

AutoFS Filesystem

The AutoFS filesystem is a virtual filesystem that provides a directory structure to enable automatic mounting of filesystems. It includes `autofskd`, a kernel-based process that periodically cleans up mounts. The filesystem interacts with the `automount` command and the `automountd` daemon to mount filesystems automatically.

The `automount` Command

This command installs the AutoFS mount-points, and associates an automount map with each mount-point. The AutoFS filesystem monitors attempts to access directories within it and notifies the `automountd` daemon. The daemon locates a filesystem using the map, and then mounts this filesystem at the point of reference within the AutoFS filesystem. The automount map specifies the location of all the AutoFS mount-points.

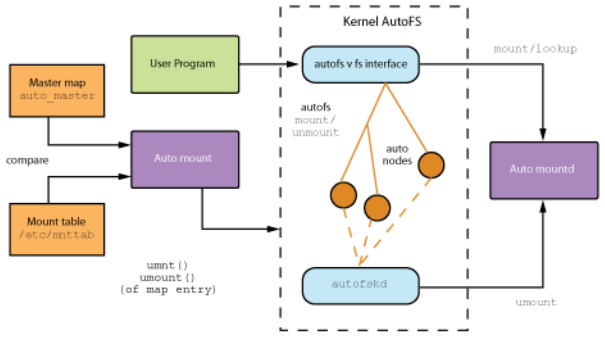
NOTE: You must run the `automount` command whenever the master map or the direct maps are updated.

The `automountd` Daemon

The `automountd` daemon is a stateless, multi-threaded daemon that mounts or unmounts directories and filesystems by accepting Remote Procedure Calls (RPC) requests.

Figure 3-1 illustrates how the AutoFS components interact.

Figure 3-1 Interaction between AutoFS Components



The `automount` command reads the AutoFS master map to create the initial set of AutoFS mount-points in the internal mount table, `/etc/mnttab`. The automounted filesystems are not mounted automatically at startup. The automounted filesystems are points under which the filesystems are mounted when the clients request access to them.

When AutoFS receives a request to mount a filesystem, it calls the `automountd` daemon, which mounts the requested filesystem. AutoFS mounts the filesystems at the configured mount-points.

The `automountd` daemon is independent of the `automount` command. This separation enables you to add, delete, or modify the AutoFS map information, without stopping and restarting the `automountd` daemon. You can modify the set of mount-points by modifying the AutoFS maps and by running the `automount` command to read them and update the mount table.

If an automounted filesystem remains idle for a specified period of time (the default is 10 minutes), AutoFS unmounts it. To change the duration, in seconds, for which a file system is to remain mounted when not in use, use the `-t` option of the `automount` command. For more information on the different options supported by `automount`, see *automount*(1M) and *automountd*(1M).

CAUTION: You must maintain filesystems managed by AutoFS, by using the `automountd` and `automount` utilities. Manually mounting and unmounting file systems managed by AutoFS can cause disruptive or unpredictable results, including but not limited to commands hanging or not returning expected results. Applications can also fail because of their dependencies on these mounted filesystems.

Maps Overview

Maps define the mount-points that AutoFS will mount. Maps are available either locally or through a distributed network Name Service, such as NIS or LDAP. AutoFS supports different types of maps.

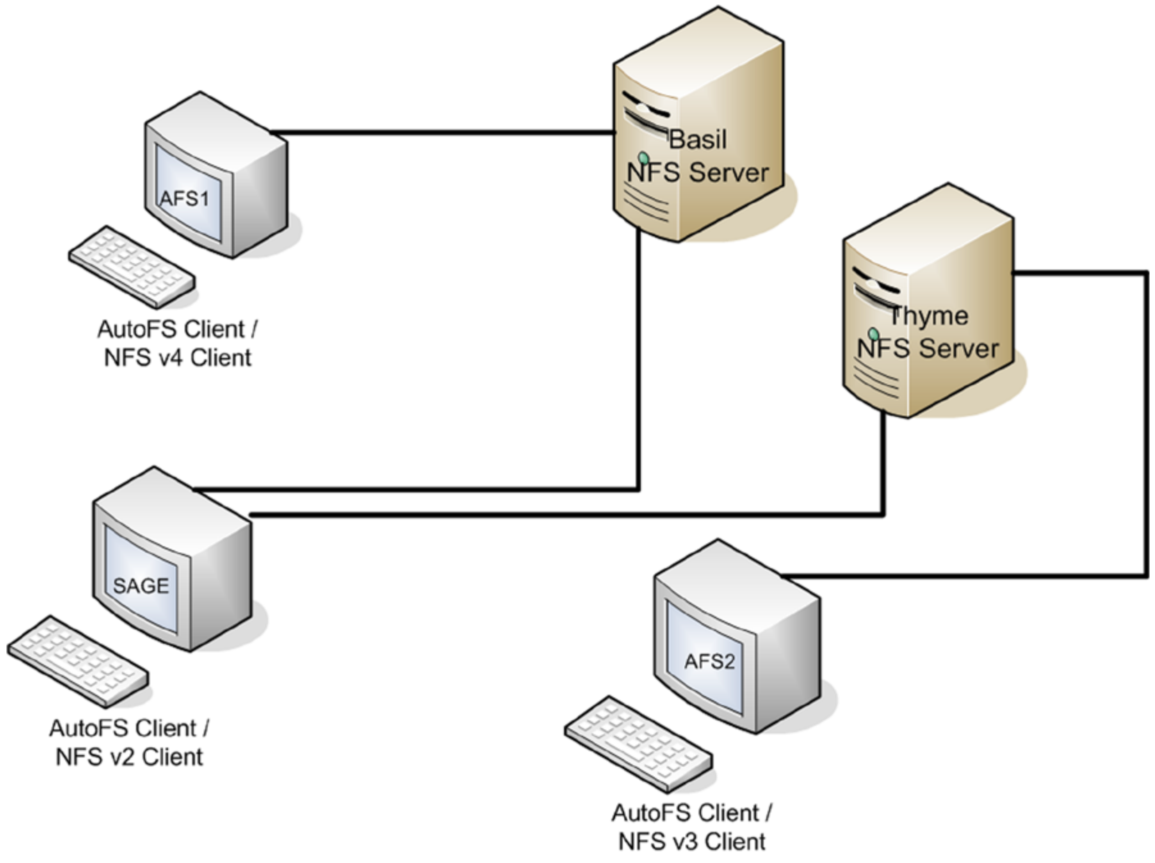
Table 3-1 lists the different types of maps supported by AutoFS.

Table 3-1 Types of AutoFS Maps

Type of Map	Description
Master Map	A master list of maps, which associates a directory with a map. The `automount` command reads the master map at system startup to create the initial set of AutoFS mounts.
Direct Map	A direct map is an entry in the AutoFS master map. In a direct map, there exists a direct association between the mount-point on a client and a directory on the server.
Indirect Map	An indirect map is an entry in the AutoFS master map. In an indirect map, there exists an association between keys representing mount-points on the client and directories on the server. A key is a representation of the first field of a map entry.
Executable Map	An executable map is a map whose entries are generated dynamically by a program or a script. Local maps that have the `execute` bit set in their file permissions will be executed by the AutoFS daemon. A direct map cannot be made executable.
Special Map	Special Maps are of two types namely, `-hosts` and `-null`.
Included Map	An included map is a map that is included within another map. The entries of the included map are read as if they are part of the existing map.

Figure 3-2 "AutoFS" displays a sample AutoFS network.

Figure 3-2 AutoFS



In this figure, `AFS1`, `AFS2`, and `Sage` are AutoFS clients. `Thyme` and `Basil` are the NFS servers. The NFS servers export directories. The AutoFS clients use maps to access the exported directories. For instance, the NFS server, `Basil`, exports the `/export` directory. If you are a user on any of the AutoFS clients, you can use maps to access the `/export` directory from the NFS server, `Basil`.

AutoFS clients can access the exported filesystem using any one of the following map types:

- Direct Map
- Indirect Map
- Special Map

The AutoFS client, `Sage`, uses a direct map to access the `/export` directory. `Sage` includes an entry similar to the following in its map:

```
/export  Basil:/export
```

`Sage` mounts the `/export` directory on the `export` mount-point.

The `AFS1` client uses an indirect map to access the `/export` directory. The `AFS1` client includes an entry similar to the following in its map:

```
/usr   /  server:/
   /export  Basil:/export
```

The `AFS1` client mounts the `/export` directory on the `/usr/export` mount-point.

The `AFS2` client uses a special map to automount the `/export` directory. The `AFS2` client includes an entry similar to the following in its map:

```
/net -host -nosuid, soft
```

The `AFS2` client mounts the `/export` directory at the `/net/Basil/export` location.

Features

This section discusses the features that AutoFS supports on systems running HP-UX 11i v3.

On-Demand Mounting

In HP-UX 11i v3, the filesystems being accessed are mounted automatically. Filesystems that are hierarchically related to the automounted filesystems are mounted only when necessary.

Consider the following scenario where the AutoFS master map, `/etc/auto_master`, and the direct map, `/etc/auto_direct`, are on the NFS client, `sage`. Following are the contents of the master map, the `/etc/auto_master` file, which contains a single direct map entry:

```
# /etc/auto_master file
# local mount-point         map name             mount options

/-                         /etc/auto_direct
```

Following are the contents of the direct map, `/etc/auto_direct`, which contains the local mount-points on the client and the directory on the server:

```
# /etc/auto_direct file
# local mount-point    mount options  remote server:directory

/auto/project/specs -nosuid thyme:/export/project/specs
/auto/project/specs/reqmnts -nosuid  thyme:/export/projects/specs/reqmnts
/auto/project/specs/design -nosuid thyme:/export/projects/specs/design
```

A user on the NFS client, `sage`, enters the following command:

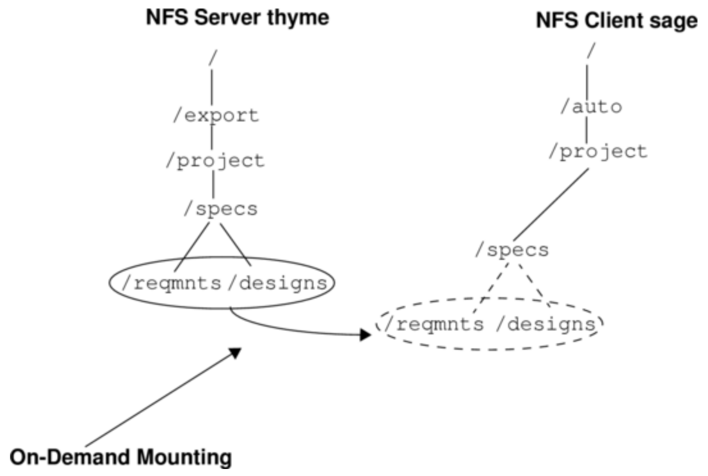
```
cd /auto/project/specs
```

Only the `/auto/project/specs` subdirectory is mounted. The `/auto/project/specs/design` subdirectory is mounted only when it is accessed, for example by using the following command:

```
cd /auto/project/specs/design
```

Figure 3-3 "Automounted Directories for On-Demand Mounting" illustrates the automounted file structure after the user enters the command.

Figure 3-3 Automounted Directories for On-Demand Mounting



Browsability for Indirect Maps

AutoFS now enables you to view the potential mount-points for indirect maps without mounting each filesystem.

Consider the following scenario where the AutoFS master map, `/etc/auto_master`, and the indirect map, `/etc/auto_indirect`, are on the NFS client, `sage`. Following are the contents of the master map, the `/etc/auto_master` file, which contains a single indirect map entry:

```
# /etc/auto_master file
# local mount-point         map name             mount options

  /nfs/desktop                  /etc/auto_indirect
```

Following are the contents of the indirect map, `/etc/auto_indirect`, which contains the local mount-points on the client and the references to the directories on the server:

```
# /etc/auto_indirect file
# local mount-point    mount options  remote server:directory

  /test                        -nosuid          thyme:/export/project/test
/apps                  -nosuid       basil:/export/apps
```

Enter the following commands to view the contents of the `/nfs/desktop` directory:

```
cd /nfs/desktop
ls
```

The `ls` command displays the following:

```
test          apps
```

The `test` and `apps` subdirectories are the potential mount-points. However, they are not currently mounted. To mount `test` and `apps`, enter the following commands:

```
cd /nfs/desktop/test
cd /nfs/desktop/apps
```

Concurrent Mount And Unmount

AutoFS performs concurrent mounts and unmounts, using the `automountd` daemon. For every mount and unmount request sent to the `automountd` daemon, a thread is created. This enables AutoFS to concurrently service multiple mounts and unmounts. This also prevents the service from hanging if a server is unavailable.

NFS Loopback Mount

By default, AutoFS uses the Loopback Filesystem (LOFS) mount for locally mounted filesystems. AutoFS provides an option to enable loopback NFS mounts for the local mount. Use the `automountd` command with the `-L` option to enable the loopback NFS mounts for locally mounted filesystems. This option is useful when AutoFS is running on a node that is part of a High Availability NFS environment.

Client-side Failover Support

AutoFS enables a mounted NFS read-only filesystem to transparently switch over to an alternate server if the current server goes down. AutoFS can make use of this feature if the NFS client supports client-side failover. For more information on client-side failover, see “Enabling Client-Side Failover” (page 38).

Secure NFS Support

AutoFS supports secure NFS filesystems if the NFS client supports mounting of secure directories. For more information on Secure NFS, see “Secure Sharing of Directories ” (page 26).

Reliable NFS Ping

In a congested network, the default timeout for an NFS ping may be too short, possibly resulting in failed mounts. AutoFS supports the *-retry*= *n* mount option for an NFS map entry to configure the `ping` timeout value. Increasing the value raises the probability for the ping to succeed.

Supported Filesystems

AutoFS enables you to mount different types of filesystems. To mount the filesystems, use the `fstype` mount option, and specify the location field of the map entry. Following is a list of supported filesystems and the appropriate map entry:

AutoFS	mount-point -fstype=autofs autofs_map_name

NOTE: You can specify another AutoFS map name in the location field of the map-entry. This would enable AutoFS to trigger other AutoFS mounts.

CacheFS	mount-point -fstype=cachefs, cachedir=/cache_directory, backfstype=nfs server:/source_directory

NOTE: You must ensure that the back-end filesystem, for example NFS, must be made available. Also, the cache directory must have been created.

HFS	mount-point -fstype=hfs :/device_path
NFS	mount-point server:/source directory

NOTE: You must ensure that the source directory has been created and exported on the server.

VxFS	mount-point /source_directory

NOTE: You must ensure that the source directory has been created.

NOTE: The examples specified are for AutoFS maps stored in plain files. You can use specific tools to convert the maps to other supported backends, such as NIS or LDAP.

To mount a CD-ROM device, follow these steps:

1. Log in as `superuser`.
2. Update the appropriate AutoFS map, as follows:

   ```
   cdrom -fstype=hfs, ro  :/dev/sr0
   ```

   For mount devices, If the mount resource begins with a “`/`” , it must be preceded by a colon. For instance in the above section, the CD-ROM device, `/dev/sr0`, is preceded by the colon.

Supported Backends (Map Locations)

AutoFS maps can be located in the following:

- Files: Local files that store the AutoFS map information for that individual system. An example of a map that can be kept on the local system is the master map. The `AUTO_MASTER` variable in `/etc/rc.config.d/nfsconf` is set to the name of the master map. The default master map name is `/etc/auto_master`.

  NOTE: To modify the master map name, use the `-f` option of the `automount` command and specify the *`filename`*. The `automount` command uses the specified file as the master map.

- Network Information Service (NIS): A service that enables you to administer the configuration of many hosts from a central location. Common configuration information, which otherwise has to be maintained separately on each host in a network without NIS, can now be stored and maintained in a central location and propagated to all the nodes in the network. For more information on NIS, see *NIS Administrator's Guide (5991–2187)* available at: http://docs.hp.com

- LDAP: A directory service that stores information, which is retrieved by clients throughout the network. To simplify HP-UX system administration, the LDAP-UX Integration product centralizes user, group, and network information management in an LDAP directory. For more information on the LDAP-UX Integration product, see the *LDAP-UX Client Services B.04.00 Administrator's Guide (J4269–90064)* available at: http://docs.hp.com

To configure the system to use one of these backends, follow these steps:

1. Configure the `/etc/nsswitch.conf` file by modifying the `automount` entry.

   This enables you to specify one or more data stores to look up information (autofs maps). You can also specify the order in which these data sources are looked up. The backends can be specified to look up AutoFS maps. For more information on configuring the `/etc/nsswitch.conf` file, see *nsswitch.conf*(4)

2. Configure the AutoFS maps on the specified data store.

Enabling LDAP Support for AutoFS

Before you enable LDAP support for AutoFS determine which schema shall be used for storing the AutoFS maps. The LDAP administrator can provide this information. HP-UX supports the following two options for schemas:

- older schema that uses `nisMAP` and `nisObject` objects to represent the maps
- new schema that uses `automountMap` and `automount` objects

To enable LDAP support for AutoFS using the older schema, follow these steps:

1. If the AutoFS maps are not already migrated, migrate your AutoFS maps to LDAP Directory Interchange Format (LDIF) files using the migration scripts.

   The migrated maps can also be used if you have chosen the older schema. For information on the specific migration scripts, see *LDAP-UX Client Services B.04.10 Administrator's Guide (J4269-90067)*.

2. Import the LDIF files into the LDAP directory server using the `ldapmodify` tool.

   For information on importing the LDIF files, see the *LDAP-UX Client Services B.04.10 Administrator's Guide (J4269-90067)*.

3. Install the following attribute/object mappings on the LDAP-UX client:

   - `objectclassmap: automount:automountMap='nisMap'`
   - `objectclassmap: automount:automount='nisObject'`
   - `attributemap: automount:automountMapName='nisMapName'`
   - `attributemap: automount:automountKey=cn`
   - `attributemap: automount:automountInformation='nisMapEntry'`

   For more information about the attribute mappings on the LDAP-UX client, see the *LDAP-UX Client Services B.04.10 Administrator's Guide (J4269-90067)*.

4. Configure the Name Service Switch (NSS). Back up the current `/etc/nsswitch.conf` file, and modify it to add LDAP as one of the backends for the automount service, as follows:

   ```
   automount: ldap
   ```

5. Enter the following command to run the AutoFS shutdown script:

   ```
   /sbin/init.d/autofs stop
   ```

6. Enter the following command to run the AutoFS startup script:

   ```
   /sbin/init.d/autofs start
   ```

To enable LDAP support for AutoFS using the older schema, follow these steps:

1. If the AutoFS maps are not already migrated, migrate your AutoFS maps to LDAP Directory Interchange Format (LDIF) files using the migration scripts.

   The LDAP-UX client only supports the new AutoFS schema, and the migration scripts will migrate the maps according to the new schema. For information on the specific migration scripts, see *LDAP-UX Client Services B.04.10 Administrator's Guide (J4269-90067)*.
2. Import the LDIF files into the LDAP directory server using the `ldapmodify` tool.

   For information on importing the LDIF files, see the *LDAP-UX Client Services B.04.10 Administrator's Guide (J4269-90067)*.
3. Enter the following command to run the AutoFS shutdown script:
   ```
   /sbin/init.d/autofs stop
   ```
4. Enter the following command to run the AutoFS startup script:
   ```
   /sbin/init.d/autofs start
   ```

AutoFS Configuration Prerequisites

Consider the following points before configuring AutoFS:

- Ensure that the local directory you configure as the mount-point is either empty or non-existent. If the local directory you configured as the mount-point is not empty, the local files or directories in it are hidden and become inaccessible if the automounted filesystem is mounted over the local directory.
- Before you mount a remote directory on your system, you must configure the remote system, where the directory is located, as an NFS server and export that directory.
- If you change the mount options, the remote server name, or the remote directory name for an existing direct or indirect mount while AutoFS is running, the changes take effect only when the directory is mounted the next time. However, if you change the local directory name in the direct or indirect map, or if you change the master map, these changes do not take effect until you run the `automount` command to force AutoFS to reread its maps.

Updating from Automounter to AutoFS

Automounter is a service that automatically mounts filesystems on reference. The service enables you to access the filesystems without taking on the superuser role and use the `mount` command. All filesystems appear to be immediately and continuously available.

Starting with HP-UX 11i v2, Automounter is no longer supported. HP recommends that you update to AutoFS. Updating to AutoFS offers the following advantages:

- AutoFS can be used to mount any type of filesystem.
- Automounter mounted the directories mounted under `/tmp_mnt` and created symbolic links from the configured mount-points to the actual ones on `/tmp_mnt`. In AutoFS, the configured mount-points are the actual mount-points.
- Automounter had to be killed and restarted when you wanted to modify maps. In AutoFS, you need not stop AutoFS when you modify the maps. Run the `automount` command after modifying the map for the change to take effect.

NOTE: Updating from Automounter to AutoFS does not require any user intervention.

The following steps are performed automatically to update all users from the Automounter configuration to AutoFS:

1. In the /etc/rc.config.d/nfsconf file, the AUTOFS variable is set to 1.
2. Any options you had specified in the AUTO_OPTIONS variable are copied to either the AUTO_OPTIONS or the AUTOMOUNTD_OPTIONS variable. Obsolete options are removed.

   Table 3-2 lists the options of the old automount command and the equivalent AutoFS command options. It also indicates which automount options are not supported in AutoFS.

Table 3-2 Old Automount Command-Line Options Used By AutoFS

Old automount Option	Equivalent AutoFS command option	Purpose
-D*variable* = *value*	automountd -D*variable* = *value*	To assign *value* to the environment *variable*.
-f*master_file*	automount -f*master_file*	To use *master_file* as local master map.
-M*mount_directory*	Obsolete with AutoFS.	To automount directories under *mount_directory* instead of /tmp_mnt.
-m	Obsolete with AutoFS.	To ignore NIS auto.master map.
-n	Obsolete with AutoFS.	To allow automounts only of previously mounted target filesystems.
-T	automountd -T	To enable automount tracing.
-tl*duration*	automount -t*duration*	To specify time before unmounting idle directories.
-tm*interval*	Obsolete with AutoFS.	To specify interval between mount attempts.
-tw*interval*	Obsolete with AutoFS.	To specify interval between unmount attempts.
-v	automount -vautomountd -v	To specify the Verbose mode.

3. Scripts that kill and restart automount are modified.

AutoFS Configuration Changes

This section describes the various methods to configure your AutoFS environment.

Configuring AutoFS Using the nfsconf File

You can use the /etc/rc.config.d/nfsconf file to configure your AutoFS environment. The /etc/rc.config.d/nfsconf file is the NFS configuration file. This file consists of the following sets of variables or parameters:

1. Core Configuration Variables
2. Remote Lock Manager Configuration Variables
3. NFS Client and Server Configuration Variables
4. AutoFS Configuration Variables

Table 3-3 lists the AutoFS configuration variables that you can use to configure your AutoFS environment.

Table 3-3 AutoFS Configuration Variables

Variable Name	Description
AUTOFS	Specifies if the system uses AutoFS. Set the value to 1 to specify that this system uses AutoFS. Set the value to 0 to specify that this system does not use AutoFS. The default value of AutoFS is 1. **NOTE:** If you set the value of *AUTOFS* to 1, the *NFS_CORE* core configuration variable must also be set to 1.
AUTOMOUNT_OPTIONS	Specifies a set of options to be passed to the `automount` command when it is run. The default value is “ ” . **NOTE:** The specified options take effect only if the `automount` command is run from the AutoFS startup script.
AUTOMOUNTD_OPTIONS	Specifies a set of options to be passed to the `automountd` daemon when it is started. The default value is “ ” . **NOTE:** The specified options take effect only if the `automountd` daemon is run from the AutoFS startup script.
AUTO_MASTER	Specifies the name of the AutoFS master map. The default value is `/etc/auto_master`. **NOTE:** The *AUTO_MASTER* option can only be used with the `/etc/rc.config.d/nfsconf` file.

When AutoFS starts up, and if the `AUTOMOUNT_OPTIONS` variable specifies a master map file with the `-f` *`filename`* option, AutoFS searches for a file by that name on the local host. It can also use the Name Service Switch (NSS) to determine which name services are in use. In addition, it determines the master maps that are available from those name services.

If the `AUTOMOUNT_OPTIONS` variable does not specify the `-f` *`filename`* option, AutoFS consults the NSS configuration, to determine where to search for the AutoFS master map.

For more information on configuring the NSS, see *nsswitch.conf* (4) and *automount*(1M) .

To configure AutoFS using the `/etc/rc.config.d/nfsconf` file, follow these steps:

1. Log in as `superuser`.
2. Edit the `/etc/rc.config.d/nfsconf` file. For instance, to change the default time for which a filesystem remains mounted when not in use, modify the *AUTOMOUNT_OPTIONS* variable, as follows:
   ```
   AUTOMOUNT_OPTIONS="-t 720"
   ```
3. Enter the following command to start AutoFS.
   ```
   /sbin/init.d/autofs start
   ```

Configuring AutoFS Using the`/etc/default/autofs` File

You can also use the `/etc/default/autofs` file to configure your AutoFS environment. The `/etc/default/autofs` file contains parameters for the `automountd` daemon and the `automount` command.

Initially, the parameters in the `/etc/default/autofs` file are commented. You must uncomment the parameter to make the value for that parameter take effect. Changes made to the `/etc/default/autofs` file can be overridden if you modify the AutoFS values from the command line.

When AutoFS starts up, and if the `AUTOMOUNT_OPTIONS` variable specifies a master map file with the `-f` *`filename`* option, AutoFS searches for a file by that name on the local host. It can also use the Name Service Switch (NSS) to determine which name services are in use. In addition, it determines the master maps that are available from those name services.

If the `AUTOMOUNT_OPTIONS` variable does not specify the `-f filename` option, AutoFS consults the NSS configuration, to determine where to search for the AutoFS master map.

For more information on configuring the NSS, see *nsswitch.conf* (4) and *automount*(1M) .

To configure AutoFS using the `/etc/default/autofs` file, follow these steps:

1. Log in as `superuser`.
2. Edit the `/etc/default/autofs` file. For instance, to change the default time for which a filesystem remains mounted when not in use, modify the *AUTOMOUNT_TIMEOUT* variable, as follows:

   ```
   AUTOMOUNT_TIMEOUT=720
   ```

   For the list of parameters supported in the `/etc/default/autofs` file, see *autofs*(1M).

NOTE: Values modified using the `/etc/default/autofs` file are updated when the command or daemon is started. This update is made irrespective of the way the command or daemon is started, Autofs startup script or command-line.

Enabling AutoFS

To enable AutoFS, follow these steps:

1. In the `/etc/rc.config.d/nfsconf` file, set the value of `AUTOFS` variables to 1, as follows:

   ```
   AUTOFS=1
   ```

2. Configure AutoFS using either the `/etc/rc.config.d/nfsconf` or the `/etc/default/autofs` configuration files. If you use a local file as the master map, ensure that the `AUTO_MASTER` variable in `/etc/rc.config.d/nfsconf` is set to the name of the master map, as follows:

   ```
   AUTO_MASTER="/etc/auto_master"
   ```

   If the `-f` option is not specified for the `automount` command then the Name Service Switch (NSS) is used to get the master map. If the master map is not found in any of the backends, then it tries the default master map name`/etc/auto_master`. If the `-f` option is specified for the `automount` command, then it uses that file irrespective of what the backend says. However, if the file is not found, then it will try the `/etc/auto_master` file. If you use an NIS AutoFS master map, ensure that `-f $AUTO_MASTER` is not present in the `AUTOMOUNT_OPTIONS` variable.

3. To run the AutoFS startup script, enter the following command:

   ```
   /sbin/init.d/autofs start
   ```

Notes on AutoFS Master Map

The AutoFS master map file, `/etc/auto_master` by default, determines the locations of all AutoFS mount-points. At system startup, the `automount` command reads the master map to create the initial set of AutoFS mounts. Subsequent to system startup, the `automount` command maybe run to install AutoFS mounts for new entries in the master map or a direct map, or to perform unmounts for entries that have been removed from these maps.

Following is a syntax of an `auto_master` file:

```
mount-point map-name  [mount-options]
```

where:

mount-point	Directory on which an AutoFS mount is made.
map-name	Map associated with mount-point specifying locations of filesystems to mount.
mount-options	Options that apply to the maps specified by `map-name`.

If the first field specifies the directory as /-, then the second field is the name of the direct map. The master map file, like any other map file, may be distributed by NIS or LDAP by modifying the appropriate configuration files and removing any existing `/etc/auto_master` master map file.

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.

You must run the `automount` command after you modify the master map or a direct map.

Following is a sample `auto_master` file:

```
# Master map for AutoFS
#
#auto_master
#mount-point  map-name            [mount-options]
/net         -hosts          -nosuid,nobrowse #Special Map
/home     auto_home       -nobrowse        #Indirect Map
/-        /etc/auto_direct                     #Direct Map
/nfs      /etc/auto_indirect                 #Indirect Map
```

The master map is a primary map that defines all other maps. In this sample, `auto_master`, the master map, contains a single entry each for a direct map and a special map. It also contains a single entry for each top-level indirect map.

Deciding Between Direct and Indirect Automounts

Before you automount a remote directory, you must decide if you want to use a direct or indirect AutoFS map.

Table 3-4 lists the advantages and disadvantages of direct and indirect AutoFS maps.

Table 3-4 Direct Versus Indirect AutoFS Map Types

Direct Map	Indirect Map
Advantage: You can view the contents of a direct-mounted directory using the `ls` command. If the contents are not currently mounted, the `ls` command mounts them.	*Advantage:* AutoFS enables you to view the available mount-points for indirect maps without actually mounting each filesystem when browsability is enabled by default.
Advantage: Direct-mounted automounted directories can export the same parent directory with local, or standard-mounted files and directories.	*Disadvantage:* An indirect map hides any local, standard-mounted, or direct-mounted files or directories under the mount-point for the map.
Disadvantage: If you add or remove mounts in a direct map, or if you change the local mount-point for an existing mount in a direct map, you must force AutoFS to reread its maps.	*Advantage:* If you modify an indirect map, AutoFS will view the changes the next time it mounts the directory. You need not force AutoFS to reread its maps.
Disadvantage: When `automount` reads a direct map, it creates an entry for each automounted directory in the internal mount table, `/etc/mnttab`. This can cause the mount table to become very large.	*Advantage:* When `automount` reads an indirect map, it creates only one entry for the entire map in the internal mount table, `/etc/mnttab`. Additional entries are created when the directories are actually mounted. The mount table takes up no more space than necessary, because only mounted directories appear in it.

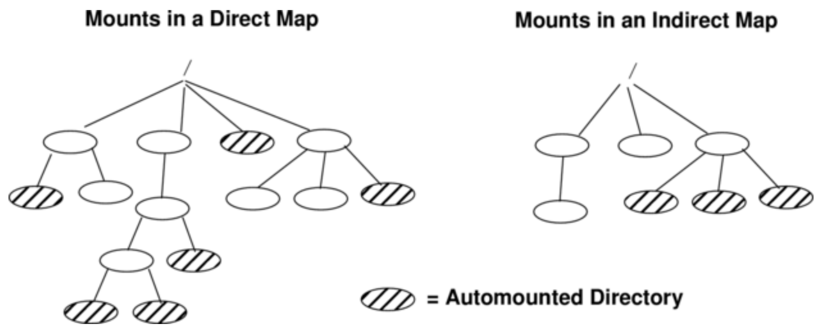
Configuring AutoFS Direct and Indirect Mounts

A direct map is an automount mount-point. It creates a direct association between a mount-point on the client and a directory on the server. Direct maps have the absolute path name. The automounts configured in a direct map can be mounted in various places in the local filesystem; they need not be located under the same parent directory.

An indirect map uses the key to establish a connection between a mount-point on the client and a directory on the server. Indirect maps are useful for accessing specific file systems, such as home directories. The automounts configured in an indirect map are mounted under the same local parent directory.

Figure 3-4 shows the difference between direct mounts and indirect mounts on an NFS client.

Figure 3-4 Difference Between Direct Mounts and Indirect Mounts



In the Mounts in a Direct Map figure, mounts are configured in various places in the local filesystem and not located under the same parent directory. In the Mounts in an Indirect Map figure, all the mounts are configured under the same parent directory.

CAUTION: Any filesystems that are being managed by AutoFS should never be manually mounted or unmounted. Even if the mount or unmount operation appears to complete successfully, the resulting change of state to the AutoFS managed filesystem can lead to disruptive or unpredictable results, including but not limited to: commands hanging or not returning expected results, and applications failing due to their dependencies on those mounted filesystems. A reboot may be necessary to resolve these issues.

Automounting a Remote Directory Using a Direct Map

This section describes how to automount a remote directory using a direct map.

To mount a remote directory using a direct map, follow these steps:

1. If you are using local files for maps, use an editor to edit the master map in the `/etc` directory. The master map is commonly called `/etc/auto_master`. If you are using NIS, open the master map on the NIS master server.

   If you are using LDAP, the map must be modified on the LDAP server. For information on modifying maps, see the *LDAP-UX Client Services B.04.00 Administrator's Guide (J4269-90064).*

   If the direct map you just modified is not listed in the master map, add the following entry to the master map:

   ```
   /-  direct_map_name  [mount_options]
   ```

2. If you are using local files for maps, use an editor to open or create a direct map in the `/etc` directory. The direct map is commonly called `/etc/auto_direct`. Add an entry to the direct map with the following syntax:

   ```
   local_directory [mount_options] server:remote_directory
   ```

   If you are using NIS or LDAP to manage maps, add an entry to the direct map on the NIS master server or the LDAP directory.

3. If you are using NIS to manage maps, rebuild the maps and push them to the slave servers. For more information on management of NIS maps, see the *NIS Administrator's Guide (5991-2187).*

4. On each host that uses the map you have modified, enter the following command to force AutoFS to read the modified map:

   ```
   /usr/sbin/automount
   ```

IMPORTANT: Do not automount a remote directory on a local directory, which is a symbolic link.

Ensure that the local mount-point specified in the AutoFS map entry is different from the exported directory on the NFS server. If it is the same, and the NFS server also acts as an NFS client and uses AutoFS with these map entries, the exported directory can attempt to mount over itself. This can result in unexpected behavior. A directory might also attempt to mount over itself if you use a single set of AutoFS maps that are distributed using NIS or LDAP, or are in a High Availability environment.

Consider the following sample entries in the `/etc/auto_master`, and `/etc/auto_direct` maps:

```
# Contents of the /etc/auto_master sample map
/net   -hosts    -nosuid,soft,nobrowse
/-      auto_direct
```

```
# Contents of the /etc/auto_direct sample map:
/tmp/export auto23:/tmp/export
```

If the NFS server also acts as an NFS client, for example `auto23` acts as the NFS server and client, and the AutoFS map references the NFS server and tries to overlay a VxFS path, it may result in unexpected behavior.

Notes on Direct Maps

The mount options that you can specify in the AutoFS maps are the same options that you use for the type of filesystems you attempt to automount. For example, if the filesystem type is NFS, then the mount options you use are identical to the ones used for standard NFS mounted directories. For more information on the different mount options, see "Changing the Default Mount Options" (page 40).

You cannot use the `bg` option for an automounted directory. The mount options configured in the direct map override the ones in the master map, if there is a conflict.

You can configure all the direct automounts in the same map. Most users use the file name, `/etc/auto_direct`, for their direct map. Following is the syntax for a direct map:

```
local mount-point    mount options   remote server:directory
```

where:

local mount-point	The path name of the mount-point
mount options	Options you want to apply to this mount.
remote server:directory	Location of the directory, on the server, that is to be mounted.

If you plan to use NIS or LDAP to manage maps, there can be only one direct map in your configuration.

If the direct map name in the master map begins with a slash (/), AutoFS considers it to be a local file. If it does not contain a slash, AutoFS uses the NSS to determine whether it is a file, LDAP, or an NIS map. For more information on using NSS, see *nsswitch.conf*(4).

Sample File Entries for NFS Direct Automounts

Following are sample entries from the AutoFS master map on the NFS client, `sage`:

```
# /etc/auto_master file
# local mount-point        map name              mount options

/-                         /etc/auto_direct
```

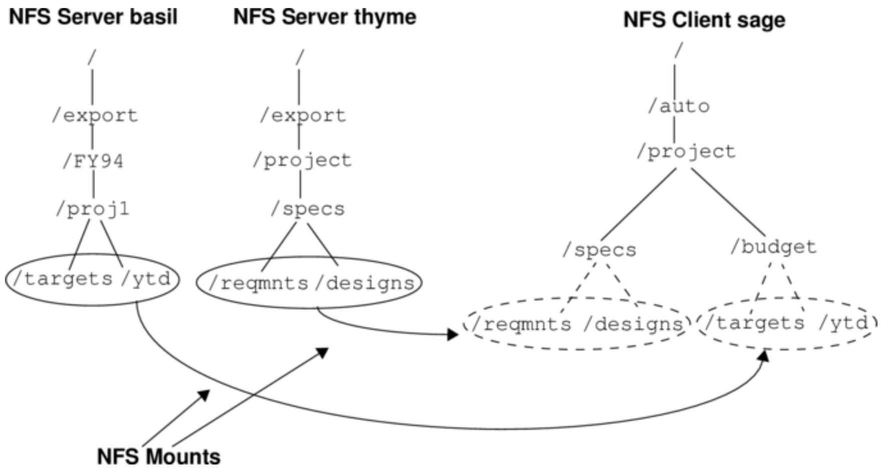
Following are sample entries from an AutoFS direct map on the NFS client, `sage`. The hash (#) symbol indicates a commented line.

```
# /etc/auto_direct file
# local mount-point    mount options  remote server:directory

/auto/project/specs  -nosuid      thyme:/export/project/specs
/auto/project/budget   -nosuid       basil:/export/FY94/proj1
```

Figure 3-5 illustrates how AutoFS sets up direct mounts.

Figure 3-5 How AutoFS Sets Up Direct Mounts



Automounting a Remote Directory Using an Indirect Map

This section describes how to automount a remote directory using an indirect map.

To automount a remote directory using an indirect map, follow these steps:

1. If you are using local files for maps, use an editor to edit the master map in the `/etc` directory. The master map is commonly called `/etc/auto_master`. If you are using NIS, open the master map on the corresponding master server.

   If you are using LDAP, the map must be modified on the LDAP server. For information on modifying the map, see the *LDAP-UX Client Services B.04.00 Administrator's Guide (J4269-90064).*

   If the indirect map you modified is not listed in the master map, add the following entry to the master map:

   ```
   local_parent_directory indirect_map_name [mount_options]
   ```

2. If you are using local files for your AutoFS maps, use an editor to open or create an indirect map in the `/etc` directory. Add a line with the following syntax, to the indirect map:

   ```
   local_subdirectory [mount_options] server:remote_directory
   ```

   If you are using NIS or LDAP to manage maps, add an entry to an indirect map on the corresponding NIS master server or the LDAP directory.

3. If you are using NIS to manage maps, rebuild the maps and push them to the slave servers. For more information on NIS maps, see the *NIS Administrator's Guide (5991-2187).*
4. If you modified the master map, enter the following command on each host that uses the map, to force AutoFS to read the modified master map:

   ```
   /usr/sbin/automount
   ```

IMPORTANT: Ensure that *`local_parent_directory`* and *`local_subdirectory`* are not already created. AutoFS creates them when it mounts the remote directory. If these directories exist, their files and directories in them are hidden when the remote directory is mounted.

Ensure that the local mount-point specified in the AutoFS map entry is different from the exported directory on the NFS server. If it is the same, and the NFS server also acts as an NFS client and uses AutoFS with these map entries, the exported directory might attempt to mount over itself. This can result in unexpected behavior. A directory might also attempt to mount over itself if you use a single set of AutoFS maps that are distributed using NIS or LDAP, or are in a High Availability environment.

Notes on Indirect Maps

The mount options that you can specify in the AutoFS maps are the same ones that you use for the type of filesystem you attempt to automount. For example, if the filesystem type is NFS, then the mount options you use are identical to the ones used for standard NFS mounted directories. For a list of mount options, see "Changing the Default Mount Options" (page 40). You cannot use the `bg` option for an automounted directory. The mount options configured in the indirect map override the ones in the master map if there is a conflict.

Indirect maps are usually called `/etc/auto_`*`name`*, where *`name`* helps you remember what is configured in the map. Following is the syntax of the indirect map:

```
local mount-point    mount options   remote server:directory
```

where:

local mount-point	Simple name in the indirect map
mount options	Options you want to apply to this mount.
remote server:directory	Location of the directory, on the server, that is to be mounted.

If the indirect map name in the AutoFS master map begins with a slash (/), AutoFS assumes that it is a local file. If it does not contain a slash, AutoFS uses the Name Service Switch to determine whether it is a file, LDAP, or an NIS map. For more information on configuring the Name Service Switch, see *nsswitch.conf*(4).

Sample File Entries for NFS Indirect Automounts

The *`local_subdirectory`* specified in an indirect map is the lowest-level subdirectory in the local directory path name. For example, if you are mounting a remote directory on `/nfs/apps/draw`, `draw` is the *`local_subdirectory`* specified in the indirect map.

Following are sample lines from the AutoFS master map on the NFS client, `sage`. The master map also includes an entry for the `/etc/auto_direct` direct map.

```
# /etc/auto_master file
# local mount-point         map name           mount options

/-                         /etc/auto_direct
/nfs/desktop                 /etc/auto_desktop
```

The *`local_parent_directory`* specified in the master map consists of directories, except the lowest-level subdirectory in the local directory pathname. For example, if you are mounting a remote directory on `/nfs/apps/draw`, `/nfs/apps` is the *`local_parent_directory`* specified in the master map.

You can configure indirect automounts in the same indirect map only if their *`local_parent_directory`*, as specified in the master map, is the same. For example, indirect mounts with the local mount-points, `/nfs/apps/draw` and `/nfs/apps/word`, can be configured in the same indirect map.

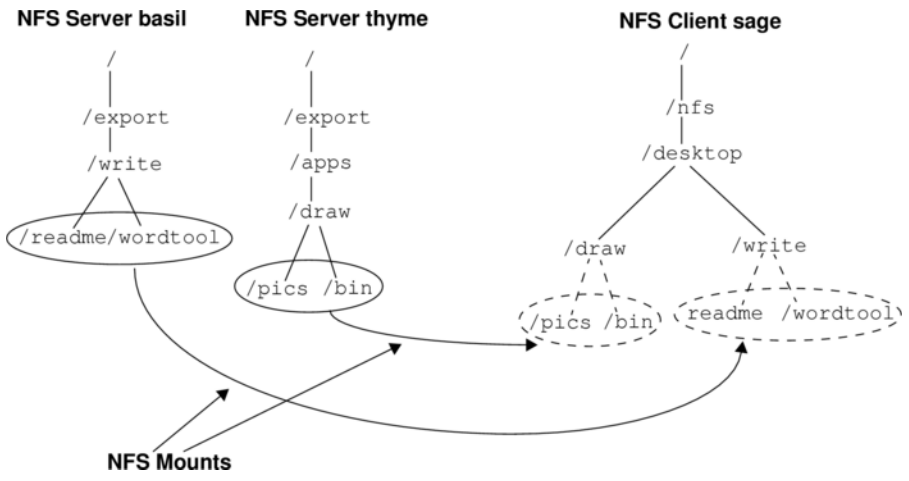
Following are sample lines from an AutoFS indirect map on the NFS client, `sage`. The hash (#) symbol indicates a commented line.

```
# /etc/auto_desktop file
# local mount-point    mount options   remote server:directory

draw                    -nosuid          thyme:/export/apps/draw
write                   -nosuid           basil:/export/write
```

Figure 3-6 illustrates how AutoFS sets up indirect mounts.

Figure 3-6 How AutoFS Sets Up NFS Indirect Mounts



Using Environment Variables as Shortcuts in AutoFS Maps

This section describes how to use environment variables as shortcuts in AutoFS maps using an example. You can create an environment variable by prefixing a dollar (`$`) to its name, or enclosing it in braces ({ }). You can use an environment variable in a direct or an indirect AutoFS map anywhere except the first field, which specifies the local mount-point.

IMPORTANT: You cannot use environment variables in the AutoFS master map.

In this example, the NFS server `basil`, contains subdirectories in its `/export/private_files` directory, which are named after the hosts in its network. Every host in the network can use the same AutoFS map and the same `AUTOMOUNTD_OPTIONS` definition to mount its private files from `basil`.

When AutoFS starts up on the host `sage`, it assigns the value `sage` to the `HOST` variable. When you request access to the local `/private_files` directory on `sage`, AutoFS mounts `/export/private_files/sage` from `basil`.

To use environment variables as shortcuts in direct and indirect maps, follow these steps:

1. To enable the `automount` command to expand the environment variable to its current value of the client's hostname, modify the direct map entry on the server, `basil`, as follows:

   ```
   /private_files  basil:/export/private_files/$HOST
   ```

   In the direct map entry, `HOST`, is the environment variable.

2. Add the `-D` option to the `AUTOMOUNTD_OPTIONS` variable in the `/etc/rc.config.d/nfsconf` file to assign a value to the variable, as follows:

   ```
   AUTOMOUNTD_OPTIONS=”-D HOST='hostname'”
   ```

NOTE: You can also use the `/etc/default/autofs` file to modify the value assigned to the variable.

You can use any environment variable that is set to a value in an AutoFS map. If you do not set the variable either with the `-D` option in `/etc/rc.config.d/nfsconf` file or by using the `/etc/default/autofs` file, AutoFS uses the current value of the environment variable on the local host.

For information on some of the pre-defined environment variables, see *automount*(1M).

Using Wildcard Characters as Shortcuts in AutoFS Maps

Using wildcard characters makes it very easy to mount all the directories from a remote server to an identically named directory on the local host.

Consider the following guidelines while using wildcard characters as shortcuts:

- Use an asterisk (*) as a wildcard character in an indirect map, to represent the local subdirectory if you want the local subdirectory to be the same as the remote system name, or the remote subdirectory.

  You cannot use the asterisk (*) wildcard in a direct map.

- Use an ampersand (&) in a direct or an indirect map as the remote system name or the remote subdirectory. The entry in the local directory name field replaces the ampersand. If you have used an asterisk to represent the local subdirectory, then the entry that replaces the asterisk (*) in the local subdirectory field also replaces the ampersand (&) in the remote system name, or remote subdirectory field.

Notes on Using Wildcard Characters as Shortcuts in Maps

The following example illustrates the automounting of the users' home directories. The home directories are physically located on the NFS server, `basil`, under the remote directory `/export/home`. On the local NFS client, the home directories are mounted under `/home`.

The following entry from the `/etc/auto_master` master map lists the indirect map, `/etc/auto_home`:

```
# /etc/auto_master file
# local mount-point       map name        mount options

/home                     /etc/auto_home      -nosuid
```

The following line from the `/etc/auto_home` indirect map mounts the user's home directories on demand:

```
# /etc/auto_home file
# local mount-point   mount options  remote server:directory

*                                        basil:/export/home/&
```

The user's home directory is configured in the `/etc/passwd` file as `/home/`*`username`*. For example, the home directory of the user `terry` is `/home/terry`. When Terry logs in, AutoFS looks up the `/etc/auto_home` map and substitutes `terry` for both the asterisk and the ampersand. AutoFS then mounts Terry's home directory from `/export/home/terry` on the server, `basil`, to `/home/terry` on the local NFS client.

You can use the ampersand character to represent both the remote server and the remote subdirectory, in the same line of the indirect map. For example, if the user's home directory is physically located on many different servers, but the directory under which the home directory is located is called `/export/home/`*`servername`* on all the servers, the following line in the `/etc/auto_home` map mounts all the user's home directories from any server:

```
*         &:/export/home/&
```

If the home directory of the user `terry` is configured in the `/etc/passwd` file as `/home/basil/terry`, AutoFS mounts the remote directory `/export/home/basil` from the server, `basil`, on the local directory `/home/basil` when Terry logs in.

The line with the asterisk must be the last line in an indirect map. AutoFS reads the lines in the indirect map sequentially until it finds a match for the requested local subdirectory. If asterisk (*) matches any subdirectory, AutoFS stops reading at the line with the asterisk.

For example, if the `/etc/auto_home` map contains the following lines,

```
*           basil:/export/home/&
charlie     thyme:/export/home/charlie
```

AutoFS attempts to mount `/export/home/charlie` from the host, `basil`. If the asterisk is a match for `charlie`, AutoFS looks no further and never reads the second line. However, if the `/etc/auto_home` map contains the following lines,

```
charlie     thyme:/export/home/charlie
*           basil:/export/home/&
```

AutoFS mounts Charlie's home directory from host `thyme` and other home directories from host `basil`.

Automounting Home Directories

To automount users' home directories, follow these steps:

1. Ensure that the systems where the users' home directories are located are set up as NFS servers and are exporting the home directories. For more information on configuring a system as an NFS server, see "Configuring and Administering an NFS Server" (page 21).
2. In the `/etc/passwd` file on the NFS clients, configure the home directory of each user as the NFS mount-point, where the user's home directory is mounted. For example, if home directories are mounted under `/home`, Claire's home directory will be configured as `/home/claire` in the `/etc/passwd` file.
3. If you are using local files for maps, create a file called `/etc/auto_home` on the NFS clients, and add an entry for each user.

   ```
   sammy       thyme:/export/home/&     -nosuid
   ```

   The ampersand (&) character takes the value of the user name in each line. User Sammy's home directory is physically located on host `thyme` in `/export/home/sammy`.
4. If you are using local files for maps, add the following entry to the AutoFS master map, `/etc/auto_master`, on the NFS clients:

   ```
   /home  /etc/auto_home
   ```
5. Enter the following command on each NFS client that uses these maps to force AutoFS to reread the maps:

   ```
   /usr/sbin/automount
   ```

Example of Automounting a User's Home Directory

User Howard's home directory is located on the NFS server, `basil`, where it is called `/export/home/howard`. Each client in the network has the following entry in the `/etc/passwd` file on all the systems in the network:

```
howard:*:700:70:Howard:/home/howard:/usr/bin/ksh
```

When Howard logs in to any NFS client, AutoFS recognizes `/home` as an AutoFS mount-point, because it is configured in the master map, as follows:

```
/home  auto_home
```

AutoFS reads the `auto_home` map to find out how to mount Howard's home directory. It finds the following line:

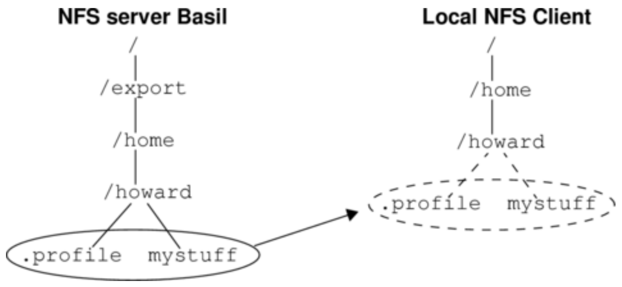
```
howard      basil:/export/home/&     -nosuid
```

AutoFS substitutes howard for the ampersand (&) character in that line:

```
howard    basil:/export/home/howard    -nosuid
```

AutoFS mounts /export/home/howard from server basil to the local mount-point /home/howard on the NFS client. Figure 3-7 illustrates this configuration.

Figure 3-7 Home Directories Automounted with Wildcards



Special Maps

There are two types of special maps: -hosts and -null.

By default, the -hosts map is used with the /net directory and assumes that the map entry is the hostname of the NFS server. The automountd daemon dynamically creates a map entry from the server's list of exported filesystems. For example, a reference to /net/Casey/usr initiates an automatic mount of all the exported filesystems from Casey which can be mounted by the client. References to the directory under /net/Casey refers to the corresponding directory relative to the root directory of Casey.

The -null map, when included in a map entry, cancels a map for the directory indicated. This is especially useful in the AutoFS master map, /etc/auto_master, for cancelling entries that would otherwise be inherited from the +auto_master include entry. To be effective, the -null entry must be inserted before the included map entry. For information on how to include a map in another map, see "Including a Map in Another Map" (page 74).

Automounting All Exported Directories from Any Host Using the -hosts Map

To automount all exported directories using the -hosts map, follow these steps:

1. If you are using local files for AutoFS maps, use an editor to add the following entry to the /etc/auto_master master map file:

   ```
   /net -hosts -nosuid,soft,nobrowse
   ```

   NOTE: The nobrowse option in the local default master map file for a newly installed system is specified for the /net map.

   If you are using NIS to manage AutoFS maps, add the previous entry to the master map file on the NIS master server. Rebuild the map and push it out to the slave servers. For more information on NIS and NIS maps, see the *NIS Administrator's Guide (5991-2187)*. If you are using LDAP, add the entry to the LDAP directory.

2. On each host that uses the map you modified, enter the following command to force AutoFS to read the modified map:

   ```
   /usr/sbin/automount
   ```

Notes on the -hosts Map

The -hosts map is a built-in AutoFS map. It enables AutoFS to mount exported directories from any NFS server found in the hosts database, whenever a user or a process requests access to one of the exported directories from that server.

CAUTION: You may inadvertently cause an NFS mount over X.25 or SLIP, which is unsupported, or through a slow router or gateway, because the -hosts map allows NFS access to any reachable remote system. Mounts over slow links can cause excessive retransmissions and degrade the performance for all users.

When a user or a process requests a directory from an NFS server, AutoFS creates a subdirectory named after the NFS server, under the local mount-point you configured in the AutoFS master map. (The conventional mount-point for the -hosts map is /net). AutoFS then mounts the exported directories from that server. These exported directories are now accessible.

The -hosts map is an indirect map. It uses the hosts database (the /etc/hosts file, the NIS hosts map, LDAP, or BIND [DNS]) to discover the host on the network. The NSS configuration determines which name services to search for host information. For more information on configuring the NSS, see *nsswitch.conf*(4).

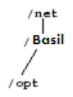
For example, if the server, Basil exports /opt and /apps, and a user on your NFS client enters the following command:

```
cd /net/basil/opt/frame
```

the subdirectory, /basil is created under /net, and /opt is mounted under /basil.

Figure 3-8 shows the automounted file structure after the user enters the command.

Figure 3-8 Automounted Directories from the -hosts Map—One Server



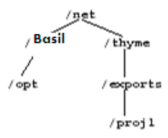
In the following example, the server thyme exports the directory /exports/proj1, and a user enters the following command:

```
more /net/thyme/exports/proj1/readme
```

The subdirectory /thyme is created under /net, and /exports/proj1 is mounted under /thyme.

Figure 3-9 shows the automounted directory structure after the user enters the second command.

Figure 3-9 Automounted Directories from the -hosts Map—Two Servers



Turning Off an AutoFS Map Using the `-null` Map

To turn off a map using the `-null` map, follow these steps:

1. Add a line with the following syntax in the AutoFS master map:

   ```
   local_directory -null
   ```

2. If AutoFS is running, enter the following command on each client that uses the map, to force AutoFS to reread its maps:

   ```
   /usr/sbin/automount
   ```

   This enables AutoFS to ignore the map entry that does not apply to your host.

Notes on the `-null` Map

The `-null` map is used to ignore mapping entries that do not apply to your host, but which would otherwise be inherited from the NIS or LDAP maps.

The `-null` option causes AutoFS to ignore AutoFS map entries that affect the specified directory. For example, an NIS (or LDAP) `auto_master` map is configured such that AutoFS mounts the `auto_home` map on `/home`, as follows:

```
/home auto_home
```

You can include the following line in the `/etc/auto_master` file, before the `+auto_master` entry, so that the NIS `auto_home` map is not used for the system:

```
/home -null
```

```
+auto_master
```

NOTE: The `-null` entry must precede the included map entry to be effective.

Using Executable Maps

An executable map is a map whose entries are generated dynamically by a program or a script. AutoFS determines whether a map is executable, by checking whether the `execute` bit is set in its permissions string. If a map is not executable, ensure that its `execute` bit is not set. An executable map is an indirect map.

When the AutoFS daemon locates a map whose `execute` bit is set, then instead of opening the file and searching for an entry, the daemon executes the file as a program and passes the key to be located within the map as an argument. Executable AutoFS maps return a map entry on the standard output. If they cannot supply a map entry for the key, they do not return anything. You can list executable AutoFS maps in the master map or include them in the local AutoFS map files. However, executable maps cannot be used with NIS and LDAP.

For example, an executable map implemented as a shell script, emulates the AutoFS built-in `-hosts` map for `/net`. It obtains a list of exported file systems from an NFS server (its name is the key argument). It then formats the path names into a multiple-mount map entry and sorts the list to order the mounts correctly into a top-down hierarchy, as follows:

```
!/bin/sh
Server=$1
showmount -e $1 | awk ‘NR > 1 {print $1 “\t'$Server’:” $1 ” \\ “}’ | sort
```

Advanced AutoFS Administration

This section presents advanced AutoFS concepts that enable you to improve mounting efficiency and also help make map building easier. This section addresses:

- “Automounting Multiple Directories (Hierarchical Mounts)” (page 72)
- “Configuring Replicated Servers for an AutoFS Directory” (page 72)

- "Including a Map in Another Map" (page 74)
- "Creating a Hierarchy of AutoFS Maps" (page 74)

Automounting Multiple Directories (Hierarchical Mounts)

AutoFS enables you to automount multiple directories simultaneously.

Use an editor to create an entry with the following format in a direct or indirect map, and if needed, create the `auto_master` entry:

```
local_dir   /local_subdirectory
[-options] \ server:remote_directory \
/local_subdirectory [-options]
server:remote_directory \
...
```

Adding these map entries does not automatically mount them. The listed remote directories are mounted only when referenced. For example, the following entry from a direct map mounts the source code and the data files for a project whenever anyone requests access to both of them:

```
/our_project   /source   -ro    basil:/opt/proj1/src  \
               /datafiles    thyme:/opt/proj1/samples/data
```

The following is another example from an indirect map.

```
chap2 -nosuid   /text       thyme:/our_book/chap2  \
                /graphics   basil:/our_book/artwork/chap2 \
                 /old         thyme:/our_book/oldfiles/chap2
```

The `nosuid` mount option applies to all three automounted directories. For more information on the `nosuid` mount option, see *automount*(1M).

Configuring Replicated Servers for an AutoFS Directory

This section describes how to configure multiple replicated servers for an AutoFS directory.

Modify the entry that mounts the remote directory to list multiple servers in the appropriate map, as follows:

- If the remote directories have the same name on every server, use a syntax such as the following from an indirect map:

  ```
  man  -ro  broccoli,cabbage,cauliflower:/usr/share/man
  ```

- You can assign a weighting factor to the servers by specifying a number in parentheses after each server name. If the server has a lower weighting number specified, it is likely to be selected. Servers that have no weight factor specified have a default weight of zero and are most likely to be selected.

  ```
  man -ro broccoli(1),cabbage(2),cauliflower(3):/usr/share/man
  ```

  However, server proximity is more important than the weighting factor you assign. A server on the same network segment as the client is more likely to be selected, than a server on another network segment, regardless of the weight you assign. The weighting factor is taken into account only when deciding between servers with the same network proximity.

- If the remote directory has a different name on different servers, use a syntax such as the following from a direct map:

  ```
  /nfs/proj2/schedule -ro \
           broccoli:/export/proj2/schedule, \
           cauliflower:/proj2/FY07/schedule
  ```

To configure multiple replicated servers for a directory, follow these steps:

1. Create and configure the `/etc/netmasks` file. AutoFS requires the `/etc/netmasks` file to determine the subnets of local clients in a replicated multiple server environment.

   The `/etc/netmasks` file contains IP address masks with IP network numbers. It supports both standard subnetting as specified in RFC-950, and variable-length subnetting as specified in RFC-1519. For standard subnetting, the file must contain a single line for each network. This provides the network number and the network mask to be used on that network. Specify the network number and the mask using the conventional IP dot (.) notation. The network number must be either a class A, B, or C network number. For example:

   ```
   network number           netmask
   128.32.0.0             255.255.255.0
   ```

   For variable length subnetting, the `/etc/netmasks` file must contain a line for each subnet, where the first field refers to the subnet and the second field refers to the netmask. The format for specifying these fields is the same as that of standard subnetting.

Example of using the `/etc/netmasks` File

The following example shows how AutoFS uses the `/etc/netmasks` file to determine the local client subnets in a multiple-server environment. In this example, servers `basil` and `thyme` export`/nfs/mount`. The IP address for server `basil` is 15.43.232.30, and the IP address for the server, `thyme` is 15.244.10.20:

```
# /etc/auto_direct file
/nfs_mount         basil,thyme:/nfs/mount
#/etc/netmasks file on the client sage:
#network number    netmask
15.43.234.210    255.255.248.0
```

AutoFS uses the `/etc/netmasks` file to determine that the masked value for the subnet of `basil` and the network number is the same (15. 42. 232. 0). This shows that the client is on the same network as `basil`. AutoFS then mounts `/nfs/mount` from `basil` on the local subnet.

Notes on Configuring Replicated Servers

Directories with multiple servers must be mounted as read-only to ensure that the versions remain the same on all servers.

The server selected for the mount is the one with the highest preference, based on a sorting order. The sorting order used gives highest preference to servers on the same local subnet. Servers on the local network are given the second strongest preference. As a result, if you configure multiple servers on both sides of a gateway, a server on the same side of the gateway as the NFS client is always preferred. For multiple servers outside the local network, and with no weighting factors assigned, the server with the lowest response time is used for the mount.

Multiple servers provide users with reliable access to a mounted directory. If one server is down, the directory can be mounted from another. Moreover, multiple servers provide some load balancing across the network; a server that is not busy responds more quickly to an AutoFS poll than one that is heavily loaded. The directory is mounted from the server that is not busy.

If the list of multiple servers contains a combination of servers that includes all versions of the NFS protocol, then AutoFS selects a subset of servers with the highest NFS protocol version configured. For example, a list contains a number of servers configured with the NFSv4 protocol, and a few servers configured with the NFSv2 protocol. AutoFS will use the subnet of servers configured with the NFSv4 protocol, unless a server configured with the NFSv2 protocol is closer.

Including a Map in Another Map

If you want your map to refer to an external map, you can do so by including the external map in your map. The entries in the external map are read as if they are part of your map.

To include the contents of an AutoFS map in another AutoFS map, add a plus (+) sign before the map name, as in the following example:

```
# /etc/auto_home file
# local mount-point mount options remote server:directory

basil -nosuid         basil:/export/home/basil
+auto_new
```

Assume that the `/etc/auto_home` map is listed in the master map with the following line:

```
/home           /etc/auto_home
```

If a user, whose home directory is in `/home/basil`, logs in, AutoFS mounts the `/export/home/basil` directory, from the host, `basil`.

If a user, whose home directory is in `/home/sage`, `/home/thyme`, or any subdirectory of `/home` other than `basil`, logs in, AutoFS consults the `auto_home` map for information on mounting the user's home directory.

The plus (+) sign instructs AutoFS to look up a different map for the information it needs to mount the directory. If the map name following the plus sign begins with a slash, AutoFS assumes that it is a local file. If the map name contains no slashes, AutoFS uses the Name Service Switch to determine whether it is a file, NIS map, or an LDAP map.

You can include an AutoFS map inside a local file, but not inside an NIS or LDAP map.

For more information on including a map in another map, see *automount*(1M).

Creating a Hierarchy of AutoFS Maps

Hierarchical AutoFS maps provide a framework that enables you to organize large exported filesystems. Together with NIS, which allows you to share information across administrative domains, hierarchical maps enable you to decentralize the maintenance of a shared namespace.

Sample Map Hierarchy

In the following example, an organization consisting of many departments, wants to organize a shared automounted directory structure. The shared top-level directory is called `/org`. The `/org` directory contains several subdirectories, which are listed in the `auto_org` map. Each department administers its own map for its subdirectory.

The AutoFS master map needs only a single entry for `/org` as in the following example:

```
# auto_master map
# Directory           Map Name
/org                  auto_org
```

The `auto_org` map appears similar to the following:

```
finance    -fstype=autofs    auto_finance
marketing  -fstype=autofs    auto_marketing
legal      -fstype=autofs    auto_legal
research   -fstype=autofs    auto_research
eng        -fstype=autofs    auto_eng
```

The engineering department map, `auto_eng`, appears similar to the following:

```
releases                      bigiron:/export/releases
tools                         mickey,minnie:/export/tools
source      -fstype=autofs    auto_eng_source
projects     -fstype=autofs     auto_eng_projects
```

A user in the blackhole project within engineering can use the following path:

```
/org/eng/projects/blackhole
```

Starting with the AutoFS mount at `/org`, the evaluation of this path dynamically creates additional AutoFS mounts at `/org/eng` and `/org/eng/projects`. No action is required for the changes to take effect on the user's system because the AutoFS mounts are created only when required. You need to run the `automount` command only when you make changes to the master map or to a direct map.

Modifying or Removing an Automounted Directory from a Map

To modify or remove an automounted directory from a map, follow these steps:

1. Enter the following command to determine whether the directory is currently in use:

   ```
   /usr/sbin/fuser -cu local_mount_point
   ```

   This command lists the process IDs and user names of everyone using the mounted directory.
2. Warn all users to exit the directory. Terminate processes that are using the directory, or wait until the processes terminate. Enter the following command to kill all the processes that are using the mounted directory:

   ```
   /usr/sbin/fuser -ck local_mount_point
   ```
3. Use an editor to modify the direct or indirect map.
4. If you removed all the entries in the direct or indirect map, remove that map’s entry in the master map.
5. If you made any changes to the master map, or if you added or modified a local mount-point in a direct map, enter the following command to force AutoFS to reread its maps:

   ```
   /usr/sbin/automount
   ```

CAUTION: You must maintain filesystems managed by AutoFS, by using the `automountd` and `automount` utilities. Manually mounting and unmounting file systems managed by AutoFS can cause disruptive or unpredictable results, including but not limited to commands hanging or not returning expected results. Applications can also fail because of their dependencies on these mounted filesystems.

Verifying the AutoFS Configuration

This section describes how to verify the AutoFS configuration.

To verify the configuration, follow these steps:

1. Enter the following command to change the current working directory to an automounted directory:

   ```
   /usr/bin/cd local_directory
   ```

   where:

   `local_directory` is the configured mount-point in the AutoFS map.
2. Enter the following command to verify that the contents of the remote directory are mounted under the local mount-point:

   ```
   /usr/bin/ls
   ```

If the directory is configured in an indirect map, entering the `ls` command from the parent directory displays potential mount-points (browsability). Changing to a subdirectory configured in the indirect map or entering the command, `ls` *`subdirectory`*, mounts the subdirectory.

The following example shows an indirect map configuration:

```
# /etc/auto_master file
# local mount-point         map name             mount options

/nfs/desktop                /etc/auto_desktop
```

```
# /etc/auto_desktop file
# local mount-point mount options remote server:directory

draw                -nosuid          thyme:/export/apps/draw
write                 -nosuid           basil:/export/write
```

Enter the following commands:

```
cd /nfs/desktop
ls
```

The `ls` command displays the following output:

```
draw         write
```

The `draw` and `write` subdirectories are the potential mount-points (browsability), but are not currently mounted. However, if you enter the following commands, both `draw` and `write` subdirectories are mounted:

```
cd /nfs/desktop/write
cd /nfs/desktop/draw
```

If AutoFS does not mount the configured directories, see "Troubleshooting NFS Services" (page 95).

Disabling AutoFS

This section describes how to disable AutoFS.

To disable AutoFS, follow these steps:

1. To run the AutoFS shutdown script, enter the following command:

   ```
   /sbin/init.d/autofs stop
   ```

2. In the `/etc/rc.config.d/nfsconf` file, set the value of `AUTOFS` variable to 0, as follows:

   ```
   AUTOFS=0
   ```

IMPORTANT: Do not disable AutoFS by terminating the `automountd` daemon with the `kill` command. It does not unmount AutoFS mount-points before it terminates. Use the `autofs stop` command instead.

Restarting AutoFS

AutoFS rarely needs to be restarted. In case you do need to restart it, follow these steps:

1. To find a list of all the automounted directories on the client, run the following script:

   ```
   for FS in $(grep autofs /etc/mnttab | awk '{print $2}')
   do
     grep 'nfs' /etc/mnttab | awk '{print $2}' | grep ^${FS}
   done
   ```

2. To determine whether each automounted directory returned by the `grep` command is currently in use, enter the following command:

   ```
   /usr/sbin/fuser -cu local_mount_point
   ```

   This command lists the process IDs and user names of all the users who are using the mounted directory.

3. Warn any users to exit the directory, and terminate any processes that are using the directory, or wait until the processes terminate. To terminate all the processes that are using the mounted directory, enter the following command:

   ```
   /usr/sbin/fuser -ck local_mount_point
   ```

4. To stop AutoFS, enter the following command:

   ```
   /sbin/init.d/autofs stop
   ```

IMPORTANT: Do not stop the `automountd` daemon with the `kill` command. It does not unmount AutoFS mount-points before it terminates. Use the `autofs stop` command instead.

5. To ensure that AutoFS is no longer active, enter the `ps` command:

   ```
   /usr/bin/ps -ef | grep automount
   ```

   If the `ps` command indicates that AutoFS is still active, ensure that all users have exited the automounted directories and then try again. Do not restart AutoFS until all the `automount` processes are terminated.

6. To start AutoFS, enter the following command:

   ```
   /sbin/init.d/autofs start
   ```

Troubleshooting AutoFS

This section describes the tools and procedures for troubleshooting AutoFS.

AutoFS Logging

AutoFS logs messages through `/usr/sbin/syslogd`. By default, `syslogd` writes messages to the file `/var/adm/syslog/syslog.log`. For more information on the `syslog` daemon, see *syslogd*(1M).

To Start AutoFS Logging

To start AutoFS Logging, follow these steps:

1. Log in as superuser to the NFS client.
2. To find a list of all the automounted directories on the client, run the following script:

   ```
   for FS in $(grep autofs /etc/mnttab | awk '{print $2}')
   do
     grep 'nfs' /etc/mnttab | awk '{print $2}' | grep ^${FS}
   done
   ```

3. For every automounted directory listed by the `grep` command, enter the following command to determine whether the directory is currently in use:

   ```
   /usr/sbin/fuser -cu local_mount_point
   ```

   This command lists the process IDs and user names of all the users who are using the mounted directory.

4. Warn users to exit the directory, and kill any processes that are using the directory, or wait until the processes terminate. Enter the following command to kill all the processes using the mounted directory:

   ```
   /usr/sbin/fuser -ck local_mount_point
   ```

5. Enter the following command to stop AutoFS:

   ```
   /sbin/init.d/autofs stop
   ```

6. Add `-v` to `AUTOMOUNTD_OPTIONS` variable in the `/etc/rc.config.d/nfsconf` file, as follows:

   ```
   AUTOMOUNTD_OPTIONS = "-v"
   ```

   This change enables AutoFS logging.

7. Enter the following commands to start AutoFS:

   ```
   /sbin/init.d/autofs start
   ```

To Stop AutoFS Logging

To stop AutoFS logging, stop AutoFS and restart it after removing the "`-v`" option from `AUTOMOUNTD_OPTIONS.`

AutoFS Tracing

AutoFS supports the following Trace levels:

Detailed (level 3)	Includes traces of all the AutoFS requests and replies, mount attempts, timeouts, and unmount attempts. You can start level 3 tracing while AutoFS is running.
Basic (level 1)	Includes traces of all the AutoFS requests and replies. You must restart AutoFS to start level 1 tracing.

To Start and Stop AutoFS Detailed Tracing

To start and stop the AutoFS tracing Level 3, follow these steps:

1. Log in as superuser to the NFS client.
2. Enter the following commands:

```
ps -ef | grep automountd
kill -SIGUSR2 PID
```

where:

`PID` Process ID returned by the `ps` command.

Level 3 tracing is appended to the `/var/adm/automount.log` file.

NOTE: The command, `kill` *-SIGUSR2 PID,* works only if tracing is not already on.

To stop level 3 tracing, enter the same commands listed above to send the `SIGUSR2` signal to `automountd`. The `SIGUSR2` signal is a toggle that turns tracing on or off depending on its current state.

If basic (level 1) tracing is turned on when you send the `SIGUSR2` signal to `automountd`, the `SIGUSR2` signal turns tracing off.

To Start AutoFS Basic Tracing

To start AutoFS tracing Level 1, follow these steps:

1. Log in as superuser to the NFS client.
2. Add `-T` to the `AUTOMOUNTD_OPTIONS` variable in the `/etc/rc.config.d/nfsconf` file, as follows:

   ```
   AUTOMOUNTD_OPTIONS="-T"
   ```

   This change appends AutoFS basic tracing messages into the `/var/adm/automount.log` file.
3. To find a list of all the automounted directories on the client, run the following script:

   ```
   for FS in $(grep autofs /etc/mnttab | awk '{print $2}')
   do
    grep 'nfs' /etc/mnttab | awk '{print $2}' | grep ^${FS}
   done
   ```
4. For each automounted directory listed by the `grep` command, enter the following command to determine whether the directory is currently in use:

   ```
   /usr/sbin/fuser -cu local_mount_point
   ```

This command lists the process IDs and user names of all users who are using the mounted directory.

5. Warn users to exit the directory, and kill processes that are using the directory, or wait until all the processes terminate. Enter the following command to kill all the processes using the mounted directory:

   `/usr/sbin/fuser -ck` *`local_mount_point`*

6. Enter the following command to stop AutoFS:

   `/sbin/init.d/autofs stop`

CAUTION: Do not kill the `automountd` daemon with the `kill` command. It does not unmount AutoFS mount-points before it dies.

---

7. Enter the following command to start AutoFS with tracing enabled:

   `/sbin/init.d/autofs start`

To Stop AutoFS Basic Tracing

To stop AutoFS tracing, kill AutoFS and restart it only after removing `-T` from `AUTOMOUNTD_OPTIONS`.

AutoFS Tracing Output

Following is a sample tracing output of the mounting and unmounting of a filesystem.

Mount Event Tracing Output

The general format of a mount event trace is: MOUNT REQUEST: `<time stamp>`

`<mount information>` ... `<other tracing>` ... `<other tracing>`

MOUNT REPLY: `<status>=mount status`The `<mount information>` trace has the following format (all on one line) :

`name=<key>[<subdirectory>] <map>= map name <opts>=mount options <path>=mount path <other tracing>`

where:

`<key>`= the key value from the map

`<subdirectory>` = subdirectory (may be blank)

`<map>` = name of map

`<opts>` = mount options

`<path>` = mount path

`<other tracing>` = other trace information

The `mount status` option in the mount reply contains 0 if the mount is successful; it has non-zero value when the mount is not successful.

The following is an example of a typical mount trace:

```
May 13 18:45:09 t5        MOUNT REQUEST:   Tue May 13 18:45:09 2003
May 13 18:45:09 t5        name=nfs127[/tmp] map=auto.indirect opts=path=/n2ktmp_8264/nfs127/tmp direct=1
May 13 18:45:09 t5        PUSH /etc/auto.indirect
May 13 18:45:09 t5        POP /etc/auto.indirect
May 13 18:45:09 t5        mapline:           hpnfs127:/         /tmp   hpnfs127:/tmp
May 13 18:45:09 t5        do_mount1:
May 13 18:45:09 t5        (nfs,nfs)     /n2ktmp_8264/nfs127/tmp                hpnfs127:/tmp   penalty=0
May 13 18:45:09 t5        nfsmount: input: hpnfs127[other]
May 13 18:45:09 t5        nfsmount: standard mount on/n2ktmp_8264/nfs127/tmp :
May 13 18:45:09 t5        hpnfs127:/tmpMay 13 18:45:09 t5        nfsmount: v3=1[0],v2=0[0] => v3.
May 13 18:45:09 t5        nfsmount: Get mount version: request vers=3min=3
May 13 18:45:09 t5        nfsmount: mount version=3
May 13 18:45:09 t5        Port numbers are 937, 937May 13 18:45:09 t5        Port match
May 13 18:45:09 t5        mount hpnfs127:/tmp /n2ktmp_8264/nfs127/tmp()
May 13 18:45:09 t5        nfs_args: hpnfs127, , 0x2004060, 0, 0, 0, 0,0, 0, 0, 0,
May 13 18:45:09 t5        args_temp: hpnfs127, , 0x3004060, 0, 0, 0, 0,0, 0, 0, 0, hpnfs127:/tmp
May 13 18:45:09 t5        mount hpnfs127:/tmp dev=44000004 rdev=0 OK
May 13 18:45:09 t5        MOUNT REPLY: status=0, AUTOFS_DONE
```

Unmount Event Tracing Output

The general format of an unmount event trace is:

UNMOUNT REQUEST:`<time stamp>`

`<other tracing>`

...

`<other tracing>`

UNMOUNT REPLY: `<status>=unmount status`

The `unmount status` in the unmount reply contains 0 if the unmount is successful; it has a non-zero value when the unmount is not successful.

The following is an example of a typical unmount trace event:

```
May 13 18:46:27 t1      UNMOUNT REQUEST: Tue May 13 18:46:27 2003
May 13 18:46:27 t1      dev=44000004 rdev=0 direct
May 13 18:46:27 t1      ping: hpnfs127 request vers=3 min=2
May 13 18:46:27 t1      pingnfs OK: nfs version=3
May 13 18:46:27 t1      nfsunmount: umount /n2ktmp_8264/nfs127/tmp
May 13 18:46:27 t1      Port numbers are 937, 937
May 13 18:46:27 t1      Port match
May 13 18:46:27 t1      nfsunmount: umount /n2ktmp_8264/nfs127/tmp OK
May 13 18:46:27 t1      unmount /n2ktmp_8264/nfs127/tmp OK
May 13 18:46:27 t1      UNMOUNT REPLY: status=0
```

4 Configuring and Administering a Cache Filesystem

This chapter introduces the Cache Filesystem (CacheFS) and the CacheFS environment. It also describes how to configure and administer CacheFS on a system running HP-UX 11i v3.

This chapter addresses the following topics:

CacheFS Overview

CacheFS is a general purpose filesystem caching mechanism that improves NFS client and server performance. CacheFS client performance is improved by caching data to a faster local filesystem instead of going over the wire to a slow server or on a slow network. This results in reduced server and network load because the clients send fewer requests to the server.

In an NFS environment, CacheFS:

- performs local disk caching of filesystems which enables the client systems to become less reliant on the server
- improves performance of clients on slow networks
- increases the number of clients that can be supported by a single server
- decreases the overall server load which can result in better server performance.

When data is read from a cached NFS-mounted filesystem for the first time, it results in some overhead when CacheFS writes the data to its local cache. After the data is written to the cache, read performance for the cached data improves significantly. CacheFS improves read performance for data that is read more than once. However, it does not improve NFS write performance. Therefore, good candidates for cached filesystems include manpages and executable programs, which are read multiple times though rarely modified.

By default, CacheFS maintains consistency between the cached filesystem and the remote filesystem, using a consistency checking model similar to that of standard NFS (polling for changes in file attributes).

NOTE: CacheFS cannot be used with NFSv4.

CacheFS Terms

The following CacheFS terms are used in this chapter:

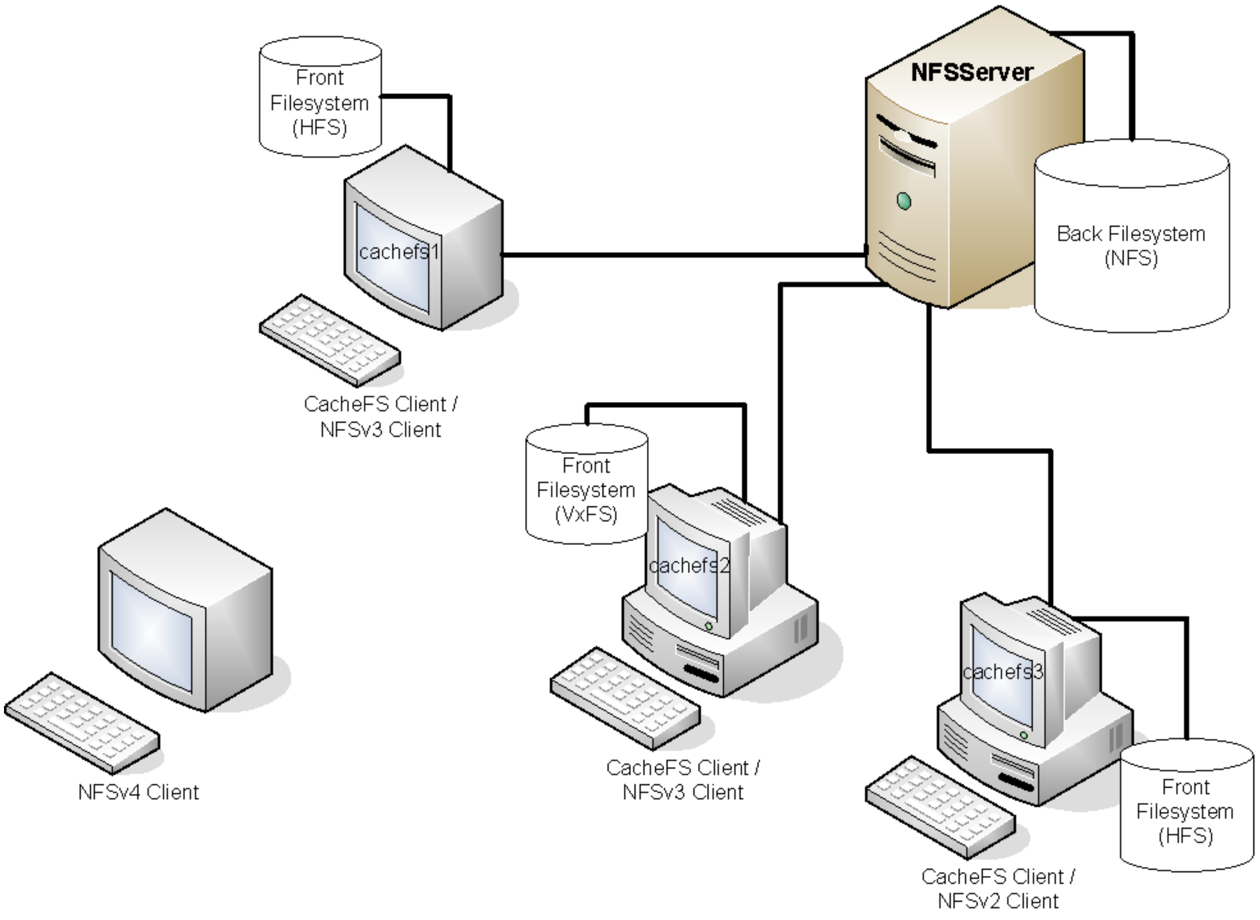
back filesystem	A filesystem that is cached is called a back filesystem. HP-UX currently supports only NFS as the back filesystem.
front filesystem	A local filesystem that is used to store the cached data is called a front filesystem. HFS and VxFS are the only supported front filesystem types.
cold cache	A cache that has not yet been populated with data from the back filesystem is called cold cache.
cache miss	An attempt to reference data that is not yet cached is called a cache miss.

warm cache	A cache that contains data in its front filesystem is called a warm cache. In this case, the cached data can be returned to the user without requiring an action from the back filesystem.
cache hit	A successful attempt to reference data that is cached is called a cache hit.

How CacheFS Works

Figure 4-1 displays a sample CacheFS environment.

Figure 4-1 Sample CacheFS Network



In the figure, `cachefs1`, `cachefs2`, and `cachefs3` are CacheFS clients. The figure also displays an NFSv4 client which is not a CacheFS client because CacheFS cannot be used with NFSv4. The NFS server, `NFSServer`, shares files and the CacheFS clients mount these files as the back filesystem.

When a user on any of the CacheFS clients, say `cachefs1`, accesses a file that is part of the back filesystem, portions of the file that were accessed are placed in the local cache.

Features of CacheFS

This section discusses the features that CacheFS supports on systems running HP-UX 11i v3.

- Cache Pre-Loading via the "cachefspack" Command

  The `cachefspack` command enables you to pre-load or pack specific files and directories in the cache, thereby improving the effectiveness of CacheFS. It also ensures that current copies of these files are always available in the cache. Packing files and directories in the cache enables you to have greater control over the cache contents.

The functionality provided by this command is an alternative to the `rpages` mount option. For information on how to pre-load or pack files, see "Packing a Cached Filesystem" (page 87).

- Complete Binary Caching via the "rpages" mount option

  CacheFS is commonly used to manage application binaries. The `rpages` mount option forces the client to cache a complete copy of the accessed application binary file. Whenever an application in a cached NFS filesystem is executed, CacheFS checks if the binary file exists in the local cache. If it does not exist, the client reads the entire binary file and automatically caches the binary file. If the client reads only a part of the binary file, the entire binary is automatically cached. For information on how to cache a complete binary, see "Caching a Complete Binary" (page 87).

- Multiple CacheFS Write Modes

  CacheFS supports two write modes, `write-around` and `non-shared`. In the `write-around` mode, writes are made to the back filesystem and the affected file is purged from the cache.

  You can use the `non-shared` mode to write to both the front and the back filesystems. All writes are made to both the front and the back filesystem and the file remains in the cache. To change the CacheFS write mode from `write-around` to the `non-shared`, you have to first unmount the CacheFS filesystem and then mount it using the `non-shared` option.

- Multiple Cache Consistency Checking Modes

  CacheFS periodically checks the files that are stored in the cache to ensure that the data in these files are kept up to date. CacheFS provides different consistency checking modes, including:

  — `demandconst`

    Verifies or checks for consistency on demand or only when explicitly requested.

    NOTE: For information on how to force a cache consistency check, see "Forcing a Cache Consistency Check" (page 89).

  — `noconst`

    Disables consistency checking. Use this option only if you know that the contents of the back filesystem are rarely modified.

  — `weakconst`

    Verifies cache consistency with the NFS client's copy of the file attributes.

NOTE: Consistency is not checked at file open time.

- Switching Mount Options

  You can also switch between mount options without deleting or rebuilding the cache. For example, you can switch from `default` to `non-shared`, or from `noconst` to `demandconst` mount options without recreating the cache. For information on how to switch mount options, see "Switching Mount Options" (page 85).

- Support for Large Files and Large Filesystems

  CacheFS supports the maximum file and filesystem sizes supported by the underlying front filesystem and the back filesystem.

  CacheFS data structures are 64-bit compliant.

- Support for ACLs

  An Access Control List (ACL) offers stronger file security by enabling the owner of the file to define file permissions for specific users and groups. This version of CacheFS on HP-UX supports ACLs with VxFS and NFS and not with HFS.

- Support for Logging

  A new command, `cachefslog`, is used to enable or disable logging for a CacheFS mount-point. If logging functionality is enabled, details about the operations performed on the CacheFS mount-point are stored in a logfile. The logfile also contains information on all the operations performed on all CacheFS mount-points that use the same cache directory. The `cachefswssize` command is used to display the amount of space taken by each of the filesystems in the same cache and the total size occupied by the cache directory (also known as the working set size). This command uses the logfile created by the `cachefslog` command to display the information.

Configuring and Administering CacheFS

You can use CacheFS to cache both manually mounted NFS filesystems or automatically mounted NFS filesystems. All CacheFS operations, except displaying CacheFS statistics, require superuser permissions.

This section describes the tasks to configure and administer CacheFS:

- "Creating a CacheFS Cache" (page 84)
- "Mounting an NFS Filesystem Using CacheFS" (page 85)
- "Automounting a Filesystem Using CacheFS" (page 86)
- "Enabling Logging in CacheFS" (page 86)
- "Disabling Logging in CacheFS" (page 87)
- "Caching a Complete Binary" (page 87)
- "Packing a Cached Filesystem" (page 87)
- "Forcing a Cache Consistency Check" (page 89)
- "Unmounting a Cache Filesystem" (page 89)
- "Checking the Integrity of a Cache" (page 89)
- "Deleting a Cache Directory" (page 91)

Creating a CacheFS Cache

This section describes how to configure a cache directory in a local filesystem. To configure a local filesystem as a cache directory, follow these steps:

1. Log in to the NFS client system as superuser.
2. Ensure that the disk partition containing the cache has enough space for the cache directory. If does not have enough space, configure and mount a new HFS or VxFS filesystem to be used as the front filesystem, where data will be cached.
3. To create a cache directory, enter the following command:

   ```
   cfsadmin -c cache_directory
   ```

   For example to create a CacheFS directory called `/disk2/cache` using the following command:

   ```
   cfsadmin -c /disk2/cache
   ```

   This creates a new directory called `cache` under the `/disk2` directory.

CacheFS allows more than one filesystem to be cached in the same cache. You need not create a separate cache directory for each CacheFS mount.

Mounting an NFS Filesystem Using CacheFS

This section describes how to mount an NFS filesystem using CacheFS. The syntax for mounting an NFS filesystem using CacheFS is as follows:

```
mount [-F cachefs] [-rqOV] -o backfstype=file_system_type
   [specific_options]  resource mount_point
```

Consider the following example where the /opt/frame directory is going to be NFS-mounted from the NFS server nfsserver to the local /opt/cframe directory.

To mount the example NFS filesystem using CacheFS manually, enter the following command on an NFS client system:

```
mount -F cachefs -o backfstype=nfs, \
cachedir=/disk2/cache    nfsserver:/opt/frame /opt/cframe
```

The /opt/frame directory can now be accessed like any other mounted filesystem. When data in /opt/cframe is referenced, it is copied into /disk2/cache.

To mount an NFS filesystem using CacheFS automatically at system boot, add the following line to the /etc/fstab file:

```
nfsserver:/opt/frame /opt/cframe cachefs \
backfstype=nfs,cachedir=/disk2/cache 0 0
```

When data in the /opt/cframe directory is accessed for the first time, the requested data is retrieved across the network from the NFS server. A copy of this data is then placed in the local cache directory. All future accesses to the data will be served from the local cache, assuming the data has not been modified on the server.

When a CacheFS filesystem is mounted, a file or CacheId is created in the cache directory for it. The name of the file is of the form:

```
NFS_server:mounted_directory:mount-point
```

Where each "/" is replaced by "_". The CacheId is unique for each CacheFS mount-point and is a link into the part of a cache directory assigned to that mount-point.

For example:

```
NFSserver      : nfss04
Mounted directory : /exp
Mount-point     : /mnt
symbolic name    : nfss04:_exp:_mnt
```

Users can explicitly specify the CacheId using the `cache ID` mount option. If you do not specify a cache ID, CacheFS constructs one.

Switching Mount Options

CacheFS is commonly used when the data on the server is read a number of times and rarely modified. For example, consider a filesystem that is initially mounted with the noconst option. The mount option has been chosen because the data on the server is rarely modified. If for some reason the data on the server is modified, you may want to change the mount options. In earlier versions of HP-UX, if you mounted a filesystem with certain mount options, you could not change those mount options without deleting or recreating the cache. Starting with HP-UX 11i v3, you can switch mount options without deleting and recreating the cache.

To change the mount option from noconst to the default option, without deleting or rebuilding the cache, enter the following commands:

```
umount /mnt1

mount -F cachefs -o backfstype=nfs,cachedir=/cache
CFS1:/tmp  /mnt1
```

To change the mount option from default to weakconst after unmounting, enter the following command:

```
mount -F cachefs -o backfstype=nfs,cachedir=/cache,weakconst
CFS2:/tmp  /mnt1
```

For more information on the various mount options of the CacheFS filesystem, see *mount_cachefs*(1M).

Automounting a Filesystem Using CacheFS

This section describes how to automount a filesystem using CacheFS.

Before you automount an NFS filesystem using CacheFS, you must configure a directory in a local filesystem as cache. For more information on how to configure a directory as cache, see "Creating a CacheFS Cache" (page 84).

To automount a filesystem using CacheFS, follow these steps:

1. Add a line to the appropriate AutoFS direct or indirect map, as in the following examples:

   Example 1

   Direct map example:

   ```
   /tmp/dist -nosuid,fstype=cachefs,backfstype=nfs, \
   cachedir=/disk2/cache  distserver:/export/dist
   ```

   Example 2

   Indirect map example:

   ```
   proj1  -nosuid,fstype=cachefs,backfstype=nfs, \
                         cachedir=/disk2/cache   \
       /src  testbox1:/export/proj1/src      \
          /data  testbox2:/export/proj1/data
   ```

2. If you have modified a direct map, enter the following command on each NFS client that uses the map, to force AutoFS to reread its maps:

   ```
   /usr/sbin/automount
   ```

NOTE: If you have modified an indirect map, the changes are read immediately. AutoFS need not be restarted and forced to reread it maps.

You can specify caching in an NIS AutoFS map, or by using LDAP, only if all the clients who use the map have their caching directory set up in the same location (`/disk2/cache` in the examples).

Enabling Logging in CacheFS

This section describes how you can enable logging in CacheFS. You can use the `cachefslog` command to enable logging for a CacheFS mount-point. Enabling the logging functionality may have a performance impact on the operations performed for all the CacheFS mount-points that are using the same cache directory.

To enable CacheFS logging, follow these steps:

1. Log in as superuser.
2. To enable logging for a CacheFS mount-point, say `/cfs_mnt1`, enter the following command:

   ```
   cachefslog -f /tmp/logfile /cfs_mnt1
   ```

   Where:

/tmp/logfile	Specifies the logfile to be used.

NOTE: When multiple mount-points use the same cache directory, enabling logging for one CacheFS mount-point automatically enables logging for all the other mount-points.

3. To verify if logging is enabled for `/cfs_mnt1`, enter the following command:

```
cachefslog /cfs_mnt1
```

If logging has been enabled, the logfile is displayed.

Disabling Logging in CacheFS

You can use the `cachefslog` command to halt or disable logging for a CacheFS mount-point.

To disable CacheFS logging, follow these steps:

1. Log in as superuser.
2. To halt or disable logging for the same CacheFS mount-point, `/cfs_mnt1`, enter the following command:

   ```
   cachefslog -h /cfs_mnt1
   ```

   NOTE: When multiple mount-points use the same cache directory, disabling logging for one CacheFS mount-point automatically disables logging for all the other mount-points.

3. To verify if logging has been halted or disabled for `/cfs_mnt1`, enter the following command:

   ```
   cachefslog /cfs_mnt1
   ```

   If logging has not been enabled or if it has been halted, the following message is displayed:

   ```
   not logged: /cfs_mnt1
   ```

Caching a Complete Binary

CacheFS is designed to work best with NFS filesystems that contain stable read-only data. One of the most common uses of CacheFS is managing application binaries. These are typically read-only and are rarely ever modified. They are modified when new versions of the application are installed or a patch containing a modified binary is installed. The `rpages` mount option enables you to cache a complete binary file.

When an application in a cached NFS filesystem is executed, CacheFS checks if the binary file is cached. If it is not cached, then the client reads the entire binary file and automatically caches it. If the `rpages` mount option is not used, only the accessed portion of the binary is cached. Using the `rpages` mount option causes slower initial load times, but subsequent executions of the application binary are significantly faster.

For example, to load the netscape binary, do the following:

1. To create a cache, if it does not exist, enter the following command:

   ```
   cfsadmin -c  /cachedir
   ```

2. To mount the cache filesystem, enter the following command:

   ```
   mount -F cachefs -o backfstype=nfs,rpages,cachedir=/cachedir \
   nfsserver:/opt/netscape  /opt/netscape
   ```

3. To run the netscape binary in `/opt/netscape`, enter the following command:

   ```
   /opt/netscape/netscape
   ```

   The netscape binary and the files that netscape needs are cached and populated in the cache directory `/cachedir`.

For more information on the `rpages` mount option, see *mount_cachefs*(1M).

Packing a Cached Filesystem

Starting with HP-UX 11i v3, the `cachefspack` command is introduced to provide greater control over the cache. This command enables you to specify files and directories to be loaded, or packed, in the cache. It also ensures that the current copies are always available in the cache.

You can pack files using one of the following methods:

- Specifying the files or directories to be packed

  Enter the following command to pack a file in the cache:

  ```
  cachefspack -p filename
  ```

  where:

`-p`	Packs the file or files in the cache.
`filename`	Name of the file or directory that is to be packed in the cache.

  NOTE: When you pack a directory, all files in that directory, subdirectories, and files in the subdirectories are packed. For instance, consider a directory `/dir1` that contains two subdirectories `/subdir1`, and `/subdir2`, as well as two files `test1`, and `test2`. When you pack `/dir1`, `/subdir1`, `/subdir2`, `test1`, and `test2` are packed.

- Using the `packing-list` file

  A `packing-list` file is an ASCII file that contains a list of files and directories that are to be pre-packed in the cache. Creating a `packing-list` file saves time, because you need not individually pack each file. It also enables you to access the `BASE`, `LIST`, and `IGNORE` options that are not available from the command-line interface. The `packing-list` file can be updated or removed when required.

  To pack files in the cache using the `packing-list` file, follow these steps:

  1. Create the `packing-list` file, if it does not exist. The file can be created using a text editor.

     NOTE: For information on the format and rules governing the creation of a `packing-list` file, see *packingrules*(4).

  2. Add an entry to the file. A sample of the `packing-list` file consists of the following entries:

     ```
     BASE /net/server/share/home/casey
     LIST work
     LIST m/sentitems
     IGNORE core *.o *.bak
     ```

     where:

`BASE`	Specifies the path to the directory that contains the files to pack.
`LIST`	Specifies the files to pack within the directory.
`IGNORE`	Specifies the files or file types or both that you do not want to pack in the cache.

  3. To pack the files specified in the `packing-list` file enter the following command:

     ```
     cachefspack -f packing-list
     ```

     where:

`packing-list`	Specifies the name of the `packing-list` file.

     The files specified in the packing list are now packed in the cache.

You can unpack files that you no longer require, using one of the following methods:

- Using the `-u` option

  To unpack a specific packed file or files from the cache directory, enter the following command:

  ```
  cachefspack -u filename
  ```

  where:

`-u`	Specifies that certain files are to be unpacked.
`filename`	Specifies the file to unpack.

- Using the `-U` option

  To unpack all the packed files in the cache, enter the following command:

  ```
  cachefspack -U cache-dir
  ```

  where:

`-U`	Specifies that you want to unpack all the packed files in the cache.
`cache-dir`	Specifies the cache directory that is to be unpacked.

  For more information about the `cachefspack` command, see *cachefspack*(1M).

Forcing a Cache Consistency Check

CacheFS periodically checks the consistency of files when a user attempts to access the file stored in the cache. This ensures that the cached directories and files are up to date with the back filesystem. To check consistency, CacheFS compares the time stamp of the cached file with the time stamp of the corresponding file on the back filesystem. If CacheFS detects a time difference, the cached data is purged, and the updated data is retrieved from the back filesystem. If you have not accessed any files, checks are not performed. Use of this option does not result in a sudden “storm” of consistency checks.

Forcing a Consistency Check for a Specific Mount-Point

By default, CacheFS verifies the consistency of the cached contents against the back filesystem every 30 seconds. However, if you want to disable these automatic consistency checks and instead perform manual consistency checks you can mount the CacheFS filesystem with the `demandconst` option, and then use the `cfsadmin` command to force a consistency check. To force a consistency check on a specific mount-point, enter the following command:

```
cfsadmin -s mount_point
```

IMPORTANT: The `-s` option works only if CacheFS is mounted with the `demandconst` option. For information and an example on how to switch between mount options, see “Switching Mount Options” (page 85)

Forcing a Consistency Check for all the Mount-Points

To request for a consistency check on all the mount-points, enter the following command:

```
cfsadmin -s all
```

For information on `cfsadmin` options, see *cfsadmin*(1M).

Unmounting a Cache Filesystem

To unmount a Cache filesystem, enter the following command:

```
umount mount-point
```

where:

`mount-point`	Specifies the CacheFS mount-point that you want to unmount.

Checking the Integrity of a Cache

You can use the `fsck` command to check the integrity of a cache. The CacheFS version of the `fsck` command checks the integrity of the cache and automatically corrects any CacheFS problems that it encounters. The CacheFS `fsck` command is run automatically by the `mount` command when the cache directory is accessed for the first time after a system reboot. The command is run either during system bootup if there is an entry in `/etc/fstab`, or the first time the cache directory is referenced as part of the mount operation.

To manually check the integrity of a cache, enter the following command:

```
fsck_cachefs -F cachefs [-m | -o noclean]
cache-directory
```

where:

`-m`	Specifies that the cache must be checked without making any repairs.
`noclean`	Forces a check on the CacheFS filesystems.
`cache-directory`	Specifies the name of the directory where the cache resides.

The cache directory must not be in use while performing a cache integrity check using the `fsck` command. To list the CacheFS mount-points that are using a specific cache directory, for example, `/disk2/cache`, enter the following command:

```
mount | grep -w "cachedir=/disk2/cache" | awk '{print $1}
```

An output similar to the following is displayed if CacheFS mount-points are using the cache directory:

```
/cfs_mnt1
/cfs_mnt2
```

You must now unmount these mount-points before you check the integrity of a cache. For information on how to unmount a cache mount-point, see "Unmounting a Cache Filesystem" (page 89).

For more information on the `fsck_cachefs` command of CacheFS, see *fsck_cachefs*(1M).

Updating Resource Parameters

Each cache has a set of parameters that determines its structure and how it behaves. When a cache directory is created, it gets created with default values for the resource parameters. Table 4-1 lists the various resource parameters and their default values.

Table 4-1 CacheFS Resource Parameters

CacheFS Resource Parameter	Default Value
maxblocks	90
minblocks	0
threshblocks	85
maxfiles	90
minfiles	0
threshfiles	85

For more information on the resource parameters, see *cfsadmin*(1M).

You can update the resource parameters using the `-u` option of the `cfsadmin` command.

NOTE: All filesystems in the cache directory must be unmounted when you use the `-u` option. Changes will be effective the next time you mount any filesystem in the specified cache directory.

For example, to update the resource parameter, *maxblocks*, you must first create the cache, and then modify the parameter as follows:

```
cfsadmin -c /cache
cfsadmin -u -o maxblocks=95 /cache
```

This updates the value of *maxblocks* from 90 which is the default value to 95.

Deleting a Cache Directory

To delete a cache directory that is no longer required you must use the `cfsadmin` command. The syntax to delete the cache directory is as follows:

```
cfsadmin -d {cacheID | all} cache-directory
```

where:

cacheID	Specifies the name of the cache filesystem.
all	Specifies that all cached filesystems in the `cache-directory` are to be deleted.
cache-directory	Specifies the name of the cache directory where the cache resides.

NOTE: The cache directory must not be in use when attempting to delete a cached filesystem or the cache directory.

To delete the cache directory, follow these steps:

1. To identify the Cache ID of each of the cached filesystems, enter the following command:

   ```
   cfsadmin -l cache-directory
   ```

   An output similar to the following is displayed:

   ```
   cfsadmin: list cache FS information
   maxblocks     90%
   minblocks      0%
   threshblocks  85%
   maxfiles      91%
   minfiles       0%
   threshfiles   85%
   maxfilesize    3MB
   srv01:_tmp:_mnt
   srv01:_tmp:_mnt1
   ```

   At the end of the output, the Cache IDs of all the cached filesystems that are using this cache directory are displayed. In this example, the`/tmp` directory of `srv01` is mounted on `mnt` and `mnt1`.

2. To list the CacheFS mount-points that are using the cache directory, enter the following command:

   ```
   mount | grep -w "cachedir=/disk2/cache" | awk '{print $1}'
   ```

   An output similar to the following is displayed if CacheFS mount-points are using the cache directory:

   ```
   /mnt
   /mnt1
   ```

3. To unmount the CacheFS mount-points, `/_mnt`, and `/_mnt1`, enter the following commands:

   ```
   umount /mnt
   umount /mnt1
   ```

4. To delete the CacheFS filesystem corresponding to the Cache ID from the specified cache directory, enter the following command:

   ```
   cfsadmin -d CacheID cache-directory
   ```

5. To verify if the CacheFS filesystem is deleted, enter the following command:

   ```
   cfsadmin -l cache-directory
   ```

   An output similar to the following is displayed:

   ```
   cfsadmin: list cache FS information
   maxblocks     90%
   minblocks      0%
   threshblocks  85%
   ```

```
maxfiles       91%
minfiles        0%
threshfiles    85%
maxfilesize     3MB
srv01:_tmp:_mnt1
```

In this example, the filesystem with CacheID `srv01:_tmp:_mnt` filesystem has been deleted.

To delete a cache directory and all the CacheFS filesystems in that directory, enter the following command:

```
cfsadmin -d all cache-directory
```

Using CacheFS

After the administrator has configured a cache directory and mounted or automounted the CacheFS filesystem, you can use CacheFS as you would any other filesystem.

The first time you access data through CacheFS, an over-the-wire call is made to the NFS server and the data is copied to your local cache. The first request is slow. However, all subsequent accesses are faster as they are served from the local cache.

Displaying CacheFS Statistics

The `cachefsstat` command displays statistical information about the mounted cache filesystem. This includes the number of cache hits, cache misses, consistency checks, and modification operations, such as writes and creates. The `cachefswssize` command displays the working set size. The working set size includes information on the amount of cache space that each filesystem in a cache directory uses. It can also be used to display the contents of the logfile. The logfile created when you enable logging is used as an input to calculate the working set size.

Viewing the CacheFS Statistics

In the following example, `/home/smh` is the CacheFS mount-point. To view the CacheFS statistics, enter the following command:

```
cachefsstat /home/smh
```

An output similar to the following is displayed:

```
/home/smh
cache hit rate:    20% (2543 hits, 9774 misses)
consistency checks:  47842 (47825 pass, 17 fail)
modifies:  85727
garbage collection:      0
```

You can run the `cachefsstat` command with the `-z` option to reinitialize CacheFS statistics. You must be a superuser to use the `-z` option.

NOTE: If you do not specify the CacheFS mount-point, statistics for all CacheFS mount-points are displayed.

For more information about the `cachefsstat` command, see *cachefsstat*(1M).

Viewing the Working Set (Cache) Size

In the following example, `/cfs_mnt1` is the CacheFS mount-point and `/tmp/logfile` refers to the logfile created when you enable logging. To view the memory usage, enter the following command:

```
cachefswssize /tmp/logfile
```

An output similar to the following is displayed:

```
/cfs_mnt1
    end size:    40k
 high water size:    40k
```

```
total for cache
  Initial size:     640k
     end size:      40k
```

To view the operations performed in ASCII format, enter the following command:

```
cachefswssize -a /tmp/logfile
```

An output similar to the following is displayed:

```
1/0  0:00  0 Mount    e000000245e31080  411 65536 256
/cfs_mnt1 (cachefs1:_cache_exp:_cfs_mnt1)
1/0  0:00  0 Mdcreate e000000245e31080  <fid> 517500609495040 1
1/0  0:00  0 Filldir e000000245e31080  <fid> 517500609495040 8192
1/0  0:00  0 Nocache e000000245e31080  <fid> 517500609495040
1/0  0:00  0 Mkdir   e000000245e31080  <fid> 517264386293760 0
```

Where:

_cfs_mnt1	Specifies the CacheFS mount-point
cachefs1	Specifies the server name
_cache_exp	Specifies the mounted directory which is displayed as `/cache_exp`

Common Problems While Using CacheFS

This section discusses the common problems you may encounter while using CacheFS. It also describes steps to overcome these problems. The following tables list the error messages, their cause, and how to resolve these errors.

Table 4-2 Common Error Messages encountered while using the `cfsadmin` command

Error Message	Possible Causes	Resolution
“cfsadmin: Cannot create lock file `/test/mnt/c/.cfs_lock`”	Indicates that you do not have write permissions to the filesystem, where the cache directory is being created.	**1.** Delete the cache. **2.** Recreate the cache directory using the `cfsadmin` command.
“mount failed No space left on device” (There could be a message in syslog that states: "WARNING: cacheFS: cache not clean. Run fsck") .	The Cache directory may not be clean.	**1.** Enter the `fsck` command. **2.** Try mount operation.
(There could be a message in syslog that states: “WARNING: cacheFS: not enough space to create <fscache directory name>).	The value of the cache parameters, used to create the cache directory, must be modified.	**1.** Increase the CacheFS parameters using the `-u` option with the `cfsadmin` command. For more information, see *cfsadmin*(1M).

Table 4-3 Common Error Messages encountered while using the `fsck` command

Error Message	Possible Causes	Resolution
"`fsck -F cachefs` Cannot open lock file `/test/c/.cfs_lock`"	This error message indicates that `/c` is not a cache directory.	1. Delete the cache. 2. Recreate the cache directory using the `cfsadmin` command.
"Cache `/c` is in use and cannot be modified".	This error message indicates that the cache directory is in use. However, the actual reason for this error may be that the cache directory does not exist, or that`/c` is not a cache directory.	1. Check if a directory named `/c` exists and also if it is a cache directory. 2. If it is not a cache directory, delete the directory. 3. If it is a cache directory, run `fsck` on the cache directory. 4. If the problem still exists, use a different cache directory.
"Cache `/c` is in use."	This error message indicates that the filesystem is mounted.	1. Unmount the filesystem. 2. Run the `fsck` command on the cache directory.

Table 4-4 Common Error Messages encountered while using the `mount` command

Error Message	Possible Causes	Resolution
"mount -F cachefs: `/test/c/.cfs_mnt_points` is not a valid cache"	The `/c` directory may not be a valid cache directory.	1. Delete the cache. 2. Recreate the cache directory using the `cfsadmin` command.

Table 4-5 Common Error Messages encountered while using the `cachefsstat` command

Error Message	Possible Causes	Resolution
"cachefsstat: Cannot zero statistics"	Only a superuser can zero-out the statistics. The user who used this command is not a superuser.	1. Log out. 2. Log in as superuser and enter the `cachefsstat` command.

5 Troubleshooting NFS Services

This chapter describes tools and procedures for troubleshooting the NFS Services. This chapter addresses the following topics:

Common Problems with NFS

This section lists the following common problems encountered with NFS and suggests ways to correct them.

NFS "Server Not Responding" Message

- □ Enter the `/usr/sbin/ping` command on the NFS client to make sure the NFS server is up and is reachable on the network. If the `ping` command fails, either the server is down, or the network has a problem. If the server is down, reboot it, or wait for it to come back up. For more information on troubleshooting network problems, see *HP-UX LAN Administrator's Guide*.
- □ Ensure that the following daemons are running on the NFS server:
  - `rpc.mountd`
  - `rpc.statd`
  - `rpc.lockd`
- □ Enter the following command on the NFS client to make sure the server is running all the NFS server processes:

  ```
  /usr/bin/rpcinfo -p servername
  ```

  The `rpcinfo` command should display the following services:
  - `rpcbind`
  - `nfs`
  - `mountd`
  - `status`
  - `nlockmgr`

  On the NFS server, check if the following processes are running:
  - `nfsd`
  - `rpc.mountd`
  - `rpc.statd`
  - `rpc.lockd`

  If any of these processes is not running, follow these steps:

1. Make sure the `/etc/rc.config.d/nfsconf` file on the NFS server contains the following lines:

   ```
   NFS_SERVER=1
   START_MOUNTD=1
   ```

2. Enter the following command on the NFS server to start all the necessary NFS processes:

   ```
   /sbin/init.d/nfs.server start
   ```

- ☐ Enter the following command on the NFS client to make sure the `rpc.mountd` process on the NFS server is available and responding to RPC requests:

  ```
  /usr/bin/rpcinfo -u servername mountd
  ```

  If the `rpcinfo` command returns `RPC_TIMED_OUT`, the `rpc.mountd` process may be hung. Enter the following commands on the NFS server to restart `rpc.mountd` (*`PID`* is the process ID returned by the `ps` command) :

  ```
  /usr/bin/ps -ef | /usr/bin/grep mountd
  /usr/bin/kill PID/usr/sbin/rpc.mountd
  ```

- ☐ You can receive "`server not responding`" messages when the server or network is heavily loaded and the RPC requests are timing out.

NOTE: For TCP, the default timeout is 600 while for UDP, the default timeout is 11.

  Try doubling the `timeo` mount option for the directory, as in the following example from the `/etc/fstab` file, which changes the `timeo` value from 600 (the default) to 1200. (The `timeo` option is in tenths of a second.)

  ```
  cabbage:/usr /usr nfs nosuid,timeo=1200 0 0
  ```

- ☐ Enter the following command on the NFS client to check that your `hosts` database returns the correct address for the NFS server:

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

  Enter the same `nslookup` command on the NFS server, and compare the address with the one returned by the `nslookup` command on the NFS client. If they are different, correct your NIS, BIND, or `/etc/hosts` configuration. For information on BIND or `/etc/hosts`, see *Installing and Administering Internet Services* (B2355-91060).

- ☐ If you are using AutoFS, enter the `ps -ef` command to make sure the `automountd` process is running on your NFS client. If it is not, follow these steps:

  1. Make sure the `AUTOFS` variable is set to 1 in the `/etc/rc.config.d/nfsconf` file on the NFS client.

     ```
     AUTOFS=1
     ```

  2. Enter the following command on the NFS client to start the AutoFS:

     ```
     /sbin/init.d/autofs start
     ```

"Access Denied" Message

- ☐ Enter the following command on the NFS client to check that the NFS server is exporting the directory you want to mount:

  ```
  /usr/sbin/showmount -e server_name
  ```

  If the server is not exporting the directory, edit the `/etc/dfs/dfstab` file on the server so that it allows your NFS client access to the directory. Then, enter the following command to force the server to read its `/etc/dfs/dfstab` file.

  ```
  shareall -F nfs
  ```

If the directory is shared with the [access_list] option, make sure your NFS client is included in the [access_list], either individually or as a member of a netgroup.

- ◻ Enter the following commands on the NFS server to make sure your NFS client is listed in its hosts database:

  ```
  nslookup client_name
  nslookup client_IP_address
  ```

“Permission Denied” Message

- ◻ Check the mount options in the /etc/fstab file on the NFS client. A directory you are attempting to write to may have been mounted as read-only.
- ◻ Enter the ls -l command to check the HP-UX permissions on the server directory and on the client directory that is the mount-point. You may not be allowed access to the directory.
- ◻ Enter the following command on the NFS server:

  ```
  /usr/sbin/share
  ```

  Or, enter the following command on the NFS client:

  ```
  /usr/sbin/showmount -e server_name
  ```

  Check the permissions on the shared directory. The directory may have been shared as read-only to your client. The administrator of the NFS server can use the remount mount option to mount the directory read/write without unmounting it. See “Changing the Default Mount Options” (page 40)“Changing the Default Mount Options” on page 51.

  If you are logged in as root to the NFS client, check the share permissions to determine whether root access to the directory is granted to your NFS client.

- ◻ If you are logged in as root to the NFS client, and your client is not allowed root access to the exported directory, check the passwd database on the NFS server to determine whether it contains an entry for user nobody. Without root access, the root user on an NFS client is given the access permissions of an anonymous user. Also, check whether anonymous users are denied access to the directory (with the anon=65535 export option).

  If your client is not allowed root access or anonymous user ID access to the exported directory, log in as a non-root user to get access to the directory.

- ◻ If you were attempting to run a program when you received the “permission denied” message, enter the ls -l command on the NFS server to check whether the program you tried to run has the setuid bit set. If it does, check /etc/fstab to determine whether the directory was mounted with the nosuid mount option. If necessary, remove the nosuid option from the /etc/fstab file, then unmount and remount the directory.

“Device Busy” Message

- ◻ If you received the “device busy” message while attempting to mount a directory, check whether it is already mounted.
- ◻ If you received the “device busy” message while attempting to unmount a directory, a user or process is currently using the directory. Wait until the process completes, or follow these steps:
  1. **1.** Enter the following command to determine who is using the mounted directory:

     ```
     /usr/sbin/fuser -cu local_mount_point
     ```

     The fuser(1M) command will return a list of process IDs and user names that are currently using the directory mounted under *local_mount_point*. This will help you decide whether to kill the processes or wait for them to complete.

  2. **2.** To kill all processes using the mounted directory, enter the following command:

```
/usr/sbin/fuser -ck local_mount_point
```

3. Try again to unmount the directory.

- Verify that the filesystem you are trying to unmount is not a mount-point of another filesystem.
- Verify that the filesystem is not exported. In HP-UX 11i v3, an exported filesystem keeps the filesystem busy.

"Stale File Handle" Message

A "`stale file handle`" occurs when one client removes an NFS-mounted file or directory that another client is accessing. The following sequence of events explains how it occurs:

Table 5-1 Stale File Handle Sequence of Events

	NFS client 1	NFS client 2
1	`% cd /proj1/source`	
2		`% cd /proj1`
3		`% rm -Rf source`

4	`% ls`
	`.:Stale File Handle`	

If a server stops exporting a directory that a client has mounted, the client will receive a `stale file handle` error. Stale file handles also occur if you restore the NFS server's file systems from a backup or randomize the inode numbers with `fsirand`(1M).

- If the stale file handle occurred because someone removed a file or directory that was in use, or because a server stopped exporting a directory that was in use, follow these steps:
  1. Enter the `/usr/bin/cd` command to move out of the NFS-mounted directory that is causing the problem, then try unmounting the directory:
     ```
     /usr/bin/cd /..
     /usr/sbin/umount directory
     ```
  2. If the directory cannot be unmounted because it is busy (in use), enter the following commands to kill the processes using the directory and to try again to unmount it:
     ```
     /usr/sbin/fuser -ck local_mount_point
     /usr/sbin/umount local_mount_point
     ```
  3. If the directory still cannot be unmounted, reboot the client.
  4. To avoid stale file handles caused by users deleting NFS-mounted files, try using a source code control system, like Revision Control System (RCS). A source code control system allows only one user at a time to modify a file or directory, so one user cannot remove files another user is accessing. For more information on the source code control system, see *rcsintro*(5).
- If someone has restored the server's file systems from backup or entered the `fsirand` command on the server, follow these steps on each of the NFS clients to prevent stale file handles by restarting NFS:
  1. Enter the `mount`(1M) command with no options, to get a list of all the mounted file systems on the client:
     ```
     /usr/sbin/mount
     ```
  2. For every NFS-mounted directory listed by the `mount` command, enter the following command to determine whether the directory is currently in use:
     ```
     /usr/sbin/fuser -cu local_mount_point
     ```
     This command lists the process IDs and user names of everyone using the mounted directory.

3. Warn any users to `cd` out of the directory, and kill any processes that are using the directory, or wait until the processes terminate. Enter the following command to kill all processes using the directory:

   ```
   /usr/sbin/fuser -ck local_mount_point
   ```

4. Enter the following command on the client to unmount all NFS-mounted directories:

   ```
   /usr/sbin/umount -aF nfs
   ```

5. Enter the following commands to restart the NFS client:

   ```
   /sbin/init.d/nfs.client stop
   /sbin/init.d/nfs.client start
   ```

A Program Hangs

- ☐ Check whether the NFS server is up and operating correctly. If you are not sure, see "NFS "Server Not Responding" Message" (page 95).

  If the server is down, wait until it comes back up, or, if the directory was mounted with the `intr` mount option (the default), you can interrupt the NFS mount, usually with `CTRL-C`.

- ☐ If the program uses file locking, enter the following commands (on either the client or the server) to make sure `rpc.statd` and `rpc.lockd` are available and responding to RPC requests:

  ```
  /usr/bin/rpcinfo -u servername status
  /usr/bin/rpcinfo -u servername nlockmgr
  /usr/bin/rpcinfo -u clientname status
  /usr/bin/rpcinfo -u clientname nlockmgr
  ```

  If any of these commands return `RPC_TIMED_OUT`, the `rpc.statd` or `rpc.lockd` process may be hung. Follow these steps to restart `rpc.statd` and `rpc.lockd` daemons:

  1. Enter the following commands, on both the NFS client and the NFS server, to kill `rpc.statd` and `rpc.lockd` (*PID* is a process ID returned by the `ps` command):

     ```
     /usr/bin/ps -ef | /usr/bin/grep rpc.statd
     /usr/bin/kill PID
     /usr/bin/ps -ef | /usr/bin/grep rpc.lockd
     /usr/bin/kill PID
     ```

  2. Enter the following commands to restart `rpc.statd` and `rpc.lockd` on both the client and the server:

     ```
     /usr/sbin/rpc.statd
     /usr/sbin/rpc.lockd
     ```

     NOTE: Always start `rpc.statd` before starting `rpc.lockd`.

  3. Enter the following commands to verify that `rpc.statd`, `rpc.lockd`, and `nfsd` are all running and responding to RPC requests:

     ```
     /usr/bin/rpcinfo -u servername status
     /usr/bin/rpcinfo -u servername nlockmgr
     /usr/bin/rpcinfo -u servername nfs
     /usr/bin/rpcinfo -u clientname status
     /usr/bin/rpcinfo -u clientname nlockmgr
     /usr/bin/rpcinfo -u clientname nfs
     ```

  4. Before retrying the mount that caused the program to hang, wait for a short while, say two minutes.
  5. If the problem persists, restart `rpc.statd` and `rpc.lockd` daemons and enable tracing.

Data is Lost Between the Client and the Server

- ☐ Make sure that the directory is not exported from the server with the `async` option. If the directory is exported with the `async` option, the NFS server will acknowledge NFS writes before actually writing data to disk.
- ☐ If users or applications are writing to the NFS-mounted directory, make sure it is mounted with the `hard` option (the default), rather than the `soft` option.
- ☐ If you have a small number of NFS applications that require absolute data integrity, add the `O_SYNC` flag to the `open()` calls in your applications. When you open a file with the `O_SYNC` flag, a `write()` call will not return until the write request has been sent to the NFS server and acknowledged. The `O_SYNC` flag degrades write performance for applications that use it.
- ☐ If multiple NFS users are writing to the same file, add the `lockf()` call to your applications to lock the file so that only one user may modify it at a time.

  If multiple users on different NFS clients are writing to the file, you must also turn off attribute caching on those clients by mounting the file with the `noac` mount option. Turning off attribute caching degrades NFS performance.

For more information, see *mount*(1M), *open*(2),*write*(2), and *lockf*(2).

“Too Many Levels of Remote in Path” Message

This message indicates that you are attempting to mount a directory from a server that has NFS-mounted the directory from another server. You cannot “chain” your NFS mounts this way. You must mount the directory from the server that has mounted its directory on a local disk.

Common Problems while using Secure NFS with Kerberos

“Permission Denied” Message

This message could be displayed because of one of the following reasons:

- The Ticket Granting Ticket (TGT) has expired

  To renew the ticket, enter the following command:

  `kinit` *`username`*

- Fully qualified hostname resolution problem

  To verify the hostname resolution, check the following files:

  — `/etc/nsswitch.conf`
  — `/etc/hosts`

  To provide a fully qualified host name, do the following:

  — Add `dns` in the host entry in the `/etc/nsswitch.conf`
  — Re-configure NIS and `/etc/hosts`

- Time mismatch of 5 minutes between Kerberos server and Kerberos client

  HP recommends that you run time server to synchronize the time between client and server.

- Improper `krb5.conf`

  This could be because the realm to domain matching is not set in either server or client’s configuration file (`krb5.conf`).

  To fix the `krb5.conf` file for proper domain name to realm matching, modify the file based on the following sample:

```
#
# Kerberos configuration
# This krb5.conf file is intended as an example only.
# see krb5.conf(4) for more details
# hostname is the fully qualified hostname(FQDN) of host on
which kdc is running
# domain_name is the fully qualified name of your domain
[libdefaults]
     default_realm = krbhost.anyrealm.com
     default_tkt_enctypes = DES-CBC-CRC
     default_tgs_enctypes = DES-CBC-CRC
     ccache_type = 2
[realms]
     krbhost.anyrealm.com = {
            kdc = krbhost.anyrealm.com:88
            admin_server = krbhost.anyrealm.com
        }
[domain_realm]
.anyrealm.com = krbhost.anyrealm.com
[logging]
        kdc = FILE:/var/log/krb5kdc.log
        admin_server = FILE:/var/log/kadmin.log
        default = FILE:/var/log/krb5lib.log
```

- The user who is trying to access the mounted filesystem has not obtained a TGT using their login.

  For example, if you are a guest user and are attempting to access the NFS mounted filesystem with Kerberos security option, you need to have a TGT.

  To identify the default principal name, enter the following command:

  ```
  klist
  ```

  If the default principal name is not 'guest', enter the following command to obtain a TGT for the guest principal:

  ```
  kinit guest
  ```

Performance Tuning

This section gives suggestions for identifying performance problems in your network and improving NFS performance on your servers and clients. This section addresses the following topics:

- "Diagnose NFS Performance Problems" (page 101)
- "Improve NFS Server Performance" (page 102)
- "Improving NFS Client Performance" (page 103)

Diagnose NFS Performance Problems

1. Enter the following command on several of your NFS clients:

   ```
   nfsstat -rc
   ```

2. If the `timeout` and `retrans` values displayed by `nfsstat -rc` are high, but the `badxid` value is close to zero, packets are being dropped before they get to the NFS server.

   Try decreasing the values of the `wsize` and `rsize` mount options to 4096 or 2048 on the NFS clients. See "Changing the Default Mount Options" (page 40)"Changing the Default Mount Options" on page 51 .

See *Installing and Administering LAN/9000 Software* for information on troubleshooting LAN problems.

3. If the `timeout` and `badxid` values displayed by `nfsstat -rc` are of the same magnitude, your server is probably slow. Client RPC requests are timing out and being retransmitted before the NFS server has a chance to respond to them.

   Try doubling the value of the `timeo` mount option on the NFS clients. See "Changing the Default Mount Options" (page 40)"Changing the Default Mount Options" on page 51.

Improve NFS Server Performance

- ☐ Enter the following command to check your server's memory utilization:

  ```
  netstat -m
  ```

  If the number of `requests for memory denied` is high, your server does not have enough memory, and NFS clients will experience poor performance. Consider adding more memory or using a different host as the NFS server.

- ☐ Put heavily used directories on different disks on your NFS servers so they can be accessed in parallel.
- ☐ Enter the following command on the NFS server:

  ```
  vmstat -n
  ```

  If the `us` and `sy` values under `cpu` are high, and the `id` (idle time) value under `cpu` is close to zero, your server's CPU is heavily loaded. Try using a faster machine as your NFS server. Do not use a gateway or a terminal server as an NFS or NIS server.

- ☐ Enter the following command to determine which processes are using the most CPU:

  ```
  /usr/bin/top
  ```

  The `top` program sorts the processes running on your system, with the most CPU-intensive process at the top of the display. It refreshes the display every five seconds. Try taking some CPU-intensive processes off the server.

  Type `q` to exit the `top` program.

- ☐ Log into the NFS server and enter the following command:

  ```
  nfsstat -s
  ```

  If the number of `readlink` calls is of the same magnitude as the number of `lookup` calls, you have a symbolic link in a filesystem that is frequently traversed by NFS clients.

  On the NFS clients that require access to the linked directory, mount the target of the link. Then, remove the link from the exported directory on the server.

  When a client requests access to a linked file or directory, two requests are sent to the server: one to look up the path to the link, and another to look up the target of the link. You can improve NFS performance by removing symbolic links from exported directories.

- ☐ If the value of `getattr` displayed by `nfsstat -s` is greater than 60%, one or more clients have either turned off attribute caching (with the `noac` mount option) or set the caching timeout values too low.

  Increase the attribute caching timeouts on the clients that have them set below the default values. See "Changing the Default Mount Options" (page 40)"Changing the Default Mount Options" on page 51.

- ☐ Share directories with the `async` option. When `async` is specified, the server acknowledges write requests from clients before writing data to a disk. Clients do not have to wait for a write request to complete before issuing another request. This can be performed only for NFSv2. The default option for NFSv3 is async.

Improving NFS Client Performance

- ☐ For files and directories that are mounted read-only and never change, set the `actimeo` mount option to 120 or greater in the `/etc/fstab` file on your NFS clients.
- ☐ If you see several "`server not responding`" messages within a few minutes, try doubling the value of the `timeo` mount option in the `/etc/fstab` file on your NFS clients.
- ☐ If you frequently see the following message when attempting access to a soft-mounted directory,

  ```
  NFS operation failed for server servername: Timed out
  ```

  try increasing the value of the `retrans` mount option in the `/etc/fstab` file on the NFS clients. Or, change the soft mount to an interruptible hard mount, by specifying the `hard` and `intr` options (the defaults).

- ☐ Enter the following command on the NFS server, to find out the block size of the server's filesystem:

  ```
  /usr/sbin/tunefs -v devicefilename
  ```

  On the NFS clients, set the `wsize` and `rsize` mount options to the `bsize` value displayed by `tunefs`.

- ☐ On the NFS clients, look in the `/etc/fstab` file for "stepping-stone" mounts (hierarchical mounts), as in the following example:

  ```
  thyme:/usr /usr nfs defaults 0 0
  basil:/usr/share /usr/share nfs defaults 0 0
  sage:/usr/share/lib /usr/share/lib nfs defaults 0 0
  ```

  Wherever possible, change these "stepping-stone" mounts so that whole directories are mounted from a single NFS server.

  Stepping-stone (hierarchical) mounts, like the one in the example above, cause more NFS requests than mounts from a single server. In the example, if a client wants access to something in `/usr/share/lib`, a request must be sent to server `thyme`, then to server `basil`, and finally to server `sage`.

  For more information on the stepping-stone mounts, see "Changing the Default Mount Options" (page 40).

Logging and Tracing of NFS Services

This section tells you how to start the following tools:

- "AutoFS Logging"
- "AutoFS Tracing"
- "Logging for the Other NFS Services"
- "Logging With `nettl` and `netfmt`"
- "Tracing With `nettl` and `netfmt`"

Logging for the Other NFS Services

You can configure logging for the following NFS services:

- `rpc.rexd`
- `rpc.rstatd`
- `rpc.rusersd`
- `rpc.rwalld`
- `rpc.sprayd`

Logging is not available for the `rpc.quotad` daemon.

Each message logged by these daemons can be identified by the date, time, host name, process ID, and name of the function that generated the message. You can direct logging messages from all these NFS services to the same file.

To Control the Size of LogFiles

Logfiles grow without bound, using up disk space. You might want to create a `cron` job to truncate your logfiles regularly. Following is an example `crontab` entry that empties the logfile at 1:00 AM every Monday, Wednesday, and Friday:

```
0 1 * * 1,3,5 cat /dev/null > log_file
```

For more information, type `man 1M cron` or `man 1 crontab` at the HP-UX prompt.

To Configure Logging for the Other NFS Services

1. Add the `-l` *`logfile`* option to the lines in `/etc/inetd.conf` for the services you want to log. In the following example, logging is turned on for `rpc.rexd` and `rpc.rstatd`:

   ```
   rpc xit tcp nowait root /usr/sbin/rpc.rexd 100017 1 \
   rpc.rexd -l /var/adm/rpc.log

   rpc dgram udp wait root /usr/lib/netsvc/rstat/rpc.rstatd \
   100001 1-3 rpc.rstatd -l /var/adm/rpc.log
   ```

2. Enter the following command to restart `inetd`:

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

If you do not specify a logfile for the other NFS services (with the `-l` option), they do not log any messages. The NFS services can all share the same logfile.

For more information, see *rexd*(1M)*, rstatd*(1M)*, rusersd*(1M)*, rwalld*(1M) , and *sprayd*(1M).

Logging With `nettl` and `netfmt`

1. Enter the following command to make sure `nettl` is running:

   ```
   /usr/bin/ps -ef | grep nettl
   ```

   If `nettl` is not running, enter the following command to start it:

   ```
   /usr/sbin/nettl -start
   ```

2. Enter the following command to start logging:

   ```
   /usr/sbin/nettl -l i w e d -e all
   ```

   The logging classes are specified following the `-l` option. They are `i` (informational), `w` (warning), `e` (error), and `d` (disaster). Disaster logging is always on. You cannot turn it off. Information logging (`i`) fills up your logfile faster than the other classes, so you might want to leave it off.

3. Recreate the event you want to log.
4. Enter the following command to turn logging off:

   ```
   /usr/sbin/nettl -l d -e all
   ```

   This command changes the logging class back to disaster only for all subsystems.

5. Enter the following command to format the binary logfile:

   ```
   /usr/sbin/netfmt -lN -f /var/adm/nettl.LOG00 > formatted_file
   ```

   where *`formatted_file`* is the name of the file where you want the formatted output from `netfmt`. The default logfile, `/var/adm/nettl.LOG`*nn*, is specified in the `nettl` configuration file, `/etc/nettlgen.conf`. If the file `/var/adm/nettl.LOG00` does not exist on your system, the default logfile may have been changed in `/etc/nettlgen.conf`.

For more information, see *nettl*(1M) and *netfmt*(1M).

Tracing With `nettl` and `netfmt`

1. Enter the following command to make sure `nettl` is running:

   ```
   /usr/bin/ps -ef | grep nettl
   ```

   If `nettl` is not running, enter the following command to start it:

   ```
   /usr/sbin/nettl -start
   ```

2. Enter the following command to start tracing:

   ```
   /usr/sbin/nettl -tn pduin pduout loopback -e all -s 1024 \
   -f tracefile
   ```

3. Recreate the event you want to trace.
4. Enter the following command to turn tracing off:

   ```
   /usr/sbin/nettl -tf -e all
   ```

5. Create the following filter file for `netfmt`:

   ```
   filter ip_saddr remote_host_IP_address
   filter ip_daddr remote_host_IP_address
   filter rpcprogram nfs
   filter rpcprogram nlockmgr
   filter rpcprogram llockmgr
   filter rpcprogram status
   filter rpcprogram mount
   filter rpcprogram rpcbind
   ```

   `remote_host_IP_address` is the IP address of the host with which your host was communicating when the event you want to trace occurred.

6. Enter the following command to format the binary trace file:

   ```
   /usr/sbin/netfmt -c filter_file -lN -f tracefile.TRC0 > formatted_file
   ```

   where *`tracefile`* is the name of the file you specified when you started tracing, and *`formatted_file`* is the name of the file where you want the formatted output from `netfmt`.

For more information, type `man 1M nettl` or `man 1M netfmt`.

Index