Managing Network File Systems in Oracle® Solaris 11.1

Part No: E28997–01
October 2012

ORACLE

121010@25097

Contents

Figures

Tables

Examples

Preface

Managing Network File Systems in Oracle Solaris 11.1 is part of a multivolume set that covers a significant part of the Oracle Solaris system administration information. This book assumes that you have already installed the Oracle Solaris operating system, and you have set up any networking software that you plan to use.

Note – This Oracle Solaris release supports systems that use the SPARC and x86 families of processor architectures. The supported systems appear in the *Oracle Solaris OS: Hardware Compatibility Lists*. This document cites any implementation differences between the platform types.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.

TABLE P–1 Typographic Conventions *(Continued)*

Typeface	Description	Example
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows UNIX system prompts and superuser prompts for shells that are included in the Oracle Solaris OS. In command examples, the shell prompt indicates whether the command should be executed by a regular user or a user with privileges.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

◆ ◆ ◆ CHAPTER 1

Managing Network File Systems (Overview)

This chapter provides an overview of the NFS service, which can be used to access file systems over the network. The chapter includes a discussion of the concepts necessary to understand the NFS service and a description of the latest features in NFS and autofs.

Note – If your system has zones enabled and you want to use this feature in a non-global zone, see *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management* for more information.

What's New With the NFS Service

This section provides information about new features in releases of the Oracle Solaris OS.

Significant Changes in This Release

The Oracle Solaris 11.1 release includes the following enhancements:

- A new property has been added to the `/network/nfs/server:default` service which controls the amount of information that the `showmount` command displays to remote clients. For more information see Example 2–3 and "`showmount` Command" on page 110.
- Support for FedFS referrals has been added. This permits referral information for several servers to be centralized in LDAP. See "Administering FedFS" on page 63 for more information.

Significant Changes in Earlier Releases

The following enhancements are included in the Oracle Solaris 11 release:

- The configuration parameters that used to be set by editing the `/etc/default/autofs` and `/etc/default/nfs` can now be set in the Service Management Facility (SMF) repository. See the descriptions of the new SMF parameters in procedures that use them, as well as the descriptions of the daemons that use them:
  - "`automount` Command" on page 94
  - "`automountd` Daemon" on page 83
  - "`lockd` Daemon" on page 84
  - "`mountd` Daemon" on page 84
  - "`nfsd` Daemon" on page 85
  - "`nfsmapid` Daemon" on page 86
- The NFS service provides support for mirror mounts. Mirror mounts enable an NFSv4 client to traverse shared file system mount points in the server namespace. For NFSv4 mounts, the automounter will perform a mount of the server namespace root and rely on mirror mounts to access its file systems. The main advantage that mirror mounts offer over the traditional automounter is that mounting a file system using mirror mounts does not require the overhead associated with administering automount maps. Mirror mounts provide these features:
  - Namespace changes are immediately visible to all clients.
  - New shared file systems are discovered instantly and mounted automatically.
  - File systems unmount automatically after a designated inactivity period.

  For more information about mirror mounts, refer to the following:
  - "How to Mount All File Systems from a Server" on page 35
  - "How Mirror Mounts Work" on page 140
- NFS referrals have been added to the NFS service. Referrals are server-based redirections that an NFSv4 client can follow to find a file system. The NFS server supports referrals created by the `nfsref(1M)`command, and the NFSv4 client will follow them to mount the file system from the actual location. This facility can be used to replace many uses of the automounter, with creation of referrals replacing the editing of automounter map. NFS referrals provide these features:
  - All of the features of mirror mounts listed above
  - Automounter-like functionality without any dependence on the automounter.
  - No setup required at either the client or server.

  For more information about NFS referrals, see:
  - "Administering NFS Referrals" on page 61
  - "How NFS Referrals Work" on page 141

- The ability to mount the per-DNS-domain root of a Federated File System name space has been added. This mount point can be used with NFS referrals to bridge from one file server to another, building an arbitrarily large namespace. For more information see:
  - "Setting up a DNS Record for a FedFS Server" on page 38
  - "Mount Point `/nfs4`" on page 144
- The `sharectl` utility is included. This utility enables you to configure and manage file sharing protocols, such as NFS. For example, this utility allows you to set client and server operational properties, display property values for a specific protocol, and obtain the status of a protocol. For more information, see the `sharectl`(1M) man page and "`sharectl` Command" on page 102.
- The way an NFS version 4 domain can be defined has changed since the initial Solaris 10 release. See "Configuring an NFS Version 4 Default Domain in the Oracle Solaris 11 Release" on page 91 for more information.

NFS Terminology

This section presents some of the basic terminology that must be understood to work with the NFS service. Expanded coverage of the NFS service is included in Chapter 3, "Accessing Network File Systems (Reference)."

NFS Servers and Clients

The terms *client* and *server* are used to describe the roles that a computer assumes when sharing file systems. Computers that share their file systems over a network are acting as servers. The computers that are accessing the file systems are said to be clients. The NFS service enables any computer to access any other computer's file systems. A computer can assume the role of client, server, or both client and server at any particular time on a network.

Clients access files on the server by mounting the server's shared file systems. When a client mounts a remote file system, the client does not make a copy of the file system. Rather, the mounting process uses a series of remote procedure calls that enable the client to access the server's shared file system transparently. The mount resembles a local mount. Users type commands as if the file systems were local. See "Mounting File Systems" on page 33 for information about tasks that mount file systems.

After a file system has been shared on a server through an NFS operation, the file system can be accessed from a client. You can mount an NFS file system automatically with autofs. See "Automatic File System Sharing" on page 30 and "Task Overview for Autofs Administration" on page 48 for tasks that involve the `share` command and autofs.

NFS File Systems

The objects that can be shared with the NFS service include any whole or partial directory tree or a file hierarchy, including a single file. A computer cannot share a file hierarchy that overlaps a file hierarchy that is already shared. Peripheral devices such as modems and printers cannot be shared.

In most UNIX system environments, a file hierarchy that can be shared corresponds to a file system or to a portion of a file system. However, NFS support works across operating systems, and the concept of a file system might be meaningless in other, non-UNIX environments. Therefore, the term *file system* refers to a file or file hierarchy that can be shared and be mounted with NFS.

About the NFS Service

The NFS service enables computers of different architectures that run different operating systems to share file systems across a network. NFS support has been implemented on many platforms that range from the MS-DOS to the VMS operating systems.

The NFS environment can be implemented on different operating systems because NFS defines an abstract model of a file system, rather than an architectural specification. Each operating system applies the NFS model to its file-system semantics. This model means that file system operations such as reading and writing function as though the operations are accessing a local file.

The NFS service has the following benefits:

- Enables multiple computers to use the same files so that everyone on the network can access the same data
- Reduces storage costs by having computers share applications instead of needing local disk space for each user application
- Provides data consistency and reliability because all users can read the same set of files
- Makes mounting of file systems transparent to users
- Makes accessing of remote files transparent to users
- Supports heterogeneous environments
- Reduces system administration overhead

The NFS service makes the physical location of the file system irrelevant to the user. You can use the NFS implementation to enable users to see all the relevant files regardless of location. Instead of placing copies of commonly used files on every system, the NFS service enables you to share the original file from the NFS server's file system. All other systems access the files across the network. Under NFS operation, remote file systems are almost indistinguishable from local file systems.

About Autofs

File systems that are shared through the NFS service can be mounted by using automatic mounting. Autofs, a client-side service, is a file-system structure that provides automatic mounting. The autofs file system is initialized by `automount`, which is run automatically when a system is booted. The automount daemon, `automountd`, runs continuously, mounting and unmounting remote directories as necessary.

Whenever a client computer that is running `automountd` tries to access a remote file or remote directory, the daemon mounts the remote file system. This remote file system remains mounted for as long as needed. If the remote file system is not accessed for a certain period of time, the file system is automatically unmounted.

Mounting need not be done at boot time, and the user no longer has to know the superuser password to mount a directory. Users do not need to use the `mount` and `umount` commands. The autofs service mounts and unmounts file systems as required without any intervention by the user.

Mounting some file hierarchies with `automountd` does not exclude the possibility of mounting other hierarchies with `mount`. A diskless computer *must* mount `/` (root), `/usr`, and `/usr/kvm` through the `mount` command and the `/etc/vfstab` file.

"Task Overview for Autofs Administration" on page 48 and "How Autofs Works" on page 147 give more specific information about the autofs service.

Features of the NFS Service

This section describes the important features that are included in the NFS service.

NFS Version 2 Protocol

Version 2 was the first version of the NFS protocol in wide use. Version 2 continues to be available on a large variety of platforms. All Oracle Solaris releases support version 2 of the NFS protocol.

NFS Version 3 Protocol

Unlike the NFS version 2 protocol, the NFS version 3 protocol can handle files that are larger than 2 Gbytes. The previous limitation has been removed. See "NFS Large File Support" on page 25.

The NFS version 3 protocol enables safe asynchronous writes on the server, which improve performance by allowing the server to cache client write requests in memory. The client does

not need to wait for the server to commit the changes to disk, so the response time is faster. Also, the server can batch the requests, which improves the response time on the server.

Many Solaris NFS version 3 operations return the file attributes, which are stored in the local cache. Because the cache is updated more often, the need to do a separate operation to update this data arises less often. Therefore, the number of RPC calls to the server is reduced, improving performance.

The process for verifying file access permissions has been improved. Version 2 generated a "write error" message or a "read error" message if users tried to copy a remote file without the appropriate permissions. In version 3, the permissions are checked before the file is opened, so the error is reported as an "open error."

The NFS version 3 protocol removed the 8-Kbyte transfer size limit. Clients and servers could negotiate whatever transfer size the clients and servers support, rather than conform to the 8-Kbyte limit that version 2 imposed. Note that in earlier Solaris implementations, the protocol defaulted to a 32-Kbyte transfer size. Starting in the Solaris 10 release, restrictions on wire transfer sizes are relaxed. The transfer size is based on the capabilities of the underlying transport.

NFS Version 4 Protocol

NFS version 4 has features that are not available in the previous versions.

The NFS version 4 protocol represents the user ID and the group ID as strings. `nfsmapid` is used by the client and the server to do the following:

- To map these version 4 ID strings to a local numeric IDs
- To map the local numeric IDs to version 4 ID strings

For more information, refer to "`nfsmapid` Daemon" on page 86.

Note that in NFS version 4, the ID mapper, `nfsmapid`, is used to map user or group IDs in ACL entries on a server to user or group IDs in ACL entries on a client. The reverse is also true. For more information, see "ACLs and `nfsmapid` in NFS Version 4" on page 128.

With NFS version 4, when you unshare a file system, all the state for any open files or file locks in that file system is destroyed. In NFS version 3 the server maintained any locks that the clients had obtained before the file system was unshared. For more information, refer to "Unsharing and Resharing a File System in NFS Version 4" on page 120.

NFS version 4 servers use a pseudo file system to provide clients with access to exported objects on the server. Prior to NFS version 4 a pseudo file system did not exist. For more information, refer to "File-System Namespace in NFS Version 4" on page 121.

In NFS version 2 and version 3 the server returned persistent file handles. NFS version 4 supports volatile file handles. For more information, refer to "Volatile File Handles in NFS Version 4" on page 123.

Delegation, a technique by which the server delegates the management of a file to a client, is supported on both the client and the server. For example, the server could grant either a read delegation or a write delegation to a client. For more information, refer to "Delegation in NFS Version 4" on page 126.

NFS version 4 does not support the LIPKEY/SPKM security flavor.

Also, NFS version 4 does not use the following daemons:

- `lockd`
- `nfslogd`
- `statd`

For a complete list of the features in NFS version 4, refer to "Features in NFS Version 4" on page 120.

For procedural information that is related to using NFS version 4, refer to "Setting Up NFS Services" on page 39.

Controlling NFS Versions

The SMF repository includes parameters to control the NFS protocols that are used by both the client and the server. For example, you can use parameters to manage version negotiation. For more information, refer to "`mountd` Daemon" on page 84 for the client parameters, "`nfsd` Daemon" on page 85 for the server parameters, or the `nfs(4)` man page.

NFS ACL Support

Access control list (ACL) support was added in the Solaris 2.5 release. An access control list (ACL) provides a finer-grained mechanism to set file access permissions than is available through standard UNIX file permissions. NFS ACL support provides a method of changing and viewing ACL entries from a Oracle Solaris NFS client to a Oracle Solaris NFS server.

The NFS version 2 and version 3 implementations support the old POSIX-draft style ACLs. POSIX-draft ACLs are natively supported by UFS. See "Using Access Control Lists to Protect UFS Files" in *Oracle Solaris 11.1 Administration: Security Services* for more information about UFS ACLs.

The NFS Version 4 protocol supports the new NFSv4 style ACLs. NFSv4 ACLs are natively supported by ZFS. For full featured NFSv4 ACL functionality, ZFS must be used as the underlying file system on the NFSv4 server. The NFSv4 ACLs have a rich set of inheritance properties, as well as a set of permission bits beyond the standard read, write and execute. See Chapter 7, "Using ACLs and Attributes to Protect Oracle Solaris ZFS Files," in *Oracle Solaris 11.1 Administration: ZFS File Systems* for an overview of the new ACLs. For more information about support for ACLs in NFS version 4, see "ACLs and `nfsmapid` in NFS Version 4" on page 128.

NFS Over TCP

The default transport protocol for the NFS protocol was changed to the Transport Control Protocol (TCP) in the Solaris 2.5 release. TCP helps performance on slow networks and wide area networks. TCP also provides congestion control and error recovery. NFS over TCP works with version 2, version 3, and version 4. Prior to the Solaris 2.5 release, the default NFS protocol was User Datagram Protocol (UDP).

Note – If InfiniBand hardware is available on the system the default transport changes from TCP to Remote Direct Memory Access (RDMA) protocol. For more information, see "NFS Over RDMA" on page 118. Note, however, that if you use the `proto=tcp` mount option, NFS mounts are forced to use TCP only.

NFS Over UDP

Starting in the Solaris 10 release, the NFS client no longer uses an excessive number of UDP ports. Previously, NFS transfers over UDP used a separate UDP port for each outstanding request. Now, by default, the NFS client uses only one UDP reserved port. However, this support is configurable. If the use of more simultaneous ports would increase system performance through increased scalability, then the system can be configured to use more ports. This capability also mirrors the NFS over TCP support, which has had this kind of configurability since its inception. For more information, refer to the *Oracle Solaris 11.1 Tunable Parameters Reference Manual*.

Note – NFS version 4 does not use UDP. If you mount a file system with the `proto=udp` option, then NFS version 3 is used instead of version 4.

Overview of NFS Over RDMA

If InfiniBand hardware is available on the system the default transport changes from TCP to Remote Direct Memory Access (RDMA) protocol. The RDMA protocol is a technology for memory-to-memory transfer of data over high speed networks. Specifically, RDMA provides remote data transfer directly to and from memory without CPU intervention. To provide this capability, RDMA combines the interconnect I/O technology of InfiniBand with the Oracle Solaris Operating System. For more information, refer to "NFS Over RDMA" on page 118.

Network Lock Manager and NFS

The network lock manager provides UNIX record locking for any files being shared over NFS. The locking mechanism allows clients to synchronize their I/O requests with each other, insuring data integrity.

Note – The Network Lock Manager is used only for NFS version 2 and version 3 mounts. File locking is built into the NFS version 4 protocol.

NFS Large File Support

The Solaris 2.6 implementation of the NFS version 3 protocol was changed to correctly manipulate files that were larger than 2 Gbytes. The NFS version 2 protocol could not handle files that were larger than 2 Gbytes.

NFS Client Failover

Dynamic failover of read-only file systems was added in the Solaris 2.6 release. Failover provides a high level of availability for read-only resources that are already replicated, such as man pages, other documentation, and shared binaries. Failover can occur anytime after the file system is mounted. Manual mounts can now list multiple replicas, much like the automounter in previous releases. The automounter has not changed, except that failover need not wait until the file system is remounted. See "How to Use Client-Side Failover" on page 36 and "Client-Side Failover" on page 132 for more information.

Kerberos Support for the NFS Service

The NFS service supports Kerberos V5 clients. The `mount` and `share` commands have been altered to support NFS version 3 mounts that use Kerberos V5 authentication. Also, the `share` command was changed to enable multiple authentication flavors for different clients. See "RPCSEC_GSS Security Flavor" on page 26 for more information about changes that involve security flavors. See "Configuring Kerberos NFS Servers" in *Oracle Solaris 11.1 Administration: Security Services* for information about Kerberos V5 authentication.

WebNFS Support

The Solaris 2.6 release also included the ability to make a file system on the Internet accessible through firewalls. This capability was provided by using an extension to the NFS protocol. One of the advantages to using the WebNFS protocol for Internet access is its reliability. The service is built as an extension of the NFS version 3 and version 2 protocol. Additionally, the WebNFS implementation provides the ability to share these files without the administrative overhead of an anonymous `ftp` site. See "Security Negotiation for the WebNFS Service" on page 26 for a description of more changes that are related to the WebNFS service. See "WebNFS Administration Tasks" on page 45 for more task information.

Note – The NFS version 4 protocol is preferred over the WebNFS service. NFS version 4 fully integrates all the security negotiation that was added to the MOUNT protocol and the WebNFS service.

RPCSEC_GSS Security Flavor

A security flavor, called RPCSEC_GSS, is supported in the Solaris 7 release. This flavor uses the standard GSS-API interfaces to provide authentication, integrity, and privacy, as well as enabling support of multiple security mechanisms. See "Kerberos Support for the NFS Service" on page 25 for more information about support of Kerberos V5 authentication. See *Developer's Guide to Oracle Solaris 11 Security* for more information about GSS-API.

Solaris 7 Extensions for NFS Mounting

The Solaris 7 release includes extensions to the `mount` command and `automountd` command. The extensions enable the mount request to use the public file handle instead of the MOUNT protocol. The MOUNT protocol is the same access method that the WebNFS service uses. By circumventing the MOUNT protocol, the mount can occur through a firewall. Additionally, because fewer transactions need to occur between the server and the client, the mount should occur faster.

The extensions also enable NFS URLs to be used instead of the standard path name. Also, you can use the `public` option with the `mount` command and the automounter maps to force the use of the public file handle. See "WebNFS Support" on page 25 for more information about changes to the WebNFS service.

Security Negotiation for the WebNFS Service

A new protocol has been added to enable a WebNFS client to negotiate a security mechanism with an NFS server in the Solaris 8 release. This protocol provides the ability to use secure transactions when using the WebNFS service. See "How WebNFS Security Negotiation Works" on page 135 for more information.

NFS Server Logging

In the Solaris 8 release, NFS server logging enables an NFS server to provide a record of file operations that have been performed on its file systems. The record includes information about which file was accessed, when the file was accessed, and who accessed the file. You can specify the location of the logs that contain this information through a set of configuration options.

You can also use these options to select the operations that should be logged. This feature is particularly useful for sites that make anonymous FTP archives available to NFS and WebNFS clients. See "How to Enable NFS Server Logging" on page 32 for more information.

Note – NFS version 4 does not support server logging.

Autofs Features

Autofs works with file systems that are specified in the local namespace. This information can be maintained in NIS or local files.

A fully multithreaded version of `automountd` is included. This enhancement makes autofs more reliable and enables concurrent servicing of multiple mounts, which prevents the service from hanging if a server is unavailable.

The `automountd` provides better on-demand mounting. Previous releases would mount an entire set of file systems if the file systems were hierarchically related. Now, only the top file system is mounted. Other file systems that are related to this mount point are mounted when needed.

The autofs service supports browsability of indirect maps. This support enables a user to see which directories could be mounted, without having to actually mount each file system. A `-nobrowse` option has been added to the autofs maps so that large file systems, such as `/net` and `/home`, are not automatically browsable. Also, you can turn off autofs browsability on each client by using the `-n` option with `automount`. See "Disabling Autofs Browsability" on page 59 for more information.

◆ ◆ ◆ CHAPTER 2

Network File System Administration (Tasks)

This chapter provides information about how to perform such NFS administration tasks as setting up NFS services, adding new file systems to share, and mounting file systems. The chapter also covers the use of the Secure NFS system and the use of WebNFS functionality. The last part of the chapter includes troubleshooting procedures and a list of some of the NFS error messages and their meanings.

Your responsibilities as an NFS administrator depend on your site's requirements and the role of your computer on the network. You might be responsible for all the computers on your local network, in which instance you might be responsible for determining these configuration items:

- Which computers should be dedicated servers
- Which computers should act as both servers and clients
- Which computers should be clients only

Maintaining a server after it has been set up involves the following tasks:

- Sharing and unsharing file systems as necessary
- Modifying administrative files to update the lists of file systems your computer mounts automatically
- Checking the status of the network
- Diagnosing and fixing NFS-related problems as they arise

- Setting up maps for autofs

Remember, a computer can be both a server and a client. So, a computer can be used to share local file systems with remote computers and to mount remote file systems.

Note – If your system has zones enabled and you want to use this feature in a non-global zone, see *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management* for more information.

Automatic File System Sharing

In the Oracle Solaris 11 release, the `share` command creates permanent shares that are automatically shared during system startup. Unlike previous releases, you will not need to edit the `/etc/dfs/dfstab` file to record the information about shares for subsequent reboots. The `/etc/dfs/dfstab` is no longer used.

TABLE 2–1 File-System Sharing Task Map

Task	Description	For Instructions
Establish automatic file system sharing	Steps to configure a server so that file systems are automatically shared when the server is rebooted	"How to Set Up Automatic File-System Sharing" on page 30
Enable WebNFS	Steps to configure a server so that users can access files by using WebNFS	"How to Enable WebNFS Access" on page 31
Enable NFS server logging	Steps to configure a server so that NFS logging is run on selected file systems	"How to Enable NFS Server Logging" on page 32

▼ How to Set Up Automatic File-System Sharing

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Define the file systems to be shared.

Use the `share` command to define each path to be shared. This information will be retained when a system is rebooted.

```
# share -F nfs -o specific-options pathname
```

See the `share_nfs(1M)` man page for a complete list of the *specific-options*.

3 **Verify that the information is correct.**

Run the `share` command to check that the correct options are listed:

```
# share -F nfs
export_share_man    /export/share/man   sec=sys,ro
export_ftp          /usr/src       sec=sys,rw=eng
usr_share_src       /export/ftp    sec=sys,ro,public
```

See Also The next step is to set up your autofs maps so that clients can access the file systems that you have shared on the server. For more information, see "Task Overview for Autofs Administration" on page 48.

How to Enable WebNFS Access

Note the following:

- By default all file systems that are available for NFS mounting are automatically available for WebNFS access. The only condition that requires the use of this procedure is one of the following:
  - To allow NFS mounting on a server that does not currently allow NFS mounting
  - To reset the public file handle to shorten NFS URLs by using the `public` option with the `share` command
  - To force a specific HTML file to be loaded by using the `index` option with the `share` command
- You can also use the `sharectl` utility to configure file-sharing protocols, such as NFS. See the `sharectl`(1M) man page and "`sharectl` Command" on page 102.

See "Planning for WebNFS Access" on page 46 for a list of issues to consider before starting the WebNFS service.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Define the file systems to be shared by the WebNFS service.**

Use the `share` command to define each file system. The `public` and `index` tags that are shown in the following example are optional.

```
# share -F nfs -o ro,public,index=index.html /export/ftp
```

See the `share_nfs`(1M) man page for a complete list of options.

3 **Verify that the information is correct.**

Run the `share` command to check that the correct options are listed:

```
# share -F nfs
export_share_man   /export/share/man   sec=sys,ro
usr_share_src      /usr/src     sec=sys,rw=eng
export_ftp        /export/ftp    sec=sys,ro,public,index=index.html
```

How to Enable NFS Server Logging

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **(Optional) Change file system configuration settings.**

In `/etc/nfs/nfslog.conf`, you can change the settings in one of two ways. You can edit the default settings for all file systems by changing the data that is associated with the `global` tag. Alternately, you can add a new tag for this file system. If these changes are not needed, you do not need to change this file. The format of `/etc/nfs/nfslog.conf` is described in the `nfslog.conf(4)` man page.

3 **Define the file systems to use NFS server logging.**

Use the `share` command to define each file system. The tag that is used with the `log=`*tag* option must be entered in `/etc/nfs/nfslog.conf`. This example uses the default settings in the `global` tag.

```
# share -F nfs -ro,log=global /export/ftp
```

4 **Verify that the information is correct.**

Run the `share` command to check that the correct options are listed:

```
# share -F nfs
export_share_man        /export/share/man   sec=sys,ro
usr_share_src           /usr/src     sec=sys,rw=eng
export_ftp              /export/ftp  public,log=global,sec=sys,ro
```

5 **Check if `nfslogd`, the NFS log daemon, is running.**

```
# ps -ef | grep nfslogd
```

6 **(Optional) Start `nfslogd`, if it is not running.**

```
# svcadm restart network/nfs/server:default
```

Mounting File Systems

You can mount file systems in several ways. File systems can be mounted automatically when the system is booted, on demand from the command line, or through the automounter. The automounter provides many advantages to mounting at boot time or mounting from the command line. However, many situations require a combination of all three methods. Additionally, several ways of enabling or disabling processes exist, depending on the options you use when mounting the file system. See the following table for a complete list of the tasks that are associated with file system mounting.

TABLE 2–2 Task Map for Mounting File Systems

Task	Description	For Instructions
Mount a file system at boot time	Steps so that a file system is mounted whenever a system is rebooted.	"How to Mount a File System at Boot Time" on page 34
Mount a file system by using a command	Steps to mount a file system when a system is running. This procedure is useful when testing.	"How to Mount a File System From the Command Line" on page 34
Mount with the automounter	Steps to access a file system on demand without using the command line.	"Mounting With the Automounter" on page 35
Mount a file system with mirror mounts	Steps to mount one or more file systems using mirror mounts.	Example 2–2
Mount all file systems with mirror mounts	Steps to mount all of the file systems from one server.	"How to Mount All File Systems from a Server" on page 35
Start client-side failover	Steps to enable the automatic switchover to a working file system if a server fails.	"How to Use Client-Side Failover" on page 36
Disable mount access for a client	Steps to disable the ability of one client to access a remote file system.	"How to Disable Mount Access for One Client" on page 36
Provide access to a file system through a firewall	Steps to allow access to a file system through a firewall by using the WebNFS protocol.	"How to Mount an NFS File System Through a Firewall" on page 37
Mount a file system by using an NFS URL	Steps to allow access to a file system by using an NFS URL. This process allows for file system access without using the MOUNT protocol.	"How to Mount an NFS File System Using an NFS URL" on page 37
Mount a FedFS file system	Process to establish a DNS record so that a FedFS file system can be accessed through the `/nfs4` mount point.	"Setting up a DNS Record for a FedFS Server" on page 38

How to Mount a File System at Boot Time

If you want to mount file systems at boot time instead of using autofs maps, follow this procedure. This procedure must be completed on every client that should have access to remote file systems.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Add an entry for the file system to `/etc/vfstab`.**

Entries in the `/etc/vfstab` file have the following syntax:

```
special  fsckdev  mountp  fstype  fsckpass  mount-at-boot  mntopts
```

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

Caution – NFS servers that also have NFS client `vfstab` entries must always specify the `bg` option to avoid a system hang during reboot. For more information, see "`mount` Options for NFS File Systems" on page 96.

Example 2–1 Entry in the Client's `vfstab` File

You want a client machine to mount the `/var/mail` directory from the server `wasp`. You want the file system to be mounted as `/var/mail` on the client and you want the client to have read-write access. Add the following entry to the client's `vfstab` file.

```
wasp:/var/mail - /var/mail nfs - yes rw
```

How to Mount a File System From the Command Line

Mounting a file system from the command line is often performed to test a new mount point. This type of mount allows for temporary access to a file system that is not available through the automounter.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Mount the file system.**

Type the following command:

```
# mount -F nfs -o ro bee:/export/share/local /mnt
```

In this instance, the `/export/share/local` file system from the server `bee` is mounted read-only on `/mnt` on the local system. Mounting from the command line allows for temporary viewing of the file system. You can unmount the file system with `umount` or by rebooting the local host.

Caution – All versions of the `mount` command do not warn about invalid options. The command silently ignores any options that cannot be interpreted. To prevent unexpected behavior, ensure that you verify all of the options that were used.

Example 2–2 Using Mirror Mounts After Mounting a File System

This release includes the mirror mount facility. This new mounting technology can be used from any NFSv4 client accessing a second file system from an NFSv4 server. Once the first file system is mounted from the server using either the `mount` command or the automounter, then any file systems that are added to that mount point may be accessed. All you have to do is try to access the file system. The mirror mount occurs automatically. For more information, see "How Mirror Mounts Work" on page 140.

Mounting With the Automounter

"Task Overview for Autofs Administration" on page 48 includes the specific instructions for establishing and supporting mounts with the automounter. Without any changes to the generic system, clients should be able to access remote file systems through the `/net` mount point. To mount the `/export/share/local` file system from the previous example, type the following:

```
% cd /net/bee/export/share/local
```

Because the automounter allows all users to mount file systems, `root` access is not required. The automounter also provides for automatic unmounting of file systems, so you do not need to unmount file systems after you are finished.

See Example 2–2 for information about how to mount additional file systems on a client.

▼ How to Mount All File Systems from a Server

This release includes the mirror mount facility, which allows a client to access all available file systems shared using NFS from a server, once one mount from that server has succeeded. For more information, see "How Mirror Mounts Work" on page 140.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Mount the root of the exported namespace of the server.**

This command mirrors the file system hierarchy from the server on the client. In this case, a `/mnt/export/share/local` directory structure is created.

```
# mount bee:/ /mnt
```

3 **Access a file system.**

This command or any other command which accesses the file system causes the file system to be mounted.

```
# cd /mnt/export/share/local
```

▼ How to Use Client-Side Failover

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **On the NFS client, mount the file system by using the `ro` option.**

You can mount from the command line, through the automounter, or by adding an entry to `/etc/vfstab` that resembles the following:

```
bee,wasp:/export/share/local  -  /usr/local  nfs  -  no  ro
```

This syntax has been allowed by the automounter. However, the failover was not available while file systems were mounted, only when a server was being selected.

Note – Servers that are running different versions of the NFS protocol cannot be mixed by using a command line or in a `vfstab` entry. Mixing servers that support NFS version 2, version 3, or version 4 protocols can only be performed with autofs. In autofs, the best subset of version 2, version 3, or version 4 servers is used.

▼ How to Disable Mount Access for One Client

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Disable mount access for one client.**

```
# share -F nfs ro=-rose:eng /export/share/man
```

`ro=-rose:eng` The access-list that allows read-only mount access to all clients in the `eng` netgroup except for the host named `rose`

`/export/share/man` The file system to be shared.

How to Mount an NFS File System Through a Firewall

To access file systems through a firewall, use the following procedure.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Manually mount the file system by using a command such as the following:**

   ```
   # mount -F nfs bee:/export/share/local /mnt
   ```

   In this example, the file system `/export/share/local` is mounted on the local client by using the public file handle. An NFS URL can be used instead of the standard path name. If the public file handle is not supported by the server `bee`, the mount operation fails.

Note – This procedure requires that the file system on the NFS server be shared by using the `public` option. Additionally, any firewalls between the client and the server must allow TCP connections on port `2049`. All file systems that are shared allow for public file handle access, so the `public` option is applied by default.

How to Mount an NFS File System Using an NFS URL

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **(Optional) Manually mount the file system.**

   ```
   # mount -F nfs nfs://bee:3000/export/share/local /mnt
   ```

   In this example, the `/export/share/local` file system is being mounted from the server `bee` by using NFS port number `3000`. The port number is not required and by default the standard NFS port number of `2049` is used. You can choose to include the `public` option with an NFS URL. Without the `public` option, the MOUNT protocol is used if the public file handle is not supported by the server. The `public` option forces the use of the public file handle, and the mount fails if the public file handle is not supported.

Note – The NFS protocol version used when mounting the file system is the highest version supported by both the client and the server. The `vers=#` option can be used to select a specific NFS protocol version.

Setting up a DNS Record for a FedFS Server

Once an appropriate DNS record is created, mounting a a file system using FedFS is completed by the automounter once the mount point has been accessed. The DNS record for the server should look something like:

```
% nslookup -q=srv _nfs-domainroot._tcp.example.com bee.example.com
Server:         bee.example.com
Address:        192.168.1.1

_nfs-domainroot._tcp.example.com        service = 1 0 2049 bee.example.com.
```

▼ How to Display Information About File Systems Available for Mounting

The `showmount` command displays information about file systems that have been remotely mounted or are available for mounting. In some environments, this information should not be viewable by all clients. See Example 2–3 for instructions.

- **Display information about mountable file systems.**

  The `-e` option is used to print a list of the shared file systems. For information about other options, see "`showmount` Command" on page 110 or the `showmount(1M)` man page.

  ```
  % /usr/sbin/showmount -e bee
  export list for bee:
  /export/share/local  (everyone)
  /export/home        tulip,lilac
  /export/home2       rose
  ```

Example 2–3 Restricting File System Information Displayed to Clients

In some environments, information about shared file systems and which systems have mounted them should not be displayed. Instead of displaying all information about shared file systems, the `showmount_info` property can be set so that a client:

- Can only see information about file systems that it is allowed to access
- Cannot see information about all file systems that are shared
- Cannot see information about which other systems have mounted the file systems

This property can be set by running the following command on the server:

```
bee# sharectl set -p showmount_info=none nfs
```

Now on the client `rose`, the following information would be displayed:

```
% /usr/sbin/showmount -e bee
export list for bee:
/export/share/local  (everyone)
/export/home2        rose
```

Note that information about the `/export/home` file system is no longer displayed.

Setting Up NFS Services

This section describes some of the tasks that are necessary to do the following:

- Start and stop the NFS server
- Start and stop the automounter
- Select a different version of NFS

Note – Starting in the Solaris 10 release, NFS version 4 is the default.

TABLE 2–3 Task Map for NFS Services

Task	Description	For Instructions
Start the NFS server	Steps to start the NFS service if it has not been started automatically.	"How to Start the NFS Services" on page 40
Stop the NFS server	Steps to stop the NFS service. Normally the service should not need to be stopped.	"How to Stop the NFS Services" on page 40
Start the automounter	Steps to start the automounter. This procedure is required when some of the automounter maps are changed.	"How to Start the Automounter" on page 40
Stop the automounter	Steps to stop the automounter. This procedure is required when some of the automounter maps are changed.	"How to Stop the Automounter" on page 41
Select a different version of NFS on the server	Steps to select a different version of NFS on the server. If you choose not to use NFS version 4, use this procedure.	"How to Select Different Versions of NFS on a Server" on page 41
Select a different version of NFS on the client	Steps to select a different version of NFS on the client by modifying SMF parameters. If you choose not to use NFS version 4, use this procedure.	"How to Select Different Versions of NFS on a Client" on page 42

TABLE 2–3 Task Map for NFS Services *(Continued)*

Task	Description	For Instructions
	Alternate steps to select a different version of NFS on the client by using the command line. If you choose not to use NFS version 4, use this alternate procedure.	"How to Use the `mount` Command to Select Different Versions of NFS on a Client" on page 43

How to Start the NFS Services

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Enable the NFS service on the server.**

   Type the following command.

   ```
   # svcadm enable network/nfs/server
   ```

   This command enables the NFS service.

Note – The NFS server starts automatically when you boot the system. Additionally, any time after the system has been booted, the NFS service daemons can be automatically enabled by sharing the NFS file system. See "How to Set Up Automatic File-System Sharing" on page 30.

How to Stop the NFS Services

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Disable the NFS service on the server.**

   Type the following command.

   ```
   # svcadm disable network/nfs/server
   ```

How to Start the Automounter

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Enable the autofs daemon.**

Type the following command:

```
# svcadm enable system/filesystem/autofs
```

How to Stop the Automounter

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Disable the autofs daemon.**

Type the following command:

```
# svcadm disable system/filesystem/autofs
```

How to Select Different Versions of NFS on a Server

If you choose not to use NFS version 4, use this procedure.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Change SMF parameters to set the NFS version numbers.**

For example, if you want the server to provide only NFS version 3, set the values for both the `server_versmax` and `server_versmin` to 3 as shown below:

```
# sharectl set -p server_versmax=3 nfs
# sharectl set -p server_versmin=3 nfs
```

Note – The NFS version that is set by default is NFS version 4.

3 **(Optional) Disable server delegation.**

If you want to disable server delegation, change the `server_delegation` property.

```
# sharectl set -p server_delegation=off nfs
```

Note – In NFS version 4, server delegation is enabled by default. For more information, see "Delegation in NFS Version 4" on page 126.

4 **(Optional) Establish a common domain.**

If you want to set a common domain for clients and servers, change the `nfsmapid_domain` property.

```
# sharectl set -p nfsmapid_domain=my.company.com nfs
```

my.company.com Provide the common domain name

For more information, refer to "`nfsmapid` Daemon" on page 86.

5 **Check if the NFS service is running on the server.**

Type the following command:

```
# svcs network/nfs/server
```

This command reports whether the NFS server service is online or disabled.

6 **(Optional) If necessary, enable the NFS service.**

If you discovered from the previous step that the NFS service is offline, type the following command to enable the service.

```
# svcadm enable network/nfs/server
```

Note – If you need to configure your NFS service, refer to "How to Set Up Automatic File-System Sharing" on page 30.

See Also "Version Negotiation in NFS" on page 119

▼ How to Select Different Versions of NFS on a Client

The following procedure shows you how to control which version of NFS is used on the client.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Change SMF parameters to set the NFS version numbers.**

For example, if you want the file system to be mounted using the NFS version 3 protocol, set the values for both the `client_versmax` and `client_versmin` to 3 as shown below:

```
# sharectl set -p client_versmax=3 nfs
# sharectl set -p client_versmin=3 nfs
```

Note – The NFS version that is set by default is NFS version 4.

---

3 **Mount NFS on the client.**

Type the following command:

```
# mount server-name:/share-point /local-dir
```

server-name Provide the name of the server.

/share-point Provide the path of the remote directory to be mounted.

/local-dir Provide the path of the local mount point.

See Also "Version Negotiation in NFS" on page 119

▼ How to Use the `mount` Command to Select Different Versions of NFS on a Client

The following procedure shows you how to use the `mount` command to control which version of NFS is used on a client for a particular mount. If you prefer to modify the NFS version for all file systems mounted by the client, see "How to Select Different Versions of NFS on a Client" on page 42.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Mount the desired version of NFS on the client.**

Type the following command:

```
# mount -o vers=value server-name:/share-point /local-dir
```

value Provide the version number.

server-name Provide the name of the server.

/share-point Provide the path of the remote directory to be mounted.

/local-dir Provide the path of the local mount point.

Note – This command overrides the client settings in the SMF repository.

See Also "Version Negotiation in NFS" on page 119

Administering the Secure NFS System

To use the Secure NFS system, all the computers that you are responsible for must have a domain name. Typically, a domain is an administrative entity of several computers that is part of a larger network. If you are running a name service, you should also establish the name service for the domain. See *Oracle Solaris Administration: Naming and Directory Services*.

Kerberos V5 authentication is supported by the NFS service. Chapter 19, "Introduction to the Kerberos Service," in *Oracle Solaris 11.1 Administration: Security Services* discusses the Kerberos service.

You can also configure the Secure NFS environment to use Diffie-Hellman authentication. Chapter 18, "Network Services Authentication (Tasks)," in *Oracle Solaris 11.1 Administration: Security Services* discusses this authentication service.

▼ How to Set Up a Secure NFS Environment With DH Authentication

1 Assign a domain name.

Make the domain name known to each computer in the domain.

2 Establish public keys and secret keys for your clients' users.

Use the `newkey` command. Have each user establish his or her own secure RPC password by using the `chkey` command.

Note – For information about these commands, see the `newkey(1M)` and the `chkey(1)` man pages.

When public keys and secret keys have been generated, the public keys and encrypted secret keys are stored in the `publickey` database.

3 Verify that the name service is responding.

For example:

- If you are running NIS, verify that the `ypbind` daemon is running.

4 Verify that the `keyserv` daemon of the key server is running.

Type the following command.

```
# ps -ef | grep keyserv
root    100      1  16    Apr 11 ?        0:00 /usr/sbin/keyserv
root   2215   2211   5  09:57:28 pts/0    0:00 grep keyserv
```

If the daemon is not running, start the key server by typing the following:

```
# svcadm enable network/rpc/keyserv
```

5 **Decrypt and store the secret key.**

Usually, the login password is identical to the network password. In this situation, `keylogin` is not required. If the passwords are different, the users have to log in, and then run `keylogin`. You still need to use the `keylogin -r` command as `root` to store the decrypted secret key in `/etc/.rootkey`.

Note – You need to run `keylogin -r` if the root secret key changes or if `/etc/.rootkey` is lost.

---

6 **Set the security mode for the file system to be shared.**

For Diffie-Hellman authentication add the `sec=dh` option to the command line.

```
# share -F nfs -o sec=dh /export/home
```

For more information about security modes, see the `nfssec(5)` man page.

7 **Update the automounter maps for the file system.**

Edit the `auto_master` data to include `sec=dh` as a mount option in the appropriate entries for Diffie-Hellman authentication:

```
/home     auto_home     -nosuid,sec=dh
```

When you reinstall, move, or upgrade a computer, remember to save `/etc/.rootkey` if you do not establish new keys or change the keys for `root`. If you do delete `/etc/.rootkey`, you can always type the following:

```
# keylogin -r
```

WebNFS Administration Tasks

This section provides instructions for administering the WebNFS system. Related tasks follow.

TABLE 2–4 Task Map for WebNFS Administration

Task	Description	For Instructions
Plan for WebNFS	Issues to consider before enabling the WebNFS service.	"Planning for WebNFS Access" on page 46
Enable WebNFS	Steps to enable mounting of an NFS file system by using the WebNFS protocol.	"How to Enable WebNFS Access" on page 31

TABLE 2–4 Task Map for WebNFS Administration *(Continued)*

Task	Description	For Instructions
Enable WebNFS through a firewall	Steps to allow access to files through a firewall by using the WebNFS protocol.	"How to Enable WebNFS Access Through a Firewall" on page 47
Browse by using an NFS URL	Instructions for using an NFS URL within a web browser.	"How to Browse Using an NFS URL" on page 47
Use a public file handle with autofs	Steps to force use of the public file handle when mounting a file system with the automounter.	"How to Use a Public File Handle With Autofs" on page 59
Use an NFS URL with autofs	Steps to add an NFS URL to the automounter maps.	"How to Use NFS URLs With Autofs" on page 59
Provide access to a file system through a firewall	Steps to allow access to a file system through a firewall by using the WebNFS protocol.	"How to Mount an NFS File System Through a Firewall" on page 37
Mount a file system by using an NFS URL	Steps to allow access to a file system by using an NFS URL. This process allows for file system access without using the MOUNT protocol.	"How to Mount an NFS File System Using an NFS URL" on page 37

Planning for WebNFS Access

To use WebNFS, you first need an application that is capable of running and loading an NFS URL (for example, `nfs://server/path`). The next step is to choose the file system that can be exported for WebNFS access. If the application is web browsing, often the document root for the web server is used. You need to consider several factors when choosing a file system to export for WebNFS access.

1. Each server has one public file handle that by default is associated with the server's root file system. The path in an NFS URL is evaluated relative to the directory with which the public file handle is associated. If the path leads to a file or directory within an exported file system, the server provides access. You can use the `public` option of the `share` command to associate the public file handle with a specific exported directory. Using this option allows URLs to be relative to the shared file system rather than to the server's root file system. The root file system does not allow web access unless the root file system is shared.
2. The WebNFS environment enables users who already have mount privileges to access files through a browser. This capability is enabled regardless of whether the file system is exported by using the `public` option. Because users already have access to these files through the NFS setup, this access should not create any additional security risk. You only need to share a file system by using the `public` option if users who cannot mount the file system need to use WebNFS access.
3. File systems that are already open to the public make good candidates for using the `public` option. Some examples are the top directory in an ftp archive or the main URL directory for a web site.

4. You can use the `index` option with the `share` command to force the loading of an HTML file. Otherwise, you can list the directory when an NFS URL is accessed.

   After a file system is chosen, review the files and set access permissions to restrict viewing of files or directories, as needed. Establish the permissions, as appropriate, for any NFS file system that is being shared. For many sites, `755` permissions for directories and `644` permissions for files provide the correct level of access.

   You need to consider additional factors if both NFS and HTTP URLs are to be used to access one web site. These factors are described in "WebNFS Limitations With Web Browser Use" on page 136.

How to Browse Using an NFS URL

Browsers that are capable of supporting the WebNFS service should provide access to an NFS URL that resembles the following:

`nfs://`*server*`<:`*port*`>/`*path*

server	Name of the file server
port	Port number to use (`2049`, default value)
path	Path to file, which can be relative to the public file handle or to the root file system

Note – In most browsers, the URL service type (for example, `nfs` or `http`) is remembered from one transaction to the next. The exception occurs when a URL that includes a different service type is loaded. After you use an NFS URL, a reference to an HTTP URL might be loaded. If such a reference is loaded, subsequent pages are loaded by using the HTTP protocol instead of the NFS protocol.

How to Enable WebNFS Access Through a Firewall

You can enable WebNFS access for clients that are not part of the local subnet by configuring the firewall to allow a TCP connection on port `2049`. Just allowing access for `httpd` does not allow NFS URLs to be used.

Task Overview for Autofs Administration

This section describes some of the most common tasks you might encounter in your own environment. Recommended procedures are included for each scenario to help you configure autofs to best meet your clients' needs.

Note – You can also use parameters in the SMF repository to configure your autofs environment. For task information, refer to "Using SMF Parameters to Configure Your Autofs Environment" on page 49.

Task Map for Autofs Administration

The following table provides a description and a pointer to many of the tasks that are related to autofs.

TABLE 2–5 Task Map for Autofs Administration

Task	Description	For Instructions
Start autofs	Start the automount service without having to reboot the system	"How to Start the Automounter" on page 40
Stop autofs	Stop the automount service without disabling other network services	"How to Stop the Automounter" on page 41
Configure your autofs environment by the autofs SMF parameters	Assign values to parameters in the SMF repository	"Using SMF Parameters to Configure Your Autofs Environment" on page 49
Access file systems by using autofs	Access file systems by using the automount service	"Mounting With the Automounter" on page 35
Modify the autofs maps	Steps to modify the master map, which should be used to list other maps Steps to modify an indirect map, which should be used for most maps Steps to modify a direct map, which should be used when a direct association between a mount point on a client and a server is required	"How to Modify the Master Map" on page 51 "How to Modify Indirect Maps" on page 52 "How to Modify Direct Maps" on page 52
Modify the autofs maps to access non-NFS file systems	Steps to set up an autofs map with an entry for a CD-ROM application Steps to set up an autofs map with an entry for a PC-DOS diskette	"How to Access CD-ROM Applications With Autofs" on page 53 "How to Access PC-DOS Data Diskettes With Autofs" on page 53

TABLE 2–5 Task Map for Autofs Administration *(Continued)*

Task	Description	For Instructions
Using `/home`	Example of how to set up a common `/home` map	"Setting Up a Common View of `/home`" on page 54
	Steps to set up a `/home` map that refers to multiple file systems	"How to Set Up `/home` With Multiple Home Directory File Systems" on page 54
Using a new autofs mount point	Steps to set up a project-related autofs map	"How to Consolidate Project-Related Files Under `/ws`" on page 55
	Steps to set up an autofs map that supports different client architectures	"How to Set Up Different Architectures to Access a Shared Namespace" on page 57
	Steps to set up an autofs map that supports different operating systems	"How to Support Incompatible Client Operating System Versions" on page 58
Replicate file systems with autofs	Provide access to file systems that fail over	"How to Replicate Shared Files Across Several Servers" on page 58
Using security restrictions with autofs	Provide access to file systems while restricting remote `root` access to the files	"How to Apply Autofs Security Restrictions" on page 58
Using a public file handle with autofs	Force use of the public file handle when mounting a file system	"How to Use a Public File Handle With Autofs" on page 59
Using an NFS URL with autofs	Add an NFS URL so that the automounter can use it	"How to Use NFS URLs With Autofs" on page 59
Disable autofs browsability	Steps to disable browsability so that autofs mount points are not automatically populated on a single client	"How to Completely Disable Autofs Browsability on a Single NFS Client" on page 60
	Steps to disable browsability so that autofs mount points are not automatically populated on all clients	"How to Disable Autofs Browsability for All Clients" on page 60
	Steps to disable browsability so that a specific autofs mount point is not automatically populated on a client	"How to Disable Autofs Browsability on a Selected File System" on page 60

Using SMF Parameters to Configure Your Autofs Environment

You can use SMF parameters to configure your autofs environment. Specifically, this facility provides an additional way to configure your autofs commands and autofs daemons. The same specifications you would make on the command line can be made with the `sharectl` command. You can make your specifications by providing values to keywords.

The following procedure shows you how to use the `sharectl` command to manage autofs parameters.

▼ How to Configure Your Autofs Environment Using SMF Parameters

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Add or modify an autofs SMF parameter.**

   For example, if you want to turn off browsing for all autofs mount points, use the following command:

   ```
   # sharectl set -p nobrowse=on autofs
   ```

   The `nobrowse` keyword is equivalent to the `-n` option to `automountd`.

3. **Restart the autofs daemon.**

   Type the following command:

   ```
   # svcadm restart system/filesystem/autofs
   ```

Administrative Tasks Involving Maps

The following tables describe several of the factors you need to be aware of when administering autofs maps. Your choice of map and name service affect the mechanism that you need to use to make changes to the autofs maps.

The following table describes the types of maps and their uses.

TABLE 2–6 Types of autofs Maps and Their Uses

Type of Map	Use
Master	Associates a directory with a map
Direct	Directs autofs to specific file systems
Indirect	Directs autofs to reference-oriented file systems

The following table describes how to make changes to your autofs environment that are based on your name service.

TABLE 2–7 Map Maintenance

Name Service	Method
Local files	Text editor
NIS	`make` files

The next table tells you when to run the `automount` command, depending on the modification you have made to the type of map. For example, if you have made an addition or a deletion to a direct map, you need to run the `automount` command on the local system. By running the command, you make the change effective. However, if you have modified an existing entry, you do not need to run the `automount` command for the change to become effective.

TABLE 2–8 When to Run the `automount` Command

Type of Map	Restart `automount`?	
	Addition or Deletion	Modification
`auto_master`	Y	Y
`direct`	Y	N
`indirect`	N	N

Modifying the Maps

The following procedures show you how to update several types of automounter maps.

▼ How to Modify the Master Map

1. **Log in as a user who has permissions to change the maps.**
2. **Make your changes to the master map.**

   The specific steps needed to change the map depends on the name service that you are using.
3. **For each client, become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.
4. **For each client, run the `automount` command to ensure that your changes become effective.**

5 **Notify your users of the changes.**

Notification is required so that the users can also run the `automount` command as superuser on their own computers. Note that the `automount` command gathers information from the master map whenever it is run.

▼ How to Modify Indirect Maps

1 **Log in as a user who has permissions to change the maps.**

2 **Make your changes to the indirect map.**

The specific steps needed to change the map depends on the name service that you are using.

▼ How to Modify Direct Maps

1 **Log in as a user who has permissions to change the maps.**

2 **Make your changes to the direct map.**

The specific steps needed to change the map depends on the name service that you are using.

3 **Notify your users of the changes.**

Notification is required so that the users can run the `automount` command as superuser on their own computers, if necessary.

Note – If you only modify or change the contents of an existing direct map entry, you do not need to run the `automount` command.

For example, suppose you modify the `auto_direct` map so that the `/usr/src` directory is now mounted from a different server. If `/usr/src` is not mounted at this time, the new entry becomes effective immediately when you try to access `/usr/src`. If `/usr/src` is mounted now, you can wait until the auto-unmounting occurs, then access the file.

Note – Use indirect maps whenever possible. Indirect maps are easier to construct and less demanding on the computers' file systems. Also, indirect maps do not occupy as much space in the mount table as direct maps.

Avoiding Mount-Point Conflicts

If you have a local disk partition that is mounted on `/src` and you plan to use the autofs service to mount other source directories, you might encounter a problem. If you specify the mount point `/src`, the NFS service hides the local partition whenever you try to reach it.

You need to mount the partition in some other location, for example, on `/export/src`. You then need an entry in `/etc/vfstab` such as the following:

```
/dev/dsk/d0t3d0s5 /dev/rdsk/c0t3d0s5 /export/src ufs 3 yes -
```

You also need this entry in `auto_src`:

```
terra        terra:/export/src
```

`terra` is the name of the computer.

Accessing Non-NFS File Systems

Autofs can also mount files other than NFS files. Autofs mounts files on removable media, such as diskettes or CD-ROM.

Instead of mounting a file system from a server, you put the media in the drive and reference the file system from the map. If you plan to access non-NFS file systems and you are using autofs, see the following procedures.

▼ How to Access CD-ROM Applications With Autofs

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Update the autofs map.**

   Add an entry for the CD-ROM file system, which should resemble the following:

   ```
   hsfs     -fstype=hsfs,ro     :/dev/sr0
   ```

   The CD-ROM device that you intend to mount must appear as a name that follows the colon.

▼ How to Access PC-DOS Data Diskettes With Autofs

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Update the autofs map.**

Add an entry for the diskette file system such as the following:

```
pcfs     -fstype=pcfs     :/dev/diskette
```

Customizing the Automounter

You can set up the automounter maps in several ways. The following tasks give details about how to customize the automounter maps to provide an easy-to-use directory structure.

Setting Up a Common View of /home

The ideal is for all network users to be able to locate their own or anyone's home directory under `/home`. This view should be common across all computers, whether client or server.

Every Oracle Solaris installation comes with a master map: `/etc/auto_master`.

```
# Master map for autofs
#
+auto_master
/net     -hosts     -nosuid,nobrowse
/home    auto_home  -nobrowse
/nfs4    -fedfs     -ro,nosuid,nobrowse
```

A map for `auto_home` is also installed under `/etc`.

```
# Home directory map for autofs
#
rusty  dragon:/export/home/&
+auto_home
```

When a new local user is created, an entry is automatically added to `/etc/auto_home`. This way, on the server named `dragon`, the home directory for `rusty` can be accessed through `/export/home/rusty` as well as `/home/rusty`.

Note – Users should not be permitted to run `setuid` executables from their home directories. Without this restriction, any user could have superuser privileges on any computer.

▼ How to Set Up /home With Multiple Home Directory File Systems

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Install home directory partitions under `/export/home`.

If the system has several partitions, install the partitions under separate directories, for example, `/export/home1` and `/export/home2`.

3 Update the `auto_home` map.

Whenever you create a new user account, type the location of the user's home directory in the `auto_home` map. Map entries can be simple, for example:

```
rusty        dragon:/export/home1/&
gwenda       dragon:/export/home1/&
charles      sundog:/export/home2/&
rich         dragon:/export/home3/&
```

Notice the use of the & (ampersand) to substitute the map key. The ampersand is an abbreviation for the second occurrence of `rusty` in the following example.

```
rusty         dragon:/export/home1/rusty
```

With the `auto_home` map in place, users can refer to any home directory (including their own) with the path `/home/`*user*. *user* is their login name and the key in the map. This common view of all home directories is valuable when logging in to another user's computer. Autofs mounts your home directory for you. Similarly, if you run a remote windowing system client on another computer, the client program has the same view of the `/home` directory.

This common view also extends to the server. Using the previous example, if `rusty` logs in to the server `dragon`, autofs there provides direct access to the local disk by loopback-mounting `/export/home1/rusty` onto `/home/rusty`.

Users do not need to be aware of the real location of their home directories. If `rusty` needs more disk space and needs to have his home directory relocated to another server, a simple change is sufficient. You need only change `rusty`'s entry in the `auto_home` map to reflect the new location. Other users can continue to use the `/home/rusty` path.

▼ How to Consolidate Project-Related Files Under /ws

Assume that you are the administrator of a large software development project. You plan to make all project-related files available under a directory that is called `/ws`. This directory is to be common across all workstations at the site.

1 Add an entry for the `/ws` directory to the site `auto_master` map.

```
/ws     auto_ws     -nosuid
```

The `auto_ws` map determines the contents of the `/ws` directory.

2 Add the `-nosuid` option as a precaution.

This option prevents users from running setuid programs that might exist in any workspaces.

3 Add entries to the `auto_ws` map.

The `auto_ws` map is organized so that each entry describes a subproject. Your first attempt yields a map that resembles the following:

```
compiler   alpha:/export/ws/&
windows    alpha:/export/ws/&
files      bravo:/export/ws/&
drivers    alpha:/export/ws/&
man        bravo:/export/ws/&
tools      delta:/export/ws/&
```

The ampersand (&) at the end of each entry is an abbreviation for the entry key. For instance, the first entry is equivalent to the following:

```
compiler        alpha:/export/ws/compiler
```

This first attempt provides a map that appears simple, but the map is inadequate. The project organizer decides that the documentation in the `man` entry should be provided as a subdirectory under each subproject. Also, each subproject requires subdirectories to describe several versions of the software. You must assign each of these subdirectories to an entire disk partition on the server.

Modify the entries in the map as follows:

```
compiler \
    /vers1.0    alpha:/export/ws/&/vers1.0 \
    /vers2.0    bravo:/export/ws/&/vers2.0 \
    /man        bravo:/export/ws/&/man
windows \
    /vers1.0    alpha:/export/ws/&/vers1.0 \
    /man        bravo:/export/ws/&/man
files \
    /vers1.0    alpha:/export/ws/&/vers1.0 \
    /vers2.0    bravo:/export/ws/&/vers2.0 \
    /vers3.0    bravo:/export/ws/&/vers3.0 \
    /man        bravo:/export/ws/&/man
drivers \
    /vers1.0    alpha:/export/ws/&/vers1.0 \
    /man        bravo:/export/ws/&/man
tools \
    /           delta:/export/ws/&
```

Although the map now appears to be much larger, the map still contains only the five entries. Each entry is larger because each entry contains multiple mounts. For instance, a reference to `/ws/compiler` requires three mounts for the `vers1.0`, `vers2.0`, and `man` directories. The backslash at the end of each line informs autofs that the entry is continued onto the next line. Effectively, the entry is one long line, though line breaks and some indenting have been used to make the entry more readable. The `tools` directory contains software development tools for all subprojects, so this directory is not subject to the same subdirectory structure. The `tools` directory continues to be a single mount.

This arrangement provides the administrator with much flexibility. Software projects typically consume substantial amounts of disk space. Through the life of the project, you might be

required to relocate and expand various disk partitions. If these changes are reflected in the `auto_ws` map, the users do not need to be notified, as the directory hierarchy under `/ws` is not changed.

Because the servers `alpha` and `bravo` view the same autofs map, any users who log in to these computers can find the `/ws` namespace as expected. These users are provided with direct access to local files through loopback mounts instead of NFS mounts.

How to Set Up Different Architectures to Access a Shared Namespace

You need to assemble a shared namespace for local executables, and applications, such as spreadsheet applications and word-processing packages. The clients of this namespace use several different workstation architectures that require different executable formats. Also, some workstations are running different releases of the operating system.

1. **Create the `auto_local` map.**

   See the *Oracle Solaris Administration: Naming and Directory Services*.

2. **Choose a single, site-specific name for the shared namespace.**

   This name makes the files and directories that belong to this space easily identifiable. For example, if you choose `/usr/local` as the name, the path `/usr/local/bin` is obviously a part of this namespace.

3. **For ease of user community recognition, create an autofs indirect map.**

   Mount this map at `/usr/local`. Set up the following entry in the NIS `auto_master` map:

   ```
   /usr/local     auto_local     -ro
   ```

   Notice that the `-ro` mount option implies that clients cannot write to any files or directories.

4. **Export the appropriate directory on the server.**

5. **Include a `bin` entry in the auto_local map.**

   Your directory structure resembles the following:

   ```
   bin     aa:/export/local/bin
   ```

6. **(Optional) To serve clients of different architectures, change the entry by adding the autofs `CPU` variable.**

   ```
   bin     aa:/export/local/bin/$CPU
   ```

   - For SPARC clients – Place executables in `/export/local/bin/sparc`.
   - For x86 clients – Place executables in `/export/local/bin/i386`.

How to Support Incompatible Client Operating System Versions

1 **Combine the architecture type with a variable that determines the operating system type of the client.**

You can combine the autofs `OSREL` variable with the `CPU` variable to form a name that determines both CPU type and OS release.

2 **Create the following map entry.**

```
bin     aa:/export/local/bin/$CPU$OSREL
```

For clients that are running version 5.6 of the operating system, export the following file systems:

- For SPARC clients – Export `/export/local/bin/sparc5.6`.
- For x86 clients – Place executables in `/export/local/bin/i3865.6`.

How to Replicate Shared Files Across Several Servers

The best way to share replicated file systems that are read-only is to use failover. See "Client-Side Failover" on page 132 for a discussion of failover.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Modify the entry in the autofs maps.**

Create the list of all replica servers as a comma-separated list, such as the following:

```
bin     aa,bb,cc,dd:/export/local/bin/$CPU
```

Autofs chooses the nearest server. If a server has several network interfaces, list each interface. Autofs chooses the nearest interface to the client, avoiding unnecessary routing of NFS traffic.

How to Apply Autofs Security Restrictions

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Create the following entry in the name service `auto_master` file:**

```
/home     auto_home     -nosuid
```

The nosuid option prevents users from creating files with the setuid or setgid bit set.

This entry overrides the entry for /home in a generic local /etc/auto_master file. See the previous example. The override happens because the +auto_master reference to the external name service map occurs before the /home entry in the file. If the entries in the auto_home map include mount options, the nosuid option is overwritten. Therefore, either no options should be used in the auto_home map or the nosuid option must be included with each entry.

Note – Do not mount the home directory disk partitions on or under /home on the server.

How to Use a Public File Handle With Autofs

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Create an entry in the autofs map such as the following:**

   ```
   /usr/local      -ro,public     bee:/export/share/local
   ```

   The public option forces the public handle to be used. If the NFS server does not support a public file handle, the mount fails.

How to Use NFS URLs With Autofs

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Create an autofs entry such as the following:**

   ```
   /usr/local      -ro     nfs://bee/export/share/local
   ```

   The service tries to use the public file handle on the NFS server. However, if the server does not support a public file handle, the MOUNT protocol is used.

Disabling Autofs Browsability

The default version of /etc/auto_master that is installed has the -nobrowse option added to the entries for /home and /net. In addition, the upgrade procedure adds the -nobrowse option to the /home and /net entries in /etc/auto_master if these entries have not been modified. However, you might have to make these changes manually or to turn off browsability for site-specific autofs mount points after the installation.

You can turn off the browsability feature in several ways. Disable the feature by using a command-line option to the `automountd` daemon, which completely disables autofs browsability for the client. Or disable browsability for each map entry on all clients by using the autofs maps. You can also disable the feature for each map entry on each client, using local autofs maps if no network-wide namespace is being used.

How to Completely Disable Autofs Browsability on a Single NFS Client

1. **Become an administrator on the NFS client.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Change the autofs SMF configuration parameter.**

   ```
   # sharectl set -p nobrowse=TRUE autofs
   ```

3. **Restart the autofs service.**

   ```
   # svcadm restart system/filesystem/autofs
   ```

How to Disable Autofs Browsability for All Clients

To disable browsability for all clients, you must employ a name service such as NIS. Otherwise, you need to manually edit the automounter maps on each client. In this example, the browsability of the `/home` directory is disabled. You must follow this procedure for each indirect autofs node that needs to be disabled.

1. **Add the `-nobrowse` option to the `/home` entry in the name service `auto_master` file.**

   ```
   /home      auto_home      -nobrowse
   ```

2. **Run the `automount` command on all clients.**

   The new behavior becomes effective after you run the `automount` command on the client systems or after a reboot.

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

How to Disable Autofs Browsability on a Selected File System

In this example, browsability of the `/net` directory is disabled. You can use the same procedure for `/home` or any other autofs mount points.

1. **Verify the search order for the automount naming services.**

   The `config/automount` property in the `name-service/switch` service shows the search order for the automount information.

   ```
   # svcprop -p config svc:/system/name-service/switch
   config/value_authorization astring solaris.smf.value.name-service.switch
   config/printer astring user\ files
   config/default astring files\ nis
   config/automount astring files\ nis
   ```

   The last entry shows that local automount files are searched first and then the NIS service is checked. The `config/default` entry specifies the search order for all naming information not specifically listed.

2. **Check the position of the `+auto_master` entry in `/etc/auto_master`.**

   For additions to the local files to have precedence over the entries in the namespace, the `+auto_master` entry must be moved to follow `/net`:

   ```
   # Master map for automounter
   #
   /net    -hosts     -nosuid
   /home   auto_home
   /nfs4    -fedfs     -ro,nosuid,nobrowse
   +auto_master
   ```

   A standard configuration places the `+auto_master` entry at the top of the file. This placement prevents any local changes from being used.

3. **Add the `nobrowse` option to the `/net` entry in the `/etc/auto_master` file.**

   ```
   /net     -hosts     -nosuid,nobrowse
   ```

4. **On all clients, run the `automount` command.**

   The new behavior becomes effective after running the `automount` command on the client systems or after a reboot.

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

Administering NFS Referrals

NFS referrals are a way for an NFSv4 server to point to file systems located on other NFSv4 servers, as a way of connecting multiple NFSv4 servers into a uniform namespace.

▼ How to Create and Access an NFS Referral

1. **On an NFS server: create a referral.**

   Add the referral on an NFS-shared file system, pointing to one or more existing NFS-shared file systems.

   ```
   server1% nfsref add /share/docs server2:/usr/local/docs server3:/tank/docs
   Created reparse point /share/docs
   ```

2. **Verify that the referral was created.**

   ```
   server1% nfsref lookup /share/docs
   /share/docs points to:
   server2:/usr/local/docs
   server3:/tank/docs
   ```

3. **On a client: mount the referral.**

   ```
   client1% pfexec mount server1:/share/docs /mnt
   ```

4. **Verify that the mount worked.**

   ```
   client1% cd /mnt/docs
   client1% df -k .
   /mnt/docs     (server2:/usr/local/docs):10372284465 blocks 10372284465 files
   ```

Example 2–4 Modifying an Existing Referral

If you wanted to add another file system, such as `server4:/tank/docs` to the existing referral, you would enter the command from step 2 above with the new file system.

```
server1% nfsref add /share/docs server2:/usr/local/docs server3:/tank/docs server4:/tank/docs
```

The add subcommand simply replaces the information in the current referral with the new information from the command. The add subcommand is how you would modify the file systems associated with an existing referral.

▼ How to Remove an NFS Referral

Follow this procedure to remove an NFS referral.

- **Remove the referral.**

  ```
  server1% nfsref remove /share/docs
  Removed svc_type 'nfs-basic' from /share/docs
  ```

Administering FedFS

The FedFS protocols can be used to construct and maintain a federated file system. This file system can include many different file servers, allowing for a multi-vendor global namespace.

▼ How to Create an Namespace Database (NSDB)

A NSDB is used to provide information about the filesets from different types of servers that are combined into a single FedFS namespace. This procedure is done on the LDAP server.

Before You Begin This procedure requires that you assume the `root` role and that you have an LDAP server installed.

1 Configure the FedFS LDAP schema.

Update the LDAP configuration file with the following entries:

```
include         /usr/lib/fs/nfs/fedfs-11.schema
suffix  dc=example,dc=org
rootdn  cn=Manager,dc=example,dc=org
rootpw  <password>
```

2 Create a distinguished name for the FedFS data.

See the `nsdb-update-nci(1M)` man page.

```
# nsdb-update-nci -l localhost -r 389 -D cn=Manager -w\
      example.org dc=example,dc=org adding new entry "dc=example,dc=org"
NCE entry created
```

▼ How to Use a Secured Connection to the NSDB

Before You Begin This procedure requires that you assume the `root` role and that you have an LDAP server installed.

1 On the LDAP server: create a certificate.

```
# mkdir /etc/openldap/certs
# mkdir /etc/openldap/certs/keys
# cd /etc/openldap/certs
# openssl req -x509 -nodes -days 3650 -newkey rsa:2048 \
  -keyout keys/ldapskey.pem -out ldapscert.pem
# chown -R openldap:openldap /etc/openldap/certs/*
# chmod 0400 keys/ldapskey.pem
```

2 On the LDAP server: add declarations to the LDAP configuration file.

```
TLSCertificateFile /etc/openldap/certs/ldapscert.pem
TLSCertificateKeyFile /etc/openldap/certs/keys/ldapskey.pem
```

3 **Copy the certificate to the NFS server and clients.**

```
# scp ldap-server:/etc/openldap/certs/keys/ldapskey.pem /etc/openldap/certs/keys/ldapskey.pem
# chmod 0400 /etc/openldap/certs/keys/ldapskey.pem
```

4 **On the NFS server and clients: update the connection entry.**

```
# nsdbparams update -f ldapscert.pem -t FEDFS_SEC_TLS localhost
```

▼ How to Create a FedFS Referral

Before You Begin This procedure requires that you assume the `root` role and that you have an NFS server installed.

1 **Create a connection entry for a NSDB.**

This command creates a connection entry between the NSDB defined on the LDAP server and an NFS server.

```
# nsdbparams update -D cn=Manager,dc=example,dc=org -w example.org nsdb.example.org
```

2 **Create a FedFS referral.**

The `-t` option selects the service type of the referral.

```
# nfsref -t nfs-fedfs add /share/docs server2:/usr/local/docs server3:/tank/docs
Created reparse point /share/doc
```

Strategies for NFS Troubleshooting

When tracking an NFS problem, remember the main points of possible failure: the server, the client, and the network. The strategy that is outlined in this section tries to isolate each individual component to find the one that is not working. In all situations, the `mountd` and `nfsd` daemons must be running on the server for remote mounts to succeed.

The `-intr` option is set by default for all mounts. If a program hangs with a `server not responding` message, you can kill the program with the keyboard interrupt Control-c.

When the network or server has problems, programs that access hard-mounted remote files fail differently than those programs that access soft-mounted remote files. Hard-mounted remote file systems cause the client's kernel to retry the requests until the server responds again. Soft-mounted remote file systems cause the client's system calls to return an error after trying for awhile. Because these errors can result in unexpected application errors and data corruption, avoid soft mounting.

When a file system is hard mounted, a program that tries to access the file system hangs if the server fails to respond. In this situation, the NFS system displays the following message on the console:

```
NFS server hostname not responding still trying
```

When the server finally responds, the following message appears on the console:

```
NFS server hostname ok
```

A program that accesses a soft-mounted file system whose server is not responding generates the following message:

```
NFS operation failed for server hostname: error # (error-message)
```

Note – Because of possible errors, do not soft-mount file systems with read-write data or file systems from which executables are run. Writable data could be corrupted if the application ignores the errors. Mounted executables might not load properly and can fail.

NFS Troubleshooting Procedures

To determine where the NFS service has failed, you need to follow several procedures to isolate the failure. Check for the following items:

- Can the client reach the server?
- Can the client contact the NFS services on the server?
- Are the NFS services running on the server?

In the process of checking these items, you might notice that other portions of the network are not functioning. For example, the name service or the physical network hardware might not be functioning. The *Oracle Solaris Administration: Naming and Directory Services* contains debugging procedures for several name services. Also, during the process you might see that the problem is not at the client end. An example is if you get at least one trouble call from every subnet in your work area. In this situation, you should assume that the problem is the server or the network hardware near the server. So, you should start the debugging process at the server, not at the client.

▼ How to Check Connectivity on an NFS Client

1 Check that the NFS server is reachable from the client. On the client, type the following command.

```
% /usr/sbin/ping bee
bee is alive
```

If the command reports that the server is alive, remotely check the NFS server. See "How to Check the NFS Server Remotely" on page 66.

2 If the server is not reachable from the client, ensure that the local name service is running.

3 **If the name service is running, ensure that the client has received the correct host information by typing the following:**

```
% /usr/bin/getent hosts bee
129.144.83.117    bee.eng.acme.com
```

4 **If the host information is correct, but the server is not reachable from the client, run the `ping` command from another client.**

If the command run from a second client fails, see "How to Verify the NFS Service on the Server" on page 67.

5 **If the server is reachable from the second client, use `ping` to check connectivity of the first client to other systems on the local net.**

If this command fails, check the networking software configuration on the client, for example, `/etc/netmasks` and the property information associated with the `svc:/system/name-service/switch` service.

6 **(Optional) Check the output of the `rpcinfo` command.**

If the `rpcinfo` command does not display `program 100003 version 4 ready and waiting`, then NFS version 4 is not enabled on the server. See Table 2–3 for information about enabling NFS version 4.

7 **If the software is correct, check the networking hardware.**

Try to move the client onto a second net drop.

▼ How to Check the NFS Server Remotely

Note that support for both the UDP and the MOUNT protocols is not necessary if you are using an NFS version 4 server.

1 **Check that the NFS services have started on the NFS server by typing the following command:**

```
% rpcinfo -s bee|egrep 'nfs|mountd'
 100003  3,2    tcp,udp,tcp6,upd6               nfs     superuser
 100005  3,2,1  ticots,ticotsord,tcp,tcp6,ticlts,udp,upd6  mountd  superuser
```

If the daemons have not been started, see "How to Restart NFS Services" on page 69.

2 **Check that the server's `nfsd` processes are responding.**

On the client, type the following command to test the UDP NFS connections from the server.

```
% /usr/bin/rpcinfo -u bee nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
```

Note – NFS version 4 does not support UDP.

If the server is running, it prints a list of program and version numbers. Using the `-t` option tests the TCP connection. If this command fails, proceed to "How to Verify the NFS Service on the Server" on page 67.

3 Check that the server's `mountd` is responding, by typing the following command.

```
% /usr/bin/rpcinfo -u bee mountd
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting
program 100005 version 3 ready and waiting
```

If the server is running, it prints a list of program and version numbers that are associated with the UDP protocol. Using the `-t` option tests the TCP connection. If either attempt fails, proceed to "How to Verify the NFS Service on the Server" on page 67.

4 Check the local autofs service if it is being used:

```
% cd /net/wasp
```

Choose a `/net` or `/home` mount point that you know should work properly. If this command fails, then as `root` on the client, type the following to restart the autofs service:

```
# svcadm restart system/filesystem/autofs
```

5 Verify that file system is shared as expected on the server.

```
% /usr/sbin/showmount -e bee
/usr/src                                    eng
/export/share/man                       (everyone)
```

Check the entry on the server and the local mount entry for errors. Also, check the namespace. In this instance, if the first client is not in the `eng` netgroup, that client cannot mount the `/usr/src` file system.

Check all entries that include mounting information in all the local files. The list includes `/etc/vfstab` and all the `/etc/auto_*` files.

▼ How to Verify the NFS Service on the Server

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Check that the server can reach the clients.

```
# ping lilac
lilac is alive
```

3. **If the client is not reachable from the server, ensure that the local name service is running.**

4. **If the name service is running, check the networking software configuration on the server, for example, `/etc/netmasks` and the property information associated with the `svc:/system/name-service/switch` service.**

5. **Type the following command to check whether the `rpcbind` daemon is running.**

```
# /usr/bin/rpcinfo -u localhost rpcbind
program 100000 version 1 ready and waiting
program 100000 version 2 ready and waiting
program 100000 version 3 ready and waiting
```

   If the server is running, it prints a list of program and version numbers that are associated with the UDP protocol.

6. **Type the following command to check whether the `nfsd` daemon is running.**

```
# rpcinfo -u localhost nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
# ps -ef | grep nfsd
    root 101328      0   0   Jul 12 ?         303:25 nfsd_kproc
    root 101327      1   0   Jul 12 ?           2:54 /usr/lib/nfs/nfsd
    root 263149 131084   0 13:59:19 pts/17      0:00 grep nfsd
```

   Note – NFS version 4 does not support UDP.

   If the server is running, it prints a list of program and version numbers that are associated with the UDP protocol. Also use the `-t` option with `rpcinfo` to check the TCP connection. If these commands fail, restart the NFS service. See "How to Restart NFS Services" on page 69.

7. **Type the following command to check whether the `mountd` daemon is running.**

```
# /usr/bin/rpcinfo -u localhost mountd
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting
program 100005 version 3 ready and waiting
# ps -ef | grep mountd
root    145      1 0 Apr 07  ?     21:57 /usr/lib/autofs/automountd
root    234      1 0 Apr 07  ?     0:04  /usr/lib/nfs/mountd
root   3084 2462 1 09:30:20 pts/3  0:00  grep mountd
```

   If the server is running, it prints a list of program and version numbers that are associated with the UDP protocol. Also use the `-t` option with `rpcinfo` to check the TCP connection. If these commands fail, restart the NFS service. See "How to Restart NFS Services" on page 69.

▼ How to Restart NFS Services

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Restart the NFS service on the server.**

   Type the following command.

   ```
   # svcadm restart network/nfs/server
   ```

Identifying Which Host Is Providing NFS File Service

Run the `nfsstat` command with the `-m` option to gather current NFS information. The name of the current server is printed after "`currserver=`".

```
% nfsstat -m
/usr/local from bee,wasp:/export/share/local
 Flags: vers=3,proto=tcp,sec=sys,hard,intr,llock,link,synlink,
        acl,rsize=32768,wsize=32678,retrans=5
 Failover: noresponse=0, failover=0, remap=0, currserver=bee
```

▼ How to Verify Options Used With the `mount` Command

No warning is issued for invalid options. The following procedure helps determine whether the options that were supplied either on the command line or through `/etc/vfstab` were valid.

For this example, assume that the following command has been run:

```
# mount -F nfs -o ro,vers=2 bee:/export/share/local /mnt
```

1. **Verify the options by running the following command.**

   ```
   % nfsstat -m
   /mnt from bee:/export/share/local
   Flags:  vers=2,proto=tcp,sec=sys,hard,intr,dynamic,acl,rsize=8192,wsize=8192,
           retrans=5
   ```

   The file system from bee has been mounted with the protocol version set to 2. Unfortunately, the `nfsstat` command does not display information about all of the options. However, using the `nfsstat` command is the most accurate way to verify the options.

2 **Check the entry in `/etc/mnttab`.**

The `mount` command does not allow invalid options to be added to the mount table. Therefore, verify that the options that are listed in the file match those options that are listed on the command line. In this way, you can check those options that are not reported by the `nfsstat` command.

```
# grep bee /etc/mnttab
bee:/export/share/local /mnt nfs    ro,vers=2,dev=2b0005e 859934818
```

Troubleshooting Autofs

Occasionally, you might encounter problems with autofs. This section should improve the problem-solving process. The section is divided into two subsections.

This section presents a list of the error messages that autofs generates. The list is divided into two parts:

- Error messages that are generated by the verbose (`-v`) option of `automount`
- Error messages that might appear at any time

Each error message is followed by a description and probable cause of the message.

When troubleshooting, start the autofs programs with the verbose (`-v`) option. Otherwise, you might experience problems without knowing the cause.

The following paragraphs are labeled with the error message you are likely to see if autofs fails, and a description of the possible problem.

Error Messages Generated by `automount -v`

`bad key` *key* `in direct map` *mapname*
Description: While scanning a direct map, autofs has found an entry key without a prefixed /.

Solution: Keys in direct maps must be full path names.

`bad key` *key* `in indirect map` *mapname*
Description: While scanning an indirect map, autofs has found an entry key that contains a `/`.

Solution: Indirect map keys must be simple names, not path names.

`can't mount` *server*`:`*pathname: reason*
Description: The mount daemon on the server refuses to provide a file handle for *server*:*pathname*.

Solution: Check the export table on the server.

`couldn't create mount point` *mountpoint*`: ` *reason*
Description: Autofs was unable to create a mount point that was required for a mount. This problem most frequently occurs when you attempt to hierarchically mount all of a server's exported file systems.

Solution: A required mount point can exist only in a file system that cannot be mounted, which means the file system cannot be exported. The mount point cannot be created because the exported parent file system is exported read-only.

`leading space in map entry` *entry* `text in` *mapname*
Description: Autofs has discovered an entry in an automount map that contains leading spaces. This problem is usually an indication of an improperly continued map entry. For example:

```
fake
/blat           frobz:/usr/frotz
```

Solution: In this example, the warning is generated when autofs encounters the second line because the first line should be terminated with a backslash (`\`).

mapname`: Not found`
Description: The required map cannot be located. This message is produced only when the `-v` option is used.

Solution: Check the spelling and path name of the map name.

`remount` *server*`:`*pathname* `on` *mountpoint*`: server not responding`
Description: Autofs has failed to remount a file system that it previously unmounted.

Solution: Contact Sun for assistance. This error message is extremely rare and has no straightforward solution.

`WARNING:` *mountpoint* `already mounted on`
Description: Autofs is attempting to mount over an existing mount point. This message means that an internal error occurred in autofs (an anomaly).

Solution: Contact Sun for assistance. This error message is extremely rare and has no straightforward solution.

Miscellaneous Error Messages

`dir` *mountpoint* `must start with '/'`
Solution: The automounter mount point must be given as a full path name. Check the spelling and path name of the mount point.

`hierarchical mountpoint:` *pathname1* `and` *pathname2*

Solution: Autofs does not allow its mount points to have a hierarchical relationship. An autofs mount point must not be contained within another automounted file system.

`host` *server* `not responding`

Description: Autofs attempted to contact *server*, but received no response.

Solution: Check the NFS server status.

hostname`: exports:` *rpc-err*

Description: An error occurred while getting the export list from *hostname*. This message indicates a server or network problem.

Solution: Check the NFS server status.

`map` *mapname*`, key` *key*`: bad`

Description: The map entry is malformed, and autofs cannot interpret the entry.

Solution: Recheck the entry. Perhaps the entry has characters that need to be escaped.

mapname`:` *nis-err*

Description: An error occurred when looking up an entry in a NIS map. This message can indicate NIS problems.

Solution: Check the NIS server status.

`mount of` *server*`:`*pathname* `on` *mountpoint*:*reason*

Description: Autofs failed to do a mount. This occurrence can indicate a server or network problem. The *reason* string defines the problem.

Solution: Contact Sun for assistance. This error message is extremely rare and has no straightforward solution.

mountpoint`: Not a directory`

Description: Autofs cannot mount itself on *mountpoint* because it is not a directory.

Solution: Check the spelling and path name of the mount point.

`nfscast: cannot send packet:` *reason*

Description: Autofs cannot send a query packet to a server in a list of replicated file system locations. The *reason* string defines the problem.

Solution: Contact Sun for assistance. This error message is extremely rare and has no straightforward solution.

`nfscast: cannot receive reply:` *reason*

Description: Autofs cannot receive replies from any of the servers in a list of replicated file system locations. The *reason* string defines the problem.

Solution: Contact Sun for assistance. This error message is extremely rare and has no straightforward solution.

`nfscast: select:` *reason*

Description: All these error messages indicate problems in attempting to check servers for a replicated file system. This message can indicate a network problem. The *reason* string defines the problem.

Solution: Contact Sun for assistance. This error message is extremely rare and has no straightforward solution.

`pathconf: no info for` *server*`:`*pathname*

Description: Autofs failed to get `pathconf` information for the path name.

Solution: See the `fpathconf(2)` man page.

`pathconf:` *server*`: server not responding`

Description: Autofs is unable to contact the mount daemon on *server* that provides the information to `pathconf()`.

Solution: Avoid using the `POSIX` mount option with this server.

Other Errors With Autofs

If the `/etc/auto*` files have the execute bit set, the automounter tries to execute the maps, which creates messages such as the following:

```
/etc/auto_home: +auto_home: not found
```

In this situation, the `auto_home` file has incorrect permissions. Each entry in the file generates an error message that is similar to this message. The permissions to the file should be reset by typing the following command:

```
# chmod 644 /etc/auto_home
```

NFS Error Messages

This section shows an error message that is followed by a description of the conditions that should create the error and at minimum one remedy.

`Bad argument specified with index option - must be a file`
Solution: You must include a file name with the `index` option. You cannot use directory names.

`Cannot establish NFS service over /dev/`*`tcp`*`: transport setup problem`
Description: This message is often created when the services information in the namespace has not been updated. The message can also be reported for UDP.

Solution: To fix this problem, you must update the services data in the namespace.

For NIS and `/etc/services`, the entries should be as follows:

```
nfsd    2049/tcp    nfs    # NFS server daemon
nfsd    2049/udp    nfs    # NFS server daemon
```

`Could not start` *`daemon`*`: ` *`error`*
Description: This message is displayed if the daemon terminates abnormally or if a system call error occurs. The *error* string defines the problem.

Solution: Contact Sun for assistance. This error message is rare and has no straightforward solution.

`Could not use public filehandle in request to` *`server`*
Description: This message is displayed if the `public` option is specified but the NFS server does not support the public file handle. In this situation, the mount fails.

Solution: To remedy this situation, either try the mount request without using the public file handle or reconfigure the NFS server to support the public file handle.

daemon `running already with pid` *pid*
Description: The daemon is already running.

Solution: If you want to run a new copy, kill the current version and start a new version.

`error locking` *lock file*
Description: This message is displayed when the *lock file* that is associated with a daemon cannot be locked properly.

Solution: Contact Sun for assistance. This error message is rare and has no straightforward solution.

`error checking` *lock file*`:` *error*
Description: This message is displayed when the *lock file* that is associated with a daemon cannot be opened properly.

Solution: Contact Sun for assistance. This error message is rare and has no straightforward solution.

`NOTICE: NFS3: failing over from` *host1* `to` *host2*
Description: This message is displayed on the console when a failover occurs. The message is advisory only.

Solution: No action required.

filename`: File too large`
Description: An NFS version 2 client is trying to access a file that is over 2 Gbytes.

Solution: Avoid using NFS version 2. Mount the file system with version 3 or version 4. Also, see the description of the `nolargefiles` option in "`mount` Options for NFS File Systems" on page 96.

`mount: ... server not responding:RPC_PMAP_FAILURE - RPC_TIMED_OUT`
Description: The server that is sharing the file system you are trying to mount is down or unreachable, at the wrong run level, or its `rpcbind` is dead or hung.

Solution: Wait for the server to reboot. If the server is hung, reboot the server.

`mount: ... server not responding: RPC_PROG_NOT_REGISTERED`
Description: The mount request registered with `rpcbind`, but the NFS mount daemon `mountd` is not registered.

Solution: Wait for the server to reboot. If the server is hung, reboot the server.

`mount: ... No such file or directory`
Description: Either the remote directory or the local directory does not exist.

Solution: Check the spelling of the directory names. Run `ls` on both directories.

`mount: ...: Permission denied`
Description: Your computer name might not be in the list of clients or netgroup that is allowed access to the file system you tried to mount.

Solution: Use `showmount -e` to verify the access list.

`NFS file temporarily unavailable on the server, retrying ...`
Description: An NFS version 4 server can delegate the management of a file to a client. This message indicates that the server is recalling a delegation for another client that conflicts with a request from your client.

Solution: The recall must occur before the server can process your client's request. For more information about delegation, refer to "Delegation in NFS Version 4" on page 126.

`NFS fsstat failed for server` *hostname*`: RPC: Authentication error`

Description: This error can be caused by many situations. One of the most difficult situations to debug is when this problem occurs because a user is in too many groups. Currently, a user can be in no more than 16 groups if the user is accessing files through NFS mounts.

Solution: An alternate does exist for users who need to be in more than 16 groups. You can use access control lists to provide the needed access privileges.

`nfs mount: NFS can't support "nolargefiles"`

Description: An NFS client has attempted to mount a file system from an NFS server by using the `-nolargefiles` option.

Solution: This option is not supported for NFS file system types.

`nfs mount: NFS V2 can't support "largefiles"`

Description: The NFS version 2 protocol cannot handle large files.

Solution: You must use version 3 or version 4 if access to large files is required.

`NFS server` *hostname* `not responding still trying`

Description: If programs hang while doing file-related work, your NFS server might have failed. This message indicates that NFS server *hostname* is down or that a problem has occurred with the server or the network.

Solution: If failover is being used, *hostname* is a list of servers. Start troubleshooting with "How to Check Connectivity on an NFS Client" on page 65.

`NFS server recovering`

Description: During part of the NFS version 4 server reboot, some operations were not permitted. This message indicates that the client is waiting for the server to permit this operation to proceed.

Solution: No action required. Wait for the server to permit the operation.

`Permission denied`

Description: This message is displayed by the `ls -l`, `getfacl`, and `setfacl` commands for the following reasons:

- If the user or group that exists in an access control list (ACL) entry on an NFS version 4 server cannot be mapped to a valid user or group on an NFS version 4 client, the user is not allowed to read the ACL on the client.

- If the user or group that exists in an ACL entry that is being set on an NFS version 4 client cannot be mapped to a valid user or group on an NFS version 4 server, the user is not allowed to write or modify an ACL on the client.
- If an NFS version 4 client and server have mismatched NFSMAPID_DOMAIN values, ID mapping fails.

For more information, see "ACLs and `nfsmapid` in NFS Version 4" on page 128.

Solution: Do the following:

- Make sure that all user and group IDs in the ACL entries exist on both the client and server.
- Make sure that the value for `nfsmapid_domain` is set correctly in the SMF Repository.

To determine if any user or group cannot be mapped on the server or client, use the script that is provided in "Checking for Unmapped User or Group IDs" on page 129.

`port` *number* `in nfs URL not the same as port` *number* `in port option`

Description: The port number that is included in the NFS URL must match the port number that is included with the `-port` option to mount. If the port numbers do not match, the mount fails.

Solution: Either change the command to make the port numbers identical or do not specify the port number that is incorrect. Usually, you do not need to specify the port number with both the NFS URL and the `-port` option.

`replicas must have the same version`

Description: For NFS failover to function properly, the NFS servers that are replicas must support the same version of the NFS protocol.

Solution: Running multiple versions is not allowed.

`replicated mounts must be read-only`

Description: NFS failover does not work on file systems that are mounted read-write. Mounting the file system read-write increases the likelihood that a file could change.

Solution: NFS failover depends on the file systems being identical.

`replicated mounts must not be soft`

Description: Replicated mounts require that you wait for a timeout before failover occurs.

Solution: The `soft` option requires that the mount fail immediately when a timeout starts, so you cannot include the `-soft` option with a replicated mount.

`share_nfs: Cannot share more than one filesystem with 'public' option`

Solution: Use the `share` command to make sure that only one file system selected to be shared with the `-public` option. Only one public file handle can be established per server, so only one file system per server can be shared with this option.

`WARNING: No network locking on` *hostname*`:`*path*`: contact admin to install server change`

Description: An NFS client has unsuccessfully attempted to establish a connection with the network lock manager on an NFS server. Rather than fail the mount, this warning is generated to warn you that locking does not work.

Solution: Upgrade the server with a new version of the OS that provides complete lock manager support.

◆ ◆ ◆ CHAPTER 3

Accessing Network File Systems (Reference)

This chapter describes the NFS commands, as well as the different parts of the NFS environment and how these parts work together.

Note – If your system has zones enabled and you want to use this feature in a non-global zone, see *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management* for more information.

NFS Files

You need several files to support NFS activities on any computer. Many of these files are ASCII, but some of the files are data files. Table 3–1 lists these files and their functions.

TABLE 3–1 NFS Files

File Name	Function
`/etc/default/fs`	Lists the default file-system type for local file systems.

TABLE 3–1 NFS Files *(Continued)*

File Name	Function
`/etc/default/nfslogd`	Lists configuration information for the NFS server logging daemon, `nfslogd`.
`/etc/dfs/dfstab`	Obsolete: Lists the local resources to be shared.
`/etc/dfs/fstypes`	Lists the default file-system types for remote file systems.
`/etc/dfs/sharetab`	Lists the local and remote resources that are shared. See the `sharetab(4)` man page. Do not edit this file.
`/etc/mnttab`	Lists file systems that are currently mounted, including automounted directories. See the `mnttab(4)` man page. Do not edit this file.
`/etc/netconfig`	Lists the transport protocols. Do not edit this file.
`/etc/nfs/nfslog.conf`	Lists general configuration information for NFS server logging.
`/etc/nfs/nfslogtab`	Lists information for log postprocessing by `nfslogd`. Do not edit this file.
`/etc/nfssec.conf`	Lists NFS security services.
`/etc/rmtab`	Lists file systems that are remotely mounted by NFS clients. See the `rmtab(4)` man page. Do not edit this file.
`/etc/vfstab`	Defines file systems to be mounted locally. See the `vfstab(4)` man page.

The first entry in `/etc/dfs/fstypes` is often used as the default file-system type for remote file systems. This entry defines the NFS file-system type as the default.

Only one entry is in `/etc/default/fs`: the default file-system type for local disks. You can determine the file-system types that are supported on a client or server by checking the files in `/kernel/fs`.

`/etc/default/nfslogd` File

This file defines some of the parameters that are used when using NFS server logging. The following parameters can be defined.

CYCLE_FREQUENCY
: Determines the number of hours that must pass before the log files are cycled. The default value is 24 hours. This option is used to prevent the log files from growing too large.

IDLE_TIME
: Sets the number of seconds `nfslogd` should sleep before checking for more information in the buffer file. This parameter also determines how often the configuration file is checked. This parameter, along with MIN_PROCESSING_SIZE, determines how often the buffer file

is processed. The default value is 300 seconds. Increasing this number can improve performance by reducing the number of checks.

MAPPING_UPDATE_INTERVAL
: Specifies the number of seconds between updates of the records in the file-handle-to-path mapping tables. The default value is 86400 seconds or one day. This parameter helps keep the file-handle-to-path mapping tables up-to-date without having to continually update the tables.

MAX_LOGS_PRESERVE
: Determines the number of log files to be saved. The default value is 10.

MIN_PROCESSING_SIZE
: Sets the minimum number of bytes that the buffer file must reach before processing and writing to the log file. This parameter, along with IDLE_TIME, determines how often the buffer file is processed. The default value is 524288 bytes. Increasing this number can improve performance by reducing the number of times the buffer file is processed.

PRUNE_TIMEOUT
: Selects the number of hours that must pass before a file-handle-to-path mapping record times out and can be reduced. The default value is 168 hours or 7 days.

UMASK
: Specifies the file mode creation mask for the log files that are created by `nfslogd`. The default value is 0137.

`/etc/nfs/nfslog.conf` File

This file defines the path, file names, and type of logging to be used by `nfslogd`. Each definition is associated with a *tag*. Starting NFS server logging requires that you identify the *tag* for each file system. The global tag defines the default values. You can use the following parameters with each tag as needed.

defaultdir=*path*
: Specifies the default directory path for the logging files. Unless you specify differently, the default directory is `/var/nfs`.

log=*path/filename*
: Sets the path and file name for the log files. The default is `/var/nfs/nfslog`.

fhtable=*path/filename*
: Selects the path and file name for the file-handle-to-path database files. The default is `/var/nfs/fhtable`.

buffer=*path/filename*
: Determines the path and file name for the buffer files. The default is `/var/nfs/nfslog_workbuffer`.

logformat=*basic*|*extended*
: Selects the format to be used when creating user-readable log files. The basic format produces a log file that is similar to some `ftpd` daemons. The extended format gives a more detailed view.

If the path is not specified, the path that is defined by `defaultdir` is used. Also, you can override `defaultdir` by using an absolute path.

To identify the files more easily, place the files in separate directories. Here is an example of the changes that are needed.

```
% cat /etc/nfs/nfslog.conf
#ident  "@(#)nfslog.conf        1.5     99/02/21 SMI"
#
  .
  .
# NFS server log configuration file.
#

global  defaultdir=/var/nfs \
        log=nfslog fhtable=fhtable buffer=nfslog_workbuffer

publicftp log=logs/nfslog fhtable=fh/fhtables buffer=buffers/workbuffer
```

In this example, any file system that is shared with `log=publicftp` uses the following values:

- The default directory is `/var/nfs`.
- Log files are stored in `/var/nfs/logs/nfslog*`.
- File-handle-to-path database tables are stored in `/var/nfs/fh/fhtables`.
- Buffer files are stored in `/var/nfs/buffers/workbuffer`.

For procedural information, refer to "How to Enable NFS Server Logging" on page 32.

NFS Daemons

To support NFS activities, several daemons are started when a system goes into run level `3` or multiuser mode. The `mountd` and `nfsd` daemons are run on systems that are servers. The automatic startup of the server daemons depends on the existence of at least one NFS share. To display the current list of NFS shares, run the `share -F nfs` command. To support NFS file locking, the `lockd` and `statd` daemons are run on NFS clients and servers. However, unlike previous versions of NFS, in NFS version 4, the daemons `lockd`, `statd`, and `nfslogd` are not used.

This section describes the following daemons.

- "`automountd` Daemon" on page 83
- "`lockd` Daemon" on page 84
- "`mountd` Daemon" on page 84

automountd Daemon

This daemon handles the mounting and unmounting requests from the autofs service. The syntax of the command is as follows:

```
automountd [ -Tnv ] [ -D name=value ]
```

The command behaves in the following ways:

- `-T` enables tracing.
- `-n` disables browsing on all autofs nodes.
- `-v` selects to log all status messages to the console.
- `-D` *name=value* substitutes *value* for the automount map variable that is indicated by *name*.

The default value for the automount map is `/etc/auto_master`. Use the `-T` option for troubleshooting.

The same specifications you would make on the command line can be made using the `sharectl` command. However, unlike the command line options, the SMF repository preserves your specifications, through service restarts, system reboots, as well as system upgrades. These are the parameters that can be set for the `automountd` daemon.

automountd_verbose
: Logs status messages to the console and is the equivalent of the `-v` argument for the `automountd` daemon. The default value is FALSE.

nobrowse
: Turns browsing on or off for all autofs mount points and is the equivalent of the `-n` argument for `automountd`. The default value is FALSE.

trace
: Expands each remote procedure call (RPC) and displays the expanded RPC on standard output. This keyword is the equivalent of the `-T` argument for `automountd`. The default value is 0. Values can range from 0 to 5.

environment
: Permits you to assign different values to different environments. This keyword is the equivalent of the `-D` argument for `automountd`. The `environment` parameter can be used multiple times. However, you must use separate entries for each environment assignment.

`lockd` Daemon

This daemon supports record-locking operations on NFS files. The `lockd` daemon manages RPC connections between the client and the server for the Network Lock Manager (NLM) protocol. The daemon is normally started without any options. You can use three options with this command. See the `lockd(1M)` man page. These options can either be used from the command line or setting parameters using the `sharectl` command. The following are descriptions of parameters that can be set.

Note – The `LOCKD_GRACE_PERIOD` keyword and the `-g` option have been deprecated. The deprecated keyword is replaced with the new `grace_period` parameter. If both keywords are set, the value for `grace_period` overrides the value for `LOCKD_GRACE_PERIOD`. See the description of `grace_period` that follows.

Like `LOCKD_GRACE_PERIOD`, the `grace_period=`*graceperiod*parameter sets the number of seconds after a server reboot that the clients have to reclaim both NFS version 3 locks, provided by NLM, and version 4 locks. Thus, the value for `grace_period` controls the length of the grace period for lock recovery, for both NFS version 3 and NFS version 4.

The `lockd_retransmit_timeout=`*timeout* parameter selects the number of seconds to wait before retransmitting a lock request to the remote server. This option affects the NFS client-side service. The default value for *timeout* is 5 seconds. Decreasing the *timeout* value can improve response time for NFS clients on a "noisy" network. However, this change can cause additional server load by increasing the frequency of lock requests. The same parameter can be used from the command line by starting the daemon with the `-t` *timeout* option.

The `lockd_servers=`*number* parameter specifies the maximum number of concurrent `lockd` requests. The default value is 1024.

All NFS clients that use UDP share a single connection with the NFS server. Under these conditions, you might have to increase the number of threads that are available for the UDP connection. A minimum calculation would be to allow two threads for each UDP client. However, this number is specific to the workload on the client, so two threads per client might not be sufficient. The disadvantage to using more threads is that when the threads are used, more memory is used on the NFS server. If the threads are never used, however, increasing *nthreads* has no effect. The same parameter can be used from the command line by starting the daemon with the `nthreads` option.

`mountd` Daemon

This daemon handles file-system mount requests from remote systems and provides access control. The `mountd` daemon checks `/etc/dfs/sharetab` to determine which file systems are available for remote mounting and which systems are allowed to do the remote mounting. You can use the `-v` option and the `-r` option with this command. See the `mountd(1M)` man page.

The `-v` option runs the command in verbose mode. Every time an NFS server determines the access that a client should be granted, a message is printed on the console. The information that is generated can be useful when trying to determine why a client cannot access a file system.

The `-r` option rejects all future mount requests from clients. This option does not affect clients that already have a file system mounted.

In addition to the command line options, several SMF parameters can be used to configure the `mountd` daemon.

client_versmin
: Sets the minimum version of the NFS protocol to be used by the NFS client. The default is 2. Other valid values include 3 or 4. Refer to "Setting Up NFS Services" on page 39.

client_versmax
: Sets the maximum version of the NFS protocol to be used by the NFS client. The default is 4. Other valid values include 2 or 3. Refer to "Setting Up NFS Services" on page 39.

`nfs4cbd` Daemon

`nfs4cbd`, which is for the exclusive use of the NFS version 4 client, manages the communication endpoints for the NFS version 4 callback program. The daemon has no user-accessible interface. For more information, see the `nfs4cbd(1M)` man page.

`nfsd` Daemon

This daemon handles other client file-system requests. You can use several options with this command. See the `nfsd(1M)` man page for a complete listing. These options can either be used from the command line or by setting the appropriate SMF parameter with the `sharectl` command.

The `listen_backlog`=*length* parameter sets the length of the connection queue over connection-oriented transports for NFS and TCP. The default value is 32 entries. The same selection can be made from the command line by starting `nfsd` with the `-l` option.

The `max_connections`=*#-conn* parameter selects the maximum number of connections per connection-oriented transport. The default value for *#-conn* is unlimited. The same parameter can be used from the command line by starting the daemon with the `-c` *#-conn* option.

The `servers`=*nservers* parameter selects the maximum number of concurrent requests that a server can handle. The default value for *nservers* is 1024. The same selection can be made from the command line by starting `nfsd` with the `nservers` option.

Unlike older versions of this daemon, `nfsd` does not spawn multiple copies to handle concurrent requests. Checking the process table with `ps` only shows one copy of the daemon running.

In addition, these SMF parameters can be used to configure the `mountd` daemon. These parameters do not have command line equivalents:

server_versmin
: Sets the minimum version of the NFS protocol to be registered and offered by the server. The default is 2. Other valid values include 3 or 4. Refer to "Setting Up NFS Services" on page 39.

server_versmax
: Sets the maximum version of the NFS protocol to be registered and offered by the server. The default is 4. Other valid values include 2 or 3. Refer to "Setting Up NFS Services" on page 39.

server_delegation
: Controls whether the NFS version 4 delegation feature is enabled for the server. If this feature is enabled, the server attempts to provide delegations to the NFS version 4 client. By default, server delegation is enabled. To disable server delegation, see "How to Select Different Versions of NFS on a Server" on page 41. For more information, refer to "Delegation in NFS Version 4" on page 126.

`nfslogd` Daemon

This daemon provides operational logging. NFS operations that are logged against a server are based on the configuration options that are defined in `/etc/default/nfslogd`. When NFS server logging is enabled, records of all RPC operations on a selected file system are written to a buffer file by the kernel. Then `nfslogd` postprocesses these requests. The name service switch is used to help map UIDs to logins and IP addresses to host names. The number is recorded if no match can be found through the identified name services.

Mapping of file handles to path names is also handled by `nfslogd`. The daemon tracks these mappings in a file-handle-to-path mapping table. One mapping table exists for each tag that is identified in `/etc/nfs/nfslogd`. After post-processing, the records are written to ASCII log files.

Note – NFS version 4 does not use this daemon.

`nfsmapid` Daemon

Version 4 of the NFS protocol (RFC3530) changed the way user or group identifiers (UID or GID) are exchanged between the client and server. The protocol requires that a file's owner and group attributes be exchanged between an NFS version 4 client and an NFS version 4 server as strings in the form of `user@nfsv4_domain` or `group@nfsv4_domain`, respectively.

For example, user `known_user` has a UID 123456 on an NFS version 4 client whose fully qualified hostname is `system.example.com`. For the client to make requests to the NFS version 4 server, the client must map the UID 123456 to `known_user@example.com` and then send this

attribute to the NFS version 4 server. The NFS version 4 server expects to receive user and group file attributes in the `user_or_group@nfsv4_domain` format. After the server receives `known_user@example.com` from the client, the server maps the string to the local UID 123456, which is understood by the underlying file system. This functionality assumes that every UID and GID in the network is unique and that the NFS version 4 domains on the client match the NFS version 4 domains on the server.

Note – If the server does not recognize the given user or group name, even if the NFS version 4 domains match, the server is unable to map the user or group name to its unique ID, an integer value. Under such circumstances, the server maps the inbound user or group name to the `nobody` user. To prevent such occurrences, administrators should avoid making special accounts that only exist on the NFS version 4 client.

The NFS version 4 client and server are both capable of performing integer-to-string and string-to-integer conversions. For example, in response to a GETATTR operation, the NFS version 4 server maps UIDs and GIDs obtained from the underlying file system into their respective string representation and sends this information to the client. Alternately, the client must also map UIDs and GIDs into string representations. For example, in response to the `chown` command, the client maps the new UID or GID to a string representation before sending a SETATTR operation to the server.

Note, however, that the client and server respond differently to unrecognized strings:

- If the user does not exist on the server, even within the same NFS version 4 domain configuration, the server rejects the remote procedure call (RPC) and returns an error message to the client. This situation limits the operations that can be performed by the remote user.
- If the user exists on both the client and server, but they have mismatched domains, the server rejects the attribute modifying operations (such as SETATTR) that require the server to map the inbound user string to an integer value that the underlying file system can understand. For NFS version 4 clients and servers to function properly, their NFS version 4 domains, the portion of the string after the `@` sign, should match.
- If the NFS version 4 client does not recognize a user or group name from the server, the client is unable to map the string to its unique ID, an integer value. Under such circumstances, the client maps the inbound user or group string to the `nobody` user. This mapping to `nobody` creates varied problems for different applications. As for NFS version 4 functionality, operations that modify file attributes will fail.

You can change the domain name for the clients and servers using the `sharectl` command with the following option.

nfsmapid_domain
: Sets a common domain for clients and servers. Overrides the default behavior of using the local DNS domain name. For task information, refer to "Setting Up NFS Services" on

page 39.

Configuration Files and `nfsmapid`

The following describes how the `nfsmapid` daemon uses the SMF configuration information found in `svc:system/name-service/switch` and in `svc:/network/dns/client`:

- `nfsmapid` uses standard C library functions to request password and group information from back-end name services. These name services are controlled by the settings in the `svc:system/name-service/switch` SMF service. Any changes to the service properties affect `nfsmapid` operations. For more information about the `svc:system/name-service/switch` SMF service, see the `nsswitch.conf(4)` man page.
- To ensure that the NFS version 4 clients are capable of mounting file systems from different domains, `nfsmapid` relies on the configuration of the DNS TXT resource record (RR), `_nfsv4idmapdomain`. For more information about configuring the `_nfsv4idmapdomain` resource record, see "`nfsmapid` and DNS TXT Records" on page 89. Also, note the following:
  - The DNS TXT RR should be explicitly configured on the DNS server with the desired domain information.
  - The `svc:system/name-service/switch` SMF service should be configured with the desired parameters to enable the `resolver` to find the DNS server and search the TXT records for client and server NFS version 4 domains.

  For more information, see the following:

  - "Precedence Rules" on page 88
  - "Configuring the NFS Version 4 Default Domain" on page 91
  - `resolv.conf(4)` man page

Precedence Rules

For `nfsmapid` to work properly, NFS version 4 clients and servers must have the same domain. To ensure matching NFS version 4 domains, `nfsmapid` follows these strict precedence rules:

1. The daemon first checks the SMF repository for a value that has been assigned to the `nfsmapid_domain` parameter. If a value is found, the assigned value takes precedence over any other settings. The assigned value is appended to the outbound attribute strings and is compared against inbound attribute strings. For procedural information, see "Setting Up NFS Services" on page 39.

Note – The use of the NFSMAPID_DOMAIN setting is not scalable and is not recommended for large deployments.

---

2. If no value has been assigned to `nfsmapid_domain`, then the daemon checks for a domain name from a DNS TXT RR. `nfsmapid` relies on directives in the `/etc/resolv.conf` file that are used by the set of routines in the `resolver`. The `resolver` searches through the configured DNS servers for the `_nfsv4idmapdomain` TXT RR. Note that the use of DNS TXT records is more scalable. For this reason, continued use of TXT records is much preferred over setting the the parameter in the SMF repository.
3. If no DNS TXT record is configured to provide a domain name, then the `nfsmapid` daemon uses the value specified by the `domain` or `search` directive in the `/etc/resolv.conf` file, with the directive specified last taking precedence.

   In the following example, where both the `domain` and `search` directives are used, the `nfsmapid` daemon uses the first domain listed after the `search` directive, which is `company.com`.

   ```
   domain example.company.com
   search company.com foo.bar.com
   ```
4. If the `/etc/resolv.conf` file does not exist, `nfsmapid` obtains the NFS version 4 domain name by following the behavior of the `domainname` command. Specifically, if the `/etc/defaultdomain` file exists, `nfsmapid` uses the contents of that file for the NFS version 4 domain. If the `/etc/defaultdomain` file does not exist, `nfsmapid` uses the domain name that is provided by the network's configured naming service. For more information, see the `domainname(1M)` man page.

`nfsmapid` and DNS TXT Records

The ubiquitous nature of DNS provides an efficient storage and distribution mechanism for the NFS version 4 domain name. Additionally, because of the inherent scalability of DNS, the use of DNS TXT resource records is the preferred method for configuring the NFS version 4 domain name for large deployments. You should configure the `_nfsv4idmapdomain` TXT record on enterprise-level DNS servers. Such configurations ensure that any NFS version 4 client or server can find its NFS version 4 domain by traversing the DNS tree.

The following is an example of a preferred entry for enabling the DNS server to provide the NFS version 4 domain name:

```
_nfsv4idmapdomain        IN        TXT            "foo.bar"
```

In this example, the domain name to configure is the value that is enclosed in double-quotes. Note that no `ttl` field is specified and that no domain is appended to `_nfsv4idmapdomain`, which is the value in the `owner` field. This configuration enables the TXT record to use the zone's `${ORIGIN}` entry from the Start-Of-Authority (SOA) record. For example, at different levels of the domain namespace, the record could read as follows:

```
_nfsv4idmapdomain.subnet.yourcorp.com.    IN    TXT    "foo.bar"
_nfsv4idmapdomain.yourcorp.com.           IN    TXT    "foo.bar"
```

This configuration provides DNS clients with the added flexibility of using the `resolv.conf` file to search up the DNS tree hierarchy. See the `resolv.conf(4)` man page. This capability provides a higher probability of finding the TXT record. For even more flexibility, lower level DNS sub-domains can define their own DNS TXT resource records (RRs). This capability enables lower level DNS sub-domains to override the TXT record that is defined by the top level DNS domain.

Note – The domain that is specified by the TXT record can be an arbitrary string that does not necessarily match the DNS domain for clients and servers that use NFS version 4. You have the option of not sharing NFS version 4 data with other DNS domains.

Checking for the NFS Version 4 Domain

Before assigning a value for your network's NFS version 4 domain, check to see if an NFS version 4 domain has already been configured for your network. The following examples provide ways of identifying your network's NFS version 4 domain.

- To identify the NFS version 4 domain from a DNS TXT RR, use either the `nslookup` or the `dig` command:

  The following provides sample output for the `nslookup` command:

  ```
  # nslookup -q=txt _nfsv4idmapdomain
  Server:         10.255.255.255
  Address:        10.255.255.255#53

  _nfsv4idmapdomain.example.company.com text = "company.com"
  ```

  See this sample output for the `dig` command:

  ```
  # dig +domain=example.company.com -t TXT _nfsv4idmapdomain
  ...
  ;; QUESTION SECTION:
  ;_nfsv4idmapdomain.example.company.com. IN    TXT

  ;; ANSWER SECTION:
  _nfsv4idmapdomain.example.company.com. 21600 IN TXT   "company.com"

  ;; AUTHORITY SECTION:
  ...
  ```

  For information about setting up a DNS TXT RR, see "`nfsmapid` and DNS TXT Records" on page 89.

- If your network is not setup with a NFS version 4 DNS TXT RR, use the following command to identify your NFS version 4 domain from the DNS domain name:

  ```
  # egrep domain /etc/resolv.conf
  domain example.company.com
  ```

- If the `/etc/resolv.conf` file is not configured to provide a DNS domain name for the client, use the following command to identify the domain from the network's NFS version 4 domain configuration:

```
# cat /system/volatile/nfs4_domain
company.com
```

- If you are using a different naming service, such as NIS, use the following command to identify the domain for the naming service configured for your network:

```
# domainname
it.example.company.com
```

For more information, see the following man pages:

- `nslookup`(1M)
- `dig`(1M)
- `resolv.conf`(4)
- `domainname`(1M)

Configuring the NFS Version 4 Default Domain

This section describes how the network obtains the desired default domain:

- For most current releases, see "Configuring an NFS Version 4 Default Domain in the Oracle Solaris 11 Release" on page 91.
- For the initial Solaris 10 release, see "Configuring an NFS Version 4 Default Domain in the Solaris 10 Release" on page 91.

Configuring an NFS Version 4 Default Domain in the Oracle Solaris 11 Release

In the Oracle Solaris 11 Release, the default NFS domain version can be set using from the command line by typing the following command:

```
# sharectl set -p nfsmapid_domain=example.com nfs
```

Note – Because of the inherent ubiquitous and scalable nature of DNS, the use of DNS TXT records for configuring the domain of large NFS version 4 deployments continues to be preferred and strongly encouraged. See "`nfsmapid` and DNS TXT Records" on page 89.

Configuring an NFS Version 4 Default Domain in the Solaris 10 Release

In the initial Solaris 10 release of NFS version 4, if your network includes multiple DNS domains, but only has a single UID and GID namespace, all clients must use one value for `nfsmapid_domain`. For sites that use DNS, `nfsmapid` resolves this issue by obtaining the domain name from the value that you assigned to `_nfsv4idmapdomain`. For more information, see "`nfsmapid` and DNS TXT Records" on page 89. If your network is not configured to use DNS, during the first system boot the OS uses the `sysidconfig` utility to provide the following prompts for an NFS version 4 domain name:

```
This system is configured with NFS version 4, which uses a
domain name that is automatically derived from the system's
name services. The derived domain name is sufficient for most
```

```
configurations. In a few cases, mounts that cross different
domains might cause files to be owned by nobody due to the
lack of a common domain name.

Do you need to override the system's default NFS verion 4 domain
name (yes/no)? [no]
```

The default response is `[no]`. If you choose `[no]`, you see the following:

```
For more information about how the NFS version 4 default domain name is
derived and its impact, refer to the man pages for nfsmapid(1M) and
nfs(4), and the System Administration Guide: Network Services.
```

If you choose `[yes]`, you see this prompt:

```
Enter the domain to be used as the NFS version 4 domain name.
NFS version 4 domain name []:
```

Note – If a value for `nfsmapid_domain` exists in the SMF repository, the `[domain_name]` that you provide overrides that value.

Additional Information About `nfsmapid`

For more information about `nfsmapid`, see the following:

- `nfsmapid(1M)` man page
- `nfs(4)` man page
- `http://www.ietf.org/rfc/rfc1464.txt`
- "ACLs and `nfsmapid` in NFS Version 4" on page 128

`reparsed` Daemon

The `reparsed` daemon interprets the data associated with a reparse point, which are used by DFS and NFS referrals on SMB and NFS file servers. This service is managed by SMF and should not be manually started.

`statd` Daemon

This daemon works with `lockd` to provide crash and recovery functions for the lock manager. The `statd` daemon tracks the clients that hold locks on an NFS server. If a server crashes, on rebooting `statd` on the server contacts `statd` on the client. The client `statd` can then attempt to reclaim any locks on the server. The client `statd` also informs the server `statd` when a client has crashed so that the client's locks on the server can be cleared. You have no options to select with this daemon. For more information, see the `statd(1M)` man page.

In the Solaris 7 release, the way that `statd` tracks the clients has been improved. In all earlier Solaris releases, `statd` created files in `/var/statmon/sm` for each client by using the client's unqualified host name. This file naming caused problems if you had two clients in different domains that shared a host name, or if clients were not resident in the same domain as the NFS server. Because the unqualified host name only lists the host name, without any domain or IP-address information, the older version of `statd` had no way to differentiate between these types of clients. To fix this problem, the Solaris 7 `statd` creates a symbolic link in `/var/statmon/sm` to the unqualified host name by using the IP address of the client. The new link resembles the following:

```
# ls -l /var/statmon/sm
lrwxrwxrwx   1 daemon          11 Apr 29 16:32 ipv4.192.168.255.255 -> myhost
lrwxrwxrwx   1 daemon          11 Apr 29 16:32 ipv6.fec0::56:a00:20ff:feb9:2734 -> v6host
--w-------   1 daemon          11 Apr 29 16:32 myhost
--w-------   1 daemon          11 Apr 29 16:32 v6host
```

In this example, the client host name is `myhost` and the client's IP address is `192.168.255.255`. If another host with the name `myhost` were mounting a file system, two symbolic links would lead to the host name.

Note – NFS version 4 does not use this daemon.

NFS Commands

These commands must be run as `root` to be fully effective, but requests for information can be made by all users:

- "`automount` Command" on page 94
- "`clear_locks` Command" on page 94
- "`fsstat` Command" on page 95
- "`mount` Command" on page 96
- "`mountall` Command" on page 102
- "`nfsref` Command" on page 111
- "`sharectl` Command" on page 102
- "`share` Command" on page 105
- "`shareall` Command" on page 110
- "`showmount` Command" on page 110
- "`umount` Command" on page 101
- "`umountall` Command" on page 102
- "`unshare` Command" on page 109
- "`unshareall` Command" on page 110

In addition, commands associated with the FedFS service are covered in "FedFS Commands" on page 112.

`automount` Command

This command installs autofs mount points and associates the information in the automaster files with each mount point. The syntax of the command is as follows:

`automount [ -t` *duration* `] [ -v ]`

`-t` *duration* sets the time, in seconds, that a file system is to remain mounted, and `-v` selects the verbose mode. Running this command in the verbose mode allows for easier troubleshooting.

If not specifically set, the value for duration is set to 5 minutes. In most circumstances, this value is good. However, on systems that have many automounted file systems, you might need to increase the duration value. In particular, if a server has many users active, checking the automounted file systems every 5 minutes can be inefficient. Checking the autofs file systems every 1800 seconds, which is 30 minutes, could be more optimal. By not unmounting the file systems every 5 minutes, `/etc/mnttab` can become large. To reduce the output when `df` checks each entry in `/etc/mnttab`, you can filter the output from `df` by using the `-F` option (see the `df`(1M) man page) or by using `egrep`.

You should consider that adjusting the duration also changes how quickly changes to the automounter maps are reflected. Changes cannot be seen until the file system is unmounted. Refer to "Modifying the Maps" on page 51 for instructions on how to modify automounter maps.

The same specifications you would make on the command line can be made using the `sharectl` command. However, unlike the command line options, the SMF repository preserves your specifications, through service restarts, system reboots, as well as system upgrades. These are the parameters that can be set for the `automount` command.

timeout
: Sets the duration for a file system to remain idle before the file system is unmounted. This keyword is the equivalent of the `-t` argument for the `automount` command. The default value is 600.

automount_verbose
: Provides notification of autofs mounts, unmounts, and other nonessential events. This keyword is the equivalent of the `-v` argument for `automount`. The default value is FALSE.

`clear_locks` Command

This command enables you to remove all file, record, and share locks for an NFS client. You must be `root` to run this command. From an NFS server, you can clear the locks for a specific client. From an NFS client, you can clear locks for that client on a specific server. The following example would clear the locks for the NFS client that is named `tulip` on the current system.

```
# clear_locks tulip
```

Using the `-s` option enables you to specify which NFS host to clear the locks from. You must run this option from the NFS client, which created the locks. In this situation, the locks from the client would be removed from the NFS server that is named `bee`.

```
# clear_locks -s bee
```

Caution – This command should only be run when a client crashes and cannot clear its locks. To avoid data corruption problems, do not clear locks for an active client.

`fsstat` Command

The `fsstat` utility enables you to monitor file system operations by file system type and by mount point. Various options allow you to customize the output. See the following examples.

This example shows output for NFS version 3, version 4, and the `root` mount point.

```
% fsstat nfs3 nfs4 /
  new     name    name    attr    attr   lookup   rddir   read   read   write   write
 file    remov   chng     get     set      ops     ops    ops  bytes     ops   bytes
3.81K      90  3.65K   5.89M   11.9K    35.5M   26.6K   109K   118M   35.0K   8.16G  nfs3
  759     503    457   93.6K   1.44K     454K   8.82K  65.4K   827M     292    223K  nfs4
25.2K    18.1K  1.12K   54.7M    1017     259M   1.76M  22.4M  20.1G   1.43M   3.77G  /
```

This example uses the `-i` option to provide statistics about the I/O operations for NFS version 3, version 4, and the `root` mount point.

```
% fsstat -i nfs3 nfs4 /
 read    read    write   write   rddir   rddir   rwlock   rwulock
  ops   bytes      ops   bytes     ops   bytes      ops       ops
 109K    118M    35.0K   8.16G   26.6K   4.45M     170K      170K  nfs3
65.4K    827M      292    223K   8.82K   2.62M    74.1K     74.1K  nfs4
22.4M   20.1G    1.43M   3.77G   1.76M   3.29G    25.5M     25.5M  /
```

This example uses the `-n` option to provide statistics about the naming operations for NFS version 3, version 4, and the `root` mount point.

```
% fsstat -n nfs3 nfs4 /
lookup   creat   remov  link   renam  mkdir  rmdir   rddir  symlnk  rdlnk
 35.5M   3.79K      90     2   3.64K      5      0   26.6K      11   136K  nfs3
  454K     403     503     0     101      0      0   8.82K     356  1.20K  nfs4
  259M   25.2K   18.1K   114    1017     10      2   1.76M      12  8.23M  /
```

For more information, see the `fsstat(1M)` man page.

`mount` Command

With this command, you can attach a named file system, either local or remote, to a specified mount point. For more information, see the `mount(1M)` man page. Used without arguments, `mount` displays a list of file systems that are currently mounted on your computer.

Many types of file systems are included in the standard Oracle Solaris installation. Each file-system type has a specific man page that lists the options to `mount` that are appropriate for that file-system type. The man page for NFS file systems is `mount_nfs(1M)`. For UFS file systems, see `mount_ufs(1M)`.

The Solaris 7 release includes the ability to select a path name to mount from an NFS server by using an NFS URL instead of the standard `server:/pathname` syntax. See "How to Mount an NFS File System Using an NFS URL" on page 37 for further information.

Caution – The version of the `mount` command does not warn about invalid options. The command silently ignores any options that cannot be interpreted. Ensure that you verify all of the options that were used so that you can prevent unexpected behavior.

`mount` Options for NFS File Systems

The subsequent text lists some of the options that can follow the `-o` flag when you are mounting an NFS file system. For a complete list of options, refer to the `mount_nfs(1M)` man page.

bg|fg
: These options can be used to select the retry behavior if a mount fails. The `bg` option causes the mount attempts to be run in the background. The `fg` option causes the mount attempt to be run in the foreground. The default is `fg`, which is the best selection for file systems that must be available. This option prevents further processing until the mount is complete. `bg` is a good selection for noncritical file systems because the client can do other processing while waiting for the mount request to be completed.

forcedirectio
: This option improves performance of large sequential data transfers. Data is copied directly to a user buffer. No caching is performed in the kernel on the client. This option is off by default.

  Previously, all write requests were serialized by both the NFS client and the NFS server. The NFS client has been modified to permit an application to issue concurrent writes, as well as concurrent reads and writes, to a single file. You can enable this functionality on the client by using the `forcedirectio` mount option. When you use this option, you are enabling this functionality for all files within the mounted file system. You could also enable this functionality on a single file on the client by using the `directio()` interface. Unless this functionality has been enabled, writes to files are serialized. Also, if concurrent writes or concurrent reads and writes are occurring, then POSIX semantics are no longer being supported for that file.

For an example of how to use this option, refer to "Using the mount Command" on page 99.

largefiles
: With this option, you can access files that are larger than 2 Gbytes. Whether a large file can be accessed can only be controlled on the server, so this option is silently ignored on NFS version 3 mounts. By default, all UFS file systems are mounted with `largefiles`. For mounts that use the NFS version 2 protocol, the `largefiles` option causes the mount to fail with an error.

nolargefiles
: This option for UFS mounts guarantees that no large files can exist on the file system. See the `mount_ufs(1M)` man page. Because the existence of large files can only be controlled on the NFS server, no option for `nolargefiles` exists when using NFS mounts. Attempts to NFS-mount a file system by using this option are rejected with an error.

nosuid|suid
: The `nosuid` option is the equivalent of specifying the `nodevices` option with the `nosetuid` option. When the `nodevices` option is specified, the opening of device-special files on the mounted file system is disallowed. When the `nosetuid` option is specified, the `setuid` bit and `setgid` bit in binary files that are located in the file system are ignored. The processes run with the privileges of the user who executes the binary file.

  The `suid` option is the equivalent of specifying the `devices` option with the `setuid` option. When the `devices` option is specified, the opening of device-special files on the mounted file system is allowed. When the `setuid` option is specified, the `setuid` bit and the `setgid` bit in binary files that are located in the file system are honored by the kernel.

  If neither option is specified, the default option is `suid`, which provides the default behavior of specifying the `devices` option with the `setuid` option.

  The following table describes the effect of combining `nosuid` or `suid` with `devices` or `nodevices`, and `setuid` or `nosetuid`. Note that in each combination of options, the most restrictive option determines the behavior.

Behavior From the Combined Options	Option	Option	Option
The equivalent of `nosetuid` with `nodevices`	`nosuid`	`nosetuid`	`nodevices`
The equivalent of `nosetuid` with `nodevices`	`nosuid`	`nosetuid`	`devices`
The equivalent of `nosetuid` with `nodevices`	`nosuid`	`setuid`	`nodevices`

Behavior From the Combined Options	Option	Option	Option
The equivalent of `nosetuid` with `nodevices`	`nosuid`	`setuid`	`devices`
The equivalent of `nosetuid` with `nodevices`	`suid`	`nosetuid`	`nodevices`
The equivalent of `nosetuid` with `devices`	`suid`	`nosetuid`	`devices`
The equivalent of `setuid` with `nodevices`	`suid`	`setuid`	`nodevices`
The equivalent of `setuid` with `devices`	`suid`	`setuid`	`devices`

The `nosuid` option provides additional security for NFS clients that access potentially untrusted servers. The mounting of remote file systems with this option reduces the chance of privilege escalation through importing untrusted devices or importing untrusted `setuid` binary files. All these options are available in all Oracle Solaris file systems.

public
: This option forces the use of the public file handle when contacting the NFS server. If the public file handle is supported by the server, the mounting operation is faster because the MOUNT protocol is not used. Also, because the MOUNT protocol is not used, the public option allows mounting to occur through a firewall.

rw|ro
: The `-rw` and `-ro` options indicate whether a file system is to be mounted read-write or read-only. The default is read-write, which is the appropriate option for remote home directories, mail-spooling directories, or other file systems that need to be changed by users. The read-only option is appropriate for directories that should not be changed by users. For example, shared copies of the man pages should not be writable by users.

sec=*mode*
: You can use this option to specify the authentication mechanism to be used during the mount transaction. The value for *mode* can be one of the following.

  - Use `krb5` for Kerberos version 5 authentication service.
  - Use `krb5i` for Kerberos version 5 with integrity.
  - Use `krb5p` for Kerberos version 5 with privacy.
  - Use `none` for no authentication.
  - Use `dh` for Diffie-Hellman (DH) authentication.
  - Use `sys` for standard UNIX authentication.

  The modes are also defined in `/etc/nfssec.conf`.

soft|hard
An NFS file system that is mounted with the `soft` option returns an error if the server does not respond. The `hard` option causes the mount to continue to retry until the server responds. The default is `hard`, which should be used for most file systems. Applications frequently do not check return values from `soft`-mounted file systems, which can make the application fail or can lead to corrupted files. If the application does check the return values, routing problems and other conditions can still confuse the application or lead to file corruption if the `soft` option is used. In most situations, the `soft` option should not be used. If a file system is mounted by using the `hard` option and becomes unavailable, an application that uses this file system hangs until the file system becomes available.

Using the `mount` Command

Refer to the following examples.

- In NFS version 2 or version 3, both of these commands mount an NFS file system from the server bee read-only.

  ```
  # mount -F nfs -r bee:/export/share/man /usr/man
  # mount -F nfs -o ro bee:/export/share/man /usr/man
  ```

  In NFS version 4, the following command line would accomplish the same mount.

  ```
  # mount -F nfs -o vers=4 -r bee:/export/share/man /usr/man
  ```

- In NFS version 2 or version 3, this command uses the `-O` option to force the man pages from the server bee to be mounted on the local system even if `/usr/man` has already been mounted. See the following.

  ```
  # mount -F nfs -O bee:/export/share/man /usr/man
  ```

  In NFS version 4, the following command line would accomplish the same mount.

  ```
  # mount -F nfs -o vers=4 -O bee:/export/share/man /usr/man
  ```

- In NFS version 2 or version 3, this command uses client failover.

  ```
  # mount -F nfs -r bee,wasp:/export/share/man /usr/man
  ```

  In NFS version 4, the following command line uses client failover.

  ```
  # mount -F nfs -o vers=4 -r bee,wasp:/export/share/man /usr/man
  ```

  Note – When used from the command line, the listed servers must support the same version of the NFS protocol. Do not use both version 2 and version 3 servers when running `mount` from the command line. You can use both servers with autofs. Autofs automatically selects the best subset of version 2 or version 3 servers.

- Here is an example of using an NFS URL with the `mount` command in NFS version 2 or version 3.

```
# mount -F nfs nfs://bee//export/share/man /usr/man
```

Here is an example of using an NFS URL with the `mount` command in NFS version 4.

```
# mount -F nfs -o vers=4 nfs://bee//export/share/man /usr/man
```

- Use the `forcedirectio` mount option to enable the client to permit concurrent writes, as well as concurrent reads and writes, to a file. Here is an example.

  ```
  # mount -F nfs -o forcedirectio bee:/home/somebody /mnt
  ```

  In this example, the command mounts an NFS file system from the server bee and enables concurrent reads and writes for each file in the directory `/mnt`. When support for concurrent reads and writes is enabled, the following occurs.

  - The client permits applications to write to a file in parallel.
  - Caching is disabled on the client. Consequently, data from reads and writes is kept on the server. More explicitly, because the client does not cache the data that is read or written, any data that the application does not already have cached for itself is read from the server. The client's operating system does not have a copy of this data. Normally, the NFS client caches data in the kernel for applications to use.

    Because caching is disabled on the client, the read-ahead and write-behind processes are disabled. A read-ahead process occurs when the kernel anticipates the data that an application might request next. The kernel then starts the process of gathering that data in advance. The kernel's goal is to have the data ready before the application makes a request for the data.

    The client uses the write-behind process to increase write throughput. Instead of immediately starting an I/O operation every time an application writes data to a file, the data is cached in memory. Later, the data is written to the disk.

    Potentially, the write-behind process permits the data to be written in larger chunks or to be written asynchronously from the application. Typically, the result of using larger chunks is increased throughput. Asynchronous writes permit overlap between application processing and I/O processing. Also, asynchronous writes permit the storage subsystem to optimize the I/O by providing a better sequencing of the I/O. Synchronous writes force a sequence of I/O on the storage subsystem that might not be optimal.
  - Significant performance degradation can occur if the application is not prepared to handle the semantics of data that is not being cached. Multithreaded applications avoid this problem.

  Note – If support for concurrent writes is not enabled, all write requests are serialized. When requests are serialized, the following occurs. When a write request is in progress, a second write request has to wait for the first write request to be completed before proceeding.

- Use the `mount` command with no arguments to display file systems that are mounted on a client. See the following.

```
% mount
/ on /dev/dsk/c0t3d0s0 read/write/setuid on Wed Apr 7 13:20:47 2004
/usr on /dev/dsk/c0t3d0s6 read/write/setuid on Wed Apr 7 13:20:47 20041995
/proc on /proc read/write/setuid on Wed Apr 7 13:20:47 2004
/dev/fd on fd read/write/setuid on Wed Apr 7 13:20:47 2004
/tmp on swap read/write on Wed Apr 7 13:20:51 2004
/opt on /dev/dsk/c0t3d0s5 setuid/read/write on Wed Apr 7 13:20:51 20041995
/home/kathys on bee:/export/home/bee7/kathys
  intr/noquota/nosuid/remote on Wed Apr 24 13:22:13 2004
```

`umount` Command

This command enables you to remove a remote file system that is currently mounted. The `umount` command supports the `-V` option to allow for testing. You might also use the `-a` option to unmount several file systems at one time. If *mount-points* are included with the `-a` option, those file systems are unmounted. If no mount points are included, an attempt is made to unmount all file systems that are listed in `/etc/mnttab` except for the "required" file systems, such as `/`, `/usr`, `/var`, `/proc`, `/dev/fd`, and `/tmp`. Because the file system is already mounted and should have an entry in `/etc/mnttab`, you do not need to include a flag for the file-system type.

The `-f` option forces a busy file system to be unmounted. You can use this option to unhang a client that is hung while trying to mount an unmountable file system.

Caution – By forcing an unmount of a file system, you can cause data loss if files are being written to.

See the following examples.

EXAMPLE 3–1 Unmounting a File System

This example unmounts a file system that is mounted on `/usr/man`:

```
# umount /usr/man
```

EXAMPLE 3–2 Using Options with `umount`

This example displays the results of running `umount -a -V`:

```
# umount -a -V
umount /home/kathys
umount /opt
umount /home
umount /net
```

Notice that this command does not actually unmount the file systems.

`mountall` Command

Use this command to mount all file systems or a specific group of file systems that are listed in a file-system table. The command provides a way of doing the following:

- Selecting the file-system type to be accessed with the `-F` *FSType* option
- Selecting all the remote file systems that are listed in a file-system table with the `-r` option
- Selecting all the local file systems with the `-l` option

Because all file systems that are labeled as NFS file-system type are remote file systems, some of these options are redundant. For more information, see the `mountall(1M)` man page.

Note that the following two examples of user input are equivalent:

```
# mountall -F nfs
```

```
# mountall -F nfs -r
```

`umountall` Command

Use this command to unmount a group of file systems. The `-k` option runs the `fuser -k` *mount-point* command to kill any processes that are associated with the *mount-point*. The `-s` option indicates that unmount is not to be performed in parallel. `-l` specifies that only local file systems are to be used, and `-r` specifies that only remote file systems are to be used. The `-h` *host* option indicates that all file systems from the named host should be unmounted. You cannot combine the `-h` option with `-l` or `-r`.

The following is an example of unmounting all file systems that are mounted from remote hosts:

```
# umountall -r
```

The following is an example of unmounting all file systems that are currently mounted from the server bee:

```
# umountall -h bee
```

`sharectl` Command

This release includes the `sharectl` utility, which is an administrative tool that enables you to configure and manage file-sharing protocols, such as NFS. You can use this command to do the following:

- Set client and server operational properties
- Display property values for a specific protocol

- Obtain the status of a protocol

The `sharectl` utility uses the following syntax:

```
# sharectl subcommand [option] [protocol]
```

The `sharectl` utility supports the following subcommands:

TABLE 3–2 Subcommands for `sharectl` Utility

Subcommand	Description
`set`	Defines the properties for a file-sharing protocol. For a list of properties and property values, see the parameters described in the `nfs(4)` man page.
`get`	Displays the properties and property values for the specified protocol.
`status`	Displays whether the specified protocol is enabled or disabled. If no protocol is specified, the status of all file-sharing protocols is displayed.

For more information about the `sharectl` utility, see the following:

- `sharectl`(1M) man page
- "`set` Subcommand" on page 103
- "`get` Subcommand" on page 104
- "`status` Subcommand" on page 104

`set` Subcommand

The `set` subcommand, which defines the properties for a file-sharing protocol, supports the following options:

`-h` Provides an online-help description

`-p` Defines a property for the protocol

The `set` subcommand uses the following syntax:

```
# sharectl set [-h] [-p property=value] protocol
```

Note –

- You must have `root` privileges to use the `set` subcommand.
- You do not need to repeat this command-line syntax for each additional property value. You can use the `-p` option multiple times to define multiple properties on the same command line.

The following example sets the minimum version of the NFS protocol for the client to `3`:

```
# sharectl set -p client_versmin=3 nfs
```

`get` Subcommand

The `get` subcommand, which displays the properties and property values for the specified protocol, supports the following options:

`-h` Provides an online-help description.

`-p` Identifies the property value for the specified property. If the `-p` option is not used, all property values are displayed.

The `get` subcommand uses the following syntax:

```
# sharectl get [-h] [-p property] protocol
```

Note – You must have `root` privileges to use the `get` subcommand.

The following example uses `servers`, which is the property that enables you to specify the maximum number of concurrent NFS requests:

```
# sharectl get -p servers nfs
servers=1024
```

In the following example, because the `-p` option is not used, all property values are displayed:

```
# sharectl get nfs
servers=1024
listen_backlog=32
protocol=ALL
servers=32
lockd_listen_backlog=32
lockd_servers=20
lockd_retransmit_timeout=5
grace_period=90
nfsmapid_domain=company.com
server_versmin=2
server_versmax=4
client_versmin=2
client_versmax=4
server_delegation=on
max_connections=-1
device=
```

`status` Subcommand

The `status` subcommand, which displays whether the specified protocol is enabled or disabled, supports the following option:

`-h` Provides an online-help description

The `status` subcommand uses the following syntax:

```
# sharectl status [-h] [protocol]
```

The following example shows the status of the NFS protocol:

```
# sharectl status nfs
nfs       enabled
```

`share` Command

With this command, you can make a local file system on an NFS server available for mounting. You can also use the `share` command to display a list of the file systems on your system that are currently shared. The NFS server must be running for the `share` command to work.

The objects that can be shared include any directory tree. However, each file system hierarchy is limited by the disk slice or partition that the file system is located on.

A file system cannot be shared if that file system is part of a larger file system that is already being shared. For example, if `/usr` and `/usr/local` are on one disk slice, `/usr` can be shared or `/usr/local` can be shared. However, if both directories need to be shared with different share options, `/usr/local` must be moved to a separate disk slice.

You can gain access to a file system that is read-only shared through the file handle of a file system that is read-write shared. However, the two file systems have to be on the same disk slice. You can create a more secure situation. Place those file systems that need to be read-write on a separate partition or separate disk slice from the file systems that you need to share as read-only.

Note – For information about how NFS version 4 functions when a file system is unshared and then reshared, refer to "Unsharing and Resharing a File System in NFS Version 4" on page 120.

Non-File-System-Specific `share` Options

Some of the options that you can include with the `-o` flag are as follows.

rw|ro
: The *pathname* file system is shared read-write or read-only for all clients.

rw=*accesslist*
: The file system is shared read-write for the clients that are listed only. All other requests are denied. Starting with the Solaris 2.6 release, the list of clients that are defined in *accesslist* has been expanded. See "Setting Access Lists With the `share` Command" on page 108 for more information. You can use this option to override an `-ro` option.

NFS-Specific `share` Options

The options that you can use with NFS file systems include the following.

aclok
: This option enables an NFS server that supports the NFS version 2 protocol to be configured to do access control for NFS version 2 clients. Without this option, all clients are given minimal access. With this option, the clients have maximal access. For instance, on file systems that are shared with the `-aclok` option, if anyone has read permissions, everyone does. However, without this option, you can deny access to a client who should have access permissions. A decision to permit too much access or too little access depends on the security systems already in place. See "Using Access Control Lists to Protect UFS Files" in *Oracle Solaris 11.1 Administration: Security Services* for more information about access control lists (ACLs).

Note – To use ACLs, ensure that clients and servers run software that supports the NFS version 3 and NFS_ACL protocols. If the software only supports the NFS version 3 protocol, clients obtain correct access but cannot manipulate the ACLs. If the software supports the NFS_ACL protocol, the clients obtain correct access and can manipulate the ACLs.

---

anon=*uid*
: You use *uid* to select the user ID of unauthenticated users. If you set *uid* to `-1`, the server denies access to unauthenticated users. You can grant root access by setting `anon=0`, but this option allows unauthenticated users to have root access, so use the `root` option instead.

index=*filename*
: When a user accesses an NFS URL, the `-index=`*filename* option forces the HTML file to load, instead of displaying a list of the directory. This option mimics the action of current browsers if an `index.html` file is found in the directory that the HTTP URL is accessing. This option is the equivalent of setting the `DirectoryIndex` option for `httpd`. For instance, suppose that `share` command reports the following:

```
export_web /export/web   nfs sec=sys,public,index=index.html,ro
```

These URLs then display the same information:

```
nfs://<server>/<dir>
nfs://<server>/<dir>/index.html
nfs://<server>//export/web/<dir>
nfs://<server>//export/web/<dir>/index.html
http://<server>/<dir>
http://<server>/<dir>/index.html
```

log=*tag*
: This option specifies the tag in `/etc/nfs/nfslog.conf` that contains the NFS server logging configuration information for a file system. This option must be selected to enable NFS server logging.

nosuid
: This option signals that all attempts to enable the `setuid` or `setgid` mode should be ignored. NFS clients cannot create files with the `setuid` or `setgid` bits on.

public
: The `-public` option has been added to the `share` command to enable WebNFS browsing. Only one file system on a server can be shared with this option.

root=*accesslist*
: The server gives root access to the hosts in the list. By default, the server does not give root access to any remote hosts. If the selected security mode is anything other than `-sec=sys`, you can only include client host names in the *accesslist*. Starting with the Solaris 2.6 release, the list of clients that are defined in *accesslist* is expanded. See "Setting Access Lists With the `share` Command" on page 108 for more information.

Caution – Granting root access to other hosts has wide security implications. Use the `-root=` option with extreme caution.

root=*client-name*
: The *client-name* value is used with AUTH_SYS authentication to check the client's IP address against a list of addresses provided by `exportfs`(1B). If a match is found, `root` access is given to the file systems being shared.

root=*host-name*
: For secure NFS modes, such as AUTH_SYS or RPCSEC_GSS, the server checks the clients' principal names against a list of host-based principal names that are derived from an access list. The generic syntax for the client's principal name is `root@`*hostname*. For Kerberos V the syntax is `root/`*hostname.fully.qualified*`@REALM`. When you use the *host-name* value, the clients on the access list must have the credentials for a principal name. For Kerberos V, the client must have a valid keytab entry for its `root/`*hostname.fully.qualified*`@REALM` principal name. For more information, see "Configuring Kerberos Clients" in *Oracle Solaris 11.1 Administration: Security Services*.

sec=*mode*[:*mode*]
: *mode* selects the security modes that are needed to obtain access to the file system. By default, the security mode is UNIX authentication. You can specify multiple modes, but use each security mode only once per command line. Each `-mode` option applies to any subsequent `-rw`, `-ro`, `-rw=`, `-ro=`, `-root=`, and `-window=` options until another `-mode` is encountered. The use of `-sec=none` maps all users to user `nobody`.

window=*value*
: *value* selects the maximum lifetime in seconds of a credential on the NFS server. The default value is 30000 seconds or 8.3 hours.

Setting Access Lists With the `share` Command

The *accesslist* can include a domain name, a subnet number, or an entry to deny access, as well as the standard `-ro=`, `-rw=`, or `-root=` options. These extensions should simplify file access control on a single server without having to change the namespace or maintain long lists of clients.

This command provides read-only access for most systems but allows read-write access for `rose` and `lilac`:

```
# share -F nfs -o ro,rw=rose:lilac /usr/src
```

In the next example, read-only access is assigned to any host in the `eng` netgroup. The client `rose` is specifically given read-write access.

```
# share -F nfs -o ro=eng,rw=rose /usr/src
```

Note – You cannot specify both `rw` and `ro` without arguments. If no read-write option is specified, the default is read-write for all clients.

To share one file system with multiple clients, you must type all options on the same line. Multiple invocations of the `share` command on the same object "remember" only the last command that is run. This command enables read-write access to three client systems, but only `rose` and `tulip` are given access to the file system as `root`.

```
# share -F nfs -o rw=rose:lilac:tulip,root=rose:tulip /usr/src
```

When sharing a file system that uses multiple authentication mechanisms, ensure that you include the `-ro`, `-ro=`, `-rw`, `-rw=`, `-root`, and `-window` options after the correct security modes. In this example, UNIX authentication is selected for all hosts in the netgroup that is named `eng`. These hosts can only mount the file system in read-only mode. The hosts `tulip` and `lilac` can mount the file system read-write if these hosts use Diffie-Hellman authentication. With these options, `tulip` and `lilac` can mount the file system read-only even if these hosts are not using DH authentication. However, the host names must be listed in the `eng` netgroup.

```
# share -F nfs -o sec=dh,rw=tulip:lilac,sec=sys,ro=eng /usr/src
```

Even though UNIX authentication is the default security mode, UNIX authentication is not included if the `-sec` option is used. Therefore, you must include a `-sec=sys` option if UNIX authentication is to be used with any other authentication mechanism.

You can use a DNS domain name in the access list by preceding the actual domain name with a dot. The string that follows the dot is a domain name, not a fully qualified host name. The following entry allows mount access to all hosts in the `eng.example.com` domain:

```
# share -F nfs -o ro=.:.eng.example.com /export/share/man
```

In this example, the single "." matches all hosts that are matched through the NIS namespace. The results that are returned from these name services do not include the domain name. The ".eng.example.com" entry matches all hosts that use DNS for namespace resolution. DNS always returns a fully qualified host name. So, the longer entry is required if you use a combination of DNS and the other namespaces.

You can use a subnet number in an access list by preceding the actual network number or the network name with "@". This character differentiates the network name from a netgroup or a fully qualified host name. You must identify the subnet in either /etc/networks or in an NIS namespace. The following entries have the same effect if the 192.168 subnet has been identified as the eng network:

```
# share -F nfs -o ro=@eng /export/share/man
# share -F nfs -o ro=@192.168 /export/share/man
# share -F nfs -o ro=@192.168.0.0 /export/share/man
```

The last two entries show that you do not need to include the full network address.

If the network prefix is not byte aligned, as with Classless Inter-Domain Routing (CIDR), the mask length can be explicitly specified on the command line. The mask length is defined by following either the network name or the network number with a slash and the number of significant bits in the prefix of the address. For example:

```
# share -f nfs -o ro=@eng/17 /export/share/man
# share -F nfs -o ro=@192.168.0/17 /export/share/man
```

In these examples, the "/17" indicates that the first 17 bits in the address are to be used as the mask. For additional information about CIDR, look up RFC 1519.

You can also select negative access by placing a "-" before the entry. Note that the entries are read from left to right. Therefore, you must place the negative access entries before the entry that the negative access entries apply to:

```
# share -F nfs -o ro=-rose:.eng.example.com /export/share/man
```

This example would allow access to any hosts in the eng.example.com domain except the host that is named rose.

unshare Command

This command allows you to make a previously available file system unavailable for mounting by clients. When you unshare an NFS file system, access from clients with existing mounts is inhibited. The file system might still be mounted on the client, but the files are not accessible. The unshare command deletes the share permanently unless the -t option is used to temporarily unshare the file system.

Note – For information about how NFS version 4 functions when a file system is unshared and then reshared, refer to "Unsharing and Resharing a File System in NFS Version 4" on page 120.

The following is an example of unsharing a specific file system:

```
# unshare /usr/src
```

`shareall` Command

This command allows for multiple file systems to be shared. When used with no options, the command shares all entries in the SMF repository. You can include a file name to specify the name of a file that lists `share` command lines.

The following is an example of sharing all file systems that are listed in a local file:

```
# shareall /etc/dfs/special_dfstab
```

`unshareall` Command

This command makes all currently shared resources unavailable. The `-F` *FSType* option selects a list of file-system types that are defined in `/etc/dfs/fstypes`. This flag enables you to choose only certain types of file systems to be unshared. The default file-system type is defined in `/etc/dfs/fstypes`. To choose specific file systems, use the `unshare` command.

The following is an example of unsharing all NFS-type file systems:

```
# unshareall -F nfs
```

`showmount` Command

This command displays one of the following:

- All clients that have remotely mounted file systems that are shared from an NFS server
- Only the file systems that are mounted by clients
- The shared file systems with the client access information

Note – The `showmount` command only shows NFS version 2 and version 3 exports. This command does not show NFS version 4 exports.

The command syntax is as follows:

`showmount [ -ade ] [ hostname ]`

`-a`	Prints a list of all the remote mounts. Each entry includes the client name and the directory.
`-d`	Prints a list of the directories that are remotely mounted by clients.
`-e`	Prints a list of the files that are shared or are exported.
hostname	Selects the NFS server to gather the information from.

If *hostname* is not specified, the local host is queried.

The following command lists all clients and the local directories that the clients have mounted:

```
# showmount -a bee
lilac:/export/share/man
lilac:/usr/src
rose:/usr/src
tulip:/export/share/man
```

The following command lists the directories that have been mounted:

```
# showmount -d bee
/export/share/man
/usr/src
```

The following command lists file systems that have been shared:

```
# showmount -e bee
/usr/src                                (everyone)
/export/share/man                    eng
```

The `nfs_props/showmount_info` property of the `/network/nfs/server:default` service controls how much information is displayed to a client by the `showmount` command. The default value is `full`. If this value is set to `none` then the client will only see those remote file systems on the server that the client can mount. No information about other clients is displayed. See Example 2–3 for instructions on how to change this property.

`nfsref` Command

The `nfsref` command is used to add, delete or list NFSv4 referrals. The command syntax is as follows:

`nfsref add` *path location* [*location* …]

`nfsref remove` *path*

`nfsref lookup` *path*

path Selects the name for the reparse point.

location Identifies one or more NFS or SMB shared file systems to be associated with the reparse point.

FedFS Commands

These are the commands associated with the FedFS service:

`nsdb-list` Lists all FedFS data stored in the LDAP server.

`nsdb-nces` Lists the naming contexts on the LDAP server and the relative distinguished name.

`nsdb-resolve-fsn` Shows the fileset location for the selected fileset name.

`nsdb-update-nci` Manages distinguished names for FedFS data.

`nsdbparams` Manages FedFS connections.

For examples of how these commands are used, see "Administering FedFS" on page 63.

Commands for Troubleshooting NFS Problems

These commands can be useful when troubleshooting NFS problems.

`nfsstat` Command

You can use this command to gather statistical information about NFS and RPC connections. The syntax of the command is as follows:

```
nfsstat [ -cmnrsz ]
```

`-c` Displays client-side information

`-m` Displays statistics for each NFS-mounted file system

`-n` Specifies that NFS information is to be displayed on both the client side and the server side

`-r` Displays RPC statistics

`-s` Displays the server-side information

`-z` Specifies that the statistics should be set to zero

If no options are supplied on the command line, the `-cnrs` options are used.

Gathering server-side statistics can be important for debugging problems when new software or new hardware is added to the computing environment. Running this command a minimum of once a week, and storing the numbers, provides a good history of previous performance.

Refer to the following example:

```
# nfsstat -s

Server rpc:
Connection oriented:
calls      badcalls   nullrecv   badlen     xdrcall    dupchecks  dupreqs
719949194  0          0          0          0          58478624   33
Connectionless:
calls      badcalls   nullrecv   badlen     xdrcall    dupchecks  dupreqs
73753609   0          0          0          0          987278     7254

Server NFSv2:
calls      badcalls   referrals  referlinks
25733      0          0          0

Server NFSv3:
calls      badcalls   referrals  referlinks
132880073  0          0          0

Server NFSv4:
calls      badcalls   referrals  referlinks
488884996  4          0          0
Version 2: (746607 calls)
null       getattr    setattr    root       lookup     readlink   read
883 0%     60 0%      45 0%      0 0%       177446 23% 1489 0%    537366 71%
wrcache    write      create     remove     rename     link       symlink
0 0%       1105 0%    47 0%      59 0%      28 0%      10 0%      9 0%
mkdir      rmdir      readdir    statfs
26 0%      0 0%       27926 3%   108 0%
Version 3: (728863853 calls)
null          getattr       setattr       lookup        access
1365467 0%    496667075 68% 8864191 1%    66510206 9%   19131659 2%
readlink      read          write         create        mkdir
414705 0%     80123469 10%  18740690 2%   4135195 0%    327059 0%
symlink       mknod         remove        rmdir         rename
101415 0%     9605 0%       6533288 0%    111810 0%     366267 0%
link          readdir       readdirplus   fsstat        fsinfo
2572965 0%    519346 0%     2726631 0%    13320640 1%   60161 0%
pathconf      commit
13181 0%      6248828 0%
Version 4: (54871870 calls)
null               compound
266963 0%          54604907 99%
Version 4: (167573814 operations)
reserved           access             close              commit
0 0%               2663957 1%         2692328 1%         1166001 0%
create             delegpurge         delegreturn        getattr
167423 0%          0 0%               1802019 1%         26405254 15%
getfh              link               lock               lockt
11534581 6%        113212 0%          207723 0%          265 0%
locku              lookup             lookupp            nverify
```

```
230430 0%           11059722 6%         423514 0%           21386866 12%
open                openattr            open_confirm        open_downgrade
2835459 1%          4138 0%             18959 0%            3106 0%
putfh               putpubfh            putrootfh           read
52606920 31%        0 0%                35776 0%            4325432 2%
readdir             readlink            remove              rename
606651 0%           38043 0%            560797 0%           248990 0%
renew               restorefh           savefh              secinfo
2330092 1%          8711358 5%          11639329 6%         19384 0%
setattr             setclientid         setclientid_confirm verify
453126 0%           16349 0%            16356 0%            2484 0%
write               release_lockowner   illegal
3247770 1%          0 0%                0 0%

Server nfs_acl:
Version 2: (694979 calls)
null        getacl      setacl      getattr     access      getxattrdir
0 0%        42358 6%    0 0%        584553 84%  68068 9%    0 0%
Version 3: (2465011 calls)
null        getacl      setacl      getxattrdir
0 0%        1293312 52% 1131 0%     1170568 47%
```

The previous listing is an example of NFS server statistics. The first five lines relate to RPC and the remaining lines report NFS activities. In both sets of statistics, knowing the average number of `badcalls` or `calls` and the number of calls per week can help identify a problem. The `badcalls` value reports the number of bad messages from a client. This value can indicate network hardware problems.

Some of the connections generate write activity on the disks. A sudden increase in these statistics could indicate trouble and should be investigated. For NFS version 2 statistics, the connections to note are `setattr`, `write`, `create`, `remove`, `rename`, `link`, `symlink`, `mkdir`, and `rmdir`. For NFS version 3 and version 4 statistics, the value to watch is `commit`. If the `commit` level is high in one NFS server, compared to another almost identical server, check that the NFS clients have enough memory. The number of `commit` operations on the server grows when clients do not have available resources.

`pstack` Command

This command displays a stack trace for each process. The `pstack` command must be run by the owner of the process or by `root`. You can use `pstack` to determine where a process is hung. The only option that is allowed with this command is the `PID` of the process that you want to check. See the `proc(1)` man page.

The following example is checking the `nfsd` process that is running.

```
# /usr/bin/pgrep nfsd
243
# /usr/bin/pstack 243
243:    /usr/lib/nfs/nfsd -a 16
```

```
ef675c04 poll     (24d50, 2, ffffffff)
000115dc ???????? (24000, 132c4, 276d8, 1329c, 276d8, 0)
00011390 main     (3, effffff14, 0, 0, ffffffff, 400) + 3c8
00010fb0 _start   (0, 0, 0, 0, 0, 0) + 5c
```

The example shows that the process is waiting for a new connection request, which is a normal response. If the stack shows that the process is still in poll after a request is made, the process might be hung. Follow the instructions in "How to Restart NFS Services" on page 69 to fix this problem. Review the instructions in "NFS Troubleshooting Procedures" on page 65 to fully verify that your problem is a hung program.

`rpcinfo` Command

This command generates information about the RPC service that is running on a system. You can also use this command to change the RPC service. Many options are available with this command. See the `rpcinfo(1M)` man page. The following is a shortened synopsis for some of the options that you can use with the command.

`rpcinfo [ -m | -s ] [` *hostname* `]`

`rpcinfo -T` *transport hostname* `[` *progname* `]`

`rpcinfo [ -t | -u ] [` *hostname* `] [` *progname* `]`

`-m`	Displays a table of statistics of the `rpcbind` operations
`-s`	Displays a concise list of all registered RPC programs
`-T`	Displays information about services that use specific transports or protocols
`-t`	Probes the RPC programs that use TCP
`-u`	Probes the RPC programs that use UDP
transport	Selects the transport or protocol for the services
hostname	Selects the host name of the server that you need information from
progname	Selects the RPC program to gather information about

If no value is given for *hostname*, the local host name is used. You can substitute the RPC program number for *progname*, but many users can remember the name and not the number. You can use the `-p` option in place of the `-s` option on those systems that do not run the NFS version 3 software.

The data that is generated by this command can include the following:

- The RPC program number
- The version number for a specific program

- The transport protocol that is being used
- The name of the RPC service
- The owner of the RPC service

The following example gathers information about the RPC services that are running on a server. The text that is generated by the command is filtered by the `sort` command to make the output more readable. Several lines that list RPC services have been deleted from the example.

```
% rpcinfo -s bee |sort -n
   program version(s) netid(s)                         service     owner
    100000  2,3,4     udp6,tcp6,udp,tcp,ticlts,ticotsord,ticots portmapper  superuser
    100001  4,3,2     udp6,udp,ticlts                   rstatd      superuser
    100003  4,3,2     tcp,udp,tcp6,udp6                 nfs         1
    100005  3,2,1     ticots,ticotsord,tcp,tcp6,ticlts,udp,udp6 mountd      superuser
    100007  1,2,3     ticots,ticotsord,ticlts,tcp,udp,tcp6,udp6 ypbind      1
    100011  1         udp6,udp,ticlts                   rquotad     superuser
    100021  4,3,2,1   tcp,udp,tcp6,udp6                 nlockmgr    1
    100024  1         ticots,ticotsord,ticlts,tcp,udp,tcp6,udp6 status      superuser
    100068  5,4,3,2   ticlts                           -           superuser
    100083  1         ticotsord                         -           superuser
    100133  1         ticots,ticotsord,ticlts,tcp,udp,tcp6,udp6 -           superuser
    100134  1         ticotsord                         -           superuser
    100155  1         ticotsord                         smserverd   superuser
    100169  1         ticots,ticotsord,ticlts           -           superuser
    100227  3,2       tcp,udp,tcp6,udp6                 nfs_acl     1
    100234  1         ticotsord                         -           superuser
    390113  1         tcp                               -           superuser
    390435  1         tcp                               -           superuser
    390436  1         tcp                               -           superuser
1073741824  1         tcp,tcp6                          -           1
```

The following two examples show how to gather information about a particular RPC service by selecting a particular transport on a server. The first example checks the `mountd` service that is running over TCP. The second example checks the NFS service that is running over UDP.

```
% rpcinfo -t bee mountd
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting
program 100005 version 3 ready and waiting
% rpcinfo -u bee nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
```

`snoop` Command

This command is often used to watch for packets on the network. The `snoop` command must be run as `root`. The use of this command is a good way to ensure that the network hardware is functioning on both the client and the server. Many options are available. See the `snoop(1M)` man page. A shortened synopsis of the command follows:

`snoop [ -d` *device* `] [ -o` *filename* `] [ host` *hostname* `]`

`-d` *device*	Specifies the local network interface
`-o` *filename*	Stores all the captured packets into the named file
hostname	Displays packets going to and from a specific host only

The `-d` *device* option is useful on those servers that have multiple network interfaces. You can use many expressions other than setting the host. A combination of command expressions with `grep` can often generate data that is specific enough to be useful.

When troubleshooting, make sure that packets are going to and from the proper host. Also, look for error messages. Saving the packets to a file can simplify the review of the data.

`truss` Command

You can use this command to check if a process is hung. The `truss` command must be run by the owner of the process or by `root`. You can use many options with this command. See the `truss(1)` man page. A shortened syntax of the command follows.

`truss [ -t` *syscall* `] -p` *pid*

`-t` *syscall*	Selects system calls to trace
`-p` *pid*	Indicates the PID of the process to be traced

The *syscall* can be a comma-separated list of system calls to be traced. Also, starting *syscall* with an `!` selects to exclude the listed system calls from the trace.

This example shows that the process is waiting for another connection request from a new client.

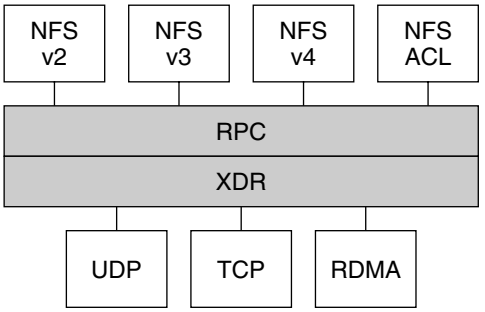
```
# /usr/bin/truss -p 243
poll(0x00024D50, 2, -1)         (sleeping...)
```

The previous example shows a normal response. If the response does not change after a new connection request has been made, the process could be hung. Follow the instructions in "How to Restart NFS Services" on page 69 to fix the hung program. Review the instructions in "NFS Troubleshooting Procedures" on page 65 to fully verify that your problem is a hung program.

NFS Over RDMA

Starting in the Oracle Solaris 11.1 release, the default transport for NFS is the Remote Direct Memory Access (RDMA) protocol, which is a technology for memory-to-memory transfer of data over high-speed networks. Specifically, RDMA provides remote data transfer directly to and from memory without CPU intervention. RDMA also provides direct data placement, which eliminates data copies and, therefore, further eliminates CPU intervention. Thus, RDMA relieves not only the host CPU, but also reduces contention for the host memory and I/O buses. To provide this capability, RDMA combines the interconnect I/O technology of InfiniBand on SPARC platforms with the Oracle Solaris operating system. The following figure shows the relationship of RDMA to other protocols, such as UDP and TCP.

FIGURE 3–1 Relationship of RDMA to Other Protocols



NFS is a family of protocols layered over RPC. The XDR (eXternal Data Representation) layer encodes RPC arguments and RPC results onto one of several RPC transports, such as UDP, TCP, and RDMA.

Because RDMA is the default transport protocol for NFS, no special `share` or `mount` options are required to use RDMA on a client or server. The existing automounter maps, vfstab and file system shares, work with the RDMA transport. NFS mounts over the RDMA transport occur transparently when InfiniBand connectivity exists on SPARC platforms between the client and the server. If the RDMA transport is not available on both the client and the server, the TCP transport is the initial fallback, followed by UDP if TCP is unavailable. Note, however, that if you use the `proto=rdma` mount option, NFS mounts are forced to use RDMA only.

To specify that TCP and UDP be used only, you can use the `proto=tcp/udp` `mount` option. This option disables RDMA on an NFS client. For more information about NFS mount options, see the `mount_nfs(1M)` man page and "`mount` Command" on page 96.

Note – RDMA for InfiniBand uses the IP addressing format and the IP lookup infrastructure to specify peers. However, because RDMA is a separate protocol stack, it does not fully implement all IP semantics. For example, RDMA does not use IP addressing to communicate with peers. Therefore, RDMA might bypass configurations for various security policies that are based on IP addresses. However, the NFS and RPC administrative policies, such as `mount` restrictions and secure RPC, are not bypassed.

How the NFS Service Works

The following sections describe some of the complex functions of the NFS software. Note that some of the feature descriptions in this section are exclusive to NFS version 4.

Note – If your system has zones enabled and you want to use this feature in a non-global zone, see *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management* for more information.

Version Negotiation in NFS

The NFS initiation process includes negotiating the protocol levels for servers and clients. If you do not specify the version level, then the best level is selected by default. For example, if both the client and the server can support version 3, then version 3 is used. If the client or the server can only support version 2, then version 2 is used.

You can set the `client_versmin`, `client_versmax`, `server_versmin` and `server_versmax` parameters using the `sharectl` command. Your specified minimum and maximum values for the server and the client would replace the default values for these keywords. For both the client and the server the default minimum value is 2 and the default maximum value is 4. To find the

version supported by the server, the NFS client begins with the setting for `client_versmax` and continues to try each version until reaching the version setting for `client_versmin`. As soon as the supported version is found, the process terminates. For example, if `client_versmax=4` and `client_versmin=2`, then the client attempts version 4 first, then version 3, and finally version 2. If `client_versmax` and `client_versmax` are set to the same value, then the client always uses this version and does not attempt any other version. If the server does not offer this version, the mount fails.

Note – You can override the values that are determined by the negotiation by using the `vers` option with the `mount` command. See the `mount_nfs(1M)` man page.

For procedural information, refer to "Setting Up NFS Services" on page 39.

Features in NFS Version 4

Many changes have been made to NFS in version 4. This section provides descriptions of these new features.

Note – Starting in the Solaris 10 release, NFS version 4 does not support the LIPKEY/SPKM security flavor. Also, NFS version 4 does not use the `mountd`, `nfslogd`, and `statd` daemons.

For procedural information related to using NFS version 4, refer to "Setting Up NFS Services" on page 39.

Unsharing and Resharing a File System in NFS Version 4

With both NFS version 3 and version 4, if a client attempts to access a file system that has been unshared, the server responds with an error code. However, with NFS version 3 the server maintains any locks that the clients had obtained before the file system was unshared. Thus, when the file system is reshared, NFS version 3 clients can access the file system as though that file system had never been unshared.

With NFS version 4, when a file system is unshared, all the state for any open files or file locks in that file system is destroyed. If the client attempts to access these files or locks, the client receives an error. This error is usually reported as an `I/O` error to the application. Note, however, that resharing a currently shared file system to change options does not destroy any of the state on the server.

For related information, refer to "Client Recovery in NFS Version 4" on page 124 or see the `unshare_nfs(1M)` man page.

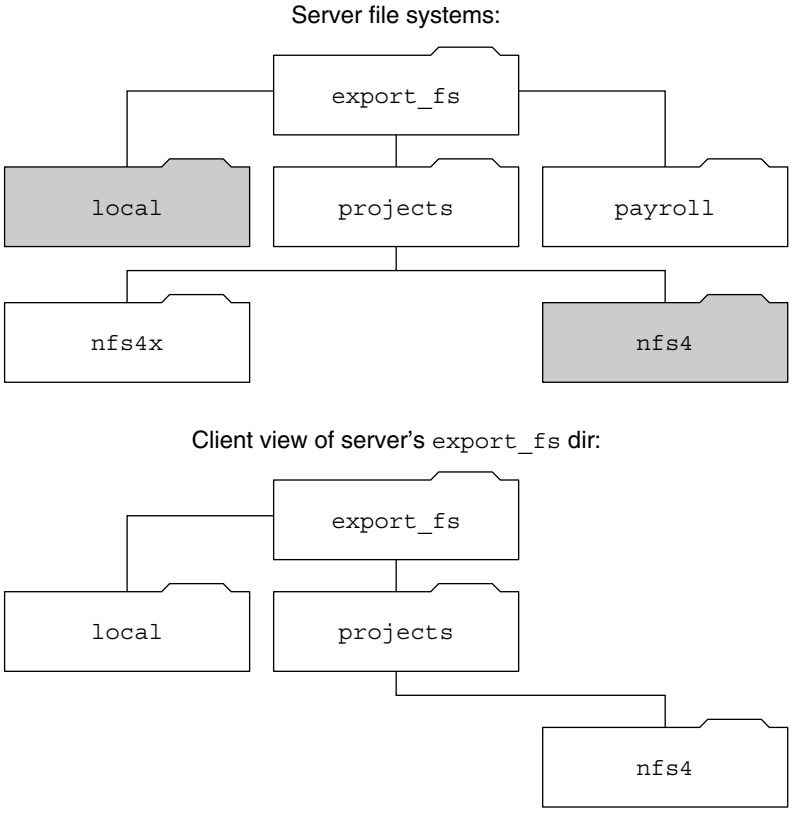
File-System Namespace in NFS Version 4

NFS version 4 servers create and maintain a pseudo-file system, which provides clients with seamless access to all exported objects on the server. Prior to NFS version 4, the pseudo-file system did not exist. Clients were forced to mount each shared server file system for access. Consider the following example.

FIGURE 3–2 Views of the Server File System and the Client File System

Server exports:
/export_fs/local
/export_fs/projects/nfs4

Server file systems:
/
/export_fs



Note that the client cannot see the `payroll` directory and the `nfs4x` directory, because these directories are not exported and do not lead to exported directories. However, the `local` directory is visible to the client, because `local` is an exported directory. The `projects` directory is visible to the client, because `projects` leads to the exported directory, `nfs4`. Thus, portions of the server namespace that are not explicitly exported are bridged with a pseudo-file system that views only the exported directories and those directories that lead to server exports.

A pseudo-file system is a structure that contains only directories and is created by the server. The pseudo-file system permits a client to browse the hierarchy of exported file systems. Thus, the client's view of the pseudo-file system is limited to paths that lead to exported file systems.

Previous versions of NFS did not permit a client to traverse server file systems without mounting each file system. However, in NFS version 4, the server namespace does the following:

- Restricts the client's file-system view to directories that lead to server exports.
- Provides clients with seamless access to server exports without requiring that the client mount each underlying file system. See the previous example. Note, however, that different operating systems might require the client to mount each server file system.

Volatile File Handles in NFS Version 4

File handles are created on the server and contain information that uniquely identifies files and directories. In NFS versions 2 and 3 the server returned persistent file handles. Thus, the client could guarantee that the server would generate a file handle that always referred to the same file. For example:

- If a file was deleted and replaced with a file of the same name, the server would generate a new file handle for the new file. If the client used the old file handle, the server would return an error that the file handle was stale.
- If a file was renamed, the file handle would remain the same.
- If you had to reboot the server, the file handles would remain the same.

Thus, when the server received a request from a client that included a file handle, the resolution was straightforward and the file handle always referred to the correct file.

This method of identifying files and directories for NFS operations was fine for most UNIX-based servers. However, the method could not be implemented on servers that relied on other methods of identification, such as a file's path name. To resolve this problem, the NFS version 4 protocol permits a server to declare that its file handles are volatile. Thus, a file handle could change. If the file handle does change, the client must find the new file handle.

Like NFS versions 2 and 3, the Oracle Solaris NFS version 4 server always provides persistent file handles. However, Oracle Solaris NFS version 4 clients that access non-Solaris NFS version 4 servers must support volatile file handles if the server uses them. Specifically, when the server tells the client that the file handle is volatile, the client must cache the mapping between path name and file handle. The client uses the volatile file handle until it expires. On expiration, the client does the following:

- Flushes the cached information that refers to that file handle
- Searches for that file's new file handle
- Retries the operation

Note – The server always tells the client which file handles are persistent and which file handles are volatile.

Volatile file handles might expire for any of these reasons:

- When you close a file
- When the filehandle's file system migrates
- When a client renames a file
- When the server reboots

Note that if the client is unable to find the new file handle, an error message is put in the `syslog` file. Further attempts to access this file fail with an `I/O` error.

Client Recovery in NFS Version 4

The NFS version 4 protocol is a stateful protocol. A protocol is stateful when both the client and the server maintain current information about the following.

- Open files
- File locks

When a failure occurs, such as a server crash, the client and the server work together to reestablish the open and lock states that existed prior to the failure.

When a server crashes and is rebooted, the server loses its state. The client detects that the server has rebooted and begins the process of helping the server rebuild its state. This process is known as client recovery, because the client directs the process.

When the client discovers that the server has rebooted, the client immediately suspends its current activity and begins the process of client recovery. When the recovery process starts, a message, such as the following, is displayed in the system error log `/var/adm/messages`.

```
NOTICE: Starting recovery server basil.example.company.com
```

During the recovery process, the client sends the server information about the client's previous state. Note, however, that during this period the client does not send any new requests to the server. Any new requests to open files or set file locks must wait for the server to complete its recovery period before proceeding.

When the client recovery process is complete, the following message is displayed in the system error log `/var/adm/messages`.

```
NOTICE: Recovery done for server basil.example.company.com
```

Now the client has successfully completed sending its state information to the server. However, even though the client has completed this process, other clients might not have completed their process of sending state information to the server. Therefore, for a period of time, the server does not accept any open or lock requests. This period of time, which is known as the grace period, is designated to permit all the clients to complete their recovery.

During the grace period, if the client attempts to open any new files or establish any new locks, the server denies the request with the GRACE error code. On receiving this error, the client must wait for the grace period to end and then resend the request to the server. During the grace period the following message is displayed.

```
NFS server recovering
```

Note that during the grace period the commands that do not open files or set file locks can proceed. For example, the commands `ls` and `cd` do not open a file or set a file lock. Thus, these commands are not suspended. However, a command such as `cat`, which opens a file, would be suspended until the grace period ends.

When the grace period has ended, the following message is displayed.

```
NFS server recovery ok.
```

The client can now send new open and lock requests to the server.

Client recovery can fail for a variety of reasons. For example, if a network partition exists after the server reboots, the client might not be able to reestablish its state with the server before the grace period ends. When the grace period has ended, the server does not permit the client to reestablish its state because new state operations could create conflicts. For example, a new file lock might conflict with an old file lock that the client is trying to recover. When such situations occur, the server returns the NO_GRACE error code to the client.

If the recovery of an open operation for a particular file fails, the client marks the file as unusable and the following message is displayed.

```
WARNING: The following NFS file could not be recovered and was marked dead
(can't reopen:  NFS status 70):  file :  filename
```

Note that the number `70` is only an example.

If reestablishing a file lock during recovery fails, the following error message is posted.

```
NOTICE: nfs4_send_siglost:  pid PROCESS-ID lost
lock on server SERVER-NAME
```

In this situation, the SIGLOST signal is posted to the process. The default action for the SIGLOST signal is to terminate the process.

For you to recover from this state, you must restart any applications that had files open at the time of the failure. Note that the following can occur.

- Some processes that did not reopen the file could receive I/O errors.
- Other processes that did reopen the file, or performed the open operation after the recovery failure, are able to access the file without any problems.

Thus, some processes can access a particular file while other processes cannot.

OPEN Share Support in NFS Version 4

The NFS version 4 protocol provides several file-sharing modes that the client can use to control file access by other clients. A client can specify the following:

- `DENY_NONE` mode permits other clients read and write access to a file.
- `DENY_READ` mode denies other clients read access to a file.
- `DENY_WRITE` mode denies other clients write access to a file.
- `DENY_BOTH` mode denies other clients read and write access to a file.

The Oracle Solaris NFS version 4 server fully implements these file-sharing modes. Therefore, if a client attempts to open a file in a way that conflicts with the current share mode, the server denies the attempt by failing the operation. When such attempts fail with the initiation of the open or create operations, the NFS version 4 client receives a protocol error. This error is mapped to the application error `EACCES`.

Even though the protocol provides several sharing modes, currently the open operation in Oracle Solaris does not offer multiple sharing modes. When opening a file, a Oracle Solaris NFS version 4 client can only use the `DENY_NONE` mode.

Also, even though the `fcntl` system call has an `F_SHARE` command to control file sharing, the `fcntl` commands cannot be implemented correctly with NFS version 4. If you use these `fcntl` commands on an NFS version 4 client, the client returns the `EAGAIN` error to the application.

Delegation in NFS Version 4

NFS version 4 provides both client support and server support for delegation. Delegation is a technique by which the server delegates the management of a file to a client. For example, the server could grant either a read delegation or a write delegation to a client. Read delegations can be granted to multiple clients at the same time, because these read delegations do not conflict with each other. A write delegation can be granted to only one client, because a write delegation conflicts with any file access by any other client. While holding a write delegation, the client would not send various operations to the server because the client is guaranteed exclusive access to a file. Similarly, the client would not send various operations to the server while holding a read delegation. The reason is that the server guarantees that no client can open the file in write mode. The effect of delegation is to greatly reduce the interactions between the server and the client for delegated files. Therefore, network traffic is reduced, and performance on the client and the server is improved. Note, however, that the degree of performance improvement depends on the kind of file interaction used by an application and the amount of network and server congestion.

The decision about whether to grant a delegation is made entirely by the server. A client does not request a delegation. The server makes decisions about whether to grant a delegation, based

on the access patterns for the file. If a file has been recently accessed in write mode by several different clients, the server might not grant a delegation. The reason is that this access pattern indicates the potential for future conflicts.

A conflict occurs when a client accesses a file in a manner that is inconsistent with the delegations that are currently granted for that file. For example, if a client holds a write delegation on a file and a second client opens that file for read or write access, the server recalls the first client's write delegation. Similarly, if a client holds a read delegation and another client opens the same file for writing, the server recalls the read delegation. Note that in both situations, the second client is not granted a delegation because a conflict now exists. When a conflict occurs, the server uses a callback mechanism to contact the client that currently holds the delegation. On receiving this callback, the client sends the file's updated state to the server and returns the delegation. If the client fails to respond to the recall, the server revokes the delegation. In such instances, the server rejects all operations from the client for this file, and the client reports the requested operations as failures. Generally, these failures are reported to the application as `I/O` errors. To recover from these errors, the file must be closed and then reopened. Failures from revoked delegations can occur when a network partition exists between the client and the server while the client holds a delegation.

Note that one server does not resolve access conflicts for a file that is stored on another server. Thus, an NFS server only resolves conflicts for files that it stores. Furthermore, in response to conflicts that are caused by clients that are running various versions of NFS, an NFS server can only initiate recalls to the client that is running NFS version 4. An NFS server cannot initiate recalls for clients that are running earlier versions of NFS.

The process for detecting conflicts varies. For example, unlike NFS version 4, because version 2 and version 3 do not have an open procedure, the conflict is detected only after the client attempts to read, write, or lock a file. The server's response to these conflicts varies also. For example:

- For NFS version 3, the server returns the `JUKEBOX` error, which causes the client to halt the access request and try again later. The client prints the message `File unavailable`.
- For NFS version 2, because an equivalent of the `JUKEBOX` error does not exist, the server makes no response, which causes the client to wait and then try again. The client prints the message `NFS server not responding`.

These conditions clear when the delegation conflict has been resolved.

By default, server delegation is enabled. You can disable delegation by setting the `server_delegation` parameter to none. For procedural information, refer to "How to Select Different Versions of NFS on a Server" on page 41.

No keywords are required for client delegation. The NFS version 4 callback daemon, `nfs4cbd`, provides the callback service on the client. This daemon is started automatically whenever a mount for NFS version 4 is enabled. By default, the client provides the necessary callback

information to the server for all Internet transports that are listed in the `/etc/netconfig` system file. Note that if the client is enabled for IPv6 and if the IPv6 address for the client's name can be determined, then the callback daemon accepts IPv6 connections.

The callback daemon uses a transient program number and a dynamically assigned port number. This information is provided to the server, and the server tests the callback path before granting any delegations. If the callback path does not test successfully, the server does not grant delegations, which is the only externally visible behavior.

Note that because callback information is embedded within an NFS version 4 request, the server is unable to contact the client through a device that uses Network Address Translation (NAT). Also, the callback daemon uses a dynamic port number. Therefore, the server might not be able to traverse a firewall, even if that firewall enables normal NFS traffic on port 2049. In such situations, the server does not grant delegations.

ACLs and `nfsmapid` in NFS Version 4

An access control list (ACL) provides better 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. On ZFS file systems, ACLs are set on the server and the client by using the `chmod` command. For UFS file systems, use the `setfacl` command. See the `chmod(1)` and the `setfacl(1)` man pages for more information. In NFS version 4, the ID mapper, `nfsmapid`, is used to map user or group IDs in ACL entries on a server to user or group IDs in ACL entries on a client. The reverse is also true. The user and group IDs in the ACL entries must exist on both the client and the server.

Reasons for ID Mapping to Fail

The following situations can cause ID mapping to fail:

- If the user or group that exists in an ACL entry on the server cannot be mapped to a valid user or group on the client, the user can read the ACL, but some of the users or groups will be shown as "unknown".

  For example, when you issue the `ls -lv` or `ls -lV` command, some of the ACL entries will have the group or user be displayed as "unknown". For more information about this command, see the `ls(1)` man page.
- If the user or group ID in any ACL entry that is set on the client cannot be mapped to a valid user or group ID on the server, the `setfacl` or the `chmod` command can fail and return the `Permission denied` error message.
- If the client and server have mismatched nfsmapid_domain values, ID mapping fails. For more information, see "`nfsmapid` Daemon" on page 86.

Avoiding ID Mapping Problems With ACLs

To avoid ID mapping problems, do the following:

- Make sure that the value for nfsmapid_domain is set correctly.
- Make sure that all user and group IDs in the ACL entries exist on both the NFS version 4 client and server.

Checking for Unmapped User or Group IDs

To determine if any user or group cannot be mapped on the server or client, use the following script:

```
#! /usr/sbin/dtrace -Fs

sdt:::nfs4-acl-nobody
{
     printf("validate_idmapping: (%s) in the ACL could not be mapped!",
stringof(arg0));
}
```

Note – The probe name that is used in this script is an interface that could change in the future. For more information, see "Stability Levels" in *Solaris Dynamic Tracing Guide*.

Additional Information About ACLs or nfsmapid

See the following:

- Chapter 7, "Using ACLs and Attributes to Protect Oracle Solaris ZFS Files," in *Oracle Solaris 11.1 Administration: ZFS File Systems*
- "`nfsmapid` Daemon" on page 86

UDP and TCP Negotiation

During initiation, the transport protocol is also negotiated. By default, the first connection-oriented transport that is supported on both the client and the server is selected. If this selection does not succeed, the first available connectionless transport protocol is used. The transport protocols that are supported on a system are listed in `/etc/netconfig`. TCP is the connection-oriented transport protocol that is supported by the release. UDP is the connectionless transport protocol.

When both the NFS protocol version and the transport protocol are determined by negotiation, the NFS protocol version is given precedence over the transport protocol. The NFS version 3 protocol that uses UDP is given higher precedence than the NFS version 2 protocol that is using

TCP. You can manually select both the NFS protocol version and the transport protocol with the `mount` command. See the `mount_nfs(1M)` man page. Under most conditions, allow the negotiation to select the best options.

File Transfer Size Negotiation

The file transfer size establishes the size of the buffers that are used when transferring data between the client and the server. In general, larger transfer sizes are better. The NFS version 3 protocol has an unlimited transfer size. The client can bid a smaller transfer size at mount time if needed, but under most conditions this bid is not necessary.

The transfer size is not negotiated with systems that use the NFS version 2 protocol. Under this condition, the maximum transfer size is set to 8 Kbytes.

You can use the `-rsize` and `-wsize` options to set the transfer size manually with the `mount` command. You might need to reduce the transfer size for some PC clients. Also, you can increase the transfer size if the NFS server is configured to use larger transfer sizes.

Note – Starting in the Solaris 10 release, restrictions on wire transfer sizes have been relaxed. The transfer size is based on the capabilities of the underlying transport. For example, the NFS transfer limit for UDP is still 32 Kbytes. However, because TCP is a streaming protocol without the datagram limits of UDP, maximum transfer sizes over TCP have been increased to 1 Mbyte.

How File Systems Are Mounted

The following description applies to NFS version 3 mounts. The NFS version 4 mount process does not include the portmap service nor does it include the `MOUNT` protocol.

When a client needs to mount a file system from a server, the client must obtain a file handle from the server. The file handle must correspond to the file system. This process requires that several transactions occur between the client and the server. In this example, the client is attempting to mount `/home/terry` from the server. A `snoop` trace for this transaction follows.

```
client -> server PORTMAP C GETPORT prog=100005 (MOUNT) vers=3 proto=UDP
server -> client PORTMAP R GETPORT port=33492
client -> server MOUNT3 C Null
server -> client MOUNT3 R Null
client -> server MOUNT3 C Mount /export/home9/terry
server -> client MOUNT3 R Mount OK FH=9000 Auth=unix
client -> server PORTMAP C GETPORT prog=100003 (NFS) vers=3 proto=TCP
server -> client PORTMAP R GETPORT port=2049
client -> server NFS C NULL3
server -> client NFS R NULL3
client -> server NFS C FSINFO3 FH=9000
```

```
server -> client NFS R FSINFO3 OK
client -> server NFS C GETATTR3 FH=9000
server -> client NFS R GETATTR3 OK
```

In this trace, the client first requests the mount port number from the portmap service on the NFS server. After the client receives the mount port number (`33492`), that number is used to test the availability of the service on the server. After the client has determined that a service is running on that port number, the client then makes a mount request. When the server responds to this request, the server includes the file handle for the file system (`9000`) being mounted. The client then sends a request for the NFS port number. When the client receives the number from the server, the client tests the availability of the NFS service (`nfsd`). Also, the client requests NFS information about the file system that uses the file handle.

In the following trace, the client is mounting the file system with the `public` option.

```
client -> server NFS C LOOKUP3 FH=0000 /export/home9/terry
server -> client NFS R LOOKUP3 OK FH=9000
client -> server NFS C FSINFO3 FH=9000
server -> client NFS R FSINFO3 OK
client -> server NFS C GETATTR3 FH=9000
server -> client NFS R GETATTR3 OK
```

By using the default public file handle (which is `0000`), all the transactions to obtain information from the portmap service and to determine the NFS port number are skipped.

Note – NFS version 4 provides support for volatile file handles. For more information, refer to "Volatile File Handles in NFS Version 4" on page 123.

Effects of the `-public` Option and NFS URLs When Mounting

Using the `-public` option can create conditions that cause a mount to fail. Adding an NFS URL can also confuse the situation. The following list describes the specifics of how a file system is mounted when you use these options.

Public option with NFS URL – Forces the use of the public file handle. The mount fails if the public file handle is not supported.

Public option with regular path – Forces the use of the public file handle. The mount fails if the public file handle is not supported.

NFS URL only – Use the public file handle if this file handle is enabled on the NFS server. If the mount fails when using the public file handle, then try the mount with the `MOUNT` protocol.

Regular path only – Do not use the public file handle. The `MOUNT` protocol is used.

Client-Side Failover

By using client-side failover, an NFS client can be aware of multiple servers that are making the same data available and can switch to an alternate server when the current server is unavailable. The file system can become unavailable if one of the following occurs.

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

The failover, under these conditions, is normally transparent to the user. Thus, the failover can occur at any time without disrupting the processes that are running on the client.

Failover requires that the file system be mounted read-only. The file systems must be identical for the failover to occur successfully. See "What Is a Replicated File System?" on page 132 for a description of what makes a file system identical. A static file system or a file system that is not changed often is the best candidate for failover.

You cannot use CacheFS and client-side failover on the same NFS mount. Extra information is stored for each CacheFS file system. This information cannot be updated during failover, so only one of these two features can be used when mounting a file system.

The number of replicas that need to be established for every file system depends on many factors. Ideally, you should have a minimum of two servers. Each server should support multiple subnets. This setup is better than having a unique server on each subnet. The process requires that each listed server be checked. Therefore, if more servers are listed, each mount is slower.

Failover Terminology

To fully comprehend the process, you need to understand two terms.

- *failover* – The process of selecting a server from a list of servers that support a replicated file system. Normally, the next server in the sorted list is used, unless it fails to respond.
- *remap* – To use a new server. Through normal use, the clients store the path name for each active file on the remote file system. During the remap, these path names are evaluated to locate the files on the new server.

What Is a Replicated File System?

For the purposes of failover, a file system can be called a *replica* when each file is the same size and has the same file size or file type as the original file system. Permissions, creation dates, and other file attributes are not considered. If the file size or file types are different, the remap fails and the process hangs until the old server becomes available. In NFS version 4, the behavior is different. See "Client-Side Failover in NFS Version 4" on page 133.

You can maintain a replicated file system by using `rsync`, `cpio`, or another file transfer mechanism. Because updating the replicated file systems causes inconsistency, for best results consider these precautions:

- Renaming the old version of the file before installing a new version of the file
- Running the updates at night when client usage is low
- Keeping the updates small
- Minimizing the number of copies

Failover and NFS Locking

Some software packages require read locks on files. To prevent these products from breaking, read locks on read-only file systems are allowed but are visible to the client side only. The locks persist through a remap because the server does not "know" about the locks. Because the files should not change, you do not need to lock the file on the server side.

Client-Side Failover in NFS Version 4

In NFS version 4, if a replica cannot be established because the file sizes are different or the file types are not the same, then the following happens.

- The file is marked dead.
- A warning is printed.
- The application receives a system call failure.

Note – If you restart the application and try again to access the file, you should be successful.

In NFS version 4, you no longer receive replication errors for directories of different sizes. In prior versions of NFS, this condition was treated as an error and would impede the remapping process.

Furthermore, in NFS version 4, if a directory read operation is unsuccessful, the operation is performed by the next listed server. In previous versions of NFS, unsuccessful read operations would cause the remap to fail and the process to hang until the original server was available.

How NFS Server Logging Works

NFS server logging provides records of NFS reads and writes, as well as operations that modify the file system. This data can be used to track access to information. In addition, the records can provide a quantitative way to measure interest in the information.

When a file system with logging enabled is accessed, the kernel writes raw data into a buffer file. This data includes the following:

- A timestamp
- The client IP address
- The UID of the requester
- The file handle of the file or directory object that is being accessed
- The type of operation that occurred

The `nfslogd` daemon converts this raw data into ASCII records that are stored in log files. During the conversion, the IP addresses are modified to host names and the UIDs are modified to logins if the name service that is enabled can find matches. The file handles are also converted into path names. To accomplish the conversion, the daemon tracks the file handles and stores information in a separate file handle-to-path table. That way, the path does not have to be identified again each time a file handle is accessed. Because no changes to the mappings are made in the file handle-to-path table if `nfslogd` is turned off, you must keep the daemon running.

Note – Server logging is not supported in NFS version 4.

How the WebNFS Service Works

The WebNFS service makes files in a directory available to clients by using a public file handle. A file handle is an address that is generated by the kernel that identifies a file for NFS clients. The *public file handle* has a predefined value, so the server does not need to generate a file handle for the client. The ability to use this predefined file handle reduces network traffic by eliminating the `MOUNT` protocol. This ability should also accelerate processes for the clients.

By default, the public file handle on an NFS server is established on the root file system. This default provides WebNFS access to any clients that already have mount privileges on the server. You can change the public file handle to point to any file system by using the `share` command.

When the client has the file handle for the file system, a `LOOKUP` is run to determine the file handle for the file to be accessed. The NFS protocol allows the evaluation of only one path name component at a time. Each additional level of directory hierarchy requires another `LOOKUP`. A WebNFS server can evaluate an entire path name with a single multi-component lookup transaction when the `LOOKUP` is relative to the public file handle. Multi-component lookup enables the WebNFS server to deliver the file handle to the desired file without exchanging the file handles for each directory level in the path name.

In addition, an NFS client can initiate concurrent downloads over a single TCP connection. This connection provides quick access without the additional load on the server that is caused by setting up multiple connections. Although web browser applications support concurrent

downloading of multiple files, each file has its own connection. By using one connection, the WebNFS software reduces the overhead on the server.

If the final component in the path name is a symbolic link to another file system, the client can access the file if the client already has access through normal NFS activities.

Normally, an NFS URL is evaluated relative to the public file handle. The evaluation can be changed to be relative to the server's root file system by adding an additional slash to the beginning of the path. In this example, these two NFS URLs are equivalent if the public file handle has been established on the `/export/ftp` file system.

```
nfs://server/junk
nfs://server//export/ftp/junk
```

Note – The NFS version 4 protocol is preferred over the WebNFS service. NFS version 4 fully integrates all the security negotiation that was added to the MOUNT protocol and the WebNFS service.

How WebNFS Security Negotiation Works

The NFS service includes a protocol that enables a WebNFS client to negotiate a selected security mechanism with a WebNFS server. The new protocol uses security negotiation multi-component lookup, which is an extension to the multi-component lookup that was used in earlier versions of the WebNFS protocol.

The WebNFS client initiates the process by making a regular multi–component lookup request by using the public file handle. Because the client has no knowledge of how the path is protected by the server, the default security mechanism is used. If the default security mechanism is not sufficient, the server replies with an `AUTH_TOOWEAK` error. This reply indicates that the default mechanism is not valid. The client needs to use a stronger default mechanism.

When the client receives the `AUTH_TOOWEAK` error, the client sends a request to the server to determine which security mechanisms are required. If the request succeeds, the server responds with an array of security mechanisms that are required for the specified path. Depending on the size of the array of security mechanisms, the client might have to make more requests to obtain the complete array. If the server does not support WebNFS security negotiation, the request fails.

After a successful request, the WebNFS client selects the first security mechanism from the array that the client supports. The client then issues a regular multi-component lookup request by using the selected security mechanism to acquire the file handle. All subsequent NFS requests are made by using the selected security mechanism and the file handle.

Note – The NFS version 4 protocol is preferred over the WebNFS service. NFS version 4 fully integrates all the security negotiation that was added to the MOUNT protocol and the WebNFS service.

WebNFS Limitations With Web Browser Use

Several functions that a web site that uses HTTP can provide are not supported by the WebNFS software. These differences stem from the fact that the NFS server only sends the file, so any special processing must be done on the client. If you need to have one web site configured for both WebNFS and HTTP access, consider the following issues:

- NFS browsing does not run CGI scripts. So, a file system with an active web site that uses many CGI scripts might not be appropriate for NFS browsing.
- The browser might start different viewers to handle files in different file formats. Accessing these files through an NFS URL starts an external viewer if the file type can be determined by the file name. The browser should recognize any file name extension for a standard MIME type when an NFS URL is used. The WebNFS software does not check inside the file to determine the file type. So, the only way to determine a file type is by the file name extension.
- NFS browsing cannot utilize server-side image maps (clickable images). However, NFS browsing can utilize client-side image maps (clickable images) because the URLs are defined with the location. No additional response is required from the document server.

Secure NFS System

The NFS environment is a powerful way and convenient way to share file systems on a network of different computer architectures and operating systems. However, the same features that make sharing file systems through NFS operation convenient also pose some security problems. Historically, most NFS implementations have used UNIX (or AUTH_SYS) authentication, but stronger authentication methods such as AUTH_DH have also been available. When using UNIX authentication, an NFS server authenticates a file request by authenticating the computer that makes the request, but not the user. Therefore, a client user can run `su` and impersonate the owner of a file. If DH authentication is used, the NFS server authenticates the user, making this sort of impersonation much harder.

With root access and knowledge of network programming, anyone can introduce arbitrary data into the network and extract any data from the network. The most dangerous attacks are those attacks that involve the introduction of data. An example is the impersonation of a user by generating the right packets or by recording “conversations” and replaying them later. These attacks affect data integrity. Attacks that involve passive eavesdropping, which is merely

listening to network traffic without impersonating anybody, are not as dangerous, because data integrity is not compromised. Users can protect the privacy of sensitive information by encrypting data that is sent over the network.

A common approach to network security problems is to leave the solution to each application. A better approach is to implement a standard authentication system at a level that covers all applications.

The Oracle Solaris operating system includes an authentication system at the level of the remote procedure call (RPC), which is the mechanism on which the NFS operation is built. This system, known as Secure RPC, greatly improves the security of network environments and provides additional security to services such as the NFS system. When the NFS system uses the facilities that are provided by Secure RPC, it is known as a Secure NFS system.

Secure RPC

Secure RPC is fundamental to the Secure NFS system. The goal of Secure RPC is to build a system that is at minimum as secure as a time-sharing system. In a time-sharing system all users share a single computer. A time-sharing system authenticates a user through a login password. With Data Encryption Standard (DES) authentication, the same authentication process is completed. Users can log in on any remote computer just as users can log in on a local terminal. The users' login passwords are their assurance of network security. In a time-sharing environment, the system administrator has an ethical obligation not to change a password to impersonate someone. In Secure RPC, the network administrator is trusted not to alter entries in a database that stores *public keys*.

You need to be familiar with two terms to understand an RPC authentication system: credentials and verifiers. Using ID badges as an example, the credential is what identifies a person: a name, address, and birthday. The verifier is the photo that is attached to the badge. You can be sure that the badge has not been stolen by checking the photo on the badge against the person who is carrying the badge. In RPC, the client process sends both a credential and a verifier to the server with each RPC request. The server sends back only a verifier because the client already “knows” the server's credentials.

RPC's authentication is open ended, which means that a variety of authentication systems can be plugged into it, such as UNIX, DH, and KERB.

When UNIX authentication is used by a network service, the credentials contain the client's host name, UID, GID, and group-access list. However, the verifier contains nothing. Because no verifier exists, a superuser could falsify appropriate credentials by using commands such as `su`. Another problem with UNIX authentication is that UNIX authentication assumes all computers on a network are UNIX computers. UNIX authentication breaks down when applied to other operating systems in a heterogeneous network.

To overcome the problems of UNIX authentication, Secure RPC uses DH authentication.

DH Authentication

DH authentication uses the Data Encryption Standard (DES) and Diffie-Hellman public-key cryptography to authenticate both users and computers in the network. DES is a standard encryption mechanism. Diffie-Hellman public-key cryptography is a cipher system that involves two keys: one public and one secret. The public keys and secret keys are stored in the namespace. NIS stores the keys in the public-key map. These maps contain the public key and secret key for all potential users. See the *Oracle Solaris Administration: Naming and Directory Services* for more information about how to set up the maps.

The security of DH authentication is based on a sender's ability to encrypt the current time, which the receiver can then decrypt and check against its own clock. The timestamp is encrypted with DES. The requirements for this scheme to work are as follows:

- The two agents must agree on the current time.
- The sender and receiver must be using the same encryption key.

If a network runs a time-synchronization program, the time on the client and the server is synchronized automatically. If a time-synchronization program is not available, timestamps can be computed by using the server's time instead of the network time. The client asks the server for the time before starting the RPC session, then computes the time difference between its own clock and the server's. This difference is used to offset the client's clock when computing timestamps. If the client and server clocks become unsynchronized the server begins to reject the client's requests. The DH authentication system on the client resynchronizes with the server.

The client and server arrive at the same encryption key by generating a random *conversation key*, also known as the *session key*, and by using public-key cryptography to deduce a *common key*. The common key is a key that only the client and server are capable of deducing. The conversation key is used to encrypt and decrypt the client's timestamp. The common key is used to encrypt and decrypt the conversation key.

KERB Authentication

Kerberos is an authentication system that was developed at MIT. Kerberos offers a variety of encryption types, including DES. Kerberos support is no longer supplied as part of Secure RPC, but a server-side and client-side implementation is included in the release. See Chapter 19, "Introduction to the Kerberos Service," in *Oracle Solaris 11.1 Administration: Security Services* for more information about the implementation of Kerberos authentication.

Using Secure RPC With NFS

Be aware of the following points if you plan to use Secure RPC:

- If a server crashes when no one is around (after a power failure, for example), all the secret keys that are stored on the system are deleted. Now no process can access secure network services or mount an NFS file system. The important processes during a reboot are usually run as `root`. Therefore, these processes would work if root's secret key were stored away, but nobody is available to type the password that decrypts it. `keylogin -r` allows `root` to store the clear secret key in `/etc/.rootkey`, which `keyserv` reads.
- Some systems boot in single-user mode, with a root login shell on the console and no password prompt. Physical security is imperative in such cases.
- Diskless computer booting is not totally secure. Somebody could impersonate the boot server and boot a devious kernel that, for example, makes a record of your secret key on a remote computer. The Secure NFS system provides protection only after the kernel and the key server are running. Otherwise, no way exists to authenticate the replies that are given by the boot server. This limitation could be a serious problem, but the limitation requires a sophisticated attack, using kernel source code. Also, the crime would leave evidence. If you polled the network for boot servers, you would discover the devious boot server's location.
- Most setuid programs are owned by `root`. If the secret key for `root` is stored in `/etc/.rootkey`, these programs behave as they always have. If a setuid program is owned by a user, however, the setuid program might not always work. For example, suppose that a setuid program is owned by `dave` and `dave` has not logged into the computer since it booted. The program would not be able to access secure network services.
- If you log in to a remote computer (using `login`, `rlogin`, or `telnet`) and use `keylogin` to gain access, you give access to your account. The reason is that your secret key is passed to that computer's key server, which then stores your secret key. This process is only a concern if you do not trust the remote computer. If you have doubts, however, do not log in to a remote computer if the remote computer requires a password. Instead, use the NFS environment to mount file systems that are shared by the remote computer. As an alternative, you can use `keylogout` to delete the secret key from the key server.
- If a home directory is shared with the `-o sec=dh` option, remote logins can be a problem. If the `/etc/hosts.equiv` or `~/.rhosts` files are not set to prompt for a password, the login succeeds. However, the users cannot access their home directories because no authentication has occurred locally. If the user is prompted for a password, the user has access to his or her home directory if the password matches the network password.

How Mirror Mounts Work

The Oracle Solaris release includes a new mounting facility called mirror mounts. Mirror mounts allow a NFSv4 client to access files in a file system as soon as the file system is shared on an NFSv4 server. The files can be accessed without the overhead of using the mount command or updating autofs maps. In effect, once a NFSv4 file system is mounted on a client, any other file systems from that server could also be mounted.

When to Use Mirror Mounts

Generally, using the mirror mount facility is optimal for your NFSv4 clients except when you:

- Need to use a different hierarchy on the client, than exists on the server
- Need to use different mount options than those of the parent file system

Mounting a File System Using Mirror Mounts

If a file system is mounted on an NFSv4 client using manual mounts or autofs, any additional file systems added to the mounted file system, may be mounted on the client using the mirror mount facility. The client requests access to the new file system using the same mount options as were used on the parent directory. If the mount fails for any reason, the normal NFSv4 security negotiations occur between the server and the client to adjust the mount options so that the mount request succeeds.

Where there is an existing automount trigger point setup for a particular server file system, the automount trigger takes precedence over mirror mounting, so a mirror mount will not occur for that file system. To use mirror mounts in this case, the automount entry would need to be removed.

In the Oracle Solaris 11 release, accessing `/net` or `/home` automount point cause a mount of the `/net` or `/home` server namespace. Access to directories or files under those directories will be given through the mirror mounts feature.

For specific instructions on how to get mirror mounts to work see:

- Example 2–2
- "How to Mount All File Systems from a Server" on page 35

Unmounting a File System Using Mirror Mounts

Mirror mounted file systems will be automatically unmounted if idle, after a certain period of inactivity. The period is set using the `timeout` parameter, which is used by the automounter for the same purpose.

If an NFS file system is manually unmounted, then any mirror mounted file systems contained within it will also be unmounted, if idle. If there is an active mirror mounted file system within, the manual unmount will fail, as though that original file system were busy. A forced unmount will, however, be propagated through to all enclosed mirror-mounted file systems.

If a file system boundary is encountered within an automounted file system, a mirror mount will occur. When the automounter unmounts the parent filesystem, any mirror-mounted file systems within it will also be automatically unmounted, if idle. If there is an active mirror mounted file system, the automatic unmount will not occur, which preserves current automount behavior.

How NFS Referrals Work

The Oracle Solaris 11.1 release includes a new NFS feature called NFS referrals. NFS referrals are a way for an NFSv4 server to point to file systems located on other NFSv4 servers, as a way of connecting multiple NFSv4 servers into a uniform namespace.

NFSv2, NFSv3 and other clients can follow a referral because it appears to them to be a symbolic link.

When to Use NFS Referrals?

NFS referrals are useful when you want to create what appears to be a single set of filenames across multiple servers, and you prefer not to use autofs to do this. Note that only NFSv4 servers may be used, and that the servers must be running the Oracle Solaris 11.1 release or later to host a referral.

Creating an NFS Referral

An NFS referral is created using the `nfsref` command. When a referral is created, if the mount point does not exist yet, a symbolic link is generated which includes a special flag which identifies object as a reparse point. If the reparse point already exists, NFS service data will be added or will replace existing NFS service data, as appropriate.

Removing an NFS Referral

An NFS referral is also removed using the `nfsref` command. It will remove NFS service data from the specified reparse point, and will remove the reparse point if there are no other types of service data present.

Autofs Maps

Autofs uses three types of maps:

- Master map
- Direct maps
- Indirect maps

Master Autofs Map

The `auto_master` map associates a directory with a map. The map is a master list that specifies all the maps that autofs should check. The following example shows what an `auto_master` file could contain.

EXAMPLE 3–3 Sample `/etc/auto_master` File

```
# Master map for automounter
#
+auto_master
/net            -hosts          -nosuid,nobrowse
/home           auto_home       -nobrowse
/nfs4           -fedfs          -ro,nosuid,nobrowse
/-              auto_direct     -ro
```

This example shows the generic `auto_master` file with one addition for the `auto_direct` map. Each line in the master map `/etc/auto_master` has the following syntax:

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

mount-point — *mount-point* is the full (absolute) path name of a directory. If the directory does not exist, autofs creates the directory if possible. If the directory exists and is not empty, mounting on the directory hides its contents. In this situation, autofs issues a warning.

The notation `/-` as a mount point indicates that this particular map is a direct map. The notation also means that no particular mount point is associated with the map.

map-name — *map-name* is the map autofs uses to find directions to locations, or mount information. If the name is preceded by a slash (/), autofs interprets the name as a local file. Otherwise, autofs searches for the mount information by using the search that is specified in the name-service switch configuration file (`/etc/nsswitch.conf`). Special maps are also used for `/net`. See "Mount Point `/net`" on page 143 for more information.

mount-options — *mount-options* is an optional, comma-separated list of options that apply to the mounting of the entries that are specified in map-name, unless the entries in map-name list other options. Options for each specific type of file system are listed in the mount man page for that file system. For example, see the `mount_nfs`(1M) man page for NFS-specific mount options. For NFS-specific mount points, the `bg` (background) and `fg` (foreground) options do not apply.

A line that begins with # is a comment. All the text that follows until the end of the line is ignored.

To split long lines into shorter ones, put a backslash (\) at the end of the line. The maximum number of characters of an entry is 1024.

Note – If the same mount point is used in two entries, the first entry is used by the `automount` command. The second entry is ignored.

Mount Point `/home`

The mount point `/home` is the directory under which the entries that are listed in `/etc/auto_home` (an indirect map) are to be mounted.

Note – Autofs runs on all computers and supports `/net` and `/home` (automounted home directories) by default. These defaults can be overridden by entries in the NIS `auto.master` map or by local editing of the `/etc/auto_master` file.

Mount Point `/net`

Autofs mounts under the directory `/net` all the entries in the special map `-hosts`. The map is a built-in map that uses only the hosts database. Suppose that the computer `gumbo` is in the hosts database and it exports any of its file systems. The following command changes the current directory to the root directory of the computer `gumbo`.

```
% cd /net/gumbo
```

Autofs can mount only the *exported* file systems of host `gumbo`, that is, those file systems on a server that are available to network users instead of those file systems on a local disk. Therefore, all the files and directories on `gumbo` might not be available through `/net/gumbo`.

With the `/net` method of access, the server name is in the path and is location dependent. If you want to move an exported file system from one server to another, the path might no longer work. Instead, you should set up an entry in a map specifically for the file system you want rather than use `/net`.

Note – Using NFSv3 and earlier protocols, autofs checks the server's export list only at mount time. After a server's file systems are mounted, autofs does not check with the server again until the server's file systems are automatically unmounted. Therefore, newly exported file systems are not "seen" until the file systems on the client are unmounted and then remounted. For systems using NFSv4, mirror mounts reflect any dynamic changes made to the list of exported file systems on the server.

Mount Point `/nfs4`

The `/nfs4` mount point uses a pseudo-map to mount the Federated File System domain root. A reference to `/nfs4/example.net` will result in an attempt to find the domain root for the DNS domain `example.net` and mount it at that location. This requires that the DNS server returns a record as described in "Setting up a DNS Record for a FedFS Server" on page 38.

Direct Autofs Maps

A direct map is an automount point. With a direct map, a direct association exists between a mount point on the client and a directory on the server. Direct maps have a full path name and indicate the relationship explicitly. The following is a typical `/etc/auto_direct` map:

```
/usr/local          -ro \
   /bin                   ivy:/export/local/sun4 \
   /share                 ivy:/export/local/share \
   /src                   ivy:/export/local/src
/usr/man            -ro   oak:/usr/man \
                          rose:/usr/man \
                          willow:/usr/man
/usr/games          -ro   peach:/usr/games
/usr/spool/news     -ro   pine:/usr/spool/news \
                          willow:/var/spool/news
```

Lines in direct maps have the following syntax:

key [*mount-options*] *location*

key *key* is the path name of the mount point in a direct map.

mount-options — *mount-options* is the options that you want to apply to this particular mount. These options are required only if the options differ from the map default. Options for each specific type of file system are listed in the mount man page for that file system. For example, see the `mount_nfs(1M)` man page for NFS specific mount options.

location — *location* is the location of the file system. One or more file systems are specified as *server*:*pathname* for NFS file systems.

Note – The *pathname* should not include an automounted mount point. The *pathname* should be the actual absolute path to the file system. For instance, the location of a home directory should be listed as *server*:`/export/home/`*username*, not as *server*:`/home/`*username*.

As in the master map, a line that begins with # is a comment. All the text that follows until the end of the line is ignored. Put a backslash at the end of the line to split long lines into shorter ones.

Of all the maps, the entries in a direct map most closely resemble the corresponding entries in `/etc/vfstab`. An entry might appear in `/etc/vfstab` as follows:

```
dancer:/usr/local - /usr/local/tmp nfs - yes ro
```

The equivalent entry appears in a direct map as follows:

```
/usr/local/tmp     -ro     dancer:/usr/local
```

Note – No concatenation of options occurs between the automounter maps. Any options that are added to an automounter map override all options that are listed in maps that are searched earlier. For instance, options that are included in the `auto_master` map would be overridden by corresponding entries in any other map.

See "How Autofs Selects the Nearest Read-Only Files for Clients (Multiple Locations)" on page 151 for other important features that are associated with this type of map.

Mount Point /–

In Example 3–3, the mount point `/-` tells autofs not to associate the entries in `auto_direct` with any specific mount point. Indirect maps use mount points that are defined in the `auto_master` file. Direct maps use mount points that are specified in the named map. Remember, in a direct map the key, or mount point, is a full path name.

An NIS `auto_master` file can have only one direct map entry because the mount point must be a unique value in the namespace. An `auto_master` file that is a local file can have any number of direct map entries if entries are not duplicated.

Indirect Autofs Maps

An indirect map uses a substitution value of a key to establish the association between a mount point on the client and a directory on the server. Indirect maps are useful for accessing specific file systems, such as home directories. The `auto_home` map is an example of an indirect map.

Lines in indirect maps have the following general syntax:

key [*mount-options*] *location*

key — *key* is a simple name without slashes in an indirect map.

mount-options — *mount-options* is the options that you want to apply to this particular mount. These options are required only if the options differ from the map default. Options for each specific type of file system are listed in the mount man page for that file system. For example, see the `mount_nfs(1M)` man page for NFS-specific mount options.

location — *location* is the location of the file system. One or more file systems are specified as *server*:*pathname*.

Note – The *pathname* should not include an automounted mount point. The *pathname* should be the actual absolute path to the file system. For instance, the location of a directory should be listed as *server*`:/usr/local`, not as *server*`:/net/`*server*`/usr/local`.

As in the master map, a line that begins with # is a comment. All the text that follows until the end of the line is ignored. Put a backslash (\) at the end of the line to split long lines into shorter ones. Example 3–3 shows an `auto_master` map that contains the following entry:

```
/home      auto_home        -nobrowse
```

`auto_home` is the name of the indirect map that contains the entries to be mounted under `/home`. A typical `auto_home` map might contain the following:

```
david                 willow:/export/home/david
rob                   cypress:/export/home/rob
gordon                poplar:/export/home/gordon
rajan                 pine:/export/home/rajan
tammy                 apple:/export/home/tammy
jim                   ivy:/export/home/jim
```

```
linda    -rw,nosuid    peach:/export/home/linda
```

As an example, assume that the previous map is on host `oak`. Suppose that the user `linda` has an entry in the password database that specifies her home directory as `/home/linda`. Whenever `linda` logs in to computer `oak`, autofs mounts the directory `/export/home/linda` that resides on the computer `peach`. Her home directory is mounted read-write, nosuid.

Assume the following conditions occur: User linda's home directory is listed in the password database as `/home/linda`. Anybody, including Linda, has access to this path from any computer that is set up with the master map referring to the map in the previous example.

Under these conditions, user linda can run `login` or `rlogin` on any of these computers and have her home directory mounted in place for her.

Furthermore, now Linda can also type the following command:

```
% cd ~david
```

autofs mounts David's home directory for her (if all permissions allow).

Note – No concatenation of options occurs between the automounter maps. Any options that are added to an automounter map override all options that are listed in maps that are searched earlier. For instance, options that are included in the `auto_master` map are overridden by corresponding entries in any other map.

On a network without a name service, you have to change all the relevant files (such as `/etc/passwd`) on all systems on the network to allow Linda access to her files. With NIS, make the changes on the NIS master server and propagate the relevant databases to the slave servers.

How Autofs Works

Autofs is a client-side service that automatically mounts the appropriate file system. The components that work together to accomplish automatic mounting are the following:

- The `automount` command
- The `autofs` file system
- The `automountd` daemon

The automount service, `svc:/system/filesystem/autofs`, which is called at system startup time, reads the master map file `auto_master` to create the initial set of autofs mounts. These autofs mounts are not automatically mounted at startup time. These mounts are points under which file systems are mounted in the future. These points are also known as trigger nodes.

After the autofs mounts are set up, these mounts can trigger file systems to be mounted under them. For example, when autofs receives a request to access a file system that is not currently mounted, autofs calls `automountd`, which actually mounts the requested file system.

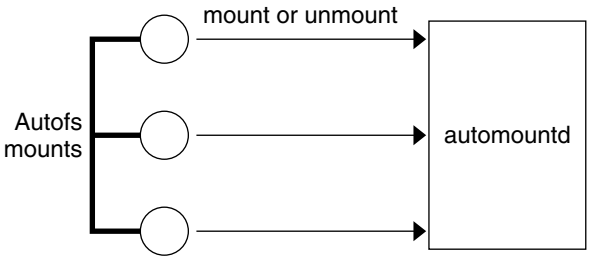
After initially mounting autofs mounts, the `automount` command is used to update autofs mounts as necessary. The command compares the list of mounts in the `auto_master` map with the list of mounted file systems in the mount table file `/etc/mnttab` (formerly `/etc/mtab`). `automount` then makes the appropriate changes. This process allows system administrators to change mount information within `auto_master` and have those changes used by the autofs processes without stopping and restarting the autofs daemon. After the file system is mounted, further access does not require any action from `automountd` until the file system is automatically unmounted.

Unlike `mount`, `automount` does not read the `/etc/vfstab` file (which is specific to each computer) for a list of file systems to mount. The `automount` command is controlled within a domain and on computers through the namespace or local files.

The following is a simplified overview of how autofs works.

The automount daemon `automountd` is started at boot time by the service `svc:/system/filesystem/autofs`. See Figure 3–3. This service also runs the `automount` command, which reads the master map and installs autofs mount points. See "How Autofs Starts the Navigation Process (Master Map)" on page 149 for more information.

FIGURE 3–3 svc:/system/filesystem/autofs Service Starts `automount`



Autofs is a kernel file system that supports automatic mounting and unmounting.

When a request is made to access a file system at an autofs mount point, the following occurs:

1. Autofs intercepts the request.
2. Autofs sends a message to the `automountd` for the requested file system to be mounted.
3. `automountd` locates the file system information in a map, creates the trigger nodes, and performs the mount.

4. Autofs allows the intercepted request to proceed.
5. Autofs unmounts the file system after a period of inactivity.

Note – Mounts that are managed through the autofs service should not be manually mounted or unmounted. Even if the operation is successful, the autofs service does not check that the object has been unmounted, resulting in possible inconsistencies. A reboot clears all the autofs mount points.

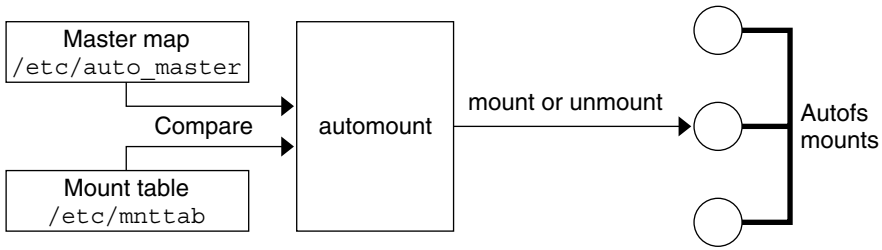
How Autofs Navigates Through the Network (Maps)

Autofs searches a series of maps to navigate through the network. Maps are files that contain information such as the password entries of all users on a network or the names of all host computers on a network. Effectively, the maps contain network-wide equivalents of UNIX administration files. Maps are available locally or through a network name service such as NIS. See "Modifying How Autofs Navigates the Network (Modifying Maps)" on page 157.

How Autofs Starts the Navigation Process (Master Map)

The `automount` command reads the master map at system startup. Each entry in the master map is a direct map name or an indirect map name, its path, and its mount options, as shown in Figure 3–4. The specific order of the entries is not important. `automount` compares entries in the master map with entries in the mount table to generate a current list.

FIGURE 3–4 Navigation Through the Master Map



Autofs Mount Process

What the autofs service does when a mount request is triggered depends on how the automounter maps are configured. The mount process is generally the same for all mounts.

However, the final result changes with the mount point that is specified and the complexity of the maps. The mount process includes the creation of the trigger nodes.

Simple Autofs Mount

To help explain the autofs mount process, assume that the following files are installed.

```
$ cat /etc/auto_master
# Master map for automounter
#
+auto_master
/net        -hosts        -nosuid,nobrowse
/home       auto_home     -nobrowse
/share      auto_share
$ cat /etc/auto_share
# share directory map for automounter
#
ws          gumbo:/export/share/ws
```

When the `/share` directory is accessed, the autofs service creates a trigger node for `/share/ws`, which is an entry in `/etc/mnttab` that resembles the following entry:

```
-hosts  /share/ws     autofs  nosuid,nobrowse,ignore,nest,dev=###
```

When the `/share/ws` directory is accessed, the autofs service completes the process with these steps:

1. Checks the availability of the server's mount service.
2. Mounts the requested file system under `/share`. Now the `/etc/mnttab` file contains the following entries.

   ```
   -hosts  /share/ws     autofs  nosuid,nobrowse,ignore,nest,dev=###
   gumbo:/export/share/ws /share/ws   nfs   nosuid,dev=####    #####
   ```

Hierarchical Mounting

When multiple layers are defined in the automounter files, the mount process becomes more complex. Suppose that you expand the `/etc/auto_shared` file from the previous example to contain the following:

```
# share directory map for automounter
#
ws       /       gumbo:/export/share/ws
         /usr    gumbo:/export/share/ws/usr
```

The mount process is basically the same as the previous example when the `/share/ws` mount point is accessed. In addition, a trigger node to the next level (`/usr`) is created in the `/share/ws` file system so that the next level can be mounted if it is accessed. In this example, `/export/share/ws/usr` must exist on the NFS server for the trigger node to be created.

Caution – Do not use the `-soft` option when specifying hierarchical layers. Refer to "Autofs Unmounting" on page 151 for an explanation of this limitation.

Autofs Unmounting

The unmounting that occurs after a certain amount of idle time is from the bottom up (reverse order of mounting). If one of the directories at a higher level in the hierarchy is busy, only file systems below that directory are unmounted. During the unmounting process, any trigger nodes are removed and then the file system is unmounted. If the file system is busy, the unmount fails and the trigger nodes are reinstalled.

Caution – Do not use the `-soft` option when specifying hierarchical layers. If the `-soft` option is used, requests to reinstall the trigger nodes can time out. The failure to reinstall the trigger nodes leaves no access to the next level of mounts. The only way to clear this problem is to have the automounter unmount all of the components in the hierarchy. The automounter can complete the unmounting either by waiting for the file systems to be automatically unmounted or by rebooting the system.

How Autofs Selects the Nearest Read-Only Files for Clients (Multiple Locations)

The example direct map contains the following:

```
/usr/local          -ro \
   /bin                   ivy:/export/local/sun4\
   /share                 ivy:/export/local/share\
   /src                   ivy:/export/local/src
/usr/man            -ro    oak:/usr/man \
                           rose:/usr/man \
                           willow:/usr/man
/usr/games          -ro    peach:/usr/games
/usr/spool/news     -ro    pine:/usr/spool/news \
                           willow:/var/spool/news
```

The mount points `/usr/man` and `/usr/spool/news` list more than one location, three locations for the first mount point, two locations for the second mount point. Any of the replicated locations can provide the same service to any user. This procedure is sensible only when you mount a file system that is read-only, as you must have some control over the locations of files that you write or modify. You want to avoid modifying files on one server on one occasion and, minutes later, modifying the "same" file on another server. The benefit is that the best available server is used automatically without any effort that is required by the user.

If the file systems are configured as replicas (see "What Is a Replicated File System?" on page 132), the clients have the advantage of using failover. Not only is the best server automatically determined, but if that server becomes unavailable, the client automatically uses the next-best server.

An example of a good file system to configure as a replica is man pages. In a large network, more than one server can export the current set of man pages. Which server you mount the man pages from does not matter if the server is running and exporting its file systems. In the previous example, multiple mount locations are expressed as a list of mount locations in the map entry.

```
/usr/man -ro oak:/usr/man rose:/usr/man willow:/usr/man
```

In this example, you can mount the man pages from the servers `oak`, `rose`, or `willow`. Which server is best depends on a number of factors, including the following:

- The number of servers that support a particular NFS protocol level
- The proximity of the server
- The weighting

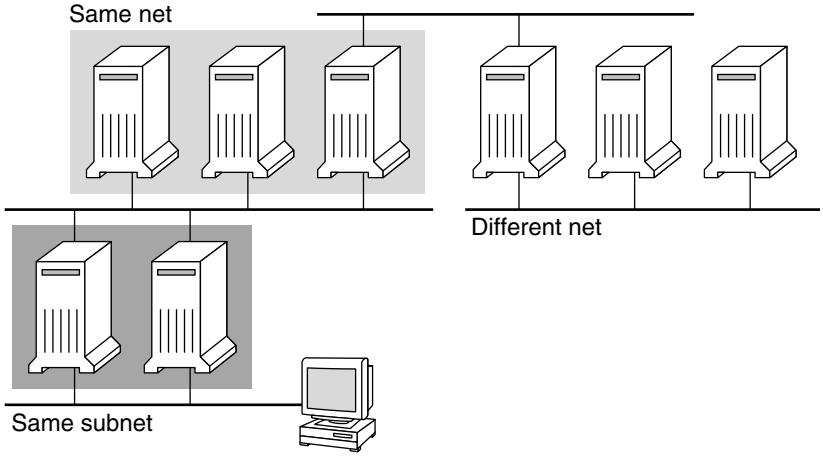
During the sorting process, a count is taken of the number of servers that support each version of the NFS protocol. Whichever version of the protocol is supported on the most servers becomes the protocol that is used by default. This selection provides the client with the maximum number of servers to depend on.

After the largest subset of servers with the same version of the protocol is found, that server list is sorted by proximity. To determine proximity IPv4 addresses are inspected. The IPv4 addresses show which servers are in each subnet. Servers on a local subnet are given preference over servers on a remote subnet. Preference for the closest server reduces latency and network traffic.

Note – Proximity cannot be determined for replicas that are using IPv6 addresses.

Figure 3–5 illustrates server proximity.

FIGURE 3–5 Server Proximity



If several servers that support the same protocol are on the local subnet, the time to connect to each server is determined and the fastest server is used. The sorting can also be influenced by using weighting (see "Autofs and Weighting" on page 154).

For example, if version 4 servers are more abundant, version 4 becomes the protocol that is used by default. However, now the sorting process is more complex. Here are some examples of how the sorting process works:

- Servers on the local subnet are given preference over servers on a remote subnet. So, if a version 3 server is on the local subnet and the closest version 4 server is on a remote subnet, the version 3 server is given preference. Likewise, if the local subnet consists of version 2 servers, they are given preference over remote subnets with version 3 and version 4 servers.
- If the local subnet consists of a varied number of version 2, version 3, and version 4 servers, more sorting is required. The automounter prefers the highest version on the local subnet. In this instance, version 4 is the highest version. However, if the local subnet has more version 3 or version 2 servers than version 4 servers, the automounter "bids down" from the highest version on the local subnet by one version. For example, if the local subnet has three servers with version 4, three servers with version 3, and ten servers with version 2, a version 3 server is selected.
- Similarly, if the local subnet consists of a varied number of version 2 and version 3 servers, the automounter first looks at the which version represents the highest version on the local subnet. Next, the automounter counts the number of servers that run each version. If the highest version on the local subnet also represents the most servers, the highest version is selected. If a lower version has more servers, the automounter bids down from the highest version on the local subnet by one version. For example, if more version 2 servers are on the local subnet than version 3 servers, a version 2 server is selected.

Note – Weighting is also influenced by parameters stored in the SMF repository. Specifically the values for `server_versmin`, `client_versmin`, `server_versmax` and `client_versmax` can make some versions be excluded from the sorting process. For more information about these parameters see "`mountd` Daemon" on page 84 and "`nfsd` Daemon" on page 85.

With failover, the sorting is checked at mount time when a server is selected. Multiple locations are useful in an environment where individual servers might not export their file systems temporarily.

Failover is particularly useful in a large network with many subnets. Autofs chooses the appropriate server and is able to confine NFS network traffic to a segment of the local network. If a server has multiple network interfaces, you can list the host name that is associated with each network interface as if the interface were a separate server. Autofs selects the nearest interface to the client.

Note – No weighting and no proximity checks are performed with manual mounts. The `mount` command prioritizes the servers that are listed from left to right.

For more information, see `automount(1M)` man page.

Autofs and Weighting

You can influence the selection of servers at the same proximity level by adding a weighting value to the autofs map. For example:

```
/usr/man -ro oak,rose(1),willow(2):/usr/man
```

The numbers in parentheses indicate a weighting. Servers without a weighting have a value of zero and, therefore, are most likely to be selected. The higher the weighting value, the lower the chance that the server is selected.

Note – All other server selection factors are more important than weighting. Weighting is only considered when selecting between servers with the same network proximity.

Variables in a Autofs Map Entry

You can create a client-specific variable by prefixing a dollar sign (`$`) to its name. The variable helps you to accommodate different architecture types that are accessing the same file-system

location. You can also use curly braces to delimit the name of the variable from appended letters or digits. Table 3–3 shows the predefined map variables.

TABLE 3–3 Predefined Map Variables

Variable	Meaning	Derived From	Example
`ARCH`	Architecture type	`uname -m`	`sun4`
`CPU`	Processor type	`uname -p`	`sparc`
`HOST`	Host name	`uname -n`	`dinky`
`OSNAME`	Operating system name	`uname -s`	`SunOS`
`OSREL`	Operating system release	`uname -r`	`5.8`
`OSVERS`	Operating system version (version of the release)	`uname -v`	`GENERIC`

You can use variables anywhere in an entry line except as a key. For instance, suppose that you have a file server that exports binaries for SPARC and x86 architectures from `/usr/local/bin/sparc` and `/usr/local/bin/x86` respectively. The clients can mount through a map entry such as the following:

```
/usr/local/bin        -ro     server:/usr/local/bin/$CPU
```

Now the same entry for all clients applies to all architectures.

Note – Most applications that are written for any of the sun4 architectures can run on all sun4 platforms. The `-ARCH` variable is hard-coded to `sun4`.

Maps That Refer to Other Maps

A map entry *+mapname* that is used in a file map causes automount to read the specified map as if it were included in the current file. If *mapname* is not preceded by a slash, autofs treats the map name as a string of characters and uses the name-service switch policy to find the map name. If the path name is an absolute path name, `automount` checks a local map of that name. If the map name starts with a dash (`-`), `automount` consults the appropriate built-in map, such as `hosts`.

The `svc:system/name-service/switch` service contains the search order for the naming services. The `automount` property in the `config` property group specifies the order that the name service databases are searched when looking for automount entries. If no specific `config/automount` property is specified, then the order defined in the `config/default` property is used. For instance:

```
# svcprop -p config svc:/system/name-service/switch
config/value_authorization astring solaris.smf.value.name-service.switch
config/printer astring user\ files
config/default astring files\ nis
config/automount astring files\ nis
```

In this example, the maps in the local files are searched before the NIS maps. The same would be true if the `config/automount` property was not specified, because the `config/default` entry would be used. Therefore, you can have a few entries in your local `/etc/auto_home` map for the most commonly accessed home directories. You can then use the switch to fall back to the NIS map for other entries.

```
bill               cs.csc.edu:/export/home/bill
bonny              cs.csc.edu:/export/home/bonny
```

After consulting the included map, if no match is found, `automount` continues scanning the current map. Therefore, you can add more entries after a + entry.

```
bill               cs.csc.edu:/export/home/bill
bonny              cs.csc.edu:/export/home/bonny
+auto_home
```

The map that is included can be a local file or a built-in map. Remember, only local files can contain + entries.

```
+/etc/auto_mystuff      # local map
+auto_home              # NIS map
+-hosts                 # built-in hosts map
```

Note – You cannot use + entries in NIS maps.

Executable Autofs Maps

You can create an autofs map that executes some commands to generate the autofs mount points. You could benefit from using an executable autofs map if you need to be able to create the autofs structure from a database or a flat file. The disadvantage to using an executable map is that the map needs to be installed on each host. An executable map cannot be included in the NIS name service.

The executable map must have an entry in the `auto_master` file.

```
/execute    auto_execute
```

Here is an example of an executable map:

```
#!/bin/ksh
#
# executable map for autofs
```

```
#

case $1 in
            src)  echo '-nosuid,hard bee:/export1' ;;
esac
```

For this example to work, the file must be installed as `/etc/auto_execute` and must have the executable bit set. Set permissions to `744`. Under these circumstances, running the following command causes the `/export1` file system from `bee` to be mounted:

```
% ls /execute/src
```

Modifying How Autofs Navigates the Network (Modifying Maps)

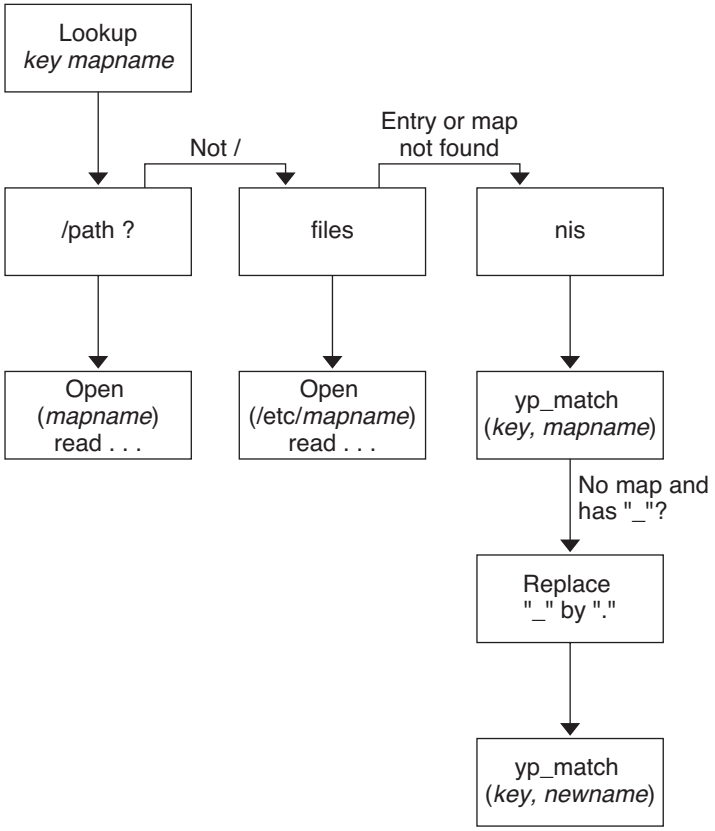
You can modify, delete, or add entries to maps to meet the needs of your environment. As applications and other file systems that users require change their location, the maps must reflect those changes. You can modify autofs maps at any time. Whether your modifications are effective the next time `automountd` mounts a file system depends on which map you modify and what kind of modification you make.

Default Autofs Behavior With Name Services

At boot time autofs is invoked by the service `svc:/system/filesystem/autofs` and autofs checks for the master `auto_master` map. Autofs is subject to the rules that are discussed subsequently.

Autofs uses the name service ordering specified in the `config/automount` property of the `svc:/system/name-service/switch` service. If the `config/automount` property is not defined then the `config/default` property is used.If NIS is selected and autofs cannot find a map that autofs needs, but finds a map name that contains one or more underscores, the underscores are changed to dots. This change allows the old NIS file names to work. Then autofs checks the map again, as shown in Figure 3–6.

FIGURE 3–6 How Autofs Uses the Name Service



The screen activity for this session would resemble the following example.

```
$ grep /home /etc/auto_master
/home           auto_home

$ ypmatch brent auto_home
Can't match key brent in map auto_home.  Reason: no such map in
server's domain.

$ ypmatch brent auto.home
diskus:/export/home/diskus1/&
```

If "files" is selected as the name service, all maps are assumed to be local files in the `/etc` directory. Autofs interprets a map name that begins with a slash (`/`) as local regardless of which name service autofs uses.

Autofs Reference

The remaining sections of this chapter describe more advanced autofs features and topics.

Autofs and Metacharacters

Autofs recognizes some characters as having a special meaning. Some characters are used for substitutions, and some characters are used to protect other characters from the autofs map parser.

Ampersand (&)

If you have a map with many subdirectories specified, as in the following, consider using string substitutions.

```
john        willow:/home/john
mary        willow:/home/mary
joe         willow:/home/joe
able        pine:/export/able
baker       peach:/export/baker
```

You can use the ampersand character (&) to substitute the key wherever the key appears. If you use the ampersand, the previous map changes to the following:

```
john        willow:/home/&
mary        willow:/home/&
joe         willow:/home/&
able        pine:/export/&
baker       peach:/export/&
```

You could also use key substitutions in a direct map, in situations such as the following:

```
/usr/man                        willow,cedar,poplar:/usr/man
```

You can also simplify the entry further as follows:

```
/usr/man                        willow,cedar,poplar:&
```

Notice that the ampersand substitution uses the whole key string. Therefore, if the key in a direct map starts with a / (as it should), the slash is included in the substitution. Consequently, for example, you could not do the following:

```
/progs                &1,&2,&3:/export/src/progs
```

The reason is that autofs would interpret the example as the following:

```
/progs                 /progs1,/progs2,/progs3:/export/src/progs
```

Asterisk (*)

You can use the universal substitute character, the asterisk (*), to match any key. You could mount the /export file system from all hosts through this map entry.

```
*                       &:/export
```

Each ampersand is substituted by the value of any given key. Autofs interprets the asterisk as an end-of-file character.

Autofs and Special Characters

If you have a map entry that contains special characters, you might have to mount directories that have names that confuse the autofs map parser. The autofs parser is sensitive to names that contain colons, commas, and spaces, for example. These names should be enclosed in double-quotes, as in the following:

```
/vms    -ro    vmsserver: -  -  - "rc0:dk1 - "
/mac    -ro    gator:/ - "Mr Disk - "
```

Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W