Prime

Guide to NFS™ on PRIMOS® Systems

Release 1.1-22.0

DOC10285-2LA

Guide to NFS™ on PRIMOS® Systems

Second Edition

Dick Frost

This manual documents the software operation of the PRIMOS operating system on 50 Series computers and their supporting systems and utilities as implemented at Master Disk Revision Level 23.0 (Rev. 23.0).

Prime Computer, Inc., Prime Park, Natick, MA 01760

Printing History

First Edition (DOC10285-1LA) March 1989 for NFS Release 1.0-22.0
Second Edition (DOC10285-2LA) April 1990 for NFS Release 1.1-22.0
First Update to Second Edition (UPD10285-21A) April 1991 for NFS Release 1.2-22.0 and 1.2-23.0

Credits

Editorial: Mary Skousgaard
Project Support: Brian Clew, Scott Reeve, Nancy LeBlang, Tom Green
Illustration: Mary Easter
Production: Judy Gordon

How to Order Technical Documents

To order copies of documents, or to obtain a catalog and price list:

United States Customers	*International*
Call Prime Telemarketing, toll free, at 1-800-343-2533, Monday through Thursday, 8:30 a.m. to 8:00 p.m., and Friday, 8:30 a.m. to 6:00 p.m. (EST).	Contact your local Prime subsidiary or distributor.

PrimeService℠

Prime provides the following toll-free number for customers in the United States needing service:

1-800-800-PRIME

For other locations, contact your Prime representative.

Surveys and Correspondence

Please comment on this manual using the Reader Response Form provided in the back of this book. Address any additional comments on this or other Prime documents to:

Technical Publications Department
Prime Computer, Inc.
500 Old Connecticut Path
Framingham, MA 01701

CONTENTS

APPENDICES

ABOUT THIS BOOK

The *Guide to NFS on PRIMOS Systems* tells you how to provide NFS™ service from a PRIMOS® host to a user on a client NFS host. NFS on PRIMOS Systems is an implementation of the Network File System (NFS) licensed from Sun Microsystems, Inc.

This guide tells the NFS Administrator on the PRIMOS system how to

- Install, configure, and run NFS on PRIMOS Systems
- Provide access for additional client hosts and additional client users of NFS on PRIMOS Systems
- Maintain reliable service for NFS on PRIMOS Systems
- Diagnose and solve problems for NFS on PRIMOS Systems

This guide also tells the client NFS user (Administrator or otherwise) about

- Particular differences (in character set, filenames, and so on) between NFS on PRIMOS Systems and other NFS implementations
- The differences in the use of the **/etc/mount** command, due to the structure of the PRIMOS file system

Finally, this guide tells the ONC™ RPC programmer how to develop networked applications on the PRIMOS system. Release 1.2 of NFS on PRIMOS Systems makes the RPC library available to the user.

Programmers on the PRIMOS system who are developing applications using the RPC library need to know the Prime-specific information in Appendix E of this book as well as the programming information in the *ONC Network Programming Guide.*

Administrators for NFS on PRIMOS Systems should be familiar with Prime® systems before reading this manual. Client users of NFS on PRIMOS Systems do not need to know about the PRIMOS operating system, although this information is useful. The *PRIMOS User's Guide* explains file system management on PRIMOS and describes essential commands and utilities. Its explanation of Access Control Lists (ACLs) is especially helpful for NFS client users.

USING THIS BOOK

Administrators for NFS on PRIMOS Systems should read the entire book. In particular, Chapter 5 provides directions to install, configure, activate, and maintain NFS on PRIMOS Systems.

Client NFS users should read Chapters 2, 3, and 4. In particular, Chapter 2 describes important differences between NFS on PRIMOS Systems and other implementations of NFS.

RELATED DOCUMENTATION

The following documentation is helpful for the Administrator of NFS on PRIMOS Systems:

- *NTS Planning and Configuration Guide* (DOC10159-1LA) and its update (UPD10159-11A) describe the hardware requirements and the configuration file for the Network Terminal Service (NTS).
- *NTS User's Guide* (DOC10117-3LA) describes how to use NTS to access 50 Series™ systems on a LAN300.
- *Operator's Guide to File System Maintenance* (DOC9300-5LA) describes the PRIMOS file system and explains how to format disk partitions, run the disk maintenance program, determine physical device numbers, and interpret disk error messages. In addition, this book discusses in detail dynamic badspot handling, mirroring, robust partitions, and record allocation.
- *Operator's Guide to Prime Networks* (DOC10114-1LA) and its update (UPD10114-11A) provide reference information about running network-related programs and monitoring network events.
- *Operator's Guide to System Commands* (DOC9304-5LA) serves as a reference guide for most PRIMOS commands used by operators and administrators.
- *Operator's Guide to the Spooler Subsystem* (DOC9303-4LA) describes how to configure and maintain the Spooler Subsystem on 50 Series systems.
- *PRIMENET Planning and Configuration Guide* (DOC7532-4LA) and its update (UPD7532-41A) describes how to install PRIMENET™ and the File Transfer Service (FTS).
- *User's Guide to Prime Network Services* (DOC10115-1LA) and its update (UPD10115-11A) describes networking services that enable users to access files remotely, transfer files, and log in to other 50 Series systems on a network.
- *ONC Network Programming Guide* (MAN13006-1LA) describes network programming using the RPC library, as well as other ONC services.

PRIMOS users can obtain a complete online list of PRIMOS technical documentation by typing the command HELP DOCUMENTS. The *Guide to Prime User Documents* (DOC6138-9PA) and its supplement provide hardcopy versions of this list, as well as summaries of documentation for other Prime systems. Lists of additional updated material are published every other month in the *Customer Technical Bulletin.*

NEW FEATURES WITH RELEASE 1.2

The major addition to Release 1.2 is the availability of the RPC library to the networked applications developer. Details are given in Appendix E. The RPC library is available, whether Release 1.2 is installed on PRIMOS Rev. 22.0 or on PRIMOS Rev. 23.0. However, the RPCGEN utility has slightly more functionality at Rev. 23.0, where support is provided for pre-processing ifdef statements. (See the *ONC Programming Guide* for an explanation of ifdef statements.)

Two new options have been added to the START_NFS command. These options handle the unlikely discovery of non-unique file system IDs among the directories being exported by NFS on PRIMOS Systems. This may occur for NFS Release 1.2 only on PRIMOS Rev. 23.0. For details see page 5-18: The -FIX_FSID and -NOCHECK_FSID Options.

Release 1.2 supports access to segment directories.

PRIMOS Revision 23.0 provides a singly-rooted file system. Release 1.2 of NFS on PRIMOS Systems now allows you to specify any directory below that root in the EXPORTS file, and you may specify it to any level on the directory tree. For details see page 5-4: Configuring the EXPORTS File.

You may also need to review how NFS clients can print files on the PRIMOS NFS server. For details see page 5-13: Designating a Remote Printer.

PRIME DOCUMENTATION CONVENTIONS

The following conventions are used throughout this document. The examples in the table illustrate the uses of these conventions.

Convention	*Explanation*	*Example*
UPPERCASE BOLD	In command formats, words in uppercase bold indicate the names of commands, options, statements, and keywords issued on a PRIMOS system. PRIMOS commands are not case sensitive. Enter them in either uppercase or lowercase.	**DATE -DOW**
lowercase bold	In command formats, words in lowercase bold indicate the names of commands, options, statements, and keywords issued on a BSD-based or SVID-compliant UNIX® system. These commands are case sensitive. A client user of NFS on PRIMOS Systems should avoid uppercase because each capital letter maps to a two-character representation on the PRIMOS server.	**mkdir SIZE_DOUBLED**
lowercase bold italic	In command formats, words in lowercase bold italic indicate variables for which you must substitute a suitable value.	**LOGIN *user-id***
lowercase italic	In text and in messages, variables are in non-bold lowercase italic.	Supply a value for *num* between 1 and 10.
Hyphen -	Wherever a hyphen appears as the first character of an option, it is a required part of that option.	**SPOOL -LIST**
Abbreviations in format statements	If a command or option has an abbreviation, the abbreviation is placed immediately below the full form.	**SET_QUOTA** **SQ**
Underscore in examples	In examples, user input is underscored but system prompts and output are not.	OK, START_NFS -NOPC (The system outputs the command banner and startup messages.) OK,

Convention	*Explanation*	*Example*
Brackets [ ]	Brackets enclose a list of one or more optional items. Choose none, one, or several of these items.	**LD** [**-BRIEF** **-SIZE**]
Braces { }	Braces enclose a list of items. Choose one and only one of these items.	**CLOSE** {*filename* **-ALL** }
Braces within brackets [{ }]	Braces within brackets enclose a list of items. Choose either none or only one of these items; do not choose more than one.	**BIND** [{*pathname* *options* }]

1

OVERVIEW

NFS on PRIMOS Systems is an implementation of the Network File System (NFS) licensed from Sun Microsystems, Inc.

NFS on PRIMOS Systems allows a client NFS user to access files on a PRIMOS machine. The client NFS host may be a BSD-based or SVID-compliant UNIX® system, or another system that implements the client side of the NFS protocol — for example, a personal computer running PC-NFS™ over its DOS-based operating system. The client NFS host must be

- Directly connected to an IEEE 802.3-compliant local area network
- Using a version of NFS that is licensed by Sun Microsystems, Inc.

NFS on PRIMOS Systems is limited to Server NFS; that is, it provides file service for files on PRIMOS systems to users on non-PRIMOS client NFS systems.

This chapter provides an overview of the following:

- Hardware and software requirements
- Differences from other NFS implementations
- Components of NFS on PRIMOS Systems
- How it works
- Handling NFS problems

HARDWARE AND SOFTWARE REQUIREMENTS

To use NFS on PRIMOS Systems, you must install it on a 50 Series™ machine that satisfies the following hardware and software requirements:

- Support IX mode (any 4-digit machine except the 2250™ system).
- Be running PRIMOS Revision 22.0 or later (Rev. 22.1 is needed for PC-NFS user authentication).
- Have its partitions formatted with a Revision 20 (or later) version of the **MAKE** command, if these partitions are to be accessible to NFS.
- Be running Release T1.4-21.0 or later of the T-family compiler libraries.
- Be connected directly to an IEEE 802.3-compliant local area network.

Note

The PRIMOS machine may also be a gateway node on PRIMENET. Other 50 Series nodes accessible through PRIMENET do not need NFS installed to make their files accessible. However, to provide NFS access they must all

  - Be running PRIMOS Revision 22.0 or later
  - Be configured for PRIMENET with remote file access but without forced user validation
  - Have partitions formatted at Revision 20 or later

- Have PRIMOS TCP/IP software installed on it. (PRIMOS systems accessible via a PRIMENET gateway do not need TCP/IP.)

Figure 1-1 illustrates a sample configuration that satisfies the hardware and software requirements for NFS on PRIMOS Systems.

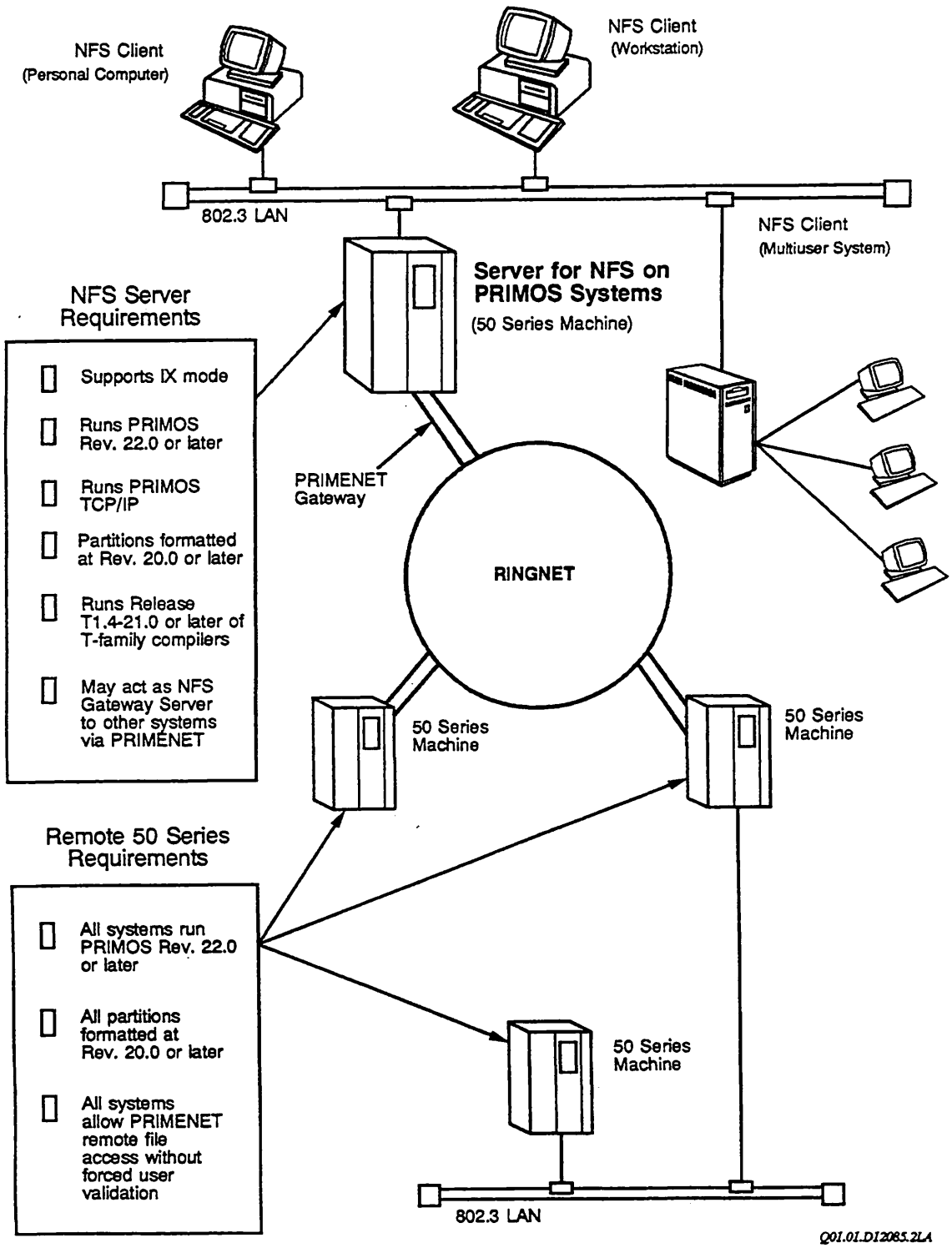

FIGURE 1-1. Hardware and Software Requirements for NFS on PRIMOS Systems

DIFFERENCES FROM OTHER NFS IMPLEMENTATIONS

There are a few areas where NFS on PRIMOS is different from other NFS implementations. These areas are

- NFS without Sun extensions
- Different rules for legal filenames
- Different text file formats
- PRIMOS ACL rights versus SVID-compliant access permissions

See Chapter 2 for details on the first three areas of difference. See Chapter 3 for details on access differences.

COMPONENTS OF NFS ON PRIMOS SYSTEMS

NFS on PRIMOS contains, or works with, the following components:

- Client commands that perform NFS file system services
- Commands to administer NFS
- NFS servers
- Access provisions for client NFS hosts and their users
- Text file conversion programs

A summary of each component follows.

Client Commands That Perform NFS File System Services

NFS on PRIMOS Systems supports three general types of commands on the client system:

- Commands that enable access to files and directories on a PRIMOS server host
- Interactive commands for manipulating PRIMOS files and directories
- Commands within programs for manipulating PRIMOS files and directories

Commands of the first type directly affect how NFS works with the PRIMOS server. These commands are explained in Chapter 4. See your client NFS system documentation for an explanation of the other command types.

Commands to Administer NFS

The Administrator for NFS on PRIMOS Systems uses five commands for daily administration. See Chapter 5 for a discussion of administrative issues and for descriptions of the commands **START_NFS**, **STOP_NFS**, **NFS_INIT_USERS**, and **NFSSTAT**. See Chapter 6 for a discussion of **RPCINFO** as a tool to diagnose NFS status on remote systems.

NFS Servers

NFS on PRIMOS Systems uses these phantoms:

- **NFS_MOUNT**
- **NFS_PORTMAP**
- **NFS_SERVER**
- **NFS_PCNFSD** (optional, to support PC clients)

You can verify that NFS is running on PRIMOS by using the command **STAT USER**. These servers for NFS should be listed as phantoms. For further details on NFS servers, see Chapter 5.

Access Provisions for Client NFS Hosts

NFS on PRIMOS does not allow any breach of PRIMOS system security while it provides access to users on client NFS hosts.

PRIMOS security is controlled by Access Control Lists (ACLs). Most versions of NFS provide security by using BSD-based or SVID-compliant access permissions. NFS on PRIMOS Systems combines these two security systems.

ACLs Providing NFS Security: The remote user with access to PRIMOS files via NFS does not need to learn about PRIMOS ACL rights to create subdirectories and subfiles.

A client NFS user can create remote objects using common BSD-based or SVID-compliant file system commands to create a directory or file. NFS on PRIMOS Systems automatically translates these into ACL rights for the object.

A client NFS user can also set specific access permissions on remote NFS objects using common BSD-based or SVID-compliant commands to change permissions. NFS on PRIMOS converts these access permissions into PRIMOS ACL rights, automatically and transparently. The client NFS user sees the new access permissions; only the local PRIMOS user sees the corresponding ACLs that have been created.

Unlike the general user of NFS on PRIMOS, the NFS Administrator needs to know how access permissions are mapped to PRIMOS ACLs. Chapter 3 provides this information. Chapter 3 also provides background and procedures for managing PRIMOS files and directories that must be accessible to both client NFS users and local PRIMOS users.

Files Providing NFS Security: When NFS on PRIMOS Systems is started, the system first starts the NFS servers and then scans three files important to NFS security.

The three files, all located within the **NFS_>ETC** directory, provide the information that makes PRIMOS files and directories available to selected users on selected client NFS hosts. The **PASSWD** file lists qualified users. The **GROUP** file lists qualified groups. The **EXPORTS** file lists the PRIMOS partitions that are available to NFS clients. NFS Administrators can refer to Chapter 5 for directions on configuring these files.

Text File Conversion Programs

NFS on PRIMOS Systems comes with several programs for converting text files that otherwise may appear garbled because of differences in operating systems. Four programs run on the PRIMOS host:

- **PTOU** and **UTOP** (for PRIMOS/UNIX conversions)
- **PTODOS** and **DOSTOP** (for PRIMOS/DOS conversions)

The UNIX or DOS client host runs the other programs, provided on PRIMOS as C source files. The client uses its own C compiler to create executable programs. Source files for the UNIX client host are **PTOU.C** and **UTOP.C.** Those for the DOS client host are **P2DOS.C** and **DOS2P.C** (note the name difference for the PC client). See Chapter 2 for details on these conversion programs.

HOW IT WORKS

All NFS implementations regard a host as a client, a server, or both. A client host can mount a file system or a directory located on a server host; thereafter, the client host can gain access to the mounted file system or directory as if it were a local file system or directory.

Mounting BSD-based or SVID-compliant File Systems

When the server host is a BSD-based or SVID-compliant operating system, a client host is able to mount the server's entire file system with a single NFS command. By mounting the root, or slash (/), directory, the client gains access to all the root subdirectories and subfiles. For example:

```
% /etc/mount -o soft xinu:/ /from_xinu
```

The pathname **/from_xinu** is created on the client as a **mountpoint** that points to the entire remote file system on the server **xinu.** The mountpoint contains enough information to make the remote object accessible.

Mounting PRIMOS Directories

PRIMOS systems previous to Revision 23.0 did not support a file system with a single source or root directory. The user saw a multi-rooted file system, and the operator directed PRIMOS to internally collate and keep track of the individual disk partitions. PRIMOS now supports a singly-rooted file system, and all disk partitions are accessible as top-level directories under root (<).

NFS on PRIMOS Systems supports root-directed mounts, but it does not support a mount of the entire PRIMOS root (<). Each NFS remote mount object on the PRIMOS file system must be fully specified to some lower level on the directory tree. You first specify this level of access in the EXPORTS file. Until the PRIMOS server exports access to a directory, no NFS client can mount the directory. See Chapter 5 for details on the EXPORTS file.

After directories are exported, a client must issue separate **mount** commands for each client mountpoint. Each remote mount object remains mounted until it is explictly unmounted. Each remote mount object has its own mountpoint on a client NFS host.

For example, the PRIMOS NFS server system called PRIME1 shares a "common file system name space" with two other PRIMOS systems, so that resident PRIMOS users on PRIME1 see twenty top-level directories under root.

Under <, three of the twenty top-level directories are to be accessed by NFS clients: **<OSCMDS**, **<APPLIX**, and **<USERS**. The NFS administrator on PRIME1 exports only these directories, because only these are needed by NFS clients. (The EXPORTS file could specify access points lower than the top-level directory on each directory tree.)

A client host, **sysx**, needs to use three NFS mount commands to gain access to the directories at or below the tree level that has been exported by PRIME1. In this situation the mountpoints go to a lower level on each directory tree:

```
% /etc/mount prime1:/oscmds/for_nfs /prime1_oscmds
% /etc/mount prime1:/applix/for_unx_nfs /prime1_applix
% /etc/mount prime1:/users/nfsusers/clients/fromsysx /prime1_users
```

To summarize, the remote mountpoint can provide access that begins at any level on the directory tree (so long as access rights or permissions are also in place). This applies on the server side (with the EXPORTS file) and on the client side (with the **mount** command). The **mount** command must specify a mountpoint that is at or lower than the exported mountpoint.

See Chapter 4 for details on the commands to mount and unmount PRIMOS directories, or to display the accessible mountpoints on the NFS server.

Accessing Remote Objects

After a remote directory tree is mounted, then a user on a client NFS system accesses an object on a mounted remote directory simply by including the name of the mountpoint in the pathname of the object. For example, assuming that an Administrator executed the **/etc/mount** commands in the preceding paragraph, a user on **sysx** can now use the pathname **/prime1_users/frost** to access a particular user directory that exists on the NFS server PRIME1. The actual directory resides on PRIME1 with the pathname of **<USERS>NFSUSERS>CLIENTS>FROMSYSX>FROST.**

Similarly, you can change your working directory using the mountpoint as follows:

```
% cd /prime1_applix
```

A mountpoint to a PRIMOS directory makes the remote directory appear to be local. To a client user, a mountpoint looks like any other directory below the root directory /. A client user with proper permissions may access lower levels of the tree structure as if it were a local file system.

See your client NFS system documentation for the local system commands that can manipulate NFS-mounted file system objects. If yours is a BSD-based or SVID-compliant system, most relevant commands should be available.

HANDLING NFS PROBLEMS

NFS on PRIMOS Systems provides assistance for solving problems that may occur with the client host, the server host, the IEEE 802.3-compliant local area network, or PRIMENET.

Chapter 6 provides some diagnostic troubleshooting procedures to help the NFS Administrator.

Appendix A provides reference information (in the form of manual pages) for generic NFS commands, files, and servers. Your client NFS system may have additional documentation.

Appendix B provides printed versions of **HELP** files for resident commands used with NFS on PRIMOS Systems. On PRIMOS you can also get online displays of these commands. Use the **HELP** command followed by one of these command names: **START_NFS, STOP_NFS, NFSSTAT, NFS_INIT_USERS, RPCGEN, RPCINFO, PTODOS, DOSTOP, PTOU,** or **UTOP.** If you install the optional RUSERS service, described in Appendix E, an online display for **RUSERS** is also available.

Appendix C lists error messages and explanations for NFS on PRIMOS Systems.

Appendix D describes filename character mapping procedures for NFS on PRIMOS Systems and discusses different formats for ASCII characters (based on differences among operating systems).

WRITING RPC PROGRAMS

With PRIMOS NFS Release 1.2, programmers can develop networked applications that use the Remote Procedure Call (RPC) library upon which PRIMOS NFS itself is built. See Appendix E for details.

2

DIFFERENCES FROM NFS ON OTHER SYSTEMS

NFS on PRIMOS Systems differs from other NFS implementations, most notably by supporting only Server NFS.

This chapter discusses the following differences between NFS on PRIMOS Systems and other implementations, especially those on BSD-based or SVID-compliant systems:

- It is NFS without Sun extensions
- It has different rules for legal filenames
- It has different text formats for ASCII files
- It uses PRIMOS ACL rights translated from BSD-based or SVID-compliant access permissions
- It has other minor differences

NFS WITHOUT SUN EXTENSIONS

NFS on PRIMOS Systems supports only the server side of Sun's Version 2 NFS protocol and Version 1 mount protocol.

Many NFS implementations include functions that are not actually part of Sun's NFS protocol definition. Sun distinguishes between NFS itself and the larger set of protocols called Open Network Computing (ONC).

The following services are *not* part of NFS on PRIMOS Systems:

- Support for the Sun RPC and XDR programming interfaces, including RPCGEN
- Yellow Pages
- Network Lock Service (**lockd** and **statd**)
- Berkeley remote services (**rex, rlogin, rwhod,** and so on)
- Secure NFS based on a secure RPC (although NFS on PRIMOS Systems supports other new NFS features in SunOS 4.0)

RULES FOR LEGAL FILENAMES

The PRIMOS rules for legal filenames are different from those on a DOS-based PC-NFS client, or those on BSD-based or SVID-compliant systems. For example, the permissible length of a filename is shorter on a PRIMOS system. PRIMOS makes no distinction between uppercase and lowercase for filenames. PRIMOS does not allow control characters within a filename. Most difficulties are solved with a system that provides character mappings for two-way conversions between SVID-compliant UNIX and PRIMOS, and also conversions between DOS and PRIMOS. Appendix D provides two tables of character mappings established for NFS on PRIMOS Systems – one for UNIX/PRIMOS and one for DOS/PRIMOS. Appendix D also provides details on the character mappings that supplement information in this chapter.

If you use client NFS on a BSD-based or SVID-compliant system to create a PRIMOS file, you see a different filename when logged in as a PRIMOS user. For example, the filename seen from the client system as **ReadMe** becomes **/READ/ME** when seen on the PRIMOS system. Likewise, the filename seen from the client system as **1988-status** becomes **/&1988-STATUS** when seen on the PRIMOS system.

A reverse transformation occurs if you create PRIMOS files while logged in to the 50 Series system, and then you look at their filenames using NFS from a DOS-based client system, or from a BSD-based or SVID-compliant system. For example, the filename created on the PRIMOS system as **TCP/IP** is seen by the UNIX client NFS user as **tcpIp.** Likewise, the PRIMOS filename **BIND&SEG** is seen as **bind?eg** (where **?** is a Control-S).

Any PRIMOS filename that includes one or more of the following character sequences cannot receive a recognizable translation when seen from a BSD-based or SVID-compliant system:

```
&/
&& (except for the sequences &&0 and &&1)
/& (except when it begins a filename and is followed by a digit)
```

Filenames that contain such character sequences are translated to **?!. See page D-7 for details. PRIMOS uses the character sequences // and && to map 8-bit characters from PC-NFS clients. See page D-5 for details.

If a client system filename is too long, NFS on PRIMOS Systems cannot map it into a filename. This may happen when a client system tries to create a new file or directory whose name is over 32 characters, or whose name becomes oversized after trying to translate too many uppercase characters or other special characters. The client system usually displays an error message or returns an error code to indicate an invalid argument. On a Sun Workstation™, you see an error message similar to `Server can't decode arguments`.

TEXT FORMATS FOR ASCII FILES

ASCII file text formats on PRIMOS differ from those on DOS-based and on BSD-based or SVID-compliant systems in

- Character set
- Space compression procedures
- End-of-line conventions

NFS on PRIMOS Systems does not convert text files from one format to another automatically. When a program on an NFS client system uses NFS to write a text file onto a PRIMOS partition, it stores the text in the format native to the client. The file maintains that format, even though it is on a PRIMOS partition. Likewise, when the client program uses NFS to read a text file already created in PRIMOS format, it sees the text in its PRIMOS format.

NFS is designed to provide file system access to many different operating systems on a network. Some NFS users on DOS-based systems, or on BSD-based or SVID-compliant systems, may not be used to seeing remote files with a format different from their local ones, and at first they may want the files converted. While the file conversion is easy enough to do, it may not be necessary. Therefore, NFS has no built-in conversion procedure, in keeping with the "receiver makes it right" axiom. The user can decide if conversion is necessary, and then use one of the Prime-supplied conversion tools to do the job.

ASCII File Conversion Tools

NFS on PRIMOS Systems is shipped with several programs that convert ASCII files between text formats with embedded operating system dependencies. These programs enable PRIMOS/UNIX conversions and PRIMOS/DOS conversions. The PRIMOS host runs four object code programs – two pairs for conversions to-and-from PRIMOS/UNIX and PRIMOS/DOS. Two additional C source code programs are provided for the UNIX-based client host. Two more are provided for the DOS-based client host. The client user must submit these source code programs to the C compiler on the client host.

All conversion tools have similar paired names and provide the same conversion service for both client and server. They are discussed as separate pairs below.

PRIMOS/UNIX Conversion Tools on PRIMOS: These conversion tools are **UTOP** and **PTOU**. **UTOP** is the UNIX-to-PRIMOS conversion tool, as used on the PRIMOS system. Its command-line format is

UTOP ***unix_format_file*** [***primos_format_file***] [**-TAB** ***tabsize***]

If the second filename is not specified, the original file is overwritten with the converted text. The **-TAB** option enables you to recreate the spacing associated with the tab in the UNIX file. PRIMOS does not recognize the tab size given in the header of the UNIX file, but assigns a default value of eight spaces to a tab, which is the usual value for a tab within a UNIX file. If the value for a tab is different within the UNIX file, then you can specify the new value after the **-TAB** option during the conversion.

PTOU, the PRIMOS-to-UNIX conversion tool used on the PRIMOS system, has the same command-line format with the arguments in reverse order:

PTOU ***primos_format_file*** [***unix_format_file***]

For each conversion tool, the first filename is the input and the second filename is the output. Use these programs when converting files to the other format. For example,

- You want to use the PRIMOS PASCAL compiler to compile a file stored in UNIX text format on the PRIMOS system. First you submit the file to the UTOP conversion tool to make a copy in PRIMOS format. Then you submit the new copy of the file to the PASCAL compiler.
- You want to make the files in the PRIMOS directory **SYSCOM** readable to BSD-based or SVID-compliant workstations. You can use the **PTOU** command to create a new PRIMOS directory with versions of those files in UNIX text format:

```
OK, PTOU SYSCOM>@@.CC  UNIX_SYSCOM>==
```

  A user on a BSD-based or SVID-compliant system can then read and (if access rights permit) update those files.

PRIMOS/DOS Conversion Tools on PRIMOS: These conversion tools are **PTODOS** and **DOSTOP**. Use these tools for conversions comparable to the UNIX/PRIMOS conversions discussed in the previous paragraph.

The command-line format for **DOSTOP**, the DOS-to-PRIMOS conversion tool, is

DOSTOP ***dos_format_file*** [***primos_format_file***] [**-TAB** ***tabsize***]

The command-line format for **PTODOS**, the PRIMOS-to-DOS conversion tool, is

PTODOS ***primos_format_file*** [***dos_format_file***]

Conversion Tools for UNIX-based NFS Clients: These tools are C source code programs: **UTOP.C** and **PTOU.C**. They are available in the PRIMOS directory **NFS_>UNIX_UTILS**.

Note

The files for the NFS client are already in UNIX text format. Do *not* convert them!

A resident PRIMOS user can read only the **README-FROM-PRIMOS** file in this directory:

```
OK, A NFS_>UNIX_UTILS
OK, LD
<MFDNM>NFS_>UNIX_UTILS (ALL access)
20 records in this directory, 20 total records out of quota of 0.

5 Files.

/MAKEFILE        PTOU.C        README        README-FROM-PRIMOS
UTOP.C
```

The client NFS user should read the file **readme** (named as seen by the client). The file provides directions for using the **ptou** and **utop** tools with a command line more appropriate to SVID-compliant and BSD-based systems. Further directions are provided within the **readme** file for using the **Makefile** shell script.

Conversion Tools for DOS-based NFS Clients: These tools are C source code programs: **DOS2P.C** and **P2DOS.C**. Notice that the base names differ slightly from the PRIMOS-resident utilities. The C source programs are available in the PRIMOS directory **NFS_>DOS_UTILS**.

Note

The files for the NFS client are already in DOS text format. Do *not* convert them!

A resident PRIMOS user can read only the **README-FROM-PRIMOS** file in this directory:

```
OK, A NFS_>DOS_UTILS
OK, LD
<MFDNM>NFS_>DOS_UTILS (ALL access)
20 records in this directory, 20 total records out of quota of 0.

5 Files.

MAKEFILE        DOS2P.C        P2DOS.C        README
README-FROM-PRIMOS
```

The client NFS user should read the file **readme** (named as seen by the client). The file provides

- Directions for using **makefile** (named as seen by the client) to create executable versions of the tools.
- Directions for using the **dos2p** and **p2dos** tools

Other ASCII Text Issues

When NFS files have been written on PRIMOS, they may have an odd number of bytes, which is not typical for native files on PRIMOS. A bit in the header of each file is now reserved to flag an odd number. However, this bit may be ignored during a PRIMOS **COPY** of the file. To prevent this from happening, issue the **COPY** command accompanied by the option **-COPY_ALL** or **-PROTECT**.

The same problem occurs when copying a file to magnetic tape. Use the **MAGSAV** and **MAGRST** commands to preserve the extra bit. The **BRMS** utilities do not provide this protection.

A related problem concerns the preservation of read/write locks on a file. A file created by a client NFS user has open locks set on it, which is different from the default for native files on PRIMOS. The problem arises when the PRIMOS user copies a file created by a client NFS user. To preserve the open locks set by NFS, the PRIMOS user must use the **COPY** command with the **-RWLOCK** option.

ACCESS RIGHTS TRANSLATION

Perhaps the most important difference between NFS on PRIMOS Systems and other versions of NFS is the way user access and security are handled. NFS on PRIMOS Systems maps BSD-based and SVID-compliant access permissions to native PRIMOS ACL rights. The translation mechanism remains transparent to the client NFS user, but the NFS Administrator should know its internals.

Chapter 3 provides a thorough explanation of the access rights translation used by NFS on PRIMOS Systems.

OTHER MINOR DIFFERENCES

Reliability Guarantees

Most implementations of NFS are stateless. In practice this means that if a client requests any operation that changes state on the server, the operation is guaranteed to be complete and its results stored on the disk before the server returns. This way, no context on the server can be lost if the server crashes just after returning a reply.

PRIMOS matches this guarantee for files, but not for directories. All file writes are recorded to disk before the server returns from the request. (You can enhance performance by overriding this default at startup: **START_NFS -NO_FORCE_WRITE**. See Chapter 6 for details.) However, the server cannot guarantee that all directory-changing operations are safely recorded on disk. It is possible that if a client does a directory-changing operation (using **mkdir**, **chmod**, **rm**, or **mv**) and the server crashes immediately after returning the reply, the results of the operation may be lost.

Remaining Minor Differences

Keep the following points in mind:

- NFS on PRIMOS Systems does not support a link, either hard or soft, from a BSD-based or SVID-compliant system. An attempt to do either a link or a symbolic link to or from a file on a PRIMOS partition fails. On a Sun Workstation, you receive a message to the effect that `NFS link [or symlink] failed for server` *server_name*`: RPC: procedure not supported.`
- PRIMOS segment directories and password directories are not accessible to clients. If you give the command **ls -l** on a directory that contains one of these, no access rights are listed for that particular object.
- You can rename (for example, using **mv**) a PRIMOS object, but its destination object must be within the same directory.
- If your client NFS implementation maps **root** to nobody, then any NFS operation using this **uid** is granted only **$REST** rights to PRIMOS objects. This happens, even if the **PASSWD** file on the PRIMOS system maps **root** to a legitimate PRIMOS user name. The NFS protocol specifies that clients limit **root** to this access in order to preserve network security.
- Remember that BSD-based or SVID-compliant systems usually include the current directory in command search rules (**$PATH**). If, as a client NFS user, you are in a PRIMOS directory which contains the file **ls** and you give the command **ls**, your host tries to execute the file. If the file is not the actual command, unpredictable results occur.

3

ACCESS CONTROL FOR NFS ON PRIMOS SYSTEMS

INTRODUCTION

Many client users of NFS on PRIMOS System will be running a version of the BSD-based or SVID-compliant UNIX operating system. Consequently, this chapter compares the PRIMOS access control mechanism with that of the other operating system, and explains the translations between them.

This chapter describes the following:

- How BSD-based and SVID-compliant systems handle access control
- How the PRIMOS system handles access control
- How NFS on PRIMOS Systems translates UNIX operating system permissions to PRIMOS access rights, and vice versa
- How NFS on PRIMOS Systems determines ownership of files and directories on the Prime file system
- How NFS on PRIMOS Systems generally manages PRIMOS ACLs
- How to handle some access problems

To test the following examples, first install and configure NFS on PRIMOS Systems, following the directions in Chapter 5.

REVIEW OF ACCESS CONTROL ON UNIX OPERATING SYSTEMS

Features of access control for the UNIX operating system that influence the PRIMOS access control mapping strategy are

- File ownership
- Access permissions
- Separate access modes (for each file system object)

File Ownership

Within the UNIX file system, every file system object (including a directory) is considered a file. Every file has two owners: one specific user and one specific user group.

A user creating a new file automatically becomes its owner. Whatever group that user belongs to at the time of file creation is the group owner of the file. Ownership may be transferred to another user or group by the current user owner; by doing this, however, the owner generally forfeits control over that file.

Whenever a user attempts to access a file, the system checks the following:

1. Is this user the owner of the file?
2. If not, is this user a member of the group that is the group owner of the file?
3. If not, this user is classified as "other" (similar to **$REST** on PRIMOS).

In this way, each file has three categories of user access assigned to it:

- *user owner*
- *group owner*
- *other*

Access Permissions

You can assign three specific permissions for a file on UNIX systems:

- `r` (*read*)
- `w` (*write*)
- `x` (*execute*)

Permissions are assigned for every file and for each category of user, in the order of *user owner*, *group owner*, and *other*. You can see the active permissions on a file or directory by issuing an ls command with the -l option. The permissions, preceded by a "d" or a "-" to indicate directory or file, take the form shown under *Display*:

d=dir	*user owner*	*group owner*	*other*		*Display*
-	rwx	rwx	rwx	=	-rwxrwxrwx (file)
d	rwx	rwx	rwx	=	drwxrwxrwx (directory)

Access permissions are also expressed as an octal number in the range of zero (0) to seven (7), where r = 4, w = 2, and x = 1. Thus — (no rights) is 0, and rwx is 7. Likewise, rw- may be expressed as 6, r-x as 5, r— as 4, and —x as 1. The three groups of permissions strung together are sometimes called *permission bits* or *permbits*.

In the following example, a file has permbits that give to the *user owner* all permissions (read, write, and execute), to the *group owner* read and execute permissions, and to *other* an execute-only permission:

```
rwxr-x--x
```

These permissions would be expressed with the octal notation 751.

The following sequence of commands shows the effect of changing permissions on a file:

```
% ls -lg file.txt
-rwxr-xr--  1 alpha    user          1060 Mar 15 13:23 file.txt

% chmod 051 file.txt

% ls -lg file.txt
----r-x--x  1 alpha    user          1060 Mar 15 13:23 file.txt
```

In this example, alpha is the *user owner* of **file.txt**, and user is the *group owner*.

Separate Access Modes

Permissions assigned to a file system object, plus other information such as whether the object is a file or a directory, make up the entire rwx string shown in the previous examples; this string is referred to as an *access mode* or *modebits*. *Each file* has its own specific access mode. This is an important distinction from a PRIMOS file system object, which may share access control with another object – usually a parent directory from which it inherits default access rights.

REVIEW OF ACCESS CONTROL ON PRIMOS SYSTEMS

The following features of PRIMOS access control are discussed:

- Access rights
- File ownership
- Default access right inheritance

Access Rights

PRIMOS uses access rights that are similar to permissions in UNIX operating systems. However, PRIMOS has two distinct sets:

- Access rights that apply to files
- Access rights that apply to directories

The file rights (**RWX**) correspond exactly to **rwx** on UNIX systems. The directory rights are **PDALU**: **P** for *protect*, **D** for *delete*, **A** for *add*, **L** for *list*, and **U** for *use*. These rights apply to operations allowed on the directory itself or on the contents of the directory.

Though it is possible to assign directory access rights to PRIMOS files, doing so has no effect on access control to those files. However, assigning file access rights to a directory is useful because of the PRIMOS default access right inheritance mechanism, described in the Default ACL Inheritance section.

In addition to file and directory-specific access rights, there is an additional access right which applies to both files and directories and which is used to show "ownership" of file system objects. This access right is **O** for *owner*. Though referred to as an access *right*, **O** actually enforces no access control and is merely advisory in the PRIMOS operating system. NFS on PRIMOS Systems, however, uses **O** to complete the access control mapping mechanism for file ownership as on a SVID-compliant system. For more about this, see the upcoming section, Determining PRIMOS File Ownership.

The string **NONE** denies all access rights, and the string **ALL** allows all access rights (**OPDALURWX**).

How Users Get Access

PRIMOS access rights are associated with specific users, with specific user groups and with everybody else (**$REST**). More than one user name and group name may have access rights specified for them. Each association of user name and access rights is referred to as a pair, and pairs are grouped into a list to form an Access Control List, or ACL. The ACL shows a list of pairs for each category of user access: first for specific user names, then for group names, and finally for **$REST**. Each category lists names alphabetically. An ACL may contain up to 32 entries (pairs), user names and group names combined, plus one additional

entry for **$REST**. ACLs are set on individual objects in the PRIMOS file system. See Figure 3-1 for an example of an ACL for a directory.

```
ACL protecting "SYSCOM":
        GUEST:          NONE
        MANAGER:        PDALURWX
        SYSTEM:         ALL
        .ADMIN$:        PDALURWX
        .BACKUP$:       LURX
        $REST:          LUR
```

FIGURE 3-1. A Typical Directory ACL

When a user tries to access a file or directory, the system checks the following:

1. Is the user's name listed in the ACL? If so, then the access rights associated with that user name are checked to see if they are sufficient to allow the requested operation.
2. If not, is the user a member of any of the groups listed in the ACL? If so, then the access rights associated with that group name are checked.
3. If neither of the above, then the access rights associated with **$REST** are checked to see if they are sufficient to allow the requested operation.

Limiting File Ownership

Since there can be many user names and group names listed in an ACL, and since any number of these names (including zero) could have the **O** access right, it is possible to have either *no* owners of a given PRIMOS file system object, or *many* owners. Consequently, it is not standard practice in PRIMOS to distinguish the single owner of a given file or directory. However, one could adopt some convention in order to do so; one such could be that the owner is the user name in the ACL that has the most access rights. Applying this convention to the ACL shown in Figure 3-1, the user **SYSTEM** would be interpreted as the owner of the directory **SYSCOM**.

Default ACL Inheritance

Unlike access modes established by permission bits, ACLs do not have to be specified for every object in the PRIMOS file system. When first created, in fact, a PRIMOS file or directory inherits the ACL associated with its parent directory, whose ACL in turn could have been inherited from its parent. The user can either set a specific ACL on the newly created object or do nothing, thereby accepting the default. This feature offers a "trickle-down" convenience: a user can modify a directory's ACL and have the modifications take effect on all the objects contained within that directory that do not have specific ACLs of their own.

Since PRIMOS has different sets of access rights for files and for directories, one might think that there would be confusion about a *file's* access rights when its ACL is inherited from its parent *directory*. One convention that solves this problem is to include file-specific access rights with a directory's rights. Files in that directory thereby inherit the proper file-specific access rights. Referring to Figure 3-1, you can see that the directory **SYSCOM** has **RWX** file access rights assigned to user names and group names, and **R** assigned to **$REST**. Any files in the directory **SYSCOM** which inherit this ACL will automatically be covered by these file-specific access rights, without the need for assigning a specific ACL to those files.

Default ACLs is a PRIMOS concept that has no counterpart on the UNIX operating system. NFS on PRIMOS Systems addresses this major difference by assigning specific ACLs to every directory and file that it creates or modifies. For details on how NFS on PRIMOS Systems manages ACLs, see the section ACL Management by NFS on PRIMOS Systems on page 3-10.

ACCESS RIGHTS MAPPING

The next two tables show the translation of access rights – from the PRIMOS system to UNIX systems and back to the PRIMOS system. Table 3-1 shows the translation for files, and Table 3-2 shows the translation for directories.

TABLE 3-1. File Access Rights Translation

PRIMOS Right	*UNIX Permission*	*PRIMOS Right*
R	r	R
W	w	W
X	x	X

Both tables are to be read left-to-right so that the translation from PRIMOS access rights to a UNIX system's permission is shown by beginning in the leftmost PRIMOS Right column and reading over to the UNIX Permission column. The translation then continues, showing how the UNIX Permission in the middle column is further translated into the rightmost PRIMOS Right column.

TABLE 3-2. Directory Access Rights Translation

PRIMOS Rights	*UNIX Permission*	*PRIMOS Rights*
L	r	LR
PDA	w	PDAW
U	x	UX

While there is a direct one-to-one correspondence between *file* access rights and permissions, making the translation for files simple, this is not the case for *directory* access translation. The directory rights mapping has two differences:

- PRIMOS **PDA** rights are needed to achieve all the privileges that the **w** permission confers on a directory.
- When translating from permissions, NFS on PRIMOS Systems augments the resulting PRIMOS *directory* rights with corresponding *file* rights. This is done to maintain the "feel" of the PRIMOS default ACL mechanism for native PRIMOS users. This way, a PRIMOS user working in a directory whose ACL was set by NFS on PRIMOS Systems would have the expected access to that directory's files created under PRIMOS.

Figure 3-2 shows examples of translations from access permissions to access rights.

DETERMINING PRIMOS FILE OWNERSHIP

As previously discussed, the NFS protocol requires ownership of files, but PRIMOS files do not necessarily have to be "owned", and could even have several "owners". Therefore, NFS on PRIMOS Systems must interpret ownership of PRIMOS files to conform to the NFS protocol, being careful to distinguish a single user owner and a single group owner for any given PRIMOS file or directory. It derives much of the information from the **PASSWD** and **GROUP** files, the owner registration files for NFS on PRIMOS Systems (see Chapter 5 for details). If their default location is accepted, these files are located in **NFS_>ETC**.

The following procedure determines a single user owner:

1. Of the PRIMOS users listed in an ACL, find the first user name with the **O** access right.
2. If this user name is listed in the PRIMOS **PASSWD** file, then this user is the owner; otherwise, repeat steps 1 and 2 until identifying a valid owner or exhausting the search.
3. If no PRIMOS users have **O**, or if those that do are not listed in the **PASSWD** file, then repeat steps 1 and 2, looking for the **P** access right.
4. If no PRIMOS users have **O** or **P**, or if those that do are not listed in the **PASSWD** file, then there is no user owner for this file or directory. NFS on PRIMOS Systems returns **uid=-2** to the client to signify this fact. (Different representations of non-ownership will be displayed by the **ls** commands in different client versions of UNIX. You may see -2, **65534**, or the strings **nobody**, **noowner**, and the like.)

NFS on PRIMOS Systems determines a single group owner by applying the previous steps to the PRIMOS group names listed in the ACL. If the group name is not in the (**GROUP**) file, there is no group owner.

Notes

- Both user names and group names are listed alphabetically in the PRIMOS ACL. This affects the determination of ownership.
- Even though the user name **NFS_SERVER** is placed in every ACL that NFS on PRIMOS Systems modifies, **NFS_SERVER** is never returned as the owner of a PRIMOS file or directory.

Figure 3-2 shows PRIMOS ACLs corresponding to UNIX access modes for a directory and a file. Within this example, both an NFS user and a PRIMOS user are checking access to the same subdirectory and file. NFS on PRIMOS maps the client users **alpha** and **omega** to the like-named PRIMOS user names. Likewise, the client groups **user** and **staff** map to the PRIMOS ACL groups **.USER** and **.STAFF**.

Refer to Figure 3-1 for a more complicated example of determining PRIMOS file ownership from an ACL originally created by a native PRIMOS user. NFS on PRIMOS Systems would pick out **SYSTEM** as the user owner and **.ADMIN$** as the group owner, provided that these names are properly listed in the **PASSWD** and **GROUP** files.

Access modes as seen by the client NFS user:

```
% ls -lg
drwxr-xr-x  2 alpha     user          554 Jan 10 13:18 sourcedir
-rwxr--r--  1 omega     staff        1060 Mar 10  1990 prog.o
```

Access rights as seen by the PRIMOS user:

```
OK, LIST_ACCESS SOURCEDIR
ACL protecting "SOURCEDIR":
        ALPHA:              ALL
        NFS_SERVER:         PDALURWX
        .USER:              OLURX
        $REST:              LURX
OK, LIST_ACCESS PROG.O
ACL protecting "PROG.O":
        NFS_SERVER:         PDALURWX
        OMEGA:              ORWX
        .STAFF:             OR
        $REST:              R
```

FIGURE 3-2. ACLs Set by NFS on PRIMOS Systems

To summarize, the user and group owners are the first user name and group name listed in an ACL that have the **O** access right. These names must also be listed in a properly configured **PASSWD** or **GROUP** file. If the files have no user names or group names with the **O** access right, then the owners are the first names with the **P** access right. If no user has the **P** access right, then there is no user or group owner.

ACL MANAGEMENT BY NFS ON PRIMOS SYSTEMS

Several client operations cause NFS on PRIMOS Systems to set or modify ACLs on PRIMOS file system objects. Some examples of these operations are

- Creation of files or directories
- Commands that modify a file system object, such as **chmod**, **chown**, or **chgrp**
- Operations that change a file's date/time stamp

Note that not all versions of UNIX implement such operations identically. Therefore, an operation for one version of UNIX may cause NFS on PRIMOS Systems to modify an ACL, whereas the same operation for another version of UNIX may not.

Specific ACLs

NFS on PRIMOS Systems does not support the PRIMOS concept of a default ACL that can be inherited by subdirectory or subfile. Any ACL set or modified by NFS on PRIMOS Systems becomes a specific ACL. New files and directories created through NFS always receive a new specific ACL, and any previously existing file or directory that inherited its parent directory's ACL now has its own specific ACL after it is modified by NFS on PRIMOS Systems. After this occurs, such a file no longer inherits any changes made to its parent directory's ACL.

PRIMOS Directory Limit: Because NFS on PRIMOS Systems sets specific ACLs, a PRIMOS directory is more likely to reach its maximum quota of file system objects. While this maximum number is in the thousands, a directory's capacity is reduced by about a third, since each specific ACL occupies space in a PRIMOS directory, along with file entries and directory entries. Because this limit is still so large (numbering in the thousands), users are unlikely to reach it, but they should be aware of the possibility.

A situation that could produce this condition would be the act of changing permissions or ownership of all objects in a directory, when the objects number in the thousands. When the directory has reached its limit, NFS on PRIMOS Systems returns to the client an error message similar to `file too big`. However, the error message eventually displayed to the client user may not refer to the directory at all, but to the particular file whose ACL creation caused the directory to overflow. In this case, the error message needs substantial interpretation to identify the real problem.

Modifying Existing ACLs

When NFS on PRIMOS Systems creates a new file or directory, it sets a minimum ACL on it similar to the ones shown in Figure 3-2. This minimum ACL is limited to entries for user owner, group owner, and **$REST** – as well as for **NFS_SERVER**.

When files and directories have previously existed with their own ACLs, NFS on PRIMOS Systems does not actually replace the ACLs, because this operation would revoke access for PRIMOS users unknown to NFS on PRIMOS Systems. In environments where the same files and directories are accessed both by native PRIMOS users and by NFS users, replacing existing ACLs could be disastrous to PRIMOS users.

Therefore, the strategy for NFS on PRIMOS Systems is to minimally disrupt the old ACLs it modifies. NFS on PRIMOS Systems merges new access control information with the old, with minimal modification of access rights for those existing users and groups not affected by NFS on PRIMOS.

As Administrator of NFS on PRIMOS Systems, you want to be sure that this strategy protects both the PRIMOS user and the NFS user. The next paragraph shows how PRIMOS users remain secure. The remaining paragraphs in this section show how NFS users maintain ownership and permissions that are secure, accurate, and reliable.

Old PRIMOS Rights Untouched by NFS Clients: As is consistent with UNIX file system procedures, the client NFS user must own a directory or file before being able to change its permissions. If a client has mounted a PRIMOS remote partition to which **NFS_SERVER** allows access, a remote user can issue a **cd** command to relocate to one of the native-made PRIMOS subdirectories. Once there, however, the remote user without ownership rights cannot successfully issue the **chown** or **chgrp** command on it.

For example, the PRIMOS directory **SYSCOM** already exists with the following ACL:

```
ACL protecting "SYSCOM":
        MANAGER:          PDALURWX
        NFS_SERVER:       PDALURWX
        SYSTEM:           ALL
        .ADMIN$:          ALL
        $REST:            LUR
```

If **SYSTEM** and **.ADMIN$** are *not* listed in the **PASSWD** and **GROUP** files for NFS on PRIMOS Systems, then a client's **ls -lg** command shows

```
d------r-x  141 nobody   nogroup     554 Mar 15 13:18 syscom
```

The client cannot successfully declare ownership of the directory:

```
% chown alpha.staff syscom
(chown:) no ownership
```

If ownership were truly necessary for the client NFS user, then a PRIMOS user could grant the necessary **O** right to a client NFS user and group.

For example, the NFS Administrator gives **O** rights to the NFS user **alpha** and NFS group **staff**, and checks the result as follows:

```
OK, EDIT_ACCESS SYSCOM ALPHA:O .STAFF:O
OK, LIST_ACCESS SYSCOM
ACL protecting "SYSCOM":
        ALPHA:          O
        MANAGER:        PDALURWX
        NFS_SERVER:     PDALURWX
        SYSTEM:         ALL
        .ADMIN$:        ALL
        .STAFF:         O
        $REST:          LUR
```

Now a client sees the following access mode for the directory **syscom**:

```
% ls -lg syscom
d------r-x  141 alpha    staff       554 Mar 15 13:18 syscom
```

Note that client user owner **alpha** and client group owner **staff** have no permissions to the directory, but **alpha** as the user owner is free to confer permissions by using the **chmod** command. Also note that the PRIMOS access rights that do not apply to the NFS client are still in effect, but invisible to the NFS client.

NFS Ownership Changes and Effects on PRIMOS ACLs: When a client has ownership of a directory or file on a PRIMOS partition, the **chown** and **chgrp** commands allow for changes in NFS user and group ownership. As above, NFS access modes are merged with any additional PRIMOS ACLs, which remain active for PRIMOS users but invisible to NFS users.

For example, a client NFS user sees the access mode of a particular file as

```
%  ls -lg file.txt
-rwxr-xr--  1 alpha    user         1060 Mar 15 13:23 file.txt
```

A native PRIMOS user sees the file's ACL as

```
OK, LIST_ACCESS FILE.TXT
ACL protecting "FILE.TXT":
        ALPHA:          ORWX
        NFS_SERVER:     PDALURWX
        .USER:          ORX
        $REST:          R
```

If the client NFS user **alpha** transfers ownership to user **beta** with the command

```
% chown beta file.txt
```

then the access mode would be

```
-rwxr-xr--  1 beta     user          1060 Mar 15 13:23 file.txt
```

and the Access Control List would be:

```
ACL protecting "FILE.TXT":
        ALPHA:              RWX
        BETA:               ORWX
        NFS_SERVER:         PDALURWX
        .USER:              ORX
        $REST:              R
```

Note that PRIMOS access rights for **ALPHA** still remain (except for **O**, which has been transferred to user **BETA**). The file still remains fully accessible to **ALPHA** as a native PRIMOS user. However, NFS on PRIMOS Systems ignores the **RWX** rights for **ALPHA**, since the remote user **alpha** is no longer the owner.

To summarize, when a client user transfers user or group ownership for existing files and directories, the following happens to the PRIMOS ACLs:

- The **O** access right is removed from the old user name and group name. If the old name only has the **O** right, then that name is removed from the ACL.
- The **O** access right is added to the new user name and group name. If these names aren't already listed in the ACL, then new entries are added for them.
- An entry for **NFS_SERVER** is placed in the ACL.

Unwanted ACL Entries: A user's ability to merge NFS permissions with PRIMOS access rights can lead to an accumulation of unwanted ACL access right entries, leaving behind undesired access rights for native PRIMOS users. The extra rights are normally harmless unless there is an excess of entries (see the next section, PRIMOS ACL Entry Limit).

For example, in the previous section, our remote user **alpha** left behind access rights for a PRIMOS user **ALPHA** that doesn't exist. If **alpha** had wanted to remove the ACL entry for **ALPHA** entirely, he should have issued the command sequence:

```
% chmod 054 file.txt          # First removing alpha's permissions
% ls -lg file.txt             #    and then checking the results...
----r-xr--  1 alpha    user          1060 Mar 15 13:23 file.txt
% chown beta file.txt         # Now transferring ownership.
```

In this case, the ACL before and after the **chown** command looks like

```
ACL protecting "FILE.TXT":                /* before...
        ALPHA:              O
        NFS_SERVER:         PDALURWX
        .USER:              ORX
        $REST:              R

ACL protecting "FILE.TXT":                /* after...
        BETA:               O
        NFS_SERVER:         PDALURWX
        .USER:              ORX
        $REST:              R
```

The **chown** command on this occasion causes two important additional changes:

1. NFS on PRIMOS Systems removes the **O** access right for **ALPHA**. As a result, no access rights remain for **ALPHA**, so this entry is removed entirely from the ACL.
2. As in the previous section's example, all access rights that **ALPHA** had were transferred to **BETA**, along with ownership. But this time, since **ALPHA** had no access rights, **BETA** also has none. Nevertheless, as the new owner, user **beta** can use the **chmod** command to acquire whatever rights are needed.

PRIMOS ACL Entry Limit

PRIMOS allows a maximum of 32 combined user names and group names to be listed in an ACL. If you attempt to add an entry that exceeds the maximum, then NFS on PRIMOS Systems adds it, but it drops an entry from the bottom of the list (above the **$REST** entry, which is never dropped). Because group names are listed after user names in an ACL, and because they are arranged alphabetically, the dropped entry is likely to be the alphabetically last group name. NFS on PRIMOS Systems issues no warning when this occurs.

SOME ACCESS ERROR MESSAGES AND PROBABLE CAUSES

Permission Denied

There are several causes for this message:

- **NFS_SERVER** has insufficient access rights.
- The **uid** (user ID number) cannot map to an appropriate PRIMOS user name within the **PASSWD** file, or the **gid** (group ID number) cannot map to a group name within the **GROUP** file. If the names map appropriately, they have insufficient access rights.
- The client **uid** or **gid** may not be unique over all hosts on your LAN. All **uids** and **gids** *must* identify the same users and groups on all hosts for proper operation of the Network File System.

No Group Owner; No User Owner

If there is no NFS-recognizable group name listed in a given directory's ACL, then files created in that directory will show no group owner and no access permissions for group owner. To solve this problem, either issue an appropriate **chown** or **chgrp** command for that directory, or, while logged into PRIMOS, enter an NFS-recognizable group name in the directory's ACL. Be sure that the PRIMOS ACL group name maps to an appropriate **gid** in the **GROUP** file.

The same condition can occur for the user owner.

chmod Confers No Group Permissions

The cause of this message is most likely the same as detailed above for No Group Owner. Issue an appropriate **chgrp** command to assign an NFS-recognizable group owner.

4

ENABLING CLIENT ACCESS TO NFS ON PRIMOS SYSTEMS

This chapter is directed to all users on an NFS client. Survey the questions and answers on the next page. If you need to mount and dismount PRIMOS directories, read the remainder of the chapter.

MOUNTING AND DISMOUNTING PRIMOS DIRECTORIES FOR NFS

The two NFS commands for manipulating remote file system access to PRIMOS directories are **mount** and **umount**. A third NFS command to display NFS network status is **showmount**.

A user on the client host issues these commands. Since the majority of NFS client hosts are UNIX systems, this document gives a SVID-compliant absolute pathname for NFS commands. If your client host is a personal computer, your NFS implementation may use a different command, but the command will have comparable options.

Client NFS Users and Mounting Directories

As an NFS client user, do you need to know the details in this chapter? The answers to the following questions may help you decide.

Who can mount and dismount directories?
If your NFS client system is a workstation or a PC, you can mount and dismount directories. However, most multiuser systems restrict this task to Administrators.

If someone else mounts a remote PRIMOS directory on my system, can I see its contents?
Yes, under two conditions:

- Your client NFS system has access to it (by being included in the PRIMOS file **NFS_>ETC>EXPORTS**)
- PRIMOS ACL rights allow you access (an Administrator on PRIMOS has configured NFS, has given you adequate access, and has provided an ACL on the partition's MFD to grant **NFS_SERVER:PDALURWX** and **NFS_MOUNT:LUR**)

If I can't reach a mounted PRIMOS directory to which I have rights, is NFS not working?
Not necessarily. It is possible that a PRIMOS user accidentally excluded NFS access to that directory when setting a specific ACL. That system's Administrator can verify and correct the problem.

After my client NFS host mounts the directory, how long is it there?
On a multiuser system, it stays there (even if you log out) until one of the following happens:

- An Administrator uses **umount** to dismount it
- Your system crashes or is shut down

Is there a way to mount directories automatically at NFS startup time?
On UNIX client NFS hosts, the system automatically mounts those directories included in the file **fstab** at NFS startup. Read your client NFS documentation for directions on formatting this file.

Does NFS on PRIMOS Systems support NFS for PCs?
Yes. You can use a PC client with PC-NFS software to read and write files, as well as to print PC files on a printer attached to your PRIMOS system. For further details, see page 5-13: Configuring for PC-NFS Access.

Does it support pathnames with 8-bit characters from PC-NFS clients?
Yes. Such 8-bit characters map to special character sequences. For further details, refer to page D-5.

Mounting a Remote PRIMOS Directory

Issue the **mount** command on the client host to mount a remote PRIMOS pathname located on the NFS server. The command is successful only for those directories that are "exported" to client hosts, as authorized in the PRIMOS file **NFS_>ETC>EXPORTS.**

See Chapter 5 for details on the **EXPORTS** file.

Format

mount [-o *options*] [*pathnames*]

Options

Three arguments to the **-o** option are used when mounting PRIMOS directories. If you specify the **-o** option without arguments, then the mount defaults to a hard mount. For a complete description of the **mount** command and all its options for your NFS client host, consult Appendix A or the command's man page.

soft
Specifies that if the NFS server fails, the client request terminates with an appropriate error message. Normally this option is followed by a comma with the **timeo** option immediately after it.

timeo=*n*
Specifies, in tenths of a second, the length of time an NFS client will wait for NFS on PRIMOS Systems to respond.

hard
Default. Specifies that the client will retransmit its request until the NFS server responds.

Pathnames

If you use the **mount** command without pathnames, you obtain a listing of all partitions and file systems mounted by the client, whether local or remote.

If you specify pathnames to the **mount** command, you must supply two of them:

host:pathname
The name of the server host, a colon (:), and the pathname onto an exported PRIMOS directory. The pathname is given in UNIX pathname terminology instead of PRIMOS terminology. For example, the client mounts **prime1:/users/nfsusers** (and not **PRIME1:<USERS>NFSUSERS**).

mtpt_name
The pathname of the local mountpoint; for example, **/prime1_users.**

Note

Various systems with NFS have different requirements on whether the mountpoint directory should pre-exist before you issue the **mount** command. See the documentation for your client NFS implementation to identify your system requirements.

Discussion

When you issue the **mount** command with pathnames *host:pathname* and *mtpt_name*, the command provides access to a specified host's remote pathname (*host:pathname*) by creating a local mountpoint (*mtpt_name*) for the remote pathname.

For example, the command

```
% mount prime1:/users/nfsusers/onsuns /prime1_users
```

allows an NFS user on a Sun Workstation to mount the PRIMOS remote pathname **<USERS>NFSUSERS>ONSUNS** located on the NFS server host PRIME1. The command creates the local mountpoint **/prime1_users.**

Note that a client does not need to mount an entire directory tree, from topmost level down. First of all, a full directory tree need not be exported by the PRIMOS NFS server; remote access may be specified as beginning at any level on the directory tree (see a discussion of the **EXPORTS** file in Chapter 5). Furthermore, the mount can specify a remote pathname that is lower still on that exported directory tree.

Note

Previous to NFS Release 1.2, the EXPORTS file required entries to be full disk partitions. This is no longer a requirement at PRIMOS Rev. 23.0, as explained above.

Client users of NFS on PRIMOS Systems could always specify a mountpoint lower than the level of access provided in the EXPORTS file. Clients were not required to "mount a partition", but that was the common practice, due to the old restriction of the EXPORTS file. "Partition" and "MFD" are now obsolete terms for NFS on PRIMOS Systems, with the availability of a singly-rooted file system for PRIMOS Rev. 23.0 and the new EXPORTS file functionality. The "MFD" terminology is no longer required, but still supported, within the EXPORTS file. See Chapter 5 for details.

Listing Mountpoints: If you give no pathname arguments to the **mount** command, it displays every pathname and file system, both local and remote, that the local host currently registers as mounted. The command returns this information by listing the current contents of the **mtab** file. For example:

```
$ mount
prime3:/notes/misc on /prime3_misc type nfs (rw)
sunny7:/usr2/project_b/rev_2q92/cleo on /sunny7_cleo type nfs (rw)
apo19:/usr/testing/gerry on /apo19_gerry type nfs (rw)
$
```

For a complete description of the **mount** command, consult the manual page for **mount/umount** in Appendix A.

Dismounting a Remote PRIMOS Directory

The **umount** command dismounts a remote PRIMOS pathname mounted on a client NFS host.

Format

umount [*-options*] [{ *host:pathname* | *mtpt_name* }]

Options

Two options are particularly relevant for dismounting PRIMOS pathnames. See Appendix A for a discussion of all **umount** options.

-a
Dismounts all remote pathnames and files systems listed in **mtab**.

-v
Displays a message when dismounting a remote pathname or file system.

Pathnames

If you do not use the **-a** option, you must specify one of these pathnames:

host:pathname
The name of the server host, a colon (:), and the remote pathname mounted from an exported PRIMOS directory; for example, **prime1:/users/nfsusers**.

mtpt_name
The pathname of the mountpoint; for example, **/prime1_users**.

Discussion

When you execute the **umount** command with *host:pathname*, the client host dismounts the remote pathname specified. You achieve the same result if you execute the **umount** command with the local *mtpt_name* for a pathname.

In most client NFS implementations, the command does the dismount by searching the contents of the file **mtab** and deleting the first entry it encounters that contains either the local *mtpt_name* pathname or *host:pathname*. For example, the following command dismounts the PRIMOS directory **<USERS>NFSUSERS** on the NFS server PRIME1.

```
% umount prime1:/users/nfsusers
```

Depending upon your client NFS implementation, the **umount** command may also delete the directory that is the local mountpoint **/prime1_users** associated with the remote pathname.

DISPLAYING STATUS INFORMATION

You can display all the remote objects mounted on your local client NFS host by issuing the **mount** command without arguments.

You can also display NFS statistics with the **nfsstat** and **rpcinfo** commands. See Chapters 5 and 6 for treatments of these administrative and diagnostic commands. For complete command descriptions, consult Appendix A.

The **showmount** command is another command of special interest to the user of NFS on PRIMOS. It enables the client user to single out a PRIMOS system that is an NFS server.

The command format relevant to the user of NFS on PRIMOS is given below.

Format

showmount [*arguments*]

Arguments

The most useful arguments for users of NFS on PRIMOS Systems are the option **-e** and the **serverhostname.**

-e
Lists the exported partitions (in **NFS_>ETC>EXPORTS**).

serverhostname
The name of the PRIMOS system that is the NFS server.

Discussion

An NFS client can mount file systems and directories from multiple NFS servers. The **showmount** command with the arguments described above allows a client NFS user to list the contents of the file **NFS_>ETC>EXPORTS** on a particular PRIMOS system that is an NFS server.

The **showmount** command lists the directories of a server host that are mounted by client hosts, displays all remote mounts in the form ***host:fsname***, and lists all exported file systems (that is, those directories and file systems listed in the **exports** file of the server host).

For example, a user on a Sun client identifies all directories exported by the NFS server PRIME1 as follows:

```
% showmount -e prime1
```

The command displays the contents of the **EXPORTS** file for PRIME1, listing it in UNIX format:

```
export list for prime1:
/cl1-3/printers  ELITE KUDZU WHELP
/cl4
/cl5    KURZ
```

The **EXPORTS** file, as a local PRIMOS user views it on the system PRIME1, reads

```
<CL1-3>PRINTERS   ELITE kudzu whelp  # Directory for these 3 clients
<cl4                                 # Top-level directory .... which
                                     # is available to ALL NFS clients
<cL5>MFD          kurz               # A disk partition that is
                                     # available to this client only
```

Note

The MFD terminology used in the final EXPORTS entry above is obsolete at PRIMOS Rev. 23.0. It remains supported by NFS 1.2 on PRIMOS Rev. 23.0 for the sake of backward compatibility.

5

NFS INSTALLATION AND ADMINISTRATION ON 50 SERIES SYSTEMS

This chapter discusses the installation and the administration of NFS on PRIMOS Systems.

Details on installation include

- Prerequisites for installing NFS on PRIMOS Systems
- Installation of software

Details on administration include

- Configuring NFS on PRIMOS Systems
- Starting, updating, and stopping NFS on PRIMOS Systems
- Checking NFS server stability

Diagnostics for NFS are provided in Chapter 6.

Note

If NFS on PRIMOS Systems is already activated and you wish to add a new client system or a new user, follow the procedures described in the section Configuring NFS on PRIMOS Systems.

PREREQUISITES FOR INSTALLING NFS ON PRIMOS SYSTEMS

You must install NFS on PRIMOS Systems software on a machine that satisfies hardware and software requirements described in Chapter 1. If your system satisfies these prerequisites, install and configure NFS on PRIMOS Systems as follows.

INSTALLATION OF SOFTWARE

Installation involves two phases:

- Installation from tape to an intermediate directory NFS that is created directly below your current location
- Installation from the intermediate NFS directory to various system directories

Installing to an Intermediate Directory

To install the software from tape, perform the following steps:

1. At the supervisor terminal, choose a location under which to install the intermediate directory. For this example, the directory is **<MASTER**.
2. Choose an available magnetic tape drive unit, such as Magnetic Tape Unit zero (MT0), and assign it to your use. Issue the command

```
OK, ASSIGN MT0
OK,
```

3. Mount the software installation tape on MT0.
4. Begin the software installation, using **MAGRST**:

```
OK, MAGRST
[MAGRST Rev. 22.1.1 Copyright (c) 1989, Prime Computer, Inc.]
Tape Unit (9 Trk): 0
Enter logical tape number: 1
Name: NFS
Date (MM DD YY): 01-04-91
Rev: 23
Reel: 1
Ready to Restore: YES
***Starting Restore***
***End Logical Tape***
***Restore Complete***
OK,
```

5. Rewind and remove the installation tape.
6. Unassign MT0.

The first phase of the installation is now complete. You have installed the intermediate directory called NFS.

Installing to System Directories

To install NFS software to its system directories, do the following:

1. Attach to the directory **NFS**:

```
OK, ATTACH NFS
OK,
```

2. List the contents of the directory:

```
OK, LD

<MASTER>NFS (ALL access)
3 records in this directory, 1114 total records out of quota of 0.

1 File.

NFS.INSTALL.CPL

9 Directories.

CMDNCO          HELP*           INFO            LIB
NFS_            RPC_SAMPLES     RPC_UTILS       SYSCOM
TOOLS

OK,
```

3. Execute the CPL file that installs NFS:

```
OK, R NFS.INSTALL.CPL
NFS Installation started at 91-01-04.11.10:12.Fri
Building the NFS_ directory and setting ACLs.
    .
    .
    .
NFS installation successfully completed at 91-01-04.11:11:02.Fri
OK,
```

4. Return to the original location before the tape installation and erase the intermediate directory **NFS**:

```
OK, ATTACH <MASTER; DELETE NFS
OK, OK to delete Directory "NFS"? YES
OK,
```

CONFIGURING NFS ON PRIMOS SYSTEMS

Before running NFS on PRIMOS Systems, the NFS Administrator must perform the following configurations:

1. Configure an **EXPORTS** file in **NFS_>ETC** that registers client NFS systems to be given access to specific PRIMOS partitions. (See below.)
2. Set ACLs on those partitions so that NFS on PRIMOS Systems allows access to the approved client NFS systems. (See page 5-6.)
3. Configure the **PASSWD** and **GROUP** files (installed by default in **NFS_>ETC**) to register authorized users on the approved clients. (See page 5-8.)
4. Provide access configuration for client users of PC-NFS and for their remote printing needs. (See page 5-12.)

Note

If your system has no PC-NFS client users, be sure to include the **-NOPC** option when you issue the **START_NFS** command. (See page 5-15.)

The best approach for the Administrator of NFS on PRIMOS Systems is to specify in the EXPORTS file an entire PRIMOS top-level directory (accessible just below <) that is dedicated to NFS clients. The Administrator then excludes resident PRIMOS users from the directory. Setting ACLs for this is simple: ACLs are set only at the top-level directory; they exclude PRIMOS users; they only need to admit the NFS servers.

The Administrator does not need to create an Initial Attach Point (a subdirectory with its own ACLs) for each remote user. After a user is registered in the **PASSWD** and **GROUP** files, then NFS on PRIMOS Systems automatically sets ACLs, as a user creates subdirectories and files using the typical commands (**mkdir**, **cp**, **mv**, and the like).

Configuring the EXPORTS File

As the Administrator of an NFS server host, you need to establish a file that lists the directories you allow to be *exported* from your host. The file also lists NFS client hosts that are allowed to *mount* these directories.

The file **NFS_>ETC>EXPORTS** lists PRIMOS directories and the names of the hosts that are allowed to access them through NFS. The format for the file is

```
#   for comments
/*  also for comments
<top_dir        # Top-level directories now are listed this way.
<mfd_name>MFD   # Neither > nor the MFD suffix is needed any longer, but
<mfd_name>MFD   host-a host-b   # they may still be used.
#    The most secure and explicit format for exporting a directory is
<top_dir>subdir-1>subdir-2>...subdir-n      host-c host-d
```

An entry without a list of hosts allows any host to access that directory. If a host list is present, only those listed can access the directory.

WARNING

Avoid making a directory or partition entry without a list of hosts. Such a directory grants, at minimum, the access rights of $REST to every host that can access your system with TCP/IP.

Note

Revision 23.0 of PRIMOS provides a singly-rooted file system. The top level of the file system is root (<). Root cannot contain files; it contains only directories. Beneath <, all those entities previously known as disk partitions (or MFDs) are now treated as full-fledged directories.

PRIMOS NFS allows the export of root-based directories (such as *<top_dir*), but not of the root itself (such as <). PRIMOS NFS continues to support the export of partitions (the obsolete "MFD" terminology) for the sake of backward compatibility.

The number of hosts allowed to mount a directory is unlimited, but each line in the **EXPORTS** file cannot exceed 256 characters. You may repeat the directory on another line and name more client hosts. The host name may match either its primary or secondary name as given in the **TCP/IP*>HOSTS** file on PRIMOS.

Note

If the PRIMOS system is functioning as a TCP/IP Domain Name Server, then the **EXPORTS** file must specify the fully-qualified name for the system.

The file **NFS_>ETC>EXPORTS.TEMPLATE** can be renamed as **EXPORTS** and then customized for your system. The template file gives a thorough explanation with examples of how to use the file. The following sample summarizes most issues:

```
<sys1>mfd                    # Everyone can mount <sys1>mfd (old terminology).
<sys1                        # This is the new terminology for above.
<sys1>chateauneuf  GUY       # The system GUY can access this directory, but
                     #  previous entries already opened it to ALL systems.
<sys2                        # Likewise, all can mount the <sys2 directory.
<part4>mfd         ARAMIS    # Just aramis can access <part4>mfd (old term).
<PART4             porthos   # Now porthos can access it too (new term).
error # in typing
<srvr2>mouton>cadet  D'artagnan ATHOS       # Both clients can access this directory.
<SRVR3>PINOT>DU>GRISE  athos porthos aramis # All three client systems can
                                            # access this directory.
```

The sample illustrates that

- Use of final angle bracket and MFD is supported, but no longer required.
- To make all server directories exportable, you must list each one on a separate line. (You cannot export <.)
- Any top-level directory not listed is invisible to a client NFS system.
- You can spread out the list of hosts across several lines, by repeating the directory and listing more hosts.

- If you list a directory but do not specify any hosts after it, the directory is available to all hosts. This is true even if that same directory (or a pathname to a lower directory) is listed on another line of the file *with* a select list of hosts.
- Lines with errors are ignored.
- Directory names and host names are case insensitive.
- The directories don't have to be local to the system running NFS on PRIMOS Systems. They can be on any (Rev. 22.0 or later) PRIMOS system accessible through remote file access via PRIMENET, as long as forced user validation is not required to access the remote partition.

You can change the **EXPORTS** file at any time while NFS on PRIMOS Systems is running, and the change takes effect immediately; any future mount requests from client hosts will be checked against the new file. However, deleting a partition or host from this file doesn't revoke access from a client host that has already mounted a partition. If a client host does not voluntarily unmount a partition, then the Administrator for NFS on PRIMOS Systems can change ACLs to deny access.

Setting ACLs for NFS on PRIMOS Systems

No client NFS user can access any level of a PRIMOS directory tree until ACL rights are established on a directory above the attempted attach point.

In order for an NFS user to reach any part of the directory tree, **NFS_MOUNT** needs **LUR** access to the top-level directory (just below <).

If the client host mounts the directory for reading and writing, **NFS_SERVER** must have **PDALURWX** access to the top-level directory and all its contents. If the client host mounts the directory for reading only, **NFS_SERVER** needs only LUR access.

Table 5-1 shows the minimum access rights on a PRIMOS directory. If the directory is dedicated to client NFS users, no further ACL settings are necessary. As the table shows, no remote system or user needs to be granted access rights. The rights are granted to the NFS servers. This is enough to provide authorized remote systems and their registered users the rights to access the directory and to create subdirectories (see Chapter 3 for details).

TABLE 5-1. Minimum Access Rights for NFS on PRIMOS Systems Set on Directories Below Root

Server:Access Rights	*Set at Top Level*	*Set at Subdirectory Level*	*To Enable*
NFS_MOUNT:LUR	Yes	No	NFS itself to access the directory tree
NFS_SERVER:LUR	Yes	Yes	an NFS user to **READ**
or			
NFS_SERVER:PDALURWX	Yes	Yes	an NFS user to **WRITE**

Note

NFS_PCNFSD automatically receives adequate access rights at startup. The PC-NFS client user also automatically receives user access if the user is registered in the **SAD.** (See page 5-12.)

ACL Settings for Non-dedicated Partitions: It is possible to have a directory tree shared by both resident PRIMOS users and client NFS users. If all users access the directory tree at separate subdirectories on the same level, then no further ACL settings are needed. As shown in Figure 5-1, this configuration avoids the risk of access conflicts between resident PRIMOS users and remote NFS users.

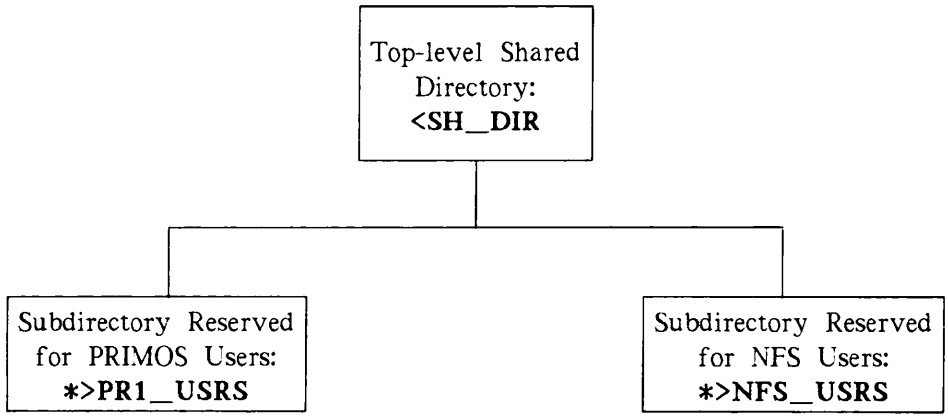

FIGURE 5-1. Shared Directory Tree for Resident and Remote Users

If, however, a PRIMOS user has a directory under which NFS users are expected to have access, then conflicts can arise. If a PRIMOS user declares a specific ACL for a directory and does not include access rights for **NFS_SERVER**, then NFS cannot provide access below this directory. If the PRIMOS user wants to make subdirectories accessible to NFS users,

then the specific ACL must include **NFS_SERVER:LUR** for read-only procedures and **NFS_SERVER:PDALURWX** for operations that allow writing.

Preventive Planning Against Access Conflicts: At times NFS users may need to access directories below directories controlled by PRIMOS users. Even if a PRIMOS user's specific ACL omits rights for **NFS_SERVER**, there is still a way for the NFS Administrator to allow NFS access lower on the directory tree, without resorting to priority ACLs.

This procedure rests on the widespread use of certain ACL groups that are given **ALL** rights, such as **.BATCH_USER$** and **.BACKUP$**. The procedure works on two conditions:

1. The PRIMOS user includes **$REST:LUR** within the specific ACL.
2. The topmost NFS subdirectory beneath that specific ACL already has an ACL that includes rights for one of the common ACL groups (**.PROJECT_ADMINISTRATORS$**, **.OPS**, **.ADMINISTRATORS**, and the like.).

You decrease the likelihood of interrupted NFS access by making **NFS_SERVER** a member of up to four of the most common ACL groups. You do this by using the **-ACLGROUP** option when you start NFS:

```
OK, START_NFS -ACLGROUP .BACKUP$ .BATCH_USER$ [-other_options]
```

If you are patient enough, you can achieve the same results without using the option by manually setting access for **NFS_SERVER** on all the topmost NFS subdirectories that are susceptible to specific ACLs from PRIMOS users.

Correcting Access Problems That Occur: If access problems still arise, the NFS Administrator can supply a blanket solution by setting a priority ACL which defines NFS access for the entire partition and overrides user-defined ACLs. The priority ACL should grant rights to the NFS servers, as follows.

```
OK, SPAC SH_PTN NFS_SERVER:PDALURWX NFS_MOUNT:LUR
```

To avoid the need for priority ACLs on numerous partitions, the NFS Administrator should encourage users to include **NFS_SERVER:PDALURWX** in their specific ACLs, unless they need to deliberately exclude client NFS users.

Configuring PASSWD and GROUP Files

Each user request from an NFS client automatically includes the user ID number (**uid**) and group ID number (**gid**). NFS on PRIMOS Systems then checks these IDs against lists of PRIMOS user names and PRIMOS group names, *provided by an Administrator* in the **PASSWD** and **GROUP** files on the PRIMOS system. NFS on PRIMOS Systems matches these PRIMOS user and group names against user and group access rights that are set on a file or directory targeted for an NFS task. If there is no match, the client user receives the PRIMOS access rights of **$REST**, which usually is very limited.

After you install NFS on PRIMOS Systems, you set up user and group IDs for client NFS

users by providing the necessary information in the two text files, **PASSWD** and **GROUP.** These may be located anywhere, but at startup the system looks for them in **NFS_>ETC** as their default location. Each file has its equivalent on a BSD-based or SVID-compliant system. The format for these PRIMOS files allows for an easy adaptation of those files from NFS clients. The installation of NFS on PRIMOS Systems provides a template version for each file in **NFS_>ETC.** If you use these templates, you must rename and customize the files.

The **PASSWD** file is the registration list that maps a client NFS user's **uid** to a PRIMOS user name. The **GROUP** file is the registration list that maps a client NFS user's **gid** to a PRIMOS ACL group. Both files are read each time you start up NFS on PRIMOS Systems.

You can change the **PASSWD** and **GROUP** files at any time.

Notes

- If you make additional entries in these files after NFS is started, the new client NFS users do not receive access until the next startup, unless you use the **NFS_INIT_USERS** command to update the NFS registration (see page 5-20).
- If you delete a user or group from these files, they maintain their access until the next startup or update of NFS on PRIMOS Systems.

If the **PASSWD** and **GROUP** files contain only comments, or are missing, or empty, then NFS on PRIMOS Systems grants only rights for **$REST** to client NFS requests.

Contents of PASSWD: The Administrator for NFS on PRIMOS Systems must list the name of *every client NFS user* in the **PASSWD** file, unless the user requires no further access rights than those for **$REST.** When NFS on PRIMOS Systems is started, the system reads the **PASSWD** file and is prepared to grant a PRIMOS user name identical to each *NFS_CLIENT-username* in the file.

Except for remote users from a PC-NFS client, no further registration of NFS client users is necessary.

Note

The System Administrator must register every PC-NFS client user in the SAD. See page 5-12.

For clients not using PC-NFS, two users with the same name (a resident PRIMOS user and an NFS user) can function simultaneously. Both resident user and remote user share the same access rights. The **uid** of the NFS user is mapped to an *NFS-CLIENT_username*, which functions as its PRIMOS user name. The user is thereby under the control of PRIMOS ACLs. If the NFS user should not have access, then access rights for **NFS_SERVER** can be removed.

The format of a line entry in **PASSWD** is

```
# comment
/* comment
NFS-CLIENT_username : (ignored) : uid : (ignored)
```

You can create a **PASSWD** file which consists solely of line entries that pair *NFS-CLIENT_username* with its **uid**, one pair per line, in the format of

```
js::37:
```

The intervening colons are necessary. If nothing (or an end-of-line comment) immediately follows the **uid**, then no final colon is needed. For example, valid entries in the PASSWD file are

```
tcg::10
frost::40 # from Sun system Ior
js::37:25
```

The value for *NFS-CLIENT_username* does not have to be identical to the home name associated with the client **uid.** You can fabricate a name if you choose, but the **PASSWD** file uses this format to allow you to copy lines of information verbatim from the **/etc/passwd** file on a BSD-based or SVID-compliant client host. No new information needs to be added to, or removed from, a line.

For example, in the **/etc/passwd** file on a Sun Workstation, one user is identified as

```
js::37:25:John Smith:/usr/desoto/js:/bin/csh
```

The Administrator on the Sun Workstation uses **ftp** to send you a copy of the **/etc/passwd** file. You then use an editor to copy the above line into your local **PASSWD** file. When NFS on PRIMOS Systems is next started, the remote user **js** with the **uid** of **37** is assigned the PRIMOS user name of **JS.**

You may need to adjust the name, especially if a client user was registered earlier on the PRIMOS system with a different user name. For example, the two significant fields above are **js** and **37.** Perhaps John Smith is the user on the remote system that uses this name and **uid.** Previously John has been registered as **SMITH** on PRIMOS. He has PRIMOS directories that he wishes to reach from his client system. You change **js** within the **PASSWD** entry to **SMITH.** When John is on the remote system, he remains **uid** 37 with the local name of **js,** but he can also use NFS, which assigns him a PRIMOS user name of **SMITH** to access his directories on PRIMOS.

Notes

- The Administrator for NFS on PRIMOS Systems should be aware that a single **uid** may represent different users on different client hosts. Remote users with identical **uids** receive the same access rights on the PRIMOS system.

  BSD-based and SVID-compliant systems that support the Yellow Pages utility for NFS have already made corrections for this possibility. Every NFS user within Yellow Pages has a unique **uid** that is registered in a single shared **/etc/passwd** file. If all NFS clients support Yellow Pages, the Administrator for NFS on PRIMOS Systems can copy this Yellow Pages version of **passwd** into the local **PASSWD** file so that the PRIMOS version is consistent with the rest of the network.

- Only the **uid** included in the **PASSWD** file has its matching *NFS-CLIENT_username* mapped to a PRIMOS user name. If a **uid** not included in your **PASSWD** file arrives in an NFS request, and its **gid** does not map to a PRIMOS ACL group, that user has the limited access rights given to **$REST** for the object being accessed.

- Do *not* assign the user name of **NFS_SERVER** within the **PASSWD** file. This grants the associated client user as much access rights as NFS itself.

Contents of GROUP: You configure **GROUP** in a similar way as for the **PASSWD** file.

The **GROUP** file has the format

```
# comment
/* comment
NFS_groupname : (ignored) : gid : (ignored)
```

The system maps the **gid** to a PRIMOS group name that is identical to *NFS_groupname* but with a preceding period (.). You need not use the BSD-based or SVID-compliant group name associated with the **gid** of a client NFS user, although the format of the file allows for this.

For example, the Administrator for NFS on PRIMOS Systems makes the following line entry in **GROUP**:

```
ntgrp1 : : 25
```

The *NFS_groupname* of **ntgrp1** maps to the PRIMOS ACL group **.NTGRP1**. All client NFS users with a **gid** of 25 inherit the rights granted to the PRIMOS ACL group **.NTGRP1**. It does not matter what group name that **gid** has on its home system. On the PRIMOS system, every **gid** of 25 maps to the PRIMOS ACL group **.NTGRP1**.

You can create the **GROUP** file from scratch, or you can copy line entries from a remote **/etc/group** file. Which procedure you choose depends on whether your BSD-based or SVID-compliant client systems support Yellow Pages. The Yellow Pages utility shares a common **/etc/group** file.

Without Yellow Pages

If the clients do not use Yellow Pages, then create the PRIMOS file from scratch.

1. Use the number that follows the **uid** in the client system's local **/etc/passwd** file.
2. Create a PRIMOS group name to associate with that number. You can choose any name.
3. Add both name and number to **GROUP**, using the format given above. The first line entry assigns a name to this **gid** number. Any later line entry cannot assign a different name to it, but it causes an error message to display at startup of NFS on PRIMOS Systems.

Note

Under these conditions, every NFS client with this local **gid** now has the PRIMOS access rights granted to the associated group name.

For example, as previously shown in **/etc/passwd**, John Smith's **uid** is 37. The next field holds 25; this is John Smith's local **gid.** You are free to create any name that you wish to be the *NFS_groupname* associated with this **gid.** You add the information to **NFS_>ETC>GROUP**, using the format:

```
NTUSRS::25:
```

With Yellow Pages

If the clients support the Yellow Pages utility, then create the **GROUP** file as follows:

1. Copy the entire **/etc/group** file from Yellow Pages into **GROUP.**
2. If there are certain groups that you wish to exclude on your system, either remove their line entries or comment them out (prepending either # or /*).

Configuring for PC-NFS Access

The following configuration steps are needed to provide PC-NFS client user access and remote spooling:

- Register each PC-NFS user in the **SAD.**
- Designate a remote printer, if NFS is installed on a PRIMOS system that uses a despooler on another PRIMOS system.

Registering in the SAD: In addition to naming PC-NFS client users in the **PASSWD** file, the System Administrator must also register them in the **SAD** as individual PRIMOS users. (See the *System Administrator's Guide, Volume III: System Access and Security* for directions on registering users.)

Note

In order to provide full authentication for a PC-NFS user, your system must be running PRIMOS Revision 22.1.

The System Administrator needs to consider what happens to a PC-NFS user whose limited password lifetime expires on the PRIMOS system. When a password expires, PRIMOS issues

a warning to change it, but the NFS protocol cannot deliver the message. As a result, the PC-NFS user loses access for no apparent reason.

If the System Administrator decides that PC-NFS users still require a limited password lifetime, then remote users need to know they can correct this situation by

1. Invoking the virtual terminal (TELNET) facility, provided with TCP/IP, and opening a connection to the PRIMOS host.
2. Logging in to the PRIMOS system and changing the expired password.

Designating a Remote Printer: A user from a PC-NFS client can print files on a PRIMOS system printer. If the despooler for the printer is on the PRIMOS NFS server, you need to make *no special adjustments* for a PC-NFS client to use it. However, you need to make an entry in the file **NFS_>SPOOLQ>SPOOL_OPTIONS** in these situations:

- If the despooler for the printer is not running on the local PRIMOS system where NFS is installed
- If you want special options set for a particular printer when it services requests from a named NFS client system

Refer to the template file **NFS_>SPOOLQ>SPOOL_OPTIONS.TEMPLATE** for directions on printer registration. A variation of it is shown below:

```
/* SPOOL_OPTIONS template
/*
/* printername [ : NFS-client-system ] spool-options
/*
/*
/* By default printer labptr1 services all clients.
/* The client PCNFS1 uses it, requesting this additional header:
labptr1 : PCNFS1 -hdr "Spooled from PCNFS1"
/*
/* This printer provides remote despooling for the designated client:
LabPtr2 : PCNFS2 -ON SYS9 -hdr "Spooled from PCNFS2"
```

The most important points are

- Uppercase and lowercase letters are interchangeable.
- The **-HDR** option allows you to identify the client system making the print request.
- The **-ON** option enables printing via a remote despooler.

The only other option likely to be useful is the **-DEFER** option. If you need further details on printer options, refer to the *Operator's Guide to the Spooler Subsystem.*

To create the **SPOOL_OPTIONS** file, you can either copy or rename the template file. Make the necessary changes in the file; NFS on PRIMOS Systems can use it immediately.

STARTING, UPDATING, AND STOPPING NFS ON PRIMOS SYSTEMS

Each procedure has its own command: **START_NFS**, **NFS_INIT_USERS**, and **STOP_NFS**. They are separately described after the following details.

After you have configured the **PASSWD**, **GROUP**, and **EXPORTS** files, provided ACLs for NFS to access those partitions named in the **EXPORTS** file, and configured access for PC-NFS users, you must check two other details before you start NFS on PRIMOS Systems.

Be sure that your PRIMOS system is

- Configured with the proper time information
- Running PRIMOS TCP/IP

Checking Time Information

NFS on PRIMOS Systems needs a site's local time zone and daylight saving time information. This system information should be configured before the startup of NFS. Time information not properly set on the server causes a client system to see false time attributes that may cause an error during file system procedures.

To verify that time information is properly configured, check that the **SET_TIME_INFO** command is included in the **PRIMOS.COMI** file. If it is not, issue the command at the supervisor terminal, following directions supplied in the *Operator's Guide to System Commands*.

For example, during the calendar year 1990 a site in Boston, Massachusetts, in Eastern United States uses the following command:

```
OK, SET_TIME_INFO -TZ -0500 -DLST YES 040190 0200 102890 0200 0100
```

Note

Some 50 Series processors may not be able to provide the default arguments of *start_date*, *end_date*, and *dlst_offset* that accompany the **-DLST YES** option, so you are urged to supply all the arguments relevant to your site, using the same format as this example. Report any problem with the **SET_TIME_INFO** command to your Customer Service Representative.

Checking TCP/IP Activity

NFS on PRIMOS Systems is built on the TCP/IP environment and its tools.

To check whether PRIMOS TCP/IP is running, issue the command

```
OK, STAT USER
```

and verify that **TCPIP_MANAGER** is one of the phantoms listed. If not, enter the command **START_TCP/IP** at the supervisor terminal and allow initialization prompts to complete. Then issue the **START_NFS** command at the supervisor terminal.

If you attempt to start NFS when TCP/IP is inactive, you receive the following prompt:

```
OK, START_NFS
[START_NFS Rev. 1.2-23.0 Copyright (c) 1991, Prime Computer Inc.]
TCP/IP is not running.  Please download the controller and start it.
OK,
```

The START_NFS Command

Format

START_NFS [*options*]

Options

-ACLGROUP *.group1, ...group4*
-AGRP
Makes **NFS_SERVER** a member of the ACL group(s) specified. Each ACL group must be preceded by a period (.). Specify a maximum of four ACL groups. (See page 5-17: The -ACLGROUP Option.)

-CACHEAGINGTIME *n*
-CTIME
Sets the duration in seconds for the likely validity of a cache entry, after which its volatile data (such as file size) is discarded and the entry is eligible for being flushed from the cache. The value for *n* can range from 3 to 300; its default value is 30. (See page 5-17: The Cache Options.)

-CACHESIZE *n*
-CSIZE
Sets the maximum number of filehandle entries stored to enable fast NFS server response. The value for *n* can range from 0 to 3000; its default value is 250. (See page 5-17: The Cache Options.)

-FILETYPE *method*
-FTYPE
Specifies that files are stored and accessed using either Sequential Access Method (SAM) or Direct Access Method (DAM). The default is DAM.

-FIX_FSID
Specifies that non-unique file system IDs be fixed. You *may* need to use this option on a few occasions: at the first activation of NFS Release 1.2 on PRIMOS Rev. 23.0, and after adding another directory to the EXPORTS file. (See page 5-18: The -FIX_FSID and -NOCHECK_FSID Options.)

-GROUPFILE *pathname*
-GFILE
Specifies the pathname location for the file that functions as the **GROUP** file on the PRIMOS system. The default location for *pathname* is **NFS_>ETC>GROUP**.

-HELP
-H
Displays the syntax and options for this command.

-NOCHECK_FSID
Directs NFS to ignore non-unique file system IDs and to provide access to them (if ACLs allow). If you plan to keep non-unique file system IDs on PRIMOS Rev. 23.0, you must use this option at each startup. (See page 5-18: The -FIX_FSID and -NOCHECK_FSID Options.)

-NO_FORCE_WRITE
-NFRCW
Overrides the default setting that otherwise dictates files are immediately written to disk. The default is stipulated by the NFS protocol, but inefficient on a multiprocessing system. (See page 5-19: The -NO_FORCE_WRITE Option.)

-NO_PCNFSD
-NOPC
Overrides the default that otherwise starts both servers for NFS client users; it suppresses startup of **NFS_PCNFSD** for PC-NFS client users. **NFS_SERVER** alone services NFS clients.

Note

If you do not have PC-NFS users, use this option to optimize system response.

-PASSWDFILE ***pathname***
-PFILE
Specifies the pathname location for the file that functions as the **PASSWD** file on the PRIMOS system. The default location for *pathname* is **NFS_>ETC>PASSWD.**

-SPOOLDIR ***pathname***
-SDIR
Specifies the pathname to the directory under which files from PC-NFS clients are queued for printing. If you need special spooling options, specify them in the **SPOOL_OPTIONS** file, and locate the file in your current spool directory. The default directory is **NFS_>SPOOLQ.**

Discussion

The **START_NFS** command, located in directory **CMDNC0**, initializes server phantoms and then terminates. The phantoms require some time to establish a stable state for NFS on PRIMOS Systems. The time varies according to CPU size and system load, from several seconds on a large system to a minute on a heavily loaded 2450™ machine.

You can have NFS on PRIMOS Systems automatically start at each cold start of PRIMOS. Within the **PRIMOS.COMI** file, be sure to issue the relevant commands in this order:

1. **SET_TIME_INFO** (with options and arguments described previously)
2. **START_TCP/IP**
3. **START_NFS**

The -ACLGROUP Option: This option offers you a tool to ensure that **NFS_SERVER** is given sufficient access rights to partitions, directories, and files. As Administrator, you can designate up to four ACL groups after this option.

When a PRIMOS user sets a specific ACL and fails to include access rights for **NFS_SERVER**, then NFS cannot access any subdirectory, including subdirectories that grant access to particular client NFS users.

The **-ACLGROUP** option allows you to reestablish rights for **NFS_SERVER**, as long as the user's specific ACL includes **$REST:LUR**. As discussed on page 5-8, there must be some common ACL groups already granted access on the topmost client NFS subdirectories susceptible to unexpected exclusion by PRIMOS users. Then, each time you start NFS, you give the name of these ACL groups as an argument to the **-ACLGROUP** option. All subdirectories beneath that point have NFS access preserved.

For example, if this command is issued

```
OK, START_NFS -ACLGROUP .BACKUP$ .OPS .ADMINISTRATORS
```

then NFS_SERVER is made a member of the three ACL groups. A subdirectory (and others beneath it) with rights for one or more of these groups can now establish rights for **NFS_SERVER**, even if a higher directory has a specific ACL that omits these rights.

The Cache Options: You can tune NFS to your system usage by adjusting the values for the **-CACHEAGINGTIME** and **-CACHESIZE** options.

The cache used by NFS on PRIMOS Systems provides a fast way to determine the pathname associated with an NFS client request. An efficient cache should be small enough for a quick search but large enough to avoid frequent cache overflow. If the cache is too large, then a search for a pathname within the cache may take almost as long as a system search, and then there is little gained by using a cache. If the cache is too small, then a cache search fails more frequently and becomes an additional delay to the system search that must be conducted anyway.

As its present default, the cache holds information about the 250 files and directories most recently accessed by NFS. When the cache becomes 80% full, the cache mechanism starts to ensure that an overflow does not occur by removing entries that have not been accessed within a given time period (twice the value of the **-CACHEAGINGTIME** option).

The cache overflows when information about a new entry cannot be added because there are no expired entries. In this case the latest NFS request uses a system search to reach a conclusion, but its results are discarded instead of being added to the cache. To avoid this situation, you can reduce the value of the **-CACHEAGINGTIME** option.

Normally, it is best to leave the **-CACHESIZE** option unchanged. If your cache is repeatedly overflowing, you may need to increase the size slightly, but you should try reducing **-CACHEAGINGTIME** first. Remember that a larger cache takes longer to search. Decreasing the value for **-CACHESIZE** on a system with low NFS activity usually gains

very little. The preferred adjustment for both heavy and light NFS use is the **-CACHEAGINGTIME** option.

If NFS service is very slow, then try to verify that there is a cache problem before changing the value for the **-CACHEAGINGTIME** option. You can find out if the cache is overflowing and losing temporary entries by issuing the **NFSSTAT** command, the diagnostic tool discussed later in this chapter. Two of the statistical fields display the condition of the cache: `fhc_ovfl` and `fhc_lost`.

The `fhc_ovfl` (filehandle cache overflow) statistic reports the number of times the cache has reached overflow. If the number of overflows is around 1% of the calls, the cache is probably being used well, as long as the cache is able to purge old entries after an overflow.

The `fhc_lost` (filehandle cache entries lost) statistic reports the number of times a temporary cache entry had to be discarded because there were no aged entries to purge after a cache overflow. When the `fhc_lost` statistic is greater than 1% of the calls, you should try lowering the value for the **-CACHEAGINGTIME** option.

However, if ensuing **NFSSTAT** displays show that this adjustment has increased the numbers for *both* `fhc_ovfl` and `fhc_lost`, then the Administrator must also increase the value for the **-CACHESIZE** option.

Note

If files and directories are being shared by native PRIMOS users and by NFS users, file access conflicts can occur, since NFS does not close a file until its cache aging interval expires. Thus, a local PRIMOS user may have to wait for access to a file no longer being used by an NFS user, to allow the cache aging interval to expire. The Administrator should therefore make the cache aging value as small as possible without eliminating the usefulness of the cache.

The -FIX_FSID and -NOCHECK_FSID Options: The NFS protocol expects a system to export each file system object with a unique file system identifier (FSID). This identifier, generated from each disk partition's date and time created, sets the object apart from any other on the system. NFS uses unique FSIDs to ensure that a user cannot mistake one file system object for another somewhere within this network.

Some older PRIMOS directories may cause difficulties with the generic NFS mechanism for providing a unique identifier. NFS on PRIMOS Systems provides a solution for this problem, but the NFS Administrator must decide how to apply the solution. If NFS on PRIMOS Systems finds any non-unique objects at startup, it does the following as the default:

1. It issues an error message about each non-unique file system object.
2. It prevents NFS from being started.

The NFS Administrator can override the above default procedure with either the -FIX_FSID option or the -NOCHECK_FSID option.

The -FIX_FSID option directs the system to submit each non-unique file system object to an additional step that will attempt to provide the object a unique FSID. This needs to be done only once for the object. Consequently, the NFS Administrator needs to use this option rarely (if ever).

Note

The PRIMOS NFS server can export a directory that resides on another PRIMOS machine. If the remote directory needs a unique FSID, then NFS_SERVER may experience access difficulties while generating the unique FSID. One solution is to make NFS_SERVER a member of the .BACKUP$ ACL group on the remote PRIMOS system.

The -NOCHECK_FSID option directs the system to allow access to all exported directories, even those that are detected to have non-unique FSIDs. Using this option might produce unexpected results, depending on how the various clients use the FSID. Nevertheless, the NFS Administrator may decide that data on the object being exported is not at risk, and the object does not warrant the issuance of a permanent unique FSID. The NFS Administrator needs to use this option each time NFS is started and such an object is being exported.

Note

The suggested procedure is to issue START_NFS and let it default to checking for unique FSIDs. If non-unique objects are identified, the NFS Administrator can then decide on which of the above options is most suitable for resolving the situation.

The -NO_FORCE_WRITE Option: The NFS protocol requires that all file operations be written immediately to disk. This allows NFS file operations to be stateless (that is, guaranteed to be complete, even if the server crashes just after returning a reply). If you use the **-NO_FORCE_WRITE** option, you allow PRIMOS to use its internal buffering and scheduling to handle these **write** procedures. You thereby increase the speed of response for NFS on PRIMOS Systems, but you sacrifice the guarantee of stateless file operations.

The NFS_INIT_USERS Command

Format

NFS_INIT_USERS [*-options*]

Options

-GROUPFILE *pathname*
-GFILE
Specifies the pathname for the file that functions as the **GROUP** file. The default is to access the file last used by NFS for a start or update.

-HELP
-H
Displays the syntax and options for this command.

-NO_PCNFSD
-NOPC
Overrides the default that otherwise starts both servers for NFS client users; it suppresses startup of **NFS_PCNFSD** for PC-NFS client users. **NFS_SERVER** alone services NFS clients.

Note

If you do not have PC-NFS users, use this option to optimize system response.

-PASSWDFILE *pathname*
-PFILE
Specifies the pathname for the file that functions as the **PASSWD** file. The default is to access the file last used by NFS for a start or update.

Discussion

The **NFS_INIT_USERS** command allows you to update the list of users and groups registered for NFS on PRIMOS Systems without having to stop and restart NFS on the system. In fact, this command executes successfully only when NFS on PRIMOS Systems is already running.

This command is located in the directory **CMDNC0.**

If you do not have PC-NFS client users, you can improve system response by limiting the update to the **NFS_SERVER** only. Do this by specifying

```
OK, NFS_INIT_USERS -NO_PCNFSD
```

If you wish to change from either of the presently active source files for **GROUP** or **PASSWD**, you can use the options provided to initialize the appropriate file. You must supply the full pathname down to the filename, but you need not call the files **GROUP** and **PASSWD.** Initialization requires a reading of both source files. So, if you give a

specific pathname for only one of these, the system by default reinitializes the last used version for the other file.

The STOP_NFS Command

Format

STOP_NFS

Discussion

You can stop and restart NFS on PRIMOS Systems at any time, even when clients are active, without losing file data. If client users are active when you stop NFS, what they see depends on whether they have done a soft mount or a hard mount of the partitions they are accessing. If it was a soft mount, the user sees the error message `RPC time out` and file access fails. If it was a hard mount, the user's access is suspended; access resumes when NFS on PRIMOS Systems is restarted.

You may issue the **STOP_NFS** command only from the supervisor terminal. The command doesn't execute until you respond `Yes` to the `Really?` prompt. Then it logs out all NFS server phantoms.

The **STOP_NFS** command is located in the directory **CMDNC0.**

CHECKING NFS SERVER STABILITY

NFS on PRIMOS Systems uses these server phantoms:

- **NFS_MOUNT** handles mount and dismount requests from the client hosts. It is the PRIMOS equivalent to the NFS server daemon **mountd.**
- **NFS_SERVER** processes each request for individual file or directory operations from client hosts, either satisfying them or returning an error message to the client. It is the PRIMOS equivalent to the NFS server daemon **nfsd.**
- **NFS_PORTMAP** tracks the port assignments of the other servers. It is the PRIMOS equivalent to the NFS server daemon **portmap.**
- **NFS_PCNFSD** allows a PC-NFS client user to share a PRIMOS user ID, with its individual access rights, and to spool files to a printer attached to the PRIMOS system. **NFS_PCNFSD** is the PRIMOS equivalent to the NFS server daemon **pcnfsd.**

If the response to a **STAT USER** command does not show all the phantoms you intended to use, give the **STOP_NFS** command and then the **START_NFS** command.

All four NFS servers keep log files, as discussed next.

NFS Log Files

NFS generates log files as part of the startup, shutdown, and recovery process. The two sets of log files useful to the NFS Administrator are

- Logs in **NFS_>LOG_FILES.** These are the Server Logs that are currently open, storing messages for the ongoing session.
- Logs in **NFS_>LOG_FILES>ARCHIVE.** These archived Server Logs are closed, holding the history of the previous session.

Note

Other files do exist in **NFS_>LOG_FILES** and in **NFS_>LOG_FILES>ARCHIVE.** These are COMO files that hold diagnostic messages intended primarily for the use of your PrimeService Representative. All information needed by an NFS Administrator is in a LOG file.

You should not need to look in a COMO file, with one exception: the RUSERD.COMO. The RUSERSD server, as an optional service available at Release 1.2, does not receive Prime support and is therefore not assigned its own log. However, if RUSERSD experiences a problem, the RUSERSD.COMO normally provides information sufficient for a solution.

Open Server Logs: NFS startup and shutdown procedures generate the following local log files, which are in the directory **NFS_>LOG_FILES:**

MOUNT_SERVER_LOG
Records the startup, shutdown, and error messages from the NFS server **NFS_MOUNT.**

PORTMAP_SERVER_LOG
Records the startup, shutdown, and error messages from the NFS server **NFS_PORTMAP.**

NFS_SERVER_LOG
Records the startup, shutdown, and error messages from the NFS server **NFS_SERVER.**

NFS_PCNFSD_LOG
Records the startup, shutdown, and error messages from the NFS server **NFS_PCNFSD.**

If you discover that any of these log files are missing after you have started NFS, you might look into its related COMO file before you contact your PrimeService Representative with the information. Its last entry should be an error message indicating the failure to open the log file; this message helps you verify that this is indeed a software problem and that the log was not deleted inadvertantly by another user.

Archived Server Logs: When you start NFS, it saves a copy of the logs from the previous session before initializing them for the current session. NFS copies the logs from the previous session into files of the same name in the directory **NFS_>LOG_FILES>ARCHIVE.** In the process, it overwrites the last log for that server. No more than one archived log exists for each server, and it is the log for the previous session.

The NFSSTAT Command

Format

NFSSTAT [*-options*]

Options

-HELP
-H
Displays the syntax and options for this command.

-ZERO
-Z
Sets the counters to zero.

Discussion

The **NFSSTAT** command allows you to check on how NFS is working. The command resides in **CMDNC0.** Any user can issue the command at any terminal. If NFS is running properly, the command returns some NFS statistics. Otherwise, you get a message such as

```
Unable to contact server to get statistics
```

Refer to Appendix C for a list of messages you may receive. You may get such a message if the servers started but have not stabilized NFS on PRIMOS Systems yet, so wait a while and try the command again.

The command returns a list of procedure requests and statistics connected with each of them. The statistics assist in analyzing NFS performance.

Each request, from `null` through `fsstat`, has two values displayed under it. The first number indicates the number of times the procedure was called. The second number indicates the percentage of the total calls dedicated to this procedure. For example, in the following display `lookup` was called 278 times out of a total of 349 calls; 79% of all calls were lookup calls.

Sample Statistics From NFSSTAT: The following is a typical display returned by the **NFSSTAT** command after it is issued on the PRIMOS system:

```
OK, NFSSTAT
[NFSSTAT Rev. 1.2-23.0 Copyright (c) 1991, Prime Computer, Inc.]
Server nfs:
calls      badcalls
349        0
null       getattr    setattr    root       lookup     readlink   read
0  0%      47 13%     0  0%      0  0%      278  79%   0  0%      1  0%
wrcache    write      create     remove     rename     link       symlink
0  0%      0  0%      0  0%      1  0%      0  0%      0  0%      0  0%
mkdir      rmdir      readdir    fsstat
0  0%      0  0%      22  6%     0  0%
fhc_ovfl   fhc_lost
0          0
```

Description of Procedures: Most of the statistical information returned by **NFSSTAT** concerns procedure requests made to NFS on PRIMOS Systems. Table 5-2 briefly describes each procedure. Note that a file system object may be either a directory or a file.

TABLE 5-2. Description of Procedures Within NFSSTAT

null: Pings, that is, tests to verify that the server is running.	**getattr:** Gets attributes of a file system object.
setattr: Sets attributes of a file system object.	**root:** (Obsolete.)
lookup: Verifies the existence of a file system object in a directory.	**readlink:** (Not supported by NFS on PRIMOS Systems.)
read: Reads data from a file.	**wrcache:** (Obsolete.)
write: Writes data to a file.	**create:** Creates a file.
remove: Deletes a file.	**rename:** Renames a file.
link: (Not supported by NFS on PRIMOS Systems.)	**symlink:** (Not supported by NFS on PRIMOS Systems.)
mkdir: Creates a directory.	**rmdir:** Removes a directory.
readdir: Reads entries from a directory.	**fsstat:** Gets file systemwide parameters.
calls: Gives a tally of valid (one of the above) procedures received by the server.	**badcalls:** Gives a tally of client requests with invalid procedure numbers (does not include the unimplemented procedures).
fhc_ovfl[1]: Gives a tally of how often the filehandle cache was full when a client request attempted to add a new cache entry.	**fhc_lost**[1]: Gives a tally of how often the NFS_SERVER was unable to add a new entry to the cache because there were no aged entries to remove.

[1]If the counts of either of these fields is greater than 1% of the total calls, consider increasing the size of the filehandle cache by using the -CACHESIZE option with START_NFS.

6

DIAGNOSING NFS PROBLEMS

USING THE RPCINFO COMMAND

You can use the **rpcinfo** command to

- Identify the registered NFS server processes on a host that is an NFS server (that is, list the servers registered with a host's **portmap** process)
- Query individual server processes on an NFS server host

The **rpcinfo** command has the pathname **/usr/etc/rpcinfo** on a Sun Workstation. The command is considered a client command, but the **RPCINFO** command is also available on the PRIMOS NFS server so that you can gather information about other servers. While most of the ensuing descriptions show the use of the command on a Sun Workstation as client, you can also use the command on your PRIMOS system (see page B-7).

Identifying Registered Servers

You use **rpcinfo** with the **-p** option to identify all the NFS server processes registered with **portmap** (known as **NFS_PORTMAP** on the PRIMOS system). For example, a user on a Sun client issues this command to identify the servers registered with **NFS_PORTMAP** on the NFS server host **prime1**:

```
% rpcinfo -p prime1
   program  vers proto   port
    100000     2   tcp    111  portmapper
    100000     2   udp    111  portmapper
    100003     2   udp   2049  nfs
    150001     1   udp   5027  pcnfsd
    100005     1   udp   5028  mountd
%
```

The response indicates that this PRIMOS system had all its NFS servers registered at the last initialization of NFS.

Instead of a list of registered servers as a response, you may receive the following error message that indicates **NFS_PORTMAP** is not running, or has not completed initialization:

```
RPC: Portmapper failure
```

Allow a few moments for initialization to complete, then retry the command.

Querying Individual Servers

While servers register with the **portmap** process at startup time, they may for some reason no longer be active. You can use the **-u** option with **rpcinfo** to have the program "ping" the other NFS servers (not the portmapper). The program requests a response to an "Are you there?" message. The remote host's targeted server is identified by a fixed program number and version, as follows:

Server Name	*Program Number*	*Version*
NFS_SERVER	100003	2
NFS_MOUNT	100005	1
NFS_PCNFSD	150001	1

In the following example, a user on a Sun client requests information about each of the above phantoms on the server host **prime1**:

```
% /usr/etc/rpcinfo -u prime1 100003 2        /* Ping NFS_SERVER
program 100003 vers 2 ready and waiting

% /usr/etc/rpcinfo -u prime1 100005 1        /* Ping NFS_MOUNT
program 100005 vers 1 ready and waiting

% /usr/etc/rpcinfo -u prime1 150001 1        /* Ping NFS_PCNFSD
program 150001 vers 1 ready and waiting

%
```

Instead of a `...ready and waiting` response, you may see `...not available` preceded by another message that gives a clue to what is wrong:

`RPC: Program not registered`
The server you pinged is not running, or has not completed initialization.

`RPC: Timed out`
The server you pinged completed its initialization, but it crashed.

If your system is BSD-based or SVID-compliant, consult its manual pages for other details on **rpcinfo**.

DIAGNOSING PROBLEMS WITH NFS ON PRIMOS SYSTEMS

This section presents some problems and their likely solutions. A problem may be assigned different error messages on different client NFS implementations, so this list is only a guideline. Most of these error messages were reported on a Sun Workstation as NFS client.

`mount failed`

From the client system, use **rpcinfo** to see if NFS and MOUNT servers are running. On PRIMOS, use **STAT USER** to check the status of all servers. Check that the **EXPORTS** file contains the partition you are trying to mount. Check that the list of hosts for that partition is either empty or contains your host name. Verify that the host name is your host's primary or secondary name in the **HOSTS** file for PRIMOS TCP/IP. Check that **NFS_MOUNT** has at least LUR access to the partition's MFD. If you tried to mount a pre-Revision 20 partition, the file **NFS_MOUNT_LOG** records an error message. Reformat the partition with a version of **MAKE** that is Rev. 20 or later. If the PRIMOS partition is remote to the server, then check that the gateway host can see the partition, that the PRIMENET remote system is up and has slaves available, and that user validation is not set between PRIMOS hosts.

`permission denied` (for access to a file system object)

Check that **NFS_SERVER** has sufficient access for the request, and has at least LU access to higher directories. Check that either your **uid** or your **gid** (preferably both) maps to its PRIMOS equivalent in the **PASSWD** and **GROUP** files. Check that those PRIMOS equivalents have adequate PRIMOS access rights. Check to see if an extensive list of ACL entries (more than 28 entries) has been truncated, after an NFS client performs an operation that modifies ownership or access permission mode. Make sure your effective **uid** is not **root**, which always maps to **nobody** for NFS on PRIMOS Systems.

`RPC timed out`

Use **rpcinfo** to see if NFS and MOUNT servers are running. If they are all running, you may need a longer time to carry out an NFS file system operation; use the **mount** command with a larger value for **timeo=*n***.

`File too big`

You may have exceeded the maximum number of file system objects that a directory can contain. Check to see if this directory contains file system objects that number in the thousands. If so, create a directory one level up, and offload to it some of the files from this directory.

`Unknown error`

There may be a conflict with RWLOCK settings on a PRIMOS file. A PRIMOS user may have the file open in Exclusive mode, preventing **NFS_SERVER** from opening the file. Refer to the file **NFS_SERVER_LOG** to identify other problems that may have occurred on the server side.

`Server can't decode arguments` or `Invalid argument`

This usually means that your filename is too long or becomes too long after being mapped to PRIMOS.

APPENDICES

A

DESCRIPTION OF COMMANDS, SERVERS, AND FILES

This appendix contains descriptions of the following NFS client commands and reference files, arranged in alphabetical order. These descriptions use terminology from the point of view of a user on an NFS client that is a BSD-based or SVID-compliant system. They do not describe the file or function as it exists on the PRIMOS system. They are here mainly to provide background information for the server implementation of NFS on PRIMOS.

If your client is a BSD-based or SVID-compliant system, you will find additional information either in the documentation or the **man** command provided by the client NFS package.

Descriptions are provided for the following:

fstab
mount/umount
mtab
nfsstat
rpcinfo
showmount

NAME

fstab – table of remote file systems used by local host

USAGE

Read by the **mount** and **umount** commands

DESCRIPTION

The **/etc/fstab** file contains information about file systems mounted by the local host. You maintain this file by producing formatted file system entries with the **mount** command and its **-p** option, then manually adding the entries to the **/etc/fstab** file with a text editor.

The **mount** and **umount** commands process the entries from top to bottom. The entries must therefore appear in proper sequence; that is, a file system mounted within another file system must appear below that file system.

An entry occupies a single line and contains the following information:

```
host:fsname dir type args 0 0
```

where

host:fsname
Remote host name, colon (:), and file system name.

dir
Mountpoint pathname (remote file systems) or temporary directory pathname (removable file systems).

type
File system type specified at mount time with the **mount** command's **-t** option (possible values are 4.2 and nfs; the default is nfs).

args
Argument(s) specified at mount time with the **mount** command's **-o** option (the default is hard).

0 0
Unused.

Entries in the **/etc/mtab** file have the same format.

EXAMPLE

bonmot:/usr/bonmot_usr nfs soft 0 0

FILES

/etc/fstab

RELATED INFORMATION

mount(8)

mtab(5)

NAMES

mount and **umount** – mount and dismount file system

USAGE

Removable file systems only:

/etc/mount [-ar] [dev dir]
/etc/umount [-a] [dev]

Remote file systems only:

/etc/mount [-afrvp] [to type options] [fsys] [dir]
/etc/umount [-av] [-root] [fsys | dir]

DESCRIPTION

The **mount** command mounts removable and remote file systems.

The **umount** command unmounts removable and remote file systems.

When mounting a removable file system located on a diskette, cartridge tape, or magtape, specify the device name for dev and a nonexistent directory pathname for dir.

When mounting a remote file system (that is, one located on another host), specify the remote host and file system in the form **host:fsname** for **fsys** and a local mountpoint name for **dir.** (The **mount** command creates a mountpoint named **dir**; the **umount** command deletes it.)

The **mount** and **umount** commands maintain a table of mounted file systems in **/etc/mtab.** If you execute the **mount** command without arguments, it prints the table. If you specify only one argument, the command searches **/etc/fstab** for an entry with the same argument; if the entry exists, the command then uses it to mount the file system.

OPTIONS

The command options can be one or more of the following items separated by a space:

-a

With **/etc/mount**, try to mount all file systems listed in the file **/etc/fstab.** (This file contains entries added manually by user.)

With **/etc/umount**, try to dismount all file systems listed in **/etc/mtab.** (This file contains an entry for every remote file system currently mounted by the local host.)

-f

Fake a new **/etc/mtab** entry without mounting the file system.

-o
Accept the following arguments separated by commas, as options. (The defaults are fg, retry=1, timeo=7, and hard.)

bg
Retry in background if server host's mountd(8c) doesn't respond.

fg
Retry in foreground.

hard
Retransmit until server responds.

port=*n*
Set NFS port number *n*.

retry=*n*
Set number of mount retries to *n*.

ro
Allow read only.

root
Mount in global root directory.

rsize=*n*
Set read request buffer size to *n* bytes.

rw
Allow read/write.

soft
Report error if server fails to respond.

timeo=*n*
Set NFS timeout to *n* tenths of a second. When attempting to access mounted file system, NFS waits specified amount of time for response; if no response, NFS multiplies *n* by 2 and retransmits request.

wsize=*n*
Set write request buffer size to *n* bytes.

-p
List the mounted file systems in the same format as entries in **/etc/fstab.**

-r
Mount the file system as read only.

-root
Dismount the file system in the global root directory.

-t
Accept the following argument as the file system type; the recognized types are 4.2 and nfs.

-v
Display a message when mounting the file system.

FILES

/etc/mtab

/etc/fstab

CAUTIONS

You must mount physically write-protected file systems (including those on magnetic tape) as read only. If not, errors develop even if no explicit write is attempted.

Mounting a removable file system that contains unreadable or otherwise invalid data can have unpredictable consequences.

RELATED INFORMATION

mount(2)

mtab(5)

fstab(5)

mkdisk(8)

Note

See Chapter 4 for options to the **mount** and **umount** commands that are supported by NFS on PRIMOS Systems.

NAME

mtab – table of mounted file systems

USAGE

Used by the **mount** and **umount** commands

DESCRIPTION

The **/etc/mtab** file identifies the file systems currently mounted by the local host. The **mount** command creates entries in this file; the **umount** command deletes them.

Each entry corresponds to a file system curently mounted by the local host. An entry occupies a single line and contains the following information:

```
host:fsname dir type args 0 0
```

where

host:fsname
Remote host name, colon (:), and file system name

dir
Mountpoint name (remote file systems) or temporary directory pathname (removable file systems)

type
File system type specified at mount time with the **mount** command's **-t** option (possible values are 4.2 and nfs; the default is nfs)

args
Argument(s) specified at mount time with the **mount** command's **-o** option (the default is hard)

0 0
Unused

EXAMPLE

```
gortek:/usr /gortek_usr nfs soft 0 0
```

FILES

/etc/mtab

RELATED INFORMATION

mount(8)

fstab(8)

NAME

nfsstat – Network File System statistics

USAGE

/etc/nfsstat [-csnrz]

DESCRIPTION

The **nfsstat** command displays statistical data about NFS; you can also use it to reinitialize the statistics. Without options, the command defaults to

```
nfsstat -csnr
```

OPTIONS

-c

Display client statistics. You can combine this option with the **-n** and **-r** options to display only client NFS or client RPC statistics.

-s

Display server statistics in the same manner as the **-c** option.

-n

Display NFS statistics for both client and server. You can combine this option with the **-c** and **-s** options to display only client or server NFS statistics.

-r

Display RPC statistics in the same manner as the **-n** option.

-z

Zero (reinitialize) statistics. You can combine this option with any of the other options to reinitialize particular sets of statistics after printing them. This option does not require super-user privileges.

Note

See Chapter 5 for options to the **nfsstat** command that are supported by NFS on PRIMOS Systems.

NAME

rpcinfo – report RPC information

USAGE

/etc/rpcinfo -p [host]

/etc/rpcinfo -u host program [version]

/etc/rpcinfo -t host program [version]

DESCRIPTION

The **rpcinfo** command makes an RPC call to an RPC server and reports the results.

The *host* argument can be either a host name or an address. The *program* argument can be either a name or a number.

If a *version* is specified, **rpcinfo** attempts to call that version of the specified *program*. Otherwise, **rpcinfo** attempts to find all the registered version numbers for the specified *program* by calling version 0. This version is presumed not to exist, and a failure to find it causes the queried host to return a list of all registered versions of the program. If version 0 does respond, then **rpcinfo** makes another attempt to identify all the registered version numbers. This time it culls the information by calling an extremely high version number. The failure to find this version should cause the list of all versions to be returned. After the version numbers are identified, **rpcinfo** attempts to call each registered version.

OPTIONS

-p
Probe the portmapper on *host*, and display a list of all registered RPC programs. If *host* is not specified, the portmapper on the local system is probed.

-u
Make an RPC call to procedure 0 of *program* on the specified *host* using UDP, and report whether a response is received.

-t
Make an RPC call to procedure 0 of *program* on the specified *host* using TCP, and report whether a response is received.

Note

This client-side utility is also available for use on the 50 Series. See **RPCINFO** in Appendix B.

NAME

showmount – show all remote mounts

USAGE

/etc/showmount [-a] [-d] [-e] [host]

DESCRIPTION

The **showmount** command lists all remote clients that have mounted file systems belonging to *host* (either specified or the local system by default). The command gets its information from the **mountd(8)** server on the host.

OPTIONS

-a

Display all remote mounts in the format

hostname:directory

where hostname is the name of the client, and directory is the root of the mounted file system.

-d

List directories that have been remotely mounted by clients.

-e

Display the list of exported file systems in **/etc/exports**.

FILES

/etc/mtab

/etc/fstab

/etc/rmtab

CAUTION

If a client crashes, its mountpoints may remain in effect until someone reboots the client and executes the **/etc/umount** command.

RELATED INFORMATION

mountd(8)

exports(8)

B

COMMAND HELP FILES FOR NFS ON PRIMOS SYSTEMS

Users on the 50 Series machine with software for NFS on PRIMOS Systems can receive online help for the following NFS commands:

DOSTOP	**PTOU**	**RUSERS**
NFS_INIT_USERS	**RPCGEN**	**START_NFS**
NFSSTAT	**RPCINFO**	**STOP_NFS**
PTODOS		**UTOP**

Issue the **HELP** command followed by one of the command names above to display its online command description. Copies of the command descriptions follow, in alphabetical order.

Note that two new HELP files are available with Release 1.2 of NFS on PRIMOS Systems. The first, for RPCGEN, is available immediately after installation of Release 1.2. However, if the System Administrator does not thereafter install the optional RUSERS service, no HELP file is made available for RUSERS. See Appendix E for further details about RUSERS.

DOSTOP

DOSTOP Converts DOS ASCII file to PRIMOS ASCII format

DOSTOP input_pathname [output_pathname] [-TAB tabsize]

DOSTOP takes a DOS ASCII source file and generates a file formatted in PRIMOS ASCII. Any user can issue this command.

If the output_pathname argument is not specified, the contents of input_pathname are converted to PRIMOS ASCII format.

DOSTOP supports one option:

-TAB tabsize
Specifies the number of spaces represented by a tab character in the DOS-formatted file. The default value is 8.

For more information on the DOSTOP command, see the Guide to NFS on PRIMOS Systems.

April 1990

NFS_INIT_USERS (Operator command)

NFS_INIT_USERS [-options]

NFS_INIT_USERS allows you to update the users registered for NFS on PRIMOS Systems without needing to stop and restart NFS on the system. The command can be used only at the supervisor terminal, and only while NFS on PRIMOS Systems is active.

NFS_INIT_USERS supports the following options:

-GROUPFILE pathname, -GFILE
Specifies the pathname for the file that functions as the GROUP file. The default is to access the file last used by NFS for a start or update.

-HELP, -H
Displays the syntax and options for this command.

-NO_PCNFSD, -NOPC
Overrides the default that otherwise starts both servers for NFS client users; it suppresses startup of NFS_PCNFSD for PC-NFS client users. NFS_SERVER alone services NFS clients.

-PASSWDFILE pathname, -PFILE
Specifies the pathname for the file that functions as the PASSWD file. The default is to access the file last used by NFS for a start or update.

For more information on the NFS_INIT_USERS command, see the Guide to NFS on PRIMOS Systems.

April 1990

NFSSTAT Prints NFS statistics

NFSSTAT [-options]

NFSSTAT displays statistics collected by the NFS server. The statistics are counts of the number of NFS procedures that have been serviced by the NFS server. Any user may issue this command.

NFSSTAT supports the following options:

-HELP, -H
Displays the syntax and options for this command.

-ZERO, -Z
Resets all of the statistics counters to zero (0) after their current counts are displayed.

For more information on the NFSSTAT command, see the Guide to NFS on PRIMOS Systems.

April 1990

PTODOS Converts PRIMOS ASCII file to DOS ASCII format

PTODOS input_pathname [output_pathname]

PTODOS takes a PRIMOS ASCII source file and generates a file formatted in DOS ASCII. Any user can issue this command.

A built-in protective feature prevents a user from accidentally converting a file that has already been converted.

If the output_pathname argument is not specified, the contents of input_pathname are converted to DOS ASCII format.

For more information on the PTODOS command, see the Guide to NFS on PRIMOS Systems.

April 1990

PTOU

PTOU Converts PRIMOS ASCII file to UNIX ASCII format

PTOU input_pathname [output_pathname]

PTOU takes a PRIMOS ASCII source file and generates a file formatted in UNIX ASCII. Any user can issue this command.

If the output_pathname argument is not specified, the contents of input_pathname are converted to UNIX ASCII format.

A built-in protective feature prevents a user from accidentally converting a file that has already been converted.

For more information on the PTOU command, see the Guide to NFS on PRIMOS Systems.

April 1990

RPCGEN An ONC/RPC protocol compiler

RPCGEN input_file [-o output_file] [options]

RPCGEN is a tool that generates C code to implement an RPC protocol. The input to rpcgen is a language similar to C known as RPC Language (Remote Procedure Call Language). Information about the syntax of the RPC Language is available in the ONC Network Programming Guide.

RPCGEN is normally used by providing just the protocol input file and generates four output files. If the input_file is named proto.x, then RPCGEN will generate a header file in proto.h, XDR routines in proto_xdr.c, server-side stubs in proto_svc.c, and client-side stubs in proto_clnt.c.

The options are used when all output files are not desired and causes only one of the four output files to be generated.

RPCGEN has the following options:

-c Compile into XDR routines.

-h Compile into C data-definitions (a header file)

-l Compile into client-side stubs.

-m Compile into server-side stubs, but do not generate a main routine. This option is useful for doing callback-routines and for people who need to write their own main routine to do initialization.

-o output_file
Specify the name of the output file. If none is specified, standard output is used (-c, -h, -l and -s modes only).

-s transport
Compile into server-side stubs, using the given transport. The supported transports are UDP and TCP. This option may be invoked more than once so as to compile a server that serves multiple transports.

December 1990

RPCINFO Reports RPC information

RPCINFO [-options]

RPCINFO returns RPC information about NFS servers. Any user may issue this command.

RPCINFO supports the following options:

-P [host]
-T host program [version_number] [-N portnum]
-U host program [version_number] [-N portnum]
-B program [version_number]

These options are described below:

-P [host]
Probe the portmapper on the specified host (local host by default). Return a list of all registered RPC programs. [host] may be either a host name or the Internet address of the target system.

-T host program [version_number] [-N portnum]
Use TCP to make an RPC call to the specified host for the specified program. If version_number is unspecified, then attempt to call all registered versions of the program. Report if a response is received. The option -N portnum directs RPCINFO to look up the program at the specified port number, instead of using the port number provided by the portmapper. host may be either a host name or the Internet address of the target system. program may be either the name or the number of the program.

-U host program [version_number] [-N portnum]
Use UDP to make an RPC call to the specified host for the specified program. If version_number is unspecified, then attempt to call all registered versions of the program. Report if a response is received. The option -N portnum directs RPCINFO to look up the program at the specified port number, instead of using the port number provided by the portmapper. host may be either a host name or the Internet address of the target system. program may be either the name or the number of the program.

-B program [version_number]
Send a broadcast over the network requesting each portmapper process to respond if the specified program with the specified version is registered.

For more information on the RPCINFO command, see the Guide to NFS on PRIMOS Systems.

February 1991

RUSERS Present information about remote system users

RUSERS [options] [host name]

The RUSERS command presents information about users on remote system. By default, it will broadcast to the network and display the responses it receives. If a host name is provided, RUSERS sends a request directly to the specified host.

A remote host can respond only if it is running the RUSERSD daemon.

The options below are used to control the format and order of the response display.

RUSERS has the following options:

-a Give a report for a machine even if no users are currently logged on.

-h Sort alphabetically by host name.

-i Sort by idle time.

-l Present the longer version of the user listing.

-u Sort by number of users.

The ONC service is provided as a sample ONC/RPC application and is not supported by Prime Computer, Inc.

To allow a 50 Series system to respond to RUSERS requests, the RUSERSD process must be running. To start the RUSERSD process, the CPL file NFS_>RUN>RUSERSD.CPL must be run, presumably as a phantom process. It is recommended that this phantom be started just after the START_NFS command is executed at system start-up.

December 1990

START_NFS (Operator command)

START_NFS [-options]

START_NFS initializes NFS on PRIMOS Systems at coldstart, or at any time that PRIMOS NFS is not running. This command can be issued only from the supervisor terminal.

START_NFS spawns up to four server processes:
NFS_SERVER, the NFS server phantom for general client users
NFS_MOUNT, the Mount server phantom
NFS_PORTMAP, the Portmap server phantom
NFS_PCNFSD, the optional server phantom for PC-NFS client users

START_NFS supports the following options:

-ACLGROUP groups, -AGRP
-CACHEAGINGTIME seconds, -CTIME
-CACHESIZE number, -CSIZE
-FILETYPE [SAM | DAM], -FTYPE
-FIX_FSID
-GROUPFILE pathname, -GFILE
-HELP, -H
-NOCHECK_FSID
-NO_FORCE_WRITE, -NFRCW
-NO_PCNFSD, -NOPC
-PASSWDFILE pathname, -PFILE
-SPOOLDIR pathname, -SDIR

These options are described below:

-ACLGROUP groups, -AGRP
Makes a server process a member of the ACL group(s) specified. No more than four ACL groups may be specified at a time.

-CACHEAGINGTIME seconds, -CTIME
Specifies the number of seconds (3 to 300) that volatile file system data information will be considered valid in the cache. The default is 30 seconds.

-CACHESIZE number, -CSIZE
Specifies the number of entries (0 to 3000) in the file system data cache used by PRIMOS NFS. The default is 250 entries.

-FILETYPE [SAM | DAM], -FTYPE
Specifies the file type created when an NFS client issues a CREATE FILE request. The default type is DAM.

-FIX_FSID
Directs NFS to attempt to fix non-unique FileSystemIDs encountered for the pathnames listed in the EXPORTS file.

-GROUPFILE pathname, -GFILE
Specifies the pathname of the file containing the information required to map UNIX gid numbers to PRIMOS groups. The default is NFS_>ETC>GROUP.

-HELP, -H
Displays the syntax and options for this command.

-NOCHECK_FSID
Directs NFS to not attempt to fix non-unique FileSystemIDs encountered for the pathnames listed in the EXPORTS file and to start NFS regardless of any FSID conflicts.

-NO_FORCE_WRITE, -NFRCW
Specifies that write requests from NFS clients are not to be forcibly written to the disk, but instead are to be updated by the standard PRIMOS file update mechanism. The default is that all write requests are written to the disk by force.

-NO_PCNFSD, -NOPC
Specifies not to start the server NFS_PCNFSD, so that NFS_SERVER alone services NFS clients. Use this option to improve system response, if there are no PC-NFS client users.

-PASSWDFILE pathname, -PFILE
Specifies the pathname of the file containing the information required to map UNIX uid numbers to PRIMOS users. The default is NFS_>ETC>PASSWD.

-SPOOLDIR pathname, -SDIR
Specifies the pathname to the directory under which files from PC-NFS clients are queued for printing. If you need special spooling options, specify them in the SPOOL_OPTIONS file, and locate the file in your current spool directory. The default directory is NFS_>SPOOLQ.

For more information on the START_NFS command, see the Guide to NFS on PRIMOS Systems.

December 1990

STOP_NFS (Operator command)

STOP_NFS

STOP_NFS logs out the server processes spawned by NFS on PRIMOS Systems: NFS_SERVER, NFS_MOUNT, NFS_PCNFSD (if activated), and NFS_PORTMAP. STOP_NFS executes successfully only at the supervisor terminal.

For more information on the STOP_NFS command, see the Guide to NFS on PRIMOS Systems.

April 1990

UTOP

UTOP Converts UNIX ASCII file to PRIMOS ASCII format

UTOP input_pathname [output_pathname] [-TAB tabsize]

UTOP takes a source file stored in UNIX ASCII text format and generates a PRIMOS-formatted version of the file. Any user may issue this command.

If the output_pathname argument is not specified, the contents of input_pathname are converted to PRIMOS ASCII format.

UTOP supports one option:

-TAB tabsize
Specifies the number of spaces represented by a tab character in the UNIX-formatted file. The default value is 8.

For more information on the UTOP command, see the Guide to NFS on PRIMOS Systems.

April 1990

C

ERROR AND STATUS MESSAGES FOR NFS ON PRIMOS SYSTEMS

Status messages and error messages for NFS on PRIMOS Systems are issued as responses:

- From the NFS server phantoms **NFS_MOUNT**, **NFS_PORTMAP**, **NFS_PCNFSD**, and **NFS_SERVER** (their responses to requests)
- To the **START_NFS**, **NFS_INIT_USERS**, and **STOP_NFS** commands
- To the **NFSSTAT** and **RPCINFO** commands
- To the **PTOU** and **UTOP** commands
- To the **DOSTOP** and **PTODOS** commands

Messages are presented in the above order within this appendix.

The messages from the server phantoms are issued either to the supervisor terminal or to the log file for that server phantom. No suggested solutions are offered for messages that give informational status and self-evident error messages.

Error responses to the commands are issued to the standard output device (usually to the supervisor terminal for the start and stop commands, and to the terminal screen of the requesting user for the others).

Your client NFS implementation may have additional error messages that do not originate from NFS on PRIMOS Systems. Refer to the client documentation for a list of these messages.

MESSAGES RECORDED IN MOUNT_SERVER_LOG

At Initialization of NFS_MOUNT

`The NFS_Mount Server was spawned on` *`date_and_time`*
Information. The first entry in the file.

`The NFS_Mount Server is ready for requests.`
Information. Server is now registered with Portmapper and can be contacted by clients.

`A failure occurred while initializing controller connections.`
`The NFS_Mount Server is terminating.`
User or internal error. Failures have occurred in the socket/UDP software. Ensure TCP/IP is running and operational.

`svc_register() error in mountd.`
Internal error. Stop and restart NFS on PRIMOS. If the error persists, contact Customer Service.

`While setting the alarm time the NFS_Mount server encountered error.`
`The NFS_Mount Server is terminating.`
Internal error. Report the problem to Customer Service.

`Cannot convert (string) to a valid PRIMOS pathname.`
User or internal error. Check the pathname of the mountpoint entered on the client NFS system.

`Unable to allocate a buffer for pathname conversion.`
Internal error. Report the problem to Customer Service.

During Operation of NFS_MOUNT

`Error while sending response to client for procedure` *`number`*
Internal server error in sending replies to client, usually a severe problem with the server or TCP/IP software. Save the log, and request Customer Service assistance in enabling a debugging trace of the server to pinpoint the events leading to the error.

`svc_run() returned with a fatal error.`
Internal error with PRIMOS. Report the problem to Customer Service.

`The arguments could not successfully be decoded for procedure` *`number`*
Indicates the server received client request packets that were improperly formatted.

`There was no entry in the host table for client address` *`internet_add_nn`*
`The mount request was rejected.`
Either correct the **HOSTS** file, or note the address of the client seeking unauthorized entry.

The NFS_Mount Server received a request to perform unknown procedure *nn*

Internal error. The client may have requested one of the unsupported items listed in Chapter 2.

The NFS_Mount Server was unable to generate a filehandle for *pathname*. The mount request was rejected. Please verify that NFS_SERVER has access to the directory or, if the request is for a remote directory, check to see if the line is up.

A bad filehandle was generated for *pathname*.
The mount request was rejected.
This partition was probably MAKEd before Rev. 20.0.
The partition must be re-MAKEd before NFS can access it.

WARNING: The NFS_Mount Server was unable to get the file system ID for *pathname*. The partition cannot be checked to see if it has a duplicate file system ID.

Some client NFS implementations require this file system ID information.

WARNING: Partition *pathname* has a duplicate file system ID of *number*.

Two PRIMOS partitions with duplicate dtc fields were mounted. Some NFS clients may be sensitive to receiving the same file system ID from two separate file systems.

The NFS_Mount Server was unable to open NFS_>ETC>EXPORTS.
WARNING: The server will use previous contents of the EXPORTS file.

See the ensuing explanation.

The NFS_Mount Server was unable to open NFS_>ETC>EXPORTS.
WARNING: The server will not be able to grant any client mount request.

Check to ensure that the **EXPORTS** file exists and that ACLs on the file are set so that **NFS_MOUNT** can access it. If the Mount server had ever successfully opened **EXPORTS**, the list of exported partitions most recently read by the server will be used to evaluate the current client mount request. If the server was never able to mount the contents of **EXPORTS**, then all client mount requests will be denied.

NFS_>ETC>EXPORTS contains an invalid pathname: *pathname*
The entry is being ignored.

Logged each time **EXPORTS** is read.

NFS_>ETC>EXPORTS contains an entry with no partition: *pathname*
The entry is being ignored.

Logged each time **EXPORTS** is read.

NFS_>ETC>EXPORTS contains an entry with illegal characters: *pathname*
The entry is being ignored.

Logged each time **EXPORTS** is read.

The *remote_host_name* was denied access to *pathname*.

Logged whenever a mount attempt fails. Can be used to monitor attempted breaches of security (clients requesting access to file systems for which they have not been granted access).

MESSAGES RECORDED IN PORTMAP_SERVER_LOG

At Initialization of NFS_PORTMAP

The NFS_Portmap Server was spawned on *date_and_time*.
First entry in log.

The NFS_Portmap Server is now ready for requests.
Server can now be contacted by clients.

A failure occurred while initializing controller connections.
The NFS_PORTMAP Server is terminating.
Internal error in socket/UDP software. Ensure TCP/IP is operational.

proc_name: Unable to determine the name bound to this socket.
Internal error in socket/UDP software. Ensure TCP/IP is operational.

svc_run() returned with a fatal error.
The NFS_PORTMAP Server is terminating.
Internal error within PRIMOS. Report the problem to Customer Service.

Cannot create log file *pathname*.
Cannot open log file *pathname*.
Log file error: [PRIMOS *error text*]
Cannot attach to log file directory *directory pathname*.
Cannot set read/write lock on *pathname*.
Any of these error messages may be issued at initialization of the log file. If no PRIMOS error text accompanies the message, the likely problem is insufficient access.

proc_name: Memory allocation error.
Insufficient dynamic memory allocated to the process executing **NFS_PORTMAP**.

Cannot get transport handle for udp.
Cannot get transport handle for tcp.
procname: Error while sending response to client.
Cannot match IP address in controller list: *internet_address*
svc_register() error in portmapper.
svc_getargs() error in procedure *proc_name/*
One or more of these internal errors may be logged. Stop and restart NFS on PRIMOS Systems. If the error persists, contact Customer Service.

MESSAGES RECORDED IN NFS_PCNFSD_LOG

At Initialization of NFS_PCNFSD

The NFS_PCNFSD Server was spawned on *date_and_time*
First entry in log.

The NFS_PCNFSD Server is now ready for requests
Server can now be contacted by clients.

Loading uid/username database from PASSWD file *pathname*
Loading gid/groupname database from GROUP file *pathname*
Whenever new information is loaded, this event and the names of the files used are logged.

The PRIMOS revision on which this version of NFS_PCNFSD is installed does not support the system call needed to provide NFS client authentication. Revision 22.1 or a later version of PRIMOS must be installed to enable NFS_PCNFSD to provide client authentication.

A failure occurred while initializing controller connections.
The NFS_PCNFSD Server is terminating.
Internal error in socket/UDP software. Ensure TCP/IP is operational.

svc_register() error in pcnfsd.
Internal error. Stop and restart NFS on PRIMOS Systems. If the error persists, contact Customer Service.

Error in setting limit$() alarm: [PRIMOS *error text*]
The NFS_PCNFSD server is terminating.
Internal error. Stop and restart NFS on PRIMOS Systems. If the error persists, contact Customer Service.

Unable to open PASSWD_file_pathname.
There will be no uid/username database. No client uids will be recognized.
Check both the pathname and the access rights on your designated **PASSWD** file. Stop and restart NFS on PRIMOS Systems. If the error persists, contact Customer Service.

Unable to open temporary PASSWD file.
There will be no uid/username database. No client uids will be recognized.
Follow the same corrective measures as above. Also check that there is write access to **NFS_>ETC** (needed to create a temporary file).

Unable to open GROUP_file_pathname.
There will be no gid/groupname database. No client gids will be recognized.
Check both the pathname and the access rights on your designated **GROUP** file. Stop and restart NFS on PRIMOS Systems. If the error persists, contact Customer Service.

```
Unable to open temporary GROUP file.
There will be no gid/groupname database.  No client gids will be
recognized.
```
Follow the same corrective measures as above.

```
Error while writing temporary PASSWD file.
There will be no uid/username database.  No client uids will be
recognized.
```
See the ensuing explanation.

```
Error while writing temporary GROUP file.
There will be no gid/groupname database.  No client gids will be
recognized.
```
One or more of these messages may be logged. User or internal error. Check for insufficient access rights for **NFS_PCNFSD** to the parent directories for **PASSWD** and **GROUP**. Check for disk full or quota exceeded.

```
Unable to open temporary PASSWD file to load the database.
There will be no uid/username database.  No client uids will be
recognized.
```
See the ensuing explanation.

```
Unable to open temporary GROUP file to load the database.
There will be no gid/groupname database.  No client gids will be
recognized.
```
One or more of these messages may be logged. User or internal error. Check for insufficient access rights for **NFS_PCNFSD** to the parent directories for **PASSWD** and **GROUP**. Check for disk full or quota exceeded.

```
Unable to add entry to name_to_id database for GROUP file.
There will be no gid/groupname database.  No client gids will be
recognized.
```
See the ensuing explanation.

```
Unable to add entry to name_to_id database for PASSWD file.
There will be no uid/username database.  No client uids will be
recognized.
```
One or more of these messages may be logged. Internal error produced by the server's internal cache management. This may indicate insufficient dynamic memory allocated to the process executing the server.

Invalid entry in *PASSWD_file_pathname*.
Ignoring line: *text_of_problem_line*.
See the ensuing explanation.

Invalid entry in *GROUP_file_pathname*.
Ignoring line: *text_of_problem_line*.
See the ensuing explanation.

Error while parsing *PASSWD_file_pathname*.
At line: *line_where_error_occurred*.
There will be no uid/username database. No client uids will be recognized.
See the ensuing explanation.

Error while parsing *GROUP_file_pathname*.
At line: *line_where_error_occurred*.
There will be no gid/groupname database. No client gids will be recognized.

One or more of these messages may be logged. User error, produced when processing the **PASSWD** and **GROUP** files. Duplicate entries are ignored; the information associated with the first occurrence of the duplicate pair is retained by the server. Invalid entries are ignored. If the **PASSWD** file is empty, then all clients using the PRIMOS NFS server are treated as having the access rights of **$REST**.

There are no valid entries in *PASSWD_file_pathname*.
There will be no uid/username database. No client uids will be recognized.

See the ensuing explanation.

There are no valid entries in *GROUP_file_pathname*.
There will be no gid/groupname database. No client gids will be recognized.

See the ensuing explanation.

Duplicate name encountered in PASSWD file: *duplicate_name_displayed*
Only the first instance of the name will be used.

See the ensuing explanation.

Duplicate name encountered in GROUP file: *duplicate_name_displayed*
Only the first instance of the name will be used.

One or more of these messages may be logged. User error, produced when processing the **PASSWD** and **GROUP** files. Duplicate entries are ignored; the information associated with the first occurrence of the duplicate pair is retained by the server. Invalid entries are ignored. If the **PASSWD** file is empty, then all clients using the PRIMOS NFS server are treated as having the access rights of **$REST**.

Unable to allocate name_to_id database for *PASSWD_file_pathname*.
There will be no uid/username database. No client uids will be recognized.

One or more of these messages may be logged. Indicates insufficient dynamic memory allocated to the process executing the server.

Unable to allocate name_to_id database for *GROUP_file_pathname*.
There will be no gid/groupname database. No client gids will be recognized.

Indicates insufficient dynamic memory allocated to the process executing the server.

Recorded During Operation of NFS_PCNFSD

Cannot convert (string) to a valid PRIMOS pathname.

User or internal error. Check the pathname of the mountpoint entered on the client NFS system.

Unable to allocate a buffer for pathname conversion.

Internal error. Report the problem to Customer Service.

svc_run() returned with a fatal error.
The NFS_PCNFSD server is terminating.
Internal error. The procedure that handles incoming RPC requests has failed. **NFS_PCNFSD** logs out after this message. Save the log and report the error to Customer Service.

Error while sending response to client for procedure *number*.
Internal error. Save the log and report the error to Customer Service.

Unable to decode the arguments for procedure *number*.
May have been user error. If it is a repeated error in the log, request that Customer Service enable a debugging trace.

The server received a request to perform unknown procedure *number*.
May have been user error. If it is a repeated error in the log, request that Customer Service enable a debugging trace.

The NFS_PCNFSD server just received a logout signal.
Status message.

The user authentication routine failed with error *number*.
The PC-NFS client was not authenticated.
Internal error. Save the log and report the error to Customer Service.

The password supplied by the PC-NFS client was too long.
The PC-NFS client was not authenticated.
This is a user error, since no password exceeding 32 characters can validly be entered in the SAD.

The spool directory, *SPOOL-directory_pathname*, was not created due to error *number*.
Error may be internal. First check existence and access rights for the specified directory. If they are valid, report the error to Customer Service.

The ACL on *SPOOL-directory_pathname* was not created due to error *number*.
Internal error. Save the log and report the error to Customer Service.

file_to_be_spooled is a null file and will not be spooled.
Status message indicating a user error.

Error *number* occurred while deleting *file_just_spooled*.
Internal error. Save the log and report the error to Customer Service.

PR_START was unable to find *file_to_be_spooled*.
Probable user error. If the file exists, report the problem to Customer Service.

Invalid entry in *GROUP_file_pathname*.
Ignoring line: *text_of_problem_line*.
See the ensuing explanation.

Error while parsing *PASSWD_file_pathname*.
At line: *line_where_error_occurred*.
There will be no uid/username database. No client uids will be recognized.
See the ensuing explanation.

Error while parsing *GROUP_file_pathname*.
At line: *line_where_error_occurred*.
There will be no gid/groupname database. No client gids will be recognized.
One or more of these messages may be logged. User error produced when processing the **PASSWD** and **GROUP** files. Duplicate entries are ignored; the information associated with the first occurrence of the duplicate pair is retained by the server. Invalid entries are ignored. If the **PASSWD** file is empty, then all clients using the PRIMOS NFS server are treated as having the access rights of **$REST**.

There are no valid entries in *PASSWD_file_pathname*.
There will be no uid/username database. No client uids will be recognized.
See the ensuing explanation.

There are no valid entries in *GROUP_file_pathname*.
There will be no gid/groupname database. No client gids will be recognized.
See the ensuing explanation.

Duplicate name encountered in PASSWD file: *duplicate_name_displayed*
Only the first instance of the name will be used.
See the ensuing explanation.

Duplicate name encountered in GROUP file: *duplicate_name_displayed*
Only the first instance of the name will be used.
One or more of these messages may be logged. User error produced when processing the **PASSWD** and **GROUP** files. Duplicate entries are ignored; the information associated with the first occurrence of the duplicate pair is retained by the server. Invalid entries are ignored. If the **PASSWD** file is empty, then all clients using the PRIMOS NFS server are treated as having the access rights of **$REST**.

Duplicate id encountered in PASSWD file: *duplicate_id_displayed*
Only the first instance of the id will be used.
See the ensuing explanation.

Duplicate id encountered in GROUP file: *duplicate_id_displayed*
Only the first instance of the id will be used.
See the ensuing explanation.

There are no valid entries in *PASSWD_file_pathname*.
There will be no uid/username database. No client uids will be recognized.
See the ensuing explanation.

There are no valid entries in *GROUP_file_pathname*.
There will be no gid/groupname database. No client gids will be recognized.

One or more of these messages may be logged. User error produced when processing the **PASSWD** and **GROUP** files. Duplicate entries are ignored; the information associated with the first occurrence of the duplicate pair is retained by the server. Invalid entries are ignored. If the **PASSWD** file is empty, then all clients using the PRIMOS NFS server are treated as having the access rights of **$REST**.

Unable to allocate id_to_name database for *PASSWD_file_pathname*.
There will be no uid/username database. No client uids will be recognized.

See the ensuing explanation.

Unable to allocate id_to_name database for *GROUP_file_pathname*.
There will be no gid/groupname database. No client gids will be recognized.

See the ensuing explanation.

Unable to allocate name_to_id database for *PASSWD_file_pathname*.
There will be no uid/username database. No client uids will be recognized.

One or more of these messages may be logged. Indicates insufficient dynamic memory allocated to the process executing the server.

Unable to allocate name_to_id database for *GROUP_file_pathname*.
There will be no gid/groupname database. No client gids will be recognized.

See the ensuing explanation.

Unable to allocate memory needed for NFSSTAT counters.
The NFS Server is terminating.

See the ensuing explanation.

Unable to allocate the buffer used to "pad" files during writes.
The NFS Server is terminating.

See the ensuing explanation.

Unable to allocate the read/write buffer.
The NFS Server is terminating.

One or more of these messages may be logged. Indicates insufficient dynamic memory allocated to the process executing the server.

Unable to get the local time-zone information.
The NFS Server is terminating.

User or internal error. Verify that the **SET_TIME_INFO** command was issued before starting NFS (see Chapter 5 for details). Report **STI** problems to Customer Service.

While setting the alarm time the NFS Server encountered error *nn*.

Internal error. Report the problem to Customer Service.

MESSAGES RECORDED IN NFS_SERVER_LOG

At Initialization of NFS_SERVER

The NFS_SERVER was spawned on *date_and_time*
First entry in log.

The NFS_SERVER is now ready for requests.
Server can now be contacted by clients.

Loading uid/username database from PASSWD file *pathname*
Loading gid/groupname database from GROUP file *pathname*
Whenever new information is loaded, this event and the names of the files used are logged.

Unable to open *PASSWD_file_pathname*
There will be no uid/username database. No client uids will be recognized.
See the ensuing explanation.

Unable to open *GROUP_file_pathname*.
There will be no gid/groupname database. No client gids will be recognized.
User or internal error. Check for insufficient access rights for **NFS_SERVER**, either to the file itself or to the parent directories for **PASSWD** and **GROUP**.

Unable to open temporary PASSWD file.
There will be no uid/username database. No client uids will be recognized.
See the ensuing explanation.

Unable to open temporary GROUP file.
There will be no gid/groupname database. No client gids will be recognized.
One or more of these messages may be logged. User or internal error. Check for insufficient access rights for **NFS_SERVER**, preventing the creation of the file in **NFS_MSG**. Check for disk full or quota exceeded.

Error while writing temporary PASSWD file.
There will be no uid/username database. No client uids will be recognized.
See the ensuing explanation.

Error while writing temporary GROUP file.
There will be no gid/groupname database. No client gids will be recognized.
One or more of these messages may be logged. User or internal error. Check for disk full or quota exceeded.

svc_register() error in nfsd.
Internal error. Stop and restart NFS on PRIMOS Systems. If the error persists, contact Customer Service.

```
Unable to open temporary PASSWD file to load the database.
There will be no uid/username database.  No client uids will be
recognized.
```
See the ensuing explanation.

```
Unable to open temporary GROUP file to load the database.
There will be no gid/groupname database.  No client gids will be
recognized.
```
One or more of these messages may be logged. User or internal error. Check for insufficient access rights for **NFS_SERVER** to the parent directories for **PASSWD** and **GROUP**. Check for disk full or quota exceeded.

```
Unable to add entry to name_to_id database for GROUP file.
There will be no gid/groupname database.  No client gids will be
recognized.
```
See the ensuing explanation.

```
Unable to add entry to name_to_id database for PASSWD file.
There will be no uid/username database.  No client uids will be
recognized.
```
One or more of these messages may be logged. Internal error produced by the server's internal cache management. May indicate insufficient dynamic memory allocated to the process executing the server.

```
Unable to add entry to id_to_name database for GROUP file.
There will be no gid/groupname database.  No client gids will be
recognized.
```
See the ensuing explanation.

```
Unable to add entry to id_to_name database for PASSWD file.
There will be no uid/username database.  No client uids will be
recognized.
```
One or more of these messages may be logged. Internal error produced by the server's internal cache management. May indicate insufficient dynamic memory allocated to the process executing the server.

```
A failure occurred while initializing controller connections.
The NFS Server is terminating.
```
User or internal error. Failures have occurred in the socket/UDP software. Ensure TCP/IP is running and operational.

```
Initialization of the filehandle cache failed.
The NFS Server is terminating.
```
Internal error. Report the problem to Customer Service.

```
Initialization of the filehandle cache overflow buffers failed.
The NFS Server is terminating.
```
Internal error. Report the problem to Customer Service.

Invalid entry in *PASSWD_file_pathname*.
Ignoring line: *text_of_problem_line*.
See the ensuing explanation.

Recorded During Operation of NFS_SERVER

The cache aging mechanism encountered error *number* while closing *filename*.

An error occurred when the NFS server tried to close a file. There may have been some file contention between a PRIMOS user and an NFS user.

Error *number* when merging a temporary cache entry into the filehandle cache.

Internal server error during cache management. Report the error to Customer Service.

svc_run() returned with a fatal error.
The NFS Server is terminating.
The cache aging mechanism encountered error *number* while closing filename.

An error occurred when the NFS server tried to close a file. There may have beeen some file contention between a PRIMOS user and an NFS user.

Error *number* when merging a temporary cache entry into the filehandle cache.

Internal server error during cache management. Report the error to Customer Service.

svc_run() returned with a fatal error.
The NFS Server is terminating.

Internal error. The procedure that handles incoming RPC requests has failed. **NFS_SERVER** logs out after this message. Report the error to Customer Service.

Received a request for which the authentication data could not be decoded. The client request was ignored.

See the ensuing explanation.

Received a request for which the arguments could not be decoded.
The client request was ignored.

One of the above messages is recorded in the log file. Indicates the server received client request packets that were improperly formatted.

An error occurred while transmitting a response to a client request.

Internal server error in sending replies to client, usually a severe problem with the server or TCP/IP software. Save the log, and request Customer Service assistance in enabling a debugging trace of the server to pinpoint the events leading to the error.

START_NFS ERROR MESSAGES

These messages are displayed at the supervisor terminal at startup. Most are also logged. Most of the messages require no explanation.

```
This is not an IX-mode machine.  NFS requires IX-mode.
```

```
This command is usable only from the system console!
```

```
Cannot open START_NFS's message file.
START_NFS is terminating.
```
See the ensuing explanation.

```
Cannot read START_NFS's message file.
START_NFS is terminating.
```
See the ensuing explanation.

```
Cannot allocate memory for START_NFS's message file.
START_NFS is terminating.
```
See the ensuing explanation.

```
Fatal error initializing START_NFS's message file.
START_NFS is terminating.
```
For the first two error messages given above, ensure that the file **NFS_>MESSAGE_TEXT>START_STOP_MSG** exists and that the user attempting to execute the command has **READ** access to the file. The last two of the above indicate that the user attempting to execute the command may not have enough dynamic memory segments.

```
The -PASSWDFILE (-PFILE) option was specified but no pathname
was provided.
```

```
The specified PASSWD file, (pathname), was not found.
```

```
The -SPOOLDIR (-SDIR) option was specified but no pathname
was provided.
```

```
The specified SPOOL directory, (pathname), was not found.
```

```
The -GROUPFILE (-GFILE) option was specified but no pathname
was provided.
```

```
The specified GROUP file, (pathname), was not found.
```

```
The cache size must be between 0 and 3000.
```

```
The cache aging interval must be between 3 and 300 seconds.
```

```
The -FILETYPE (-FTYPE) option was specified a filetype indicator.
```

```
The -FILETYPE (-FTYPE) option was specified with an invalid
filetype - (filetype)
```

The -ACLGROUP (-AGRP) option was specified without any ACL groups.

pathname is not a legal group name.

TCP/IP is not running. Please download the controller and start it.

NFS servers are already running. Run STOP_NFS before you restart.

Archiving NFS_>COMI>@@.COMO to NFS_>COMI>ARCHIVE>== failed.

NFS_>COMI>@@.COMO files archived at NFS_>COMI>ARCHIVE.

Archiving NFS_>LOG_FILES>@@LOG to NFS_>LOG_FILES>ARCHIVE>== failed.

NFS_>LOG_FILES files archived at NFS_>LOG_FILES>ARCHIVE.

The attempt to spawn the NFS_PORTMAP server failed with error *nn*

The attempt to spawn the NFS_MOUNT server failed with error *nn*

The attempt to spawn the NFS_PCNFSD server failed with error *nn*

The attempt to spawn the NFS_SERVER server failed with error *nn*

This version of PRIMOS will not support NFS. NFS requires Rev. 22.0 or later.

TCP/IP is not installed on this system. NFS cannot run without TCP/IP.

NFS_PORTMAP server successfully spawned.
NFS_MOUNT server successfully spawned.
NFS_PCNFSD server successfully spawned.
NFS_SERVER server successfully spawned.

These messages should be displayed and logged by **START_NFS** to indicate that NFS on PRIMOS Systems has been successfully started. The message for **NFS_PCNFSD** is optional, depending on whether it is being activated.

NFS_INIT_USERS MESSAGES

These messages are displayed at the supervisor terminal at the time of the NFS update. Many of the messages require no explanation.

```
Cannot open NFS_INIT_USERS's message file.
NFS_INIT_USERS is terminating.
```

```
Cannot read NFS_INIT_USERS's message file.
NFS_INIT_USERS is terminating.
```

```
Cannot allocate memory for NFS_INIT_USERS's message file.
NFS_INIT_USERS is terminating.
```

```
Fatal error initializing NFS_INIT_USERS's message file.
NFS_INIT_USERS is terminating.
```

For the first two of the error messages given above, ensure that the file **NFS_>MESSAGE_TEXT>GEN_MSG** exists and that the user attempting to execute the command has **READ** access to the file. The last two messages indicate that the user attempting to execute the command may not have enough dynamic memory segments.

```
This command is usable only from the system console!  (NFS_INIT_USERS)
```

Self-explanatory.

```
A failure occurred while initializing controller connections.
Make sure TCP/IP has been started, ICE, and try again.
```

```
Unable to get this system's host address.
```

Indicates that there was a problem looking up the system's Internet address in the **TCP/IP*>HOSTS** file. Verify that the file exists and that the user executing the command has **READ** acces to the file.

```
Unable to get NFS_SERVER's port number from the portmapper.
```

See the ensuing explanation.

```
Unable to get NFS_PCNFSD's port number from the portmapper.
```

The server is not registered with the portmapper. The server may have failed, or been logged out, or had its startup suppressed with the **-NOPC** option. The portmapper itself may also not be up. Stop and restart NFS. If the problem persists, contact Customer Service.

```
Unable to allocate resources for communications with NFS_SERVER.
Unable to allocate resources for communications with NFS_PCNFSD.
```

One or both of these messages may be logged, along with preceding messages. They indicate that a low-level communications (TCP/IP socket library routine) failure occurred. The TCP/IP log files may contain other information which is useful in pinpointing this problem. Contact Customer Service.

```
RPC: failure_message
Unable to contact NFS_SERVER to reinitialize its user information.
```

See the ensuing explanation.

RPC: *failure_message*
Unable to contact NFS_PCNFSD to reinitialize its user information.
One or both of the above messages may be logged. They indicate that a low-level communications (TCP/IP socket library routine) failure occurred. The TCP/IP log files may contain other information which is useful in pinpointing this problem. Contact Customer Service.

NFS_SERVER's user information reinitialized.
NFS_INIT_USERS was successful for **NFS_SERVER**.

NFS_PCNFSD's user information reinitialized.
NFS_INIT_USERS was successful for **NFS_PCNFSD**.

NFS_SERVER's user information not reinitialized.
See server log file for more information.
See the ensuing explanation.

NFS_PCNFSD's user information not reinitialized.
See server log file for more information.
Server could not reinitialize user information. Possibly nonexistent (or no access to) **PASSWD** and/or **GROUP** files. Server log files contain more details on why the attempt failed.

STOP_NFS MESSAGES

All messages are logged to the standard output device, which is usually the supervisor terminal. Most messages require no explanation.

This is not an IX-mode machine. NFS requires IX-mode.

Cannot open STOP_NFS's message file.
STOP_NFS is terminating.
See the ensuing explanation.

Cannot read STOP_NFS's message file.
STOP_NFS is terminating.
See the ensuing explanation.

Cannot allocate memory for STOP_NFS's message file.
STOP_NFS is terminating.
See the ensuing explanation.

Fatal error initializing STOP_NFS's message file.
STOP_NFS is terminating.
For the first two of these error messages, ensure that the file **NFS_>MESSAGE_TEXT>START_STOP_MSG** exists and that the user attempting to execute the command has **READ** access to this file. This and the former message indicate that the user attempting to execute the command may not have enough dynamic memory segments.

This command is usable only from the system console!

The attempt to logout the NFS_PORTMAP server failed with error *number*.
See the ensuing explanation.

The attempt to logout the NFS_MOUNT server failed with error *number*.
See the ensuing explanation.

The attempt to logout the NFS_PCNFSD server failed with error *number*.
See the ensuing explanation.

The attempt to logout the NFS_SERVER server failed with error *number*.
An internal error occurred while attempting to log out the indicated server. Retry **STOP_NFS** to see if the server can be successfully logged out. If this error persists, contact Customer Service.

The NFS_PORTMAP phantom does not exist.
The NFS_MOUNT phantom does not exist.
The NFS_PCNFSD phantom does not exist.
The NFS_SERVER phantom does not exist.
One or more of these messages may be logged. The indicated server was already logged out. This logout may have occurred because **STOP_NFS** was previously executed, because the server was never initialized, or because of a fatal error encountered by the server. Check the indicated server's log file for further information.

Error in obtaining NFS_PORTMAP phantom number.
Error in obtaining NFS_MOUNT phantom number.
Error in obtaining NFS_PCNFSD phantom number.
Error in obtaining NFS_SERVER phantom number.
One or more of these messages may be logged. An internal error occurred while attempting to log out the indicated server. Execute **STAT US** to see if the indicated server is still logged in. If so, retry **STOP_NFS** to see if the server can be successfully logged out. If the error persists, contact Customer Service.

NFS_PORTMAP successfully logged out.
NFS_MOUNT successfully logged out.
NFS_PCNFSD successfully logged out.
NFS_SERVER successfully logged out.
These messages are logged by **STOP_NFS** to indicate that there were no problems during the termination of NFS on PRIMOS Systems. (If **NFS_PCNFSD** had not been started, its termination message will not be issued.)

NFSSTAT MESSAGES

These messages are all logged to the standard output device, normally the terminal of the user making the request.

```
Cannot open NFSSTAT's message file.
NFSSTAT is terminating.
```
See the ensuing explanation.

```
Cannot read NFSSTAT's message file.
NFSSTAT is terminating.
```
See the ensuing explanation.

```
Cannot allocate memory for NFSSTAT's message file.
NFSSTAT is terminating.
```
See the ensuing explanation.

```
Fatal error initializing NFSSTAT's message file.
NFSSTAT is terminating.
```
For the first two of these error messages, ensure that the file **NFS_>MESSAGE_TEXT>GEN_MSG** exists and that the user attempting to execute the command has **READ** access to the file. The last two messages indicate that the user attempting to execute the command may not have enough dynamic memory segments.

```
A failure occurred while initializing controller connections.
Make sure TCP/IP has been started, ICE, and try again.
```

```
Unable to get this system's host address.
```
Indicates that there was a problem looking up the system's Internet address in the **TCP/IP*>HOSTS** file. Verify that the file exists and that the user executing the command has **READ** access to the file.

```
Unable to allocate resources for communications with NFS_SERVER.
Unable to contact server to get statistics.
Unable to contact server to initialize NFSSTAT counts.
```
One or more of these messages may be logged. They indicate that a low-level communications (TCP/IP socket library routine) failure occurred. The TCP/IP log files may contain other information which is useful in pinpointing this problem. Contact Customer Service.

RPCINFO MESSAGES

Cannot open RPCINFO's message file.
RPCINFO is terminating.
See the ensuing explanation.

Cannot read RPCINFO's message file.
RPCINFO is terminating.
See the ensuing explanation.

Cannot allocate memory for RPCINFO's message file.
RPCINFO is terminating.
See the ensuing explanation.

Fatal error initializing RPCINFO's message file.
RPCINFO is terminating.
For the first two of these error messages, ensure that the file **NFS_>MESSAGE_TEXT>GEN_MSG** exists and that the user attempting to execute the command has **READ** access to the file. The last two messages indicate that the user attempting to execute the command may not have enough dynamic memory segments.

A failure occurred while initializing controller connections.
Make sure TCP/IP has been started, ICE, and try again.

Unable to get this system's host address.
Unable to get *hostname*'s host address.
One or both of these messages may be logged. Indicates that there was a problem looking up the system's Internet address in the **TCP/IP*>HOSTS** file. Verify that the file exists and that the user executing the command has **READ** access to the file.

program_name is an unknown service.
The caller probably specified an argument to the program that was a name (instead of a number), and it was not recognized. The caller must have access to the **NFS_>ETC>RPC** file for a translation of program names to program numbers.

Unable to allocate resources for communications with service.
Indicates that a low-level communications (TCP/IP socket library routine) failure occurred. The TCP/IP log files may contain other information which is useful in pinpointing this problem. Contact Customer Service.

RPC: *failure_message*
Unable to contact portmapper on *host*.
Request to probe the portmapper failed. First verify that portmapper is running properly on the remote host, then retry. If the problem persists, contact Customer Service.

RPC: *failure_message*
program *pr_number* is not available.
See the ensuing explanation.

RPC: *failure_message*
program *pr_number* version *v_number* is not available.
The *failure_message* usually indicates the problem source. Refer to Chapter 6 for a diagnosis of some problems and solutions. If the problem persists, contact Customer Service.

PTOU MESSAGES

All messages are logged to the standard output device, normally the terminal of the user issuing the command. Most messages need no explanation.

`Cannot open PTOU's message file.`
`PTOU is terminating.`
See the ensuing explanation.

`Cannot read PTOU's message file.`
`PTOU is terminating.`
See the ensuing explanation.

`Cannot allocate memory for PTOU's message file.`
`PTOU is terminating.`
See the ensuing explanation.

`Fatal error initializing PTOU's message file.`
`PTOU is terminating.`
For the first two of these error messages, ensure that the file **NFS_>MESSAGE_TEXT>GEN_MSG** exists and that the user attempting to execute the command has **READ** access to the file. The last two messages indicate that the user attempting to execute the command may not have enough dynamic memory segments.

`Unable to open the input file:` *pathname*.
`Unable to open the output file:` *pathname*.
One of these errors may be logged. Ensure that the user executing the command has **READ** access to the input file and **WRITE** access to the output file, and also **ADD** rights to the parent directory of the output file if the output file does not already exist.

`Could not open temporary file.`
This message will only be logged if the user is attempting to modify a file in place. Ensure that the user executing the command has **ADD** rights to the directory containing the source or target file.

`Unable to obtain parent directory of destination file.`
`Could not attach to parent directory of destination file.`
One of these messages may be logged. Ensure that the user executing the command has **USE** rights to the parent directory of the destination file.

`Encountered error while reading` *pathname*.
`Encountered error while writing into the output file:` *pathname*.
`Encountered error while writing into the temporary file.`
`Unable to read the destination directory entry.`
`A failure occurred when attempting to set the odd-byte file attribute.`
One of these messages may be logged. Ensure that the user executing the command has **PROTECT** rights on the parent directory of the destination file.

Could not process file. File possibly not in PRIMOS format.

Encountered error while deleting the source file: *pathname*.

Encountered error while changing the name of the destination file.

Encountered error while returning to the home directory.

UTOP MESSAGES

All messages are logged to the standard output device, normally the terminal of the user issuing the command. Most messages need no explanation.

Cannot open UTOP's message file.
UTOP is terminating.
See the ensuing explanation.

Cannot read UTOP's message file.
UTOP is terminating.
See the ensuing explanation.

Cannot allocate memory for UTOP's message file.
UTOP is terminating.
See the ensuing explanation.

Fatal error initializing UTOP's message file.
UTOP is terminating.
For the first two of these error messages, ensure that the file **NFS_>MESSAGE_TEXT>GEN_MSG** exists and that the user attempting to execute the command has **READ** access to the file. The last two messages indicate that the user attempting to execute the command may not have enough dynamic memory segments.

Invalid argument: *bad_argument*.

The -TAB option was specified without also specifying a tab-size.

The tab-size contained an invalid character: *bad_character*.

invalid_size exceeds the maximum tab size of 256.

The tab size is set to *value*.
The tab size is set to (default) 8.
One of these messages is always logged to indicate the tab value used in the destination file. If the user did not specify a tab value, it indicates that the default was used.

Unable to open the input file: *pathname*.
Unable to open the output file: *pathname*.
One of these errors may be logged. Ensure that the user executing the command has **READ** access to the input file and **WRITE** access to the output file, and also **ADD** rights to the parent directory of the output file if the output file does not already exist.

`Could not open temporary file`

This message will only be logged if the user is attempting to modify a file in place. Ensure that the user executing the command has **ADD** rights to the directory containing the source or target file.

`Unable to obtain parent directory of destination file.`
`Could not attach to parent directory of destination file.`

One of these messages may be logged. Ensure that the user executing the command has **USE** rights to the parent directory of the destination file.

`Encountered error while reading` *pathname*.
`Encountered error while writing into the output file:` *pathname*.
`Encountered error while writing into the temporary file.`
`Encountered error while deleting the source file:` *pathname*.
`Encountered error while changing the name of the destination file.`
`Encountered error while returning to the home directory.`

One of these messages may be logged. Ensure that the user executing the command has **PROTECT** rights on the parent directory of the destination file.

DOSTOP MESSAGES

All messages are logged to the standard output device, normally the terminal of the user issuing the command. Most messages need no explanation.

```
Cannot open DOSTOP's message file.
DOSTOP is terminating.
```
See the ensuing explanation.

```
Cannot read DOSTOP's message file.
DOSTOP is terminating.
```
See the ensuing explanation.

```
Cannot allocate memory for DOSTOP's message file.
DOSTOP is terminating.
```
See the ensuing explanation.

```
Fatal error initializing DOSTOP's message file.
DOSTOP is terminating.
```
For the first two of the above messages, ensure that the file **NFS_>MESSAGE_TEXT>GEN_MSG** exists and that the user attempting to execute the command has **READ** access to the file. The last two messages indicate that the user attempting to execute the command may not have enough dynamic memory segments.

Invalid argument: *bad_argument*.

The -TAB option was specified without also specifying a tab-size.

The tab-size contained an invalid character: *bad_character*.

invalid_size exceeds the maximum tab size of 256.

The tab size is set to *value*.
The tab size is set to (default) 8.

One of these messages is always logged to indicate the tab value used in the destination file. If the user did not specify a tab value, it indicates that the default was used.

Unable to open the input file: *pathname*.
Unable to open the output file: *pathname*.

One of these errors may be logged. Ensure that the user executing the command has **READ** access to the input file and **WRITE** access to the output file, and also **ADD** rights to the parent directory of the output file if the output file does not already exist.

Could not open temporary file

This message will only be logged if the user is attempting to modify a file in place. Ensure that the user executing the command has **ADD** rights to the directory containing the source or target file.

Unable to obtain parent directory of destination file.
Could not attach to parent directory of destination file.

One of these messages may be logged. Ensure that the user executing the command has **USE** rights to the parent directory of the destination file.

Encountered error while reading *pathname*.
Encountered error while writing into the output file: *pathname*.
Encountered error while writing into the temporary file.
Encountered error while deleting the source file: *pathname*.
Encountered error while changing the name of the destination file.
Encountered error while returning to the home directory.

One of these messages may be logged. Ensure that the user executing the command has **PROTECT** rights on the parent directory of the destination file.

PTODOS MESSAGES

All messages are logged to the standard output device, normally the terminal of the user issuing the command. Many messages need no explanation.

Cannot open PTODOS's message file.
PTODOS is terminating.

See the ensuing explanation.

Cannot read PTODOS's message file.
PTODOS is terminating.

See the ensuing explanation.

Cannot allocate memory for PTODOS's message file.
PTODOS is terminating.

See the ensuing explanation.

Fatal error initializing PTODOS's message file.
PTODOS is terminating.

For the first two of the above messages, ensure that the file **NFS_>MESSAGE_TEXT>GEN_MSG** exists and that the user attempting to execute the command has **READ** access to the file. The last two messages indicate that the user attempting to execute the command may not have enough dynamic memory segments.

Unable to open the input file: *pathname*.
Unable to open the output file: *pathname*.

One of these errors may be logged. Ensure that the user executing the command has **READ** access to the input file and **WRITE** access to the output file, and also **ADD** rights to the parent directory of the output file if the output file does not already exist.

Could not open temporary file

This message will only be logged if the user is attempting to modify a file in place. Ensure that the user executing the command has **ADD** rights to the directory containing the source or target file.

Unable to obtain parent directory of destination file.
Could not attach to parent directory of destination file.

One of these messages may be logged. Ensure that the user executing the command has **USE** rights to the parent directory of the destination file.

Encountered error while reading *pathname*.
Encountered error while writing into the output file: *pathname*.
Encountered error while writing into the temporary file.
Unable to read the destination directory entry.
A failure occurred when attempting to set the odd-byte file attribute.
One of these messages may be logged. Ensure that the user executing the command has **PROTECT** rights on the parent directory of the destination file.

Could not process file. File possibly not in PRIMOS format.

Encountered error while deleting the source file: *pathname*.

Encountered error while changing the name of the destination file.

Encountered error while returning to the home directory.

D

FILENAME CHARACTER MAPPING FOR NFS ON PRIMOS SYSTEMS

NFS on PRIMOS Systems needs to map filename characters from their BSD-based or SVID-compliant UNIX formats to their PRIMOS formats. For this purpose, NFS on PRIMOS Systems has adopted the mapping used for PRIMIX™ filename characters and their PRIMOS equivalents. Several UNIX characters, including all uppercase alphabetic characters, translate to two PRIMOS characters (for example, A = /A).

Table D-1 shows the character mappings between UNIX and PRIMOS.

NFS on PRIMOS Systems also needs to map filename characters between DOS-based formats and PRIMOS formats, for which additional character sequences are reserved. Table D-2 shows the additional character mappings between DOS and PRIMOS.

TABLE D-1. Mapping UNIX to PRIMOS

Printing Characters *Left_Entry (UNIX) = Right_Entry (PRIMOS)*			
A = /A	N = /N	a = A	n = N
B = /B	O = /O	b = B	o = O
C = /C	P = /P	c = C	p = P
D = /D	Q = /Q	d = D	q = Q
E = /E	R = /R	e = E	r = R
F = /F	S = /S	f = F	s = S
G = /G	T = /T	g = G	t = T
H = /H	I = /U	h = H	u = U
I = /I	V = /V	i = I	v = V
J = /J	W = /W	j = J	w = W
K = /K	X = /X	k = K	x = X
L = /L	Y = /Y	l = L	y = Y
M = /M	Z = /Z	m = M	z = Z
	0 = 0[1]	5 = 5	
	1 = 1	6 = 6	
	2 = 2	7 = 7	
	3 = 3	8 = 8	
	4 = 4	9 = 9	

[1]**If any digit (0 - 9) begins a UNIX filename, it maps to that digit preceded by the character sequence /& in the PRIMOS filename. The sequence /& is illegal in any other position within a PRIMOS filename. For example, the UNIX filename 24theshow maps to the PRIMOS filename /&24THESHOW.**

TABLE D-1 (continued). Mapping UNIX to PRIMOS

Special Characters *Left_Entry (UNIX) = Right_Entry (PRIMOS)*	
* = *	, = /4
# = #	: = /5
$ = $	; = /6
- = -	< = /7
. = .	= = /8
_ = _	> = /9
/ = Illegal operation[2]	? = /_
\ = /*	@ = /#
! = &6	^ = /.
" = &7	` = &#
% = &8	[= /$
& = &9	] = /-
' = /0	{ = &$
(= /1	} = &-
) = /2	| = &*
+ = /3	~ = &.

[2] **A / character cannot be used within a UNIX filename. Therefore it has no mapping to characters within a PRIMOS filename.**

TABLE D-1 (continued). Mapping UNIX to PRIMOS

Nonprinting Characters *Left_Entry (UNIX) = Right_Entry (PRIMOS)*	
del = &_	del = &P
nul = nul[3]	dc1 = &Q
soh = &A	dc2 = &R
stx = &B	dc3 = &S
etx = &C	dc4 = &T
eot = &D	nak = &U
enq = &E	syn = &V
ack = &F	etb = &W
bel = &G	can = &X
bs = &H	em = &Y
ht = &I	sub = &Z
nl = &J	esc = &0
vt = &K	fs = &1
np = &L	gs = &2
cr = &M	rs = &3
so = &N	us = &4
si = &O	sp = &5

[3]**The null character (nul) is illegal for both systems.**

PRIMOS MAPPINGS FOR 8-BIT CHARACTERS FROM PC-NFS

PRIMOS reserves two special character sequences to enable the conversion of 8-bit characters from PC-NFS: // and &&. Each 8-bit character maps to some sequence involving one of these two. The sequence is then embedded in a valid PRIMOS pathname.

Whenever NFS on PRIMOS receives a character from any client, it

- Verifies whether the eighth-bit in the character is set.
- If the eighth bit is set, PRIMOS identifies the ASCII value in the first seven bits.
    - o If it is the value for the special character (/), PRIMOS records the character sequence **&&1**.
    - o If it is the value for the nonprinting null character (nul), then PRIMOS records the character sequence **&&0**.
    - o If it is the value for any other valid ASCII character, then PRIMOS records the character sequence // before the character associated with those seven bits.
- If the eighth bit is not set, then PRIMOS immediately records the character associated with those seven bits. (See the values in Table D-1).

Note

In this way the 7-bit (/) slash character remains illegal from a UNIX NFS client, but the 8-bit character (set-8th-bit plus slash) is legal from a PC-NFS client. Likewise, the 7-bit null character remains illegal, but the 8-bit character (set-8th-bit plus nul) can be read from a PC-NFS client and converted into a legal PRIMOS character sequence.

Table D-2 illustrates how an 8-bit character from PC-NFS is mapped to a character sequence on PRIMOS.

ILLEGAL FILENAME CONVERSIONS

Problems in filename conversions within NFS on PRIMOS Systems can occur because a

- UNIX pathname can have more characters than a PRIMOS pathname
- Pathname generated by an NFS client (either UNIX or DOS) may be converted into an expanded pathname too large for PRIMOS

Filename lengths vary among UNIX implementations. PRIMOS filenames may be 32 characters long. If a problem occurs, it is more likely to be an issue with converting a pathname rather than a filename. UNIX pathnames can be longer than PRIMOS pathnames, which may be 128 characters long. Likewise, DOS filenames are normally quite short, but conversion of full pathnames may cause a problem, as described in the next paragraph. See the previous section and Table D-2 for DOS/PRIMOS conversion details. UNIX filenames that are themselves longer than the character limits for PRIMOS are immediately rejected, with an error message to indicate that PRIMOS cannot understand the name.

TABLE D-2. Mapping DOS to PRIMOS

PC-NFS Character With Eighth Bit Set	*Mapped to PRIMOS Character Sequence*
Slash character (/)	&&1
Null character (nul)	&&0
All other PC-NFS characters with eighth bit set (examples below):	Characters as shown in Table D-1 but preceded by //:
A	///A
t	//T
O	///O
z	//Z
(	///1
0	//0
-	//-
9	//9
)	///2

A problem may occur when a client pathname with a length acceptable on the client system is translated into a longer PRIMOS pathname. For example, the client NFS user has successfully created various subdirectories on the PRIMOS server, the names of which have been translated into a PRIMOS pathname just short of 128 characters. While connected to the lowest subdirectory, the client user creates a file successfully and finishes a session. No problem has occurred; however, now the full pathname is greater than 128 characters. At some later time, the NFS client user tries to access the file, using the full pathname. The attempt fails. What the user sees depends upon the client operating system. An NFS user on a Sun Workstation receives an error message similar to `Server can't decode arguments`. The NFS user on an Apollo system sees `Invalid argument`.

MAPPING PRIMOS FILENAMES INTO UNIX FILENAMES

Table D-1 shows how a UNIX name is translated into a PRIMOS name. Read the table in reverse to see how PRIMOS names map to UNIX names. Most mapping rules can be summarized as follows:

- Letters map to lowercase letters.
- Digits and all printable special characters (except for / and &) map to themselves.
- Both / and & normally start a two-character sequence. The character / followed by a letter maps to that uppercase letter. See Table D-1 for special characters and mappings for the other sequences.
- The sequence /& may only begin the PRIMOS filename and be followed by a digit; it maps to that digit alone within the UNIX filename.
- The sequence // indicates the high-order bit set on the following character, which itself might require a two-character sequence in the PRIMOS filename.

Note

Many UNIX systems either ignore the high-order bit or treat it as an unprintable character. A PRIMOS filename (probably created by a PC-NFS client) may include the sequence // to indicate a character with its eighth bit set. This file may be unreachable from such a UNIX system, since the NFS client user cannot enter the actual name of the file.

UNTRANSLATABLE UNIX CHARACTER SEQUENCES

Users on a PRIMOS system can create filenames that include the following character sequences, but NFS on PRIMOS Systems cannot translate them so that a UNIX system can read them. All PRIMOS filenames containing these sequences are mapped to the same four-character filename shown below:

&& &/ /&	}	**?!

Notes

- && is translatable in the sequences &&0 and &&1 if the UNIX system can handle 8-bit characters.
- /& is translatable when it starts a name and is followed by a digit.

For example, a PRIMOS directory contains the following files:

```
/&WALDO     HELLO/WORLD     TEST//FILE     T&MRUN
```

A particular UNIX system cannot recognize 8-bit characters. A user on this system is a client of NFS on PRIMOS Systems. The user sees the contents of this PRIMOS directory as

```
helloWorld    **?!    testfile    t?run
```

The UNIX client user can see `testfile` but cannot access it, because this UNIX system cannot give its correct filename, which has the eighth bit set in the character `f` (see the preceding note). The UNIX client user also sees the untranslatable filename `**?`. Once again, the user cannot access the file because the system cannot provide a name that translates into its PRIMOS-resident name, `/&WALDO`. Even the command `ls -l` returns an error message indicating `**?! not found` before the other entries are displayed.

Note

The sequence `&M` in the PRIMOS filename `T&MRUN` maps to a Control-M. The client host displays a filename `t?run`, listing for the character ? whatever that UNIX system displays for a Control-M. (Many UNIX systems do display the ? character.)

E

RPC PROGRAMMING FOR PRIMOS SYSTEMS

CONTENTS OF RELEASE 1.2

As of Release 1.2 for NFS on PRIMOS Systems, you can design and execute your own programs that make use of the ONC™ RPC library. Release 1.2 provides

- The RPC library (RPCLIB). Programmers of network applications access the library through standard ONC RPC calls.
- The RPCGEN utility, an RPC protocol compiler that generates C code to implement the RPC protocol. Using RPCGEN the programmer can develop networked programs without the need to address low-level network connection issues in network calls; high-level protocol specifications are directed through RPCGEN, which generates the low-level calls.
- An online sample program, RUSERS, in both executable form and in source code. You can install the executable form to test RPC immediately. You can also study the source code, examining how the Prime-specific procedures are integrated with the other code, and then practicing with RPCGEN to generate C code output, which you then C-compile and Bind.
- Additional Prime-specific procedures needed to support programmer access to Prime's implementation of both TCP/IP and the RPC library.

This appendix describes both the RUSERS sample program and the Prime-specific procedures (the third and fourth bullets above).

The description of the sample program here touches only on the highlights of RPC programming, particularly the ease of use provided by the RPCGEN facility.

For complete information on the RPC library and RPCGEN (the first and second bullet), refer to the *ONC Network Programming Guide* (MAN13006-1LA). Prime distributes this manual, developed by Sun Microsystems.

WRITING RPC APPLICATIONS ON PRIMOS SYSTEMS

The RPC library allows any system supporting the library to make calls to any other system supporting the library. Each system can be either a client or server of services.

Release 1.2 of NFS on PRIMOS Systems makes the RPC library available to network programmers so that they can develop customized network applications providing access to or from remote systems. Each system can act as both client and server of a given service or operation. A system can also limit itself to one side of the operation, the client side or the server side.

A service may be a standard ONC service or an original service defined by a programmer. The RPC library allows development to be done without the overhead of building a low-level interface. The programmer makes high-level calls to routines in the RPC library to carry out low-level network procedures.

The RUSERS program is provided as a sample program. It is discussed here to illustrate the use of the RPC library, and most especially the RPCGEN utility. RUSERS is a standard ONC service, thoroughly discussed in the *ONC Network Programming Guide.*

SAMPLE PROGRAM

The RUSERS program enables a user on a client system to request a list of users on one or more remote systems. The program has both a client side and a server side. You can access this program online, after you have installed Release 1.2 of NFS on PRIMOS Systems. The program itself requires further installation, as described later in this chapter.

The sample program RUSERS is not a program supported by Prime Computer. It is a functional program, but it is provided mainly to illustrate Prime's version of a networked application using the ONC RPC library.

The components of the RUSERS program are located in two subdirectories below the directory NFS_:

- The subdirectory RPC_UTILS
- The subdirectory RPC_SAMPLES

The Subdirectory RPC_UTILS

This subdirectory holds the executable version of the program, ready for installation and activation. You are encouraged to add other executable programs to this directory as you develop them. Currently, the directory holds two RUN files and two CPL files.

- RUSERS.RUN is the client side of the RUSERS application. It makes a request for remote user information. A user invokes the program with the name of the targeted remote host as an optional argument:

  ```
  OK, RUSERS [hostname]
  ```

  If no host is specified, the request is broadcast to all nodes on the network. Each host with a server side of the RUSERS application attempts to respond with its user information.

- RUSERSD.RUN is the server side of the RUSERS application. This "daemon" is activated by phantoming the RUSERSD.CPL program. The runfile remains active as a background process, waiting for RUSERS requests from clients. When it encounters a request, it returns information about the users currently active on the system.

- RUSERSD.CPL is the program, executed as a phantom, to activate the daemon (RUSERSD.RUN). If your system is to service RUSERS requests, issue PHANTOM RUSERSD.CPL only *once* after a given PRIMOS boot. (Ensuing execution attempts simply fail without explanation.) If you are unsure whether the service is already active, issue the **RPCINFO -P** command to verify whether program number 100002 is registered with the **portmapper** on the local system.

- RUSERS.INSTALL.CPL copies the RUSERS.RUN program to CMDNC0 so that all users can call the client side of the program. The RUSERS.INSTALL.CPL also stages the server side of the program for activation; it copies both the RUSERSD.CPL file and the RUSERSD.RUN file into the directory NFS_>RUN. You may delete RUSERS.INSTALL.CPL after you execute it the first time.

Because this sample program is unsupported, the server (daemon) is not automatically activated each time NFS on PRIMOS Systems is started. If this 50 Series machine is to act as server for RUSERS requests, then the recommended time for starting the server is immediately after the START_NFS command with the command line:

```
OK, PHANTOM NFS_>RUN>RUSERSD.CPL
```

The Subdirectory RPC_SAMPLES

The subdirectory RPC_SAMPLES carries the source code components for RUSERS. You are encouraged to add other source code samples to this directory as you develop them.

The subdirectory currently holds the following, in the order of their use:

- RUSERS.C is the source code for the client side of this program on the 50 Series machine. It contains mostly setup and control information, as discussed in the next paragraph. This source code is combined with other RPC-related C modules to generate an information request about users on a *remote* machine.
- RUSERS.X is the protocol file to be submitted to the RPCGEN compiler. Called the remote program interface definition, this source is written in RPC language, which at first glance looks like C code. When you RPCGEN this file, your output is a collection of C source modules that have embedded within them the appropriate RPC calls. Details are provided later in The RUSERS.X File section.
- RUSERS.BUILD.CPL is the program that builds the runfiles for both the client side and the server side of this service. This CPL program processes the RUSERS.X file through the RPCGEN compiler, then it submits the appropriate C modules to the C compiler, and finally submits the resulting binary code to a BIND that creates the Run files. See the later section Running RUSERS.BUILD.CPL for further background information before you execute this BUILD.
- RUSERSD.C is the source code for the server side of this program on the 50 Series system. This code is one of the modules processed by the RUSERS.BUILD.CPL program.
- TIME_UTL.C and DIV_64.PMA are PRIMOS-specific code modules, not directly under discussion here. RUSERS.BUILD.CPL integrates them into the runfiles.

The following paragraphs describe the first three files listed above.

The RUSERS.C File: This module implements the main driver for the client side of the RUSERS service. The module contains important definitions of local and RPC-based structures, and it also makes calls to some important procedures:

- PRIMOS-specific calls to open and close the controller environment for RPC processing. The top part of Figure E-1 shows the RUSERS.C module's call to the procedure cntlr_open( ), with one of its arguments specifying RPC_CNTLR_MOST. RUSERS thereby requests to talk to all controllers, but the unavailability of one controller does not stop initialization of the others. See the full description of cntlr_open( ) later in this appendix.
- The procedure each_system( ), which basically martials all the information that RPC gathers from "each system" specified by the procedure. RUSERS may request information from all hosts (that is, make a broadcast), or it may request information from particular hosts. The bottom part of Figure E-1 shows the central code that martials information within each_system( ), omitting the coded IFs that branch between a broadcast call or a call to particular servers.

```
/* RUSERS.C, NFS>NFS_>RPC_SAMPLES, UNIX/DTI, 09/12/90
    .
    .
    .

    cntlr_open(0, RPC_CNTLR_MOST, (FILE * )0);

    .
    .
    .
/*
 * Function: each_system(): process the ruser data for a given system,
 *           which basically means to format and display the utmpidlearr
 *           structure as requested via the command line options.
 */
bool_t each_system(names, raddr)
    utmpidlearr *names;
    struct sockaddr_in *raddr;
{
    .
    .
    .
    if ((opts & OPT_A) || names->utmpidlearr_len > 0)
    {
        if (opts & OPT_L)
          .
          .
          .
        else
          .
          .
          .
        disp_sys(sys_buffer, rmt_name, names->utmpidlearr_len, 0);
    }
    return(0); /* Continue processing broadcast results */
}
```

FIGURE E-1. Excerpts From the RUSERS.C File

The RUSERS.X File: The RUSERS.X file contains the protocol definition of the RUSERS service, after a collection of C-language-styled definitions of structures. The definition of the actual protocol for RUSERS contains the three subdefinitions of

- program
- version
- procedure

Figure E-2 shows the entire definition of the protocol for RUSERS.

The program subdefinition for the protocol specifies both name and number for the program with the syntax:

```
program RUSERSPROG {
   .
   .
   .
} = 100002;
```

All requests and responses to the RUSERS service are identified with the program number of 100002.

The version subdefinitions, two of which exist for this program, follow a similar syntax. For example, the definition for the second version takes the form:

```
version RUSERSVERS_IDLE {
   .
   .
   .
} = 2;
```

The procedure subdefinitions are specified within each version. Each procedure defines one input and one output only. For example:

```
utmpidlearr
RUSERSPROC_NAMES(void) = 2
```

The above defines the RUSERSPROC_NAMES procedure as 2. The procedure takes a 'void' input and returns a 'utmpidlearr' output.

In summary, a call for RUSERS as the (program, version, procedure) of (100002, 2, 2) would execute the RUSERSPROC_NAMES procedure for the program RUSERSPROG, using the RUSERSVERS_IDLE version of the protocol.

Running RUSERS.BUILD.CPL: This program performs a multi-stepped procedure that finally creates both the client and server runfiles for the RUSERS service.

RUSERS.BUILD.CPL first submits RUSERS.X to the RPCGEN utility, resulting in four output files:

- rusers.h -- the header file that contains the PRIMOS-specific values discussed above, along with other values. The command line within the CPL program that creates this output is

  ```
  rpcgen rusers.x -h -o rusers.h
  ```

- rusers_svc.c -- the C source that is later compiled, with rusers.h included, to form the binary file for the server side of RUSERS, rusers_svc.bin. The command line within the CPL that creates this output, requesting use of the UDP protocol, is

  ```
  rpcgen rusers.x -s udp -o rusers_svc.c
  ```

```
/* RUSERS.X, NFS>NFS_>RPC_SAMPLES, UNIX/DTI, 09/12/90
/* Remote Users RPC protocol definition.

.
.
.

program RUSERSPROG {
    /*
     * Includes idle information
     */
    version RUSERSVERS_IDLE {
        int
        RUSERSPROC_NUM(void) = 1;

        utmpidlearr
        RUSERSPROC_NAMES(void) = 2;

        utmpidlearr
        RUSERSPROC_ALLNAMES(void) = 3;
    } = 2;

    /*
     * Old version does not include idle information
     */
    version RUSERSVERS_ORIG {
        int
        RUSERSPROC_NUM(void) = 1;

        utmparr
        RUSERSPROC_NAMES(void) = 2;

        utmparr
        RUSERSPROC_ALLNAMES(void) = 3;
    } = 1;
} = 100002;
```

FIGURE E-2. Excerpts From RUSERS.X

- rusers_clnt.c -- the C source that is later compiled, with rusers.h included, to form the binary file for the client side of RUSERS, rusers_clnt.bin. The command line within the CPL that creates this output is

  ```
  rpcgen rusers.x -l -o rusers_clnt.c
  ```

- rusers_xdr.c -- the C source that is later compiled, with rusers.h included, to form the binary file for the XDR procedures used in RUSERS, rusers_xdr.bin. The command line within the CPL that creates this output is

  ```
  rpcgen rusers.x -c -o rusers_xdr.c
  ```

RUSERS.BUILD.CPL next submits all C source files to the C compiler, both the C modules that were outputs from RPCGEN and those other C modules that were originally provided in the RPC_SAMPLES directory.

As the final step, RUSERS.BUILD.CPL submits the relevant binary files to two separate BIND sessions. The two BIND sessions generate RUSERS.RUN and RUSERSD.RUN. These runfiles are identical to those found in the directory RPC_UTILS, which was discussed earlier.

You can now verify that both the RPCGEN utility and the RPC library itself are functioning as a programming environment on your system. Execute RUSERS.BUILD.CPL. You can then substitute the newly-produced runfiles for those originally supplied in RPC_UTILS.

PRIME ROUTINES FOR RPC APPLICATIONS

Additional Prime-specific procedures support programmer access to the RPC library. These procedures manage

- Controller environment
- Authentication services

Procedures for Controller Environments

The RPC library has its access to the TCP/IP socket environment on PRIMOS initialized and terminated by two procedures, cntlr_open( ) and cntlr_close( ). Each of these is separately described on the following pages.

CNTLR_OPEN( )

This routine initializes the PRIMOS TCP/IP socket environment for subsequent use via the standard RPC library calls.

Usage:

```
int cntlr_open(so_privileged, open_mode, logfile)
int so_privileged;
int open_mode;
FILE *logfile;
```

Parameters:

so_privileged
INPUT. This argument corresponds to the first argument in the PRIMOS TCP/IP call, TUP$setup_socket_env. The argument determines whether or not privileged ports can be used. Argument values may be

0
Indicates that non-priveleged port requests are to be expected.

SO_PRIVILEGED
Defined in SOCKET.H. Indicates that privileged port requests can be made.

See the *PRIMOS TCP/IP Guide* for details on TUP$setup_socket_env.

open_mode
INPUT. Indicates the way to treat controllers. Values, defined in RPC.H, may be

RPC_CNTLR_ALL
All controllers are to be treated as one. For example, if at initialization one controller is down, all controllers are treated so. No services can be registered if any controller is down at the time of registration.

RPC_CNTLR_MOST
Each controller is treated separately. For example, if at initialization one controller is down, the socket environment tries to register the others. When the controller tries to come back up, the socket environment primarily maintains the stabilized controllers, and secondarily attempts to register the recovering controller.

logfile
INPUT. Indicates the file to which RPC informational and error messages are logged. As input, it is either a valid C file pointer, returned from a prior fopen() call, or a null pointer. When it is submitted, RPC logs to that file pointer, unless it is null. If the file pointer is null, RPC logs to STDERR, which defaults to the user terminal.

Discussion: This call must be issued prior to calling other TCP/IP gates or any of the standard ONC RPC gates. The call must be made by every application, whether client or server.

CNTLR_CLOSE( )

This routine provides an orderly close to all sockets opened for RPC calls on the controllers. The routine then closes the PRIMOS TCP/IP socket environment.

Usage:

```
void cntlr_close( )
```

Procedures for Authentication Services

Clients of NFS on PRIMOS Systems are provided all the UNIX-style authentication that normally comes with NFS. This authentication is provided by a mechanism on PRIMOS that maps PRIMOS user IDs to/from UNIX uids, and maps PRIMOS group IDs to/from UNIX gids. (See Chapter 3 of this manual for further details.) This mechanism is adequate for general NFS and PCNFS use. Additional routines are provided to enable RPC applications on PRIMOS to access this information. The additional routines are

Routine	*Functional Description*
getprmids( )	Provides the UNIX uids and gids, requests their corresponding PRIMOS user ID and group ID
getunxids( )	Provides the PRIMOS user IDs and group IDs, requests their corresponding UNIX uid and gid
opengetids( )	Informs the RPC library of upcoming multiple get-ID requests
closegetids( )	Informs the RPC library that there are no more multiple get-ID requests

Data structures are defined for passing the PRIMOS user/group IDs and the UNIX uids/gids to the RPC library mapping routines. These are defined in GETIDS.H.

Each routine is separately described on the following pages.

GETPRMIDS( )

With UNIX uids and gids as input, this routine finds each corresponding PRIMOS user ID and group ID. This function returns the information for one or more requests.

Usage:

```
returned_primos_id_info primos_id;
unix_id_info unix_id;
int status;
getprmids(unix_id, &primos_id);
status = getprmids(unix_id, &primos_id)
```

Parameters:

unix_id
INPUT. A structure containing the UNIX uids and gids to be looked up. This variable has the data type of the structure unix_id_info, which is defined in GETIDS.H.

&primos_id
OUTPUT. The address of the structure containing those PRIMOS user IDs and group IDs mapped as output from each UNIX uid and gid. The variable primos_id has the data type of the structure returned_primos_id which is defined in GETIDS.H.

status
RETURNED VALUE. This is the completion status returned by the function. It may have two possible values:

0
Indicates that the function executed to completion, even if the specified UNIX uid and/or gid could not be found in the PASSWD and GROUP files.

-1
Indicates that the lookup facility encountered an error. The global errno is set to nonzero.

Discussion: This function finds the PRIMOS information associated with the UNIX uids and gids that are provided. The information is found in the PASSWD and GROUP files that are located in the directory NFS_>ETC.

If a uid is not provided, the returned user ID has length 0. If a gid is not provided, the returned group ID has length 0. The returned group list contains entries only for the gids found in the database. If 10 gids are specified, but only 5 are in the GROUP file, then the returned group list contains only 5 entries, and the returned group_count is 5.

GETUNXIDS( )

With PRIMOS user IDs and group IDs as input, this routine finds each corresponding UNIX uid and gid.

Usage:

```
input_primos_id_info primos_id;
unix_id_info unix_id;
int status;
status = getunxids(primos_id, &unix_id);
```

Parameters:

primos_id
INPUT. A structure containing the PRIMOS user ID and the associated group IDs to be looked up. This variable has the data type of the structure primos_id, which is defined in GETIDS.H.

&unix_id
OUTPUT. The address of the value for each UNIX uid and gid that is mapped from the structure of PRIMOS user ID and group IDs, provided as input. The variable unix_id has the data type of the structure unix_id_info which is defined in GETIDS.H.

status
RETURNED VALUE. This is the completion status returned by the function. It may have two possible values:

0
Indicates that the function executed to completion, even if the specified UNIX uid and/or gid could not be found in the PASSWD and GROUP files.

-1
Indicates that the lookup facility encountered an error. The global errno is set to nonzero.

Discussion: This function returns the information for one user ID or multiple groups since a user may be a member of multiple groups. The function finds the UNIX information that corresponds with the PRIMOS user ID and group IDs that are provided. The information is found in the PASSWD and GROUP files that are located in the directory NFS_>ETC.

If a user ID is not found, the returned uid is -2. The returned gid list contains entries only for the groups found in the database. If 10 groups are specified, but only 5 are in the database, then the returned gid list contains only 5 entries, and the returned group_count is 5. The first gid in the list is returned as the effective gid; if none is found, the effective gid returned is -2.

OPENGETIDS( )

This routine tells the RPC library to prepare for one or more get-ID requests.

Usage:

void opengetids( )

Discussion: This is an optional routine. RPC programmers are encouraged to use it, because it enables the RPC library to optimize the servicing of ensuing multiple get-ID requests. For example, if your server application needs to perform frequent access checks (given the UNIX authentication of clients), then this routine optimizes response time on the PRIMOS system.

CLOSEGETIDS( )

This routine tells the RPC library that this application has completed all calls to getprmids( ) and getunxids( ). Any resources maintained by the RPC library are thereby released.

Usage:

void closegetids( )

Discussion: An RPC programmer that issues an opengetids( ) call should use this routine to provide a clean termination for RPC library resources.

The original call to opengetids( ) establishes a static onunit to do the cleanup, in case the programmer does not call closegetids( ).

INDEX

INDEX

O

P

R

S

T

U

Y

SURVEYS

READER RESPONSE FORM

Guide to NFS on PRIMOS Systems
DOC10285-2LA

Your feedback will help us continue to improve the quality, accuracy, and organization of our publications.

1. How do you rate this document for overall usefulness?

   ☐ *excellent* ☐ *very good* ☐ *good* ☐ *fair* ☐ *poor*

2. What features of this manual did you find most useful?

   _____
   _____
   _____
   _____
   _____
   _____

3. What faults or errors in this manual gave you problems?

   _____
   _____
   _____
   _____
   _____
   _____

4. How does this manual compare to equivalent manuals produced by other computer companies?

   ☐ *Much better* ☐ *Slightly better* ☐ *About the same*
   ☐ *Much worse* ☐ *Slightly worse* ☐ *Can't judge*

5. Which other companies' manuals have you read?

   _____
   _____

Name:_____ Position:_____

Company:_____

Address:_____

_____

_____ Postal Code:_____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

First Class Permit #531 Natick, Massachusetts 01760

BUSINESS REPLY MAIL

Postage will be paid by:

Prime™

Attention: Technical Publications
Bldg 10
Prime Park, Natick, Ma. 01760