Customer Order Number 424510771-430
NSC Publication Number 424510771-430A
December 1986

Series 32000®

GENIX V.3™ Remote File Sharing Utilities Notes

© 1986 National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, California 95052-8090

REVISION RECORD

REVISION	RELEASE DATE	SUMMARY OF CHANGES
A	12/86	First Release. *Series 32000*® GENIX V.3™ Remote File Sharing Utilities Notes NSC Publication Number 424510771-430A.

PREFACE

Remote File Sharing (RFS) is a software package that allows computers running the GENIX V.3™ System to share resources (files, directories, devices, and named pipes) selectively across a network. Unlike other distributed file systems used with the GENIX V.3 operating system, the Remote File System is built into the operating system itself. The advantages provided by this approach are described within this document. This document also covers the contents of the RFS software release as well as installation and administration procedures.

The information contained in this manual is for reference only and is subject to change without notice.

No part of this document may be reproduced in any form or by any means without the prior written consent of National Semiconductor Corporation.

GENIX V.3, ISE, ISE16, ISE32, and SYS32 are trademarks of National Semiconductor Corporation.

Series 32000 is a registered trademark of National Semiconductor Corporation.

The GENIX V.3 Operating System is derived from AT&T's UNIX System V.3 Operating System. Portions of the documentation for the GENIX V.3 Operating System are derived from AT&T coprighted UNIX V.3 Operating System documentation and reproduced under License from AT&T.

UNIX is a registered trademark of AT&T.

CONTENTS

Chapter 1

THE REMOTE FILE SHARING UTILITIES PACKAGE

1.1 SOFTWARE DESCRIPTION

Remote File Sharing (RFS) is a software package that allows computers running the GENIX V.3™ System to share resources selectively (files, directories, devices, and named pipes) across a network. Administrators for computers on an RFS network can choose which directories on their systems they want to share and add them to a list of available resources on the network. From this list, they can choose resources from remote hosts that they would like to use on their computers.

Each host computer on a Remote File Sharing system can be grouped with others in a "domain" or operate as an independent domain. The domain can provide a central point for administering a group of hosts. Unlike other distributed file systems used with the GENIX V.3 operating system, Remote File Sharing is built into the operating system itself. This approach has several advantages:

Compatibility — Once you mount a remote resource on your system it will look to your users as though it is part of the local system. You will be able to use most standard GENIX V.3 system features on the resource. Standard commands and system calls, as well as features like File and Record Locking, work the same on remote resources as they do locally. Applications should be able to work on remote resources without modification.

Security — Standard GENIX V.3 system file security measures will be available to protect your resources. Special means for verifying host access to your resources and restricting remote users' permissions have been added for Remote File Sharing.

Flexibility — Since you can mount a remote resource on any directory on your system, you have a lot of freedom to set up your computer's view of the world. You do not have to open up all your files to every host on the network. Likewise, you do not have to make all files on the network available to your computer's users.

1.2 CONTENTS OF THE RELEASE

The Remote File Sharing Utilities come on one floppy diskette. The files contained on that floppy are listed below.

/etc/rmount
/etc/rmountall
/etc/rumountall
/etc/master.d/du
/etc/master.d/dufst
/etc/master.d/sp
/etc/init.d/adv
/etc/init.d/rfs
/etc/init.d/fumounts
/etc/init.d/rumounts
/usr/bin/adv
/usr/bin/rmntstat
/usr/bin/rfpasswd
/usr/bin/rfstart
/usr/bin/rfstop
/usr/bin/rfuadmin
/usr/bin/rfudaemon
/usr/bin/fumount
/usr/bin/fusage
/usr/bin/idload
/usr/bin/dname
/usr/bin/rfadmin
/usr/bin/nsquery
/usr/bin/unadv
/usr/nserve/auth.info
/usr/nserve/nserve
/usr/net/servers/rfs/rfsetup
/usr/options/rfs.name

1.3 INSTALLATION AND BUILD PROCEDURES

1.3.1 Prerequisites

Before you can install Remote File Sharing Utilities, you must complete the following prerequisites.

Software

You must make sure the following software is installed before you install Remote File Sharing Utilities:

- Essential (GENIX V.3 release) Utilities
- Directory and File Management Utilities
- Networking Support Utilities
- NPACK NETWORK or some other Transport Interface-compatible network

You must also have the following space available on your system:

- / (root file system) needs 863 blocks of free space.
- **/usr** needs 1110 blocks of free space.

NOTE: 513 of the blocks needed in root are just temporary space that is used to reconfigure the system. After the package is installed, only about 350 blocks are used up.

1.3.2 Hardware

Remote File Sharing can be run on a *Series 32000®* Computer. The following hardware is required:

- Minimum of 2 megabytes of memory. (If you are upgrading to 2 megabytes of memory, make sure you update your tunable parameters to match that amount of memory. See Chapter 6 of the *Administrator's Guide* for more information.)
- Any communications hardware required by the network you are using.

1.3.3 Installation Procedure

This procedure describes Remote File Sharing installation. The Remote File Sharing Utilities for a *Series 32000* Computer are distributed on one floppy diskette. Most of the utilities are object code files. However, some are shell scripts that you can modify to suit your operating environment. To begin this procedure, you must:

- Place your computer in system state—2 (multi-user) or 1 (single-user). You must **mount /usr** to run this procedure in single-user mode.
- Be at the computer to insert and remove diskettes.

- Login as root.

1.3.4 Run Sysadm Installpkg

Step 1: To install the Remote File Sharing Utilities, use the direct access method of the System Administration menu as follows:

 # sysadm installpkg

This executes the sysadm subcommand **installpkg.**

Step 2: Insert the Remote File Sharing Utilities floppy diskette and press **RETURN** as instructed. Once you have hit **RETURN**, your *Series 32000* Computer will display the full path names of the files as they are copied from the floppy diskette to the hard disk. Be patient, this will take several minutes.

Step 3: When instructed, remove the floppy diskette and store it with your other utilities diskettes. You must then type **q** to return to the shell.

Step 4: As instructed, you should shutdown the system and bring it back up again. This is done as follows:

 # shutdown —i6 —g0 —y

Once you receive the **Console Login:** prompt your system will be ready.

Remote File Sharing installation is complete. You must now configure your Remote File Sharing system as described in Procedures 10.1 through 10.4 in the *Administrator's Guide.* Detailed information on Remote File Sharing is covered in Chapter 11 of the *Administrator's Guide.*

Chapter 2

REMOTE FILE SHARING SYSTEM ADMINISTRATION

2.1 OVERVIEW

This chapter presents a cookbook approach to the basics of RFS installation and administration. The following topics are covered:

- RFS network definition
- RFS network security
- starting and stopping an RFS network
- adding machines to an RFS network
- advertising and using resources over an RFS network
- automating RFS network administration
- software notes

The *GENIX V.3 Administrator's Guide*, contains a more detailed explanation of the RFS concepts and procedures.

2.1.1 Definitions

- client — a machine that remotely accesses a shared resource on another machine
- domain — a logical grouping of RFS machines that will share resources across a network
- host — a GENIX V.3 machine
- name service — an RFS feature that translates a user-defined symbolic name to an actual name of a resource. With name service, the user's view of resources is independent of the underlying RFS network.
- nodename — the name you give a GENIX V.3 host
- primary name server — a machine that provides name service for your domain
- secondary name server — a machine that can take over from the primary name server
- server — a machine on which a shared resource resides that is made available to one or more clients

2.2 BEFORE YOU START

These procedures assume that the GENIX V.3 System (specifically, the Essential Utilities and System Administration Utilities packages) has been successfully installed on any machine that will be part of your RFS network. You must also be logged in as **root** to perform the procedures.

Boldface items must be entered as shown. *Italics* are used for items supplied by the user; you must substitute a real name for the word in italics.

2.2.1 Check Machine Nodenames

Each machine in your RFS network must have a unique nodename. To find out a machine's nodename, type:

 # uname -n

If a machine does not have a nodename, you must assign one. See your *Administrator's Guide* or *Owner/Operator Manual* for details.

2.2.2 Install Networking Software

To use RFS, you must first install the following software in order:

- the Networking Support Utilities package
- your chosen network package (for example, NPACK)
- the Remote File Sharing Utilities package (note that this must be installed after the other two packages, and that your machine must be rebooted after installing RFS)

2.2.3 Check Network Installation

Installing the RFS utilities will configure the network listener, which is responsible for receiving requests from client machines for your machine's services. To make sure that the configuration is correct, make the following tests:

1. Verify that the listener is running:

   # nlsadmin -x

   If the listener is running, you should see:

   network ACTIVE

   (for NPACK, the *network* is "npack").

   If you see:

   network INACTIVE

your listener has not been started. To start it, type:

 # nlsadmin -s *network*

where *network* is 1 to 14 characters (upper/lower case letters, digits, hyphens, and underscores). For NPACK, you should enter:

 # nlsadmin -s npack

If you see neither of the above messages or cannot start your listener, your network package has not been installed properly.

2. After verifying that the listener is running, type the following to make sure that it knows about RFS:

   # nlsadmin -v *network*

   For example:

   # nlsadmin -v npack

   You should see the following (all on one line):

   105 ENABLED NOMODULES /usr/net/servers/rfs/rfsetup RFS server

   If you do not see this, re-install the Remote File Sharing Utilities again and repeat these two steps.

2.3 RFS NETWORK DEFINITION

2.3.1 Choosing a Domain

Choose your RFS domain name. The domain name is 1 to 14 characters (upper/lower case letters, digits, hyphens, and underscores). For the example following, the domain name is **finance.**

2.3.2 Choosing a Primary Name Server

Choose one machine to be responsible for providing name service to your domain. The machine you choose is called your primary name server and should be a machine that will be operational whenever your RFS network is operational. In the example following, the primary machine's nodename is **acct.**

For added reliability, you may want to add a secondary name server to your RFS network. See the *Administrator's Guide*, Chapter 11, for more information.

2.3.3 Setting Up the Primary Machine

From the machine that will be your primary, perform the following steps:

1. Set the domain name.

   # dname -D *domain*

   For example: **dname -D finance**

2. Define the network that you will be using.

   # dname -N *network*

   For NPACK, you should enter: **dname -N npack**

3. Specify yourself as the primary machine in the RFS master file.

   # ed /usr/nserve/rfmaster
   a
   domain → **p** → *domain.nodename*
   domain.nodename → **a** → *network_address*
   .
   w
   q

   where → is a single tab or space character; *domain* is the name you specified for your RFS community; *nodename* is the name of your chosen primary machine; *network_address* is the address of your machine on the network (for example, *nodename*.serve for an NPACK machine). For more information, see **rfmaster**(4).

   In our example, lines 3 and 4 would be

   ccs p ccs.compt
   ccs.compt a \ x060000080014105317000 0

4. Next, add the primary machine to the network:

   # rfadmin -a *domain.nodename*

   In this example:

   # rfadmin -a ccs.compt

   You will be prompted for the machine's password. Push <CR>.

5. The first time you start RFS on the primary machine, you must enter the following:

   # rfstart

   You will be prompted for your machine's password.
   Push <CR>.
   # rfstop
   # init 3

   You should wait for your machine to print out an RFS initialization message before you proceed.

   If you want to use passwords to protect your network, see the *Administrator's Guide*, Chapter 11.

6. RFS should now be up and running on the primary machine. To verify that, type:

   # nsquery

   If the output from this command includes the line:

   nsquery: RFS is not running

   RFS has not been successfully started on the primary. See the *Administrator's Guide* for troubleshooting assistance.

2.4 DAILY OPERATION

In normal course of operation, use the following command to start RFS on a machine:

 # init 3

2.5 ADDING MACHINES TO THE RFS NETWORK

To add machines to your RFS network, you must issue commands from your primary machine and from the machine you are adding.

2.5.1 From Your Primary Machine

From your primary machine, with RFS operational, do the following for each new machine you want to add to this domain:

 # rfadmin -a *domain.nodename*

For example, to add a machine with nodename of **budget:**

 # rfadmin -a finance.budget

You will be prompted for that machine's password. Push <CR>.

2.5.2 From the Other (Non-Primary) Machine

From each of your non-primary machines, you must do the following to add that machine to the RFS network:

1. Set the machine's domain name.

   # dname -D *domain*

   In this example:

   # dname -D finance

2. Set the machine's network.

   # dname -N *network*

   In this example:

   # dname -N npack

3. Start RFS initially by giving the network address of your chosen primary machine.

   # rfstart -p *network_address*

   Example:

   # rfstart -p acct.serve

   You will be prompted for your machine's password. Push <CR>

2.6 STOPPING THE RFS NETWORK

If you started RFS with **rfstart**, type the following:

 # rfstop

If you started RFS with **init 3**, type the following:

 # init 2

2.7 SHARING RESOURCES

2.7.1 Advertising Resources

A machine (server) may register a directory (and all files below it) for the use of other machines (clients) by advertising it. To advertise a given directory, you choose a symbolic name under which you will want clients to know the resource, decide if you will want to restrict their access to read-only, and decide if you want to restrict which clients can get access to it. For example,

 # adv BUDGET /usr/budget

will advertise the directory **/usr/budget** under the name **BUDGET**, so any client on the network that can access your machine can attach it to its file system tree. Alternatively,

 # adv -r SOURCE /usr/mysource mary john

will allow only client machines **mary** and **john** to attach your directory **/usr/mysource**, under the name **SOURCE**, but not to change any data there.

2.7.2 Using Resources

Before it can use an advertised resource, a machine (client) must attach (mount) that resource to its file system tree. For example,

 # mount -d BUDGET /usr/budget

or

 # mount -d -r SOURCE /usr/yoursource

will attach the two previously advertised resources to **/usr/budget** and **/usr/yoursource**, respectively, on the client machine. The **-d** option specifies that the resource is remote, and the **-r** option specifies that the resource will be read-only.

2.8 AUTOMATING RFS NETWORK ADMINISTRATION

The following sections explain how to automatically administer your RFS network.

2.8.1 Auto-Start

If you want to have RFS started automatically every time you boot your machine, you must edit the **/etc/inittab** file. This operation will automatically bring your machine into initialized (init) state 3, the RFS operational state for your machine.

```
# ed /etc/inittab
/is:/s/:2/:3/
w
q
```

2.8.2 Auto-Share

If you want to automatically make files available to other machines every time your machine enters init state 3, you must edit the **/etc/rstab** file:

```
# ed /etc/rstab
a
adv_command
.
w
q
```

where *adv_command* is the **adv** command, for example:

```
adv BUDGET /usr/budget
```

2.8.3 Auto-Mount

If you want to automatically use files that other machines have made available to you every time your machine enters initialized state 3, you must edit the **/etc/fstab** file:

```
# ed /etc/fstab
a
resource   directory_path   options
.
w
q
```

where *resource* is the name under which the other machine advertised this resource, *directory_path* is the location in your file system you want to attach that resource, and *options* are options to the **mount** command. For example, the line

```
BUDGET   /usr/budget   -d
```

will attach (mount) the resource advertised under the name **BUDGET** as **/usr/budget** on your machine.

2.9 ACCESSING FILES

By default, all remote access to shared files in an RFS network will have user-ID and group-ID permission set to "other." If you want remote file creation and access permissions to behave like they do on a standalone GENIX V.3 system machine, each user must have a unique user-ID across all machines on your RFS network. To make this happen, you have to modify the files that control user-ID and group-ID mapping as follows: first, make sure that RFS is not operational; second, edit the **uid.rules** and **gid.rules** files on each machine.

```
# ed /usr/nserve/auth.info/uid.rules
a
global
default transparent
exclude 0-99
.
w
q
# ed /usr/nserve/auth.info/gid.rules
a
global
default transparent
exclude 0-5
exclude 7-99
.
w
q
```

The **exclude** entries will prevent administrative permissions (for example, root) across your machines. If you need more flexibility, omit those lines for complete permission transparency.

If you encounter unexpected file access permissions on your RFS network, the **uid.rules** and/or **gid.rules** files may be missing or incorrect. See your *Administrator's Guide* for a complete description of uid-gid mapping.

2.10 RFS ADMINISTRATIVE COMMANDS

Following is a brief summary of the RFS administrative commands; for more information, see the corresponding manual pages in the *Administrator's Reference Manual.*

2.10.1 Sharing Resources

- **adv [-r] [-d "***desc***"]** *resource pathname* [*clients* ...]

  where **-r** restricts access to read-only; **-d** allows from 0 to 32 characters of descriptive text; *resource* is the symbolic name given to *pathname*; *pathname* is the full pathname of the directory you wish to share; and *clients* is an optional list of machines you wish to restrict this resource to.

- **adv -m** *resource* **-d "***desc***"** [*clients* ...]

  adv -m *resource* [**-d "***desc***"**] *clients* ...

  is used to modify an already advertised resource's description and/or restricted machine list.

- **mount [-r] -d** *resource pathname*

where **-r** is read-only access; *resource* is the advertised symbolic name; and *pathname* is the directory on which you wish to mount it. This will give your machine access to the shared resource.

- **unadv** *resource*

  where *resource* is the symbolic name you wish to prevent additional machines from mounting.

- **unadv** *domain.resource*

  will allow the primary machine to unadvertise another machine's resources (typically used when the advertising machine has failed).

- **umount -d** *resource*

  where *resource* is the symbolic name used when mounting the resource. This will sever access to that resource.

2.10.2 Monitoring and Status

- **fusage** [*advertised_directory*]

  reports status on how extensively hosts are using your *advertised_directory*.

- **nsquery [-h]** [*name*]

  reports status on currently advertised resources. If *name* is used as an argument, only resources advertised by *name* will be displayed.

- **rmntstat [-h]** [*resource*]

  reports on who currently has access to (has mounted) your advertised resources. If *resource* is used as an argument, only machines having access to that *resource* will be displayed.

- **sar -Du**

  prints the percent of CPU time used for RFS requests. You must have the Performance Measurement Utilities package installed to use **sar.**

- **sar -Dc**

  prints how many system calls were used by RFS.

- **sar -S**

  prints information on processes that handle requests from other machines to your data.

2.10.3 Primary Machine Administration

- **rfadmin [-a]** *domain.nodename*

  rfadmin [-r] *domain.nodename*

  will add (-a) or remove (-r) the given machine (*domain.nodename*) from an RFS domain.

2.10.4 Host Administration

- **dname [-aNDnd]** [*argument*]

  where **-N** will set your RFS network; **-D** will set your RFS domain; **-n** will display your RFS network; **-d** will display your RFS domain name; and **-a** will display both domain name and network name.

- **fumount [-w** *sec*] *resource*

  will sever all remote machine access to this advertised *resource* in *sec* seconds. The *resource* will be automatically unadvertised.

- **init 2**

  will stop RFS if RFS was started by an **init 3.** All resources will be automatically unadvertised and remote resources unmounted. All remote machine access to your resources will be severed.

- **init 3**

  will start RFS. All entries placed in **/etc/rstab** will be automatically advertised. All entries placed in **/etc/fstab** will be automatically mounted. If they are not currently available on the network, attempts will be periodically made to mount them automatically.

- **rfadmin** will display your current primary machine.

- **rfpasswd** will prompt you to change your RFS password, similar to the GENIX V.3 system **passwd** command.

- **idload -n** will display the current user mapping data. For other options see the **idload**(1M) manual page in the *Administrator's Reference Manual.*

- **rfstart** will start RFS (use only for initial RFS network configuration).

- **rfstop** will stop RFS (use only after initial **rfstart**).

2.11 SOFTWARE NOTES

This section describes problems that may occur with Remote File Sharing and, in some cases, workarounds to those problems.

2.11.1 Acct

The accounting file passed to the **acct**(2) system call cannot be remote. This restriction applies to user software that uses the system call directly and to the software in the optional processing accounting package. RFS does not allow the **acct** system call; if passed a remote pathname, **acct** will return an **errno** of **EINVAL.**

2.11.2 Adv, Unadv

Concurrent execution of the **adv** and **unadv** commands corrupts the advertise table. For example, executing the following two commands:

adv USR /usr &
unadv USR &

causes subsequent discrepancies in the resources printed by **adv** (which accesses the advertise table maintained on the local host) and **nsquery** (which accesses the advertise table maintained by the domain name server).

To avoid this, the **unadv** command should not be executed until all **adv**'s have completed.

2.11.3 Cd

The following is a scenario for a multi-hop in an RFS environment: Machine A advertises / as *RES1* and machine B advertises / as *RES2.* Then A mounts *RES2* on **/mpd,** and B mounts *RES1* on **/mnt.** If B tries to **cd** to **/mnt/mpd,** the error message given is "bad directory," instead of the appropriate multi-hop error message.

2.11.4 Cu

The Basic Networking Utilities (BNU) package allows the use of remote devices for operations such as **cu.** However, if a user on machine alpha has a device **/dev/xxx** and remotely mounts a **/dev** directory from machine beta, which also has an **xxx** entry, any use of either device will cause the other line to appear locked. The problem here is that the lock files used by BNU (in **/usr/spool/locks**) are associated with individual devices through the **basename** of the device, and the BNU commands are unable to distinguish between devices that have the same name but are in different directories.

When remotely mounting **/dev** directories,, check for entries duplicated in the local **/dev** directories. If there are duplicate names, follow these steps:

1. Make an alias name using the **ln** command. For example:

   ln /dev/culd0 /dev/*system-name***.culd0)**

2. Change all references in the BNU files to the *new* alias name (for example, in **/usr/lib/uucp/Devices**).

This will enable BNU to work as well as any locally written commands that make use of the conventional names for networking devices.

If RFS is in effect, when machine A advertises devices (**/dev**) to other machines (for example, machine B and machine C), there is no way for **cu** to know when a device is being accessed remotely. Both machine B and machine C will mount machine A's **/dev** directory under a directory such as **/rdev.** If a user from machine B **cu's** to a particular device, such as **/dev/xxx**), and a user from machine C tries to **cu** to that same device, both users will experience problems, for example, garbled information sent to the display screen. This problem will not occur if two users from the same machine try to access the same device, because the system locks the device for the first user. Unfortunately, the lock information resides only on the one machine; other machines in an RFS network have no way of knowing when a particular device is being accessed.

System administrators should be extremely cautious when advertising a directory like **/dev.**

2.11.5 Df

If **df** is used without options, it lists each occurrence of a remote resource that is mounted on a system, and places an asterisk next to the word blocks for the second and each subsequent resource that was advertised under the same remote filesystem (for example, **/usr/mail** and **/usr/bin**). This signifies that the identical block counts for the resources reside under the same file system.

The problem is that if **df** is used with multiple remote resources passed as arguments, the asterisk never appears. In this example the asterisk does not appear:

$ df USRMAIL USRBIN

This problem can be misleading because the user may think that although the block and inode count for the two resources are identical, they are not part of the same filesystem, since the asterisk that indicates they are is not present.

2.11.6 Fumount

The **—w** option to the **fumount** command allows a user to specify a grace period between warning clients that a resource is to be removed and actually removing the resource. The **atoi** subroutine (**strtol**(3C)) calculates the number of seconds. This routine looks for an initial numeric string and converts it to an integer. Any non-numeric character in the argument terminates the argument. For example, the argument **—w 123abc** gives a grace period of 123 seconds. Missing arguments and arguments without an initial numeric string produce an error message.

The usage message from **fumount** says that the range of the wait time is 0 to 3600 seconds. The correct range is 1 to 3600 seconds.

2.11.7 Fuser

The **fuser** command can fail to find processes that are using a resource, so an attempt to unmount the resource can fail because the resource is busy. For example, **fuser** does not find processes that are executing a text file in the target resource. Also, if a process tries to access a fifo, but blocks in **open**(2) waiting for a read or a write, **fuser** does not find the process. Finally, **fuser** may miss a process if that process gets a reference to the resource after **fuser** has begun its search.

In all these cases, the offending process can be killed explicitly with the **kill** command. When all processes using the resource are gone, the resource can be unmounted.

2.11.8 Idload

When **idload** is run with the **—n** option, it shows how it will interpret the specified rules file. However, in showing the interpretation of a **default** statement, **idload** does not list the name of the user or group specified, even if the **default** statement identified the user or group by name rather than by numeric ID.

In processing a rules file describing user ID or group ID mapping, **idload** will give a warning message if it encounters an ID in a **map** statement that previously appeared in an **exclude** statement. However, the warning that it gives is misleading, and says that the ID was previously mapped, rather than that it was previously excluded.

When using the **map** instruction for the **idload** command, one can map a remote ID to a local numeric ID. However, if the local numeric ID is greater than 65535 the value that is actually used is the local ID modulo 65536. A valid mapping request on a *Series 32000* Computer would not include a local ID this large anyway, but such a request could be made due to a typing mistake or misunderstanding on the part of a system administrator.

In the **idload** command, the **map** instruction may be given arguments of the form *X*:*Y*, meaning to map remote ID *X* into local ID *Y*. However, if the second ID is omitted (leaving **2:**), then remote ID *X* will be mapped into local ID 0 (root).

If you **exclude root** as recommended, the case described above will never happen. If root is excluded, *X*: would map *X* into ID number **MAXUID**+1 (60001 by default).

If the **idload** command is given a **map** line longer than 256 characters, it fails with an unclear error message. The error message states that it found an invalid token *X*, where *X* is actually the last character processed on that line. The message should identify the problem as having been given too long a line.

If this problem occurs, the offending **map** line should be broken up into two or more lines.

2.11.9 Labelit

On mounting a **ctc** device remotely under a local directory other than a subdirectory of **/dev**, **labelit** of the tape does not work on the remote machine because it expects the special file to begin with **/dev** and not with the pathname on which the raw device was mounted.

2.11.10 Logs

The following log files contain information relating to Remote File Sharing activities:

/usr/adm/rfuadmin.log
/usr/net/servers/rfs.log
/usr/net/servers/rfs/rfs.log
/usr/net/nls/*netspec***/log**

These files are for Remote File Sharing use only! Customers should not rely on the contents of these files because the information may change or the file may be deleted in future releases. Any tool written that takes advantage of the information contained in these files is not guaranteed to work in the future. (*netspec* above is replaced by the transport provider used by RFS. For the NPACK NETWORK, this will be **npack**.)

2.11.11 Mount

When a mount fails because of a password mismatch, the error message can be confusing. The following error messages result from a remote mount failure due to mismatched passwords:

negotiate: An event requires attention
mount: negotiations failed
mount: possible cause: machine password incorrect
mount: could not connect to remote machine

When a remote resource is disconnected by a **fumount** or a broken link, the default action in the (client) **rfuadmin** script is to try to remount the resource as it was mounted before. Therefore, if a resource that was originally read/write is readvertised read-only the automatic mount will never succeed.

An administrator can always enter the **mount** directly using the latest advertised mode.

2.11.12 Name Server

When the primary and secondary name servers are under heavy load, the normal passing of name server information between these hosts may cause the machines to hang because the 1K Streams buffers have been depleted. There is one long term and one short term solution to the problem. For the long term, you can increase the number of 1K Streams buffers in **/etc/master.d/kernel.** The parameter is NBLK1024. The short term solution is that you can stop Remote File Sharing on any secondary name server that is hung and then bring it back up again. That will clear the NBLK1024 buffers.

2.11.13 Nsquery

The resource list printed by **nsquery** does not always reflect the current state of the domain. If a resource is advertised and the server goes down, a subsequent **nsquery** from a client may still list the resource as being available, even though it is not. An attempt to mount the resource will fail, because it is unable to contact the server.

The **nsquery** command will return only the first 100 advertised resources even if there are more advertised.

2.11.14 Process

The last process slot is traditionally reserved for a super-user process. However, an RFS server can take the last slot, making it impossible for any new process to start. If this server, or any other process, exits, as would normally happen, new processes can start.

2.11.15 Programs

If a program creating remote directories or files loses its link to the remote machine, and the remote resource is unmounted, the program may begin to create *local* directories and files. For example, if you are using the **find** command piped to **cpio** to a remote machine and the link to the remote machine goes down and the resource is then unmounted, **cpio** may begin writing on the local machine — the target directory now looks just like an ordinary local directory.

2.11.16 Ps

When a client mounts a remote executable file and runs it in the background, a **ps** command on the client machine will sometimes show a "null" process running or the mounted directory name as a running process. Using **ps** —**f** will give the correct information.

2.11.17 Rfadmin

The command **rfadmin —r** *domain.machine* is used to remove the entry for *domain.machine* from the domain membership list. When this command is executed, the **rfmaster** file is checked to make sure that the machine being removed is not a backup (secondary) name server. However, there is no check to make sure that the machine being removed is not the primary name server. This can result in the entry for the primary machine being removed.

When executed with no options, the **rfadmin** command returns the name of the current domain name server. If this command is executed on a machine on which RFS is running, but no domain name server is available, the following message is printed:

rfadmin: cannot obtain the name of the primary name server for domain *dom*

If RFS is not running on the local machine, the same message is output, even if the domain name server is operating normally. This is misleading.

2.11.18 Rfmaster

The acting domain name server is responsible for distributing important name service information to all other accessible (secondary) name servers that are serving the same domain, with no more than a 15 minute lag, so that if the acting name server should fail, another host could assume the name server role with a minimal loss of information. However, changes to the **rfmaster** file after **rfstart** has been run are not included in the information that is distributed in this way. Because the designation of hosts as primary and secondary name servers is made in the **rfmaster** file, this has the consequence of not allowing a change to the configuration of which hosts are the primary and secondary name servers for a domain without stopping and re-starting RFS on the affected hosts. (For example, adding a new secondary name server to the **rfmaster** file will not take effect until RFS is taken down on all of the existing (primary and secondary) name servers, as well as the newly designated secondary and then re-started.)

This limitation should not be confused with the temporary transfer of name server responsibility to another one of the hosts already listed in the **rfmaster** file as a primary or secondary name server; this temporary transfer is performed with the **rfadmin —p** command.

2.11.19 Rfpasswd

The **rfpasswd** command is used to change the host password used for RFS, and is intended to parallel the **passwd** command in the way it prompts for old and new passwords. However, if a host has no password (for example it has a null password), the **rfpasswd** command will still prompt for the old password before asking for the new one, although it should ask only for the new one. Furthermore, the prompt for the old password is "Password," although it should be "Old password."

2.11.20 Rfs

If the administrator of a server machine explicitly kills all of the server processes, all of the client processes that have requests on a server will hang forever waiting for a response from the server.

2.11.21 Rfstart

RFS is initiated on a machine by executing the **rfstart** command. Under some circumstances, **rfstart** will prompt for a password. In this case, RFS has actually been started on the local machine before **rfstart** completes, and RFS commands executed on another terminal can succeed while **rfstart** is still waiting for the password to be entered. However, if any resources are advertised from the local host before **rfstart** completes, those advertised will be erased from the local advertise table when the **rfstart** does complete.

This problem can be avoided by not issuing additional RFS commands until the **rfstart** completes and exits to the shell.

The **rfstart** command can take up to a minute or more to return when executed from the shell, depending on response from the network. When possible, **rfstart** should be run automatically by using **init 3.**

2.11.22 Rfudaemon

User-level recovery of resources that are disconnected gracefully (the remote system shuts down) may fail if the number of lost resources exceeds half of the value of the tunable parameter **MAXGDP** in **/etc/master.d/du.** By default, **MAXGDP** is 24. The failure is accompanied by one or more of the following messages:

rfs user-daemon queue overflow:
make sure rfudaemon is running

2.11.23 Rmount

The **/etc/rmount** script loops forever if the system administrator manually mounts the remote resource during the **rmount** sleep interval. If this occurs, do a **ps —ef** to get the process ID of **rmount,** and then kill the process manually by issuing **kill —9** *pid#.*

2.11.24 Stat

Some commands, such as **ls —l,** can be used to identify which users have access to a file. These commands obtain data on a file via the **stat** system call. When referencing a remote file, the ownership information returned by **stat** is converted to local UIDs/GIDs by computing the inverse of the UID/GID mapping on the server machine.

Under some UID/GID mapping specifications, it will erroneously appear that local users have access to a remote file when, in fact, the remote file is inaccessible to all local users. For example:

UID Mapping:

```
global
default transparent
exclude 0
```

Command:

```
$ ls —l /remote/etc/passwd
-rw-r--r--  1 root  sys  32449 Jan 22 19:40 /remote/etc/passwd
```

Despite what is reported by the **ls** command in this example, the local user **root** (UID=0) cannot write the remote file **/remote/etc/passwd.**

2.11.25 Sticky Bit

If a program that has the sticky bit set is shared through RFS, and is executed by a user on a client machine, then it becomes impossible to remove the program (for example, with the **rm** command). Attempts to remove the file will generate an error message of "text busy," even though no one on any machine is currently running the program. Removing the sticky bit and re-running the program has no effect. A program with the sticky bit set that has never been run across RFS can be removed without complaints, even if it has been run locally.

To remove such a "stuck" sticky-bit program, it is necessary to unmount the remote resource. This can be done either from the server, with the **fumount** command, or from the client, with the **umount** command.

2.11.26 Streams

The three system calls related to STREAMS, **getmsg**, **putmsg**, and **poll**, will not operate with a file descriptor associated with a remote file. If this is attempted, the system call will fail with **errno** equal to **EINVAL.**

2.11.27 Swap

Swap devices cannot be remote. This includes the swap device configured initially, and any swap devices added using the **swap**(1M) command.

2.11.28 Umount

If a client loses its link to a server, any attempt to **umount** one of that server's file systems from the client tree will fail until recovery runs. Recovery from a link failure is handled by **rfuadmin**(1M) and **rfudaemon**(1M).

Recovery runs automatically when the link breaks, but not until someone tries to access the link, or until at most 11 minutes have passed. (The 11 minute time interval applies if you are using NPACK NETWORK. The time may be different for other transport providers.)

If the **umount** fails because the link is gone, the unmount will start recovery. After recovery runs, a second **umount** will succeed.

2.11.29 Unadv

If **unadv** is invoked for a resource that was not advertised from this host, and this host is not the acting domain name server, then the command fails (as it should), but with a vague error message:

```
unadv: Permission for operation denied
```

INDEX

S

U

National
Semiconductor

MICROCOMPUTER
SYSTEMS DIVISION

READER'S COMMENT FORM

In the interest of improving our documentation, National Semiconductor invites your comments on this manual.

Please restrict your comments to the documentation. Technical Support may be contacted at:

(800) 538-1866 - U.S. non CA
(800) 672-1811 - CA only
(800) 223-3248 - Canada only

Please rate this document according to the following categories. Include your comments below.

	EXCELLENT	GOOD	ADEQUATE	FAIR	POOR
Readability (style)	☐	☐	☐	☐	☐
Technical Accuracy	☐	☐	☐	☐	☐
Fulfills Needs	☐	☐	☐	☐	☐
Organization	☐	☐	☐	☐	☐
Presentation (format)	☐	☐	☐	☐	☐
Depth of Coverage	☐	☐	☐	☐	☐
Overall Quality	☐	☐	☐	☐	☐

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

Do you require a response? ☐ Yes ☐ No PHONE _____

Comments:

_____

_____

_____

_____

_____

_____

FOLD, STAPLE, AND MAIL

424510771-430A

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 409 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

National Semiconductor Corporation
Microcomputer Systems Division
Technical Publications Dept. 8278, M/S 7C261
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-9968