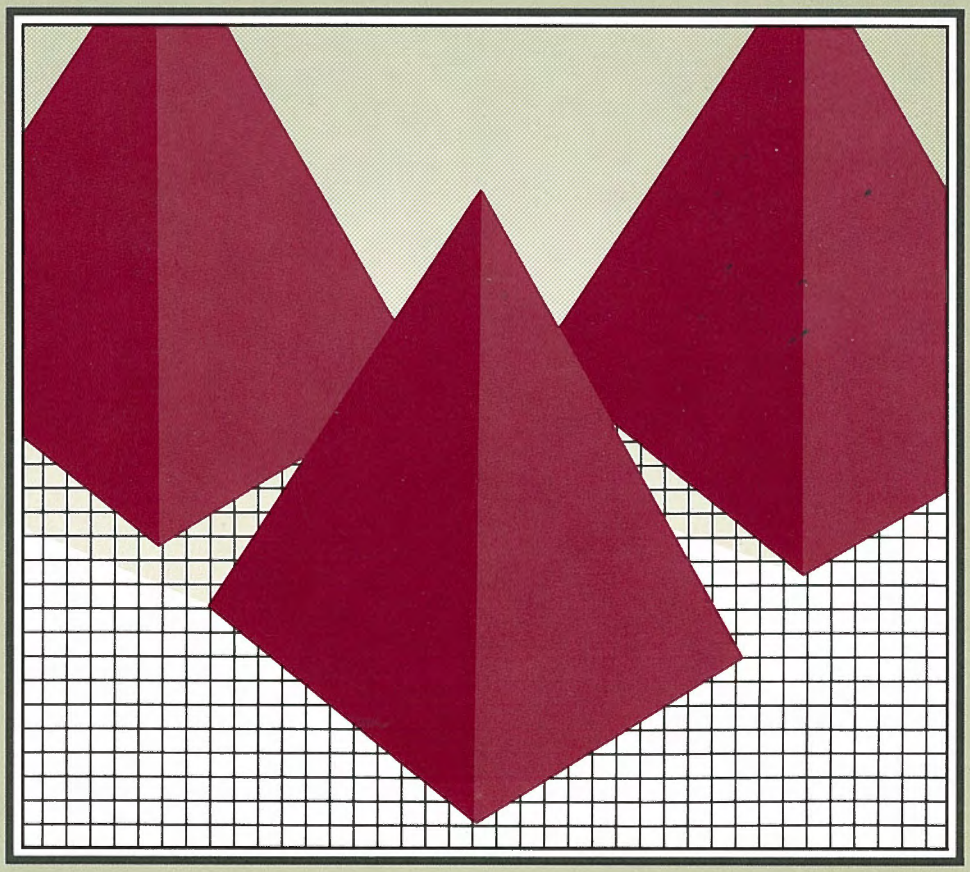
HEWLETT-PACKARD

HP 9000 Series 300 Computers

Using and Administering NFS Services



Using and Administering NFS Services

HP 9000 Series 300

Manual Part Number: 50969-90001

Printed in U.S.A., January 1989

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104.9(a).

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

DEC and VAX are registered trademarks of Digital Equipment Corp.

MS-DOS® is a U.S. registered trademark of Microsoft Corporation.

Hewlett-Packard Co.
3404 E. Harmony Rd.
Fort Collins, CO 80525 U.S.A.

Printing History

June 1988 . . . Edition 1.

January 1989 . . . Edition 2.

Table of Contents

Chapter 1: Documentation Overview

Contents 1-2
Conventions 1-5
Documentation Guide 1-6

Chapter 2: NFS Services Overview

NFS Services 2-3
- NFS Remote File Access 2-4
- Remote Execution Facility (REX) 2-8
- Remote Procedure Call (RPC) 2-8
- Remote Procedure Call Protocol Compiler (RPCGEN) 2-9
- External Data Representation (XDR) 2-11
- Network Lock Manager 2-12
- Yellow Pages (YP) 2-12
- Virtual Home Environment (VHE) 2-18

Chapter 3: Common Commands

Key Terms 3-2
NFS Commands 3-6
- NFS Remote File Access 3-6
- rpcinfo 3-9
- rup 3-11
- rusers 3-12
- showmount 3-13
- Remote Execution Facility (REX) 3-14

Yellow Pages Overview 3-15
- YP Maps 3-15
- YP Servers and YP Clients 3-16

Chapter 3: Common Commands (Continued)

YP Domains 3-16
YP Master and Slave Servers 3-18
YP Commands 3-19
domainname 3-19
ypcat 3-20
ypmatch 3-21
yppasswd 3-21
ypwhich 3-24

Chapter 4 Installation

Key Terms 4-3
Prepare the System 4-5
Install Software 4-7
Use Update 4-7
Configure a New Kernel 4-11
Add a Computer to the Network 4-13
Relink Programs 4-14

Chapter 5: NFS Configuration and Maintenance

Key Terms 5-2
Guidelines 5-6
Network Memory 5-6
Configuration Files 5-7
Daemons 5-9
Servers 5-11
NFS Configuration 5-13
1. Compare /etc/newconfig Files to Existing Files 5-14
2. Set UIDs and GIDs 5-15
3. Create an NFS Server 5-16
4. Create an NFS Client 5-34
5. Configure YP (optional) 5-47
6. Configure VHE (optional) 5-47
7. Execute /etc/netnfsrc 5-47

Chapter 5: NFS Configuration and Maintenance (Continued)

Maintenance 5-48
Remove NFS File Access 5-48
Update Software 5-52
Clock Skew 5-53

Chapter 6: Remote Execution Facility (REX)

Introduction 6-1
The on Command 6-2
Configuration Requirements 6-4
Environment Simulation 6-5
Configuring rexd 6-6
Security Considerations 6-9
Diagnostics 6-11
The on Command Error Messages 6-11
rexd Error Messages 6-12

Chapter 7: The Network Lock Manager

Introduction 7-1
The Locking Protocol 7-4
The Network Status Monitor 7-5

Chapter 8: YP Configuration and Maintenance

Key Terms 8-2
YP Databases 8-7
Local and Global Maps 8-8
Netgroups 8-12
Files Related to YP 8-16
YP Commands 8-18
YP Configuration 8-21
Before Configuring YP 8-21
1. Create a YP Master Server 8-23
2. Create a YP Client 8-27
3. Create a YP Slave Server 8-33
4. Propagate YP Maps 8-36

Chapter 8: YP Configuration and Maintenance (Continued)

Verify YP 8-39
YP Maintenance 8-40
Disable YP 8-40
Modify YP Maps 8-41
YP Maintenance 8-42
Add New YP Servers 8-45
Add New Users to a Node 8-45
Make a Different Node the YP Master 8-46
Create or Change YP Password 8-47
Log Files 8-49
Create Non-standard YP Maps 8-51

Chapter 9: VHE Configuration and Maintenance

Configuration Overview 9-2
Configuration 9-3
1. Complete Preparation Steps 9-3
2. Compare /etc/newconfig Files to Existing Files 9-4
3. Determine File Systems and Mount Point Directories 9-5
4. Create /etc/vhe_list 9-6
5. Update /etc/passwd 9-9
6. Update /etc/exports 9-10
7. Distribute /etc/vhe_list and /etc/passwd 9-11
8. Execute /usr/etc/vhe/vhe_mounter 9-11
9. Verify that VHE is Correctly Configured 9-13
Configuration Refinements 9-15
NFS mounts in the Background 9-15
VHE Maintenance 9-16
Unmounting file systems 9-16
Adding or Deleting VHE Nodes 9-17

Chapter 9: VHE Configuration and Maintenance (Continued)

Advanced Usage 9-18
Adding altlogin and mounter Logins 9-18
$HOME 9-20
$ROOT 9-20
Alternate Mount Points 9-21
Using VHE for Mail 9-21

Chapter 10: Troubleshooting

Key Terms 10-2
Troubleshooting References 10-6
Power Up and Connectivity Testing 10-6
Troubleshooting Sections 10-7
Guidelines 10-8
Common Network Problems 10-8
Initial Troubleshooting 10-9
Error Messages 10-12
Unsolved Problems 10-13
Flowchart Format 10-14
Troubleshoot NFS 10-16
Mount Fails (Flowchart 2) 10-21
Server Not Responding (Flowchart 3.1) 10-25
Restricted Access (Flowchart 4) 10-33
Programs Hang (Flowchart 5) 10-37
Performance Problems (Flowchart 6) 10-39
Troubleshoot Yellow Pages 10-40
Troubleshoot VHE 10-52

Appendix A: HP NFS Services vs. Local HP-UX

Appendix B: Migrating from RFA to NFS

Why Migrate to NFS Services? B-1
Similarities B-2
Differences B-2
Changing Scripts from RFA to NFS B-4

Appendix B: Migrating from RFA to NFS (Continued)
Shell Scripts that Accept Different Paths B-4
Shell Scripts with Hard-coded Paths B-5
RFA through NFS B-7
Appendix C: NFS in an HP-UX Cluster Environment
HP-UX Cluster Terms C-1
NFS Configuration and Maintenance C-2
YP Configuration and Maintenance C-3
Troubleshooting C-3
Appendix D: Password Security
Glossary
Index

1

Documentation Overview

Before reading this manual, you should be familiar with HP-UX and have access to *HP-UX Reference* manuals.

You will find this manual helpful if you have any of the following responsibilities for the NFS (Network File System) Services product.

- Installation
- Initial configuration of NFS, YP (Yellow Pages) or VHE (Virtual Home Environment) services
- Routine administration and maintenance of NFS, YP or VHE
- Troubleshooting common NFS, YP or VHE problems

Note

If you are using NFS Services, but have no administrative responsibilities, you will need to use the "Common Commands" chapter.

Contents

Refer to the following list for a brief description of the information contained in each chapter and appendix.

Chapter 1: Documentation Overview

This chapter describes who should use this manual, what is in this manual, and where to go for more information.

Chapter 2: NFS Services Overview

This chapter provides a brief overview of the NFS Services product, particularly the NFS, RPC, RPCGEN, REX, Network Lock Manager, YP, and VHE services. It also describes common terms and concepts.

Chapter 3: Common Commands

This chapter provides brief explanations of remote file access via NFS and common NFS and YP commands.

Chapter 4: Installation

This chapter explains how to install the NFS Services product.

Chapter 5: NFS Configuration and Maintenance

The first section explains how to set up your files in the correct configuration. It also describes NFS daemons, servers, and file systems.

The second section explains procedures for maintaining an efficient system. It includes topics such as NFS file access removal and clock skew problems.

Chapter 6: Remote Execution Facility (REX)

This chapter explains how to configure and use the Remote Execution Facility (REX). You can use REX to execute commands on a remote host.

Chapter 7: Network Lock Manager

The Network Lock Manager and the Status Monitor permits cooperating processes to synchronize access to shared files via System V file locking primitives. This chapter describes the Lock Manager in detail.

Chapter 8: YP Configuration and Maintenance

The first section explains how to set up your files in a configuration that allows you to centrally administer your YP databases.

The second section explains procedures for administrating and maintaining the YP service. It includes topics such as modifying your system to use YP and changing your YP password.

Chapter 9: VHE Configuration and Maintenance

This chapter explains how to configure your system to use the Virtual Home Environment (VHE) service. VHE allows you to set up remote login environments to resemble home node login environments.

Chapter 10: Troubleshooting

This chapter describes how to locate and eliminate network problems, specifically those related to the NFS, YP and VHE services.

Appendix A: HP NFS Services vs. Local HP-UX

This appendix describes the basic differences between NFS Services and local HP-UX operations.

Appendix B: Migrating from RFA to NFS

This appendix describes how to translate RFA applications to NFS applications.

Appendix C: NFS in an HP-UX Cluster Environment

This appendix lists the interactions between NFS Services and HP-UX cluster nodes.

Appendix D: Password Security

This appendix explains the use of encrypted passwords and password security.

Glossary

The glossary lists and defines terms used in this manual that may not be familiar to you.

Index

The index provides a page reference to the subjects contained within this manual.

Conventions

This manual uses the following format for all entry instructions and examples.

Bold Text	emphasizes the word or point.
`Computer Text`	specifies a literal entry. You should enter the text exactly as shown.
Italic Text	indicates you should enter information according to your requirements.

Example:

`domainname`	*domain_name*
Enter the word domainname	Enter the name of your YP domain.

Note

Except for the "YP Configuration and Maintenance" chapter, all references to servers and clients apply to NFS servers and clients unless otherwise specified.

Documentation Guide

For More Information	Read
ARPA Services: Daily Use	*Using ARPA Services*
ARPA Services: System Administration	*Installing and Maintaining NS-ARPA Services*
C Programming Language	*C Programming Guide*, Jack Purdum, Que Corporation, Indianapolis, Indiana *The C Programming Language*, Brian W. Kernighan, Dennis M. Ritchie; Prentice-Hall, Inc.
Commands and System Calls	*ARPA/Berkeley Services Reference Pages* *HP-UX Reference* Manuals *NFS Services Reference Pages* *Network Services Reference Pages*
HP 92223A Repeater	*HP 92223A Repeater Installation Manual*
HP-UX: Installation	*HP-UX Installation Manual*

For More Information	Read
HP-UX: Operating System (HP 9000)	*HP-UX Concepts and Tutorials* *HP-UX Installation Manual* *HP-UX Reference* Manuals *HP-UX Series 300 System Administrator Manual* *Beginner's Guide* series for HP-UX *Introducing UNIX System V*
HP-UX: System Administration	*HP-UX Series 300 System Administrator Manual*
LAN Hardware for Series 300: Installation	*HP 98643A LAN/300 Link* *LANIC Installation Manual* *LAN Cable and Accessories Installation Manual*
Networking: General Information	*Networking Overview: NS-ARPA, NFS Services*, and X.25
NFS Services: Common Commands	*Using and Administering NFS Services* "Common Commands" Chapter **only**
NFS Services: Programming and Protocols	*Programming and Protocols for NFS Services*

For More Information	Read
NFS Services: System Administration • Configuration • Installation • Maintenance • Migrating from NFS to RFA • NFS in an HP-UX Cluster Environment • NFS Services vs. Local HP-UX • Network Lock Manager • Remote Execution Facility (REX) • Troubleshooting • Virtual Home Environment • Yellow Pages	*Using and Administering NFS Services*
NS-ARPA: System Administration	*Installing and Maintaining NS-ARPA Services*

2

NFS Services Overview

HP's NFS (Network File System) Services product allows many systems to share the same files. It is an independent networking product, not a distributed operating system. NFS differs from distributed operating systems by not limiting its use to specific hardware and software. Rather, it operates on heterogeneous nodes and in operating systems from a variety of vendors. Explicit file transfers across the network to your local node are unnecessary. Since access techniques are transparent, remote file access remains similar to local file access.

With NFS all network nodes are either **clients** or **servers** or both.

- A **client** is any node or process that accesses a network service.

  An NFS client can also be configured as any combination of an NFS server, YP (Yellow Pages) client, or YP server. (A YP server **must** also be configured as a YP client.)

- A **server** is any node that provides one of the network services. A single node can provide more than one service.

  An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

- Servers are passive in that they always wait for clients to call them.

  The degree to which clients **bind** to their server varies with each of the network services. However, the client always initiates the binding. The server completes the binding subject to access control rules specific to each service.

- NFS servers are **stateless**; they do not maintain information relating to each client being served. Each file request goes to the appropriate server with the parameters attached to it locally (e.g., read and write privileges). One advantage is that you can reboot servers without adverse consequences to the client.

NFS Services

The NFS Services product includes the following components.

- NFS Remote File Access
- Remote Execution Facility (REX)
- Remote Procedure Calls (RPC)
- Remote Procedure Call Protocol Compiler (RPCGEN)
- External Data Representation (XDR)
- Network Lock Manager
- Yellow Pages (YP)
- Virtual Home Environment (VHE)

The NFS, REX, Lock Manager, and YP functionalities are built on top of RPC and XDR library routines.

Note

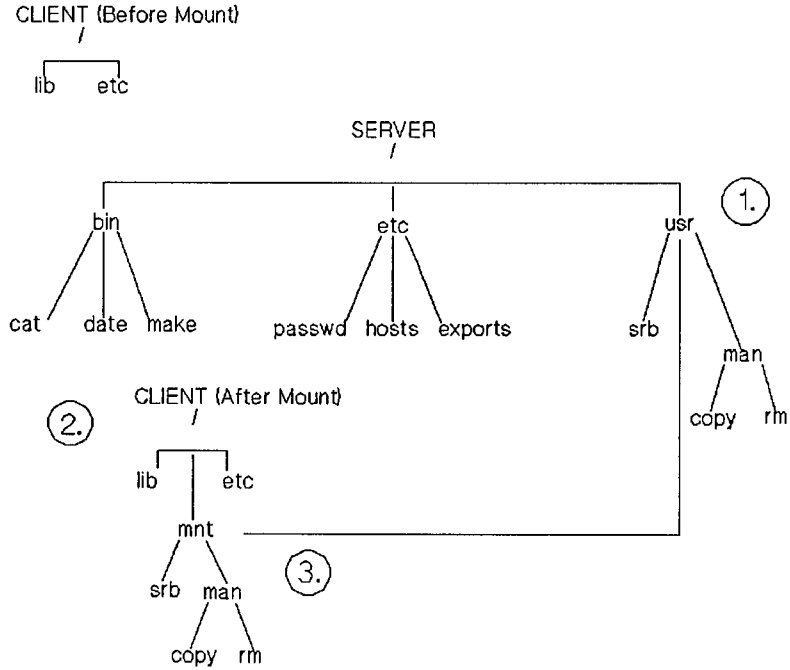
You must recompile programs that access remote directories. Otherwise, these programs will not be able to access remote directories mounted through NFS since the old directory routines use a *read* call instead of a *getdirentries* call to access those directories.

NFS Remote File Access

Before the client can access remote files,

- on the server, the super-user must export the file system (i.e., make it available) to the client and
- on the client, the super-user must mount (import) the file system.

Access to remote files is the same as for local files. You need to include either the complete path name starting with / (slash) or the path name relative to the current directory.



EXAMPLE:

1. The super-user edits the server's */etc/exports* file to make the */usr* file system available to the client.

```
server super-user% cat /etc/exports
/usr client_name
```

2. On the client, the super-user creates a mount point */mnt* (empty directory) and mounts the file system.

```
client super-user% mkdir /mnt
client super-user% mount server:/usr /mnt
```

3. The client reads the files in the */mnt* directory.

```
client% more /mnt/man/copy
```

Two very important features of NFS Remote File Access are **named pipes** and **device files**. The following sections explain the details of these two features.

Named Pipes

A named pipe is a special type of object in the HP-UX file system. A named pipe is one of the many ways in HP-UX that unrelated processes can communicate. HP-UX processes executing on the same client system are able to communicate using named pipes. You can use named pipes via normal file operations, e.g. *open()*, *close()*, *read()*, *write()*. Typically, one process will open the named pipe for reading and another process will open it for writing.

To illustrate named pipes, consider the following example:

EXAMPLE:

C1 and C2 are processes executing on system C. Also assume host C has mounted file system / from host S on */mnt*. C1 opens */mnt/FIFO* for reading and C2 opens */mnt/FIFO* for writing. C1 can now read what C2 wrote to the named pipe.

Next, assume a third process (process D3) is running on another client D which also has / from S mounted on /mnt (on system D), and it opened */mnt/FIFO* for reading. Is process D3 able to read what process C2 wrote to this named pipe? No, because no actual NFS activity occurs between the NFS client and NFS server for named pipe reads and writes. These are handled entirely by the client.

Note

In certain cases there would be NFS activity. For example, if you do a *chown(2)* on the named pipe, the request will go to the server to change the owner.

---

mknod()

Named pipes are created with *mknod()*. Any user can create a named pipe with *mknod()*. Use of *mknod()* to create device files requires super-user privileges.

Note

If you attempt to make a directory or a network special file over NFS, *mknod()* will fail and will return with *errno* set to EINVAL.

Device Files

Device files are another type of object in the file system, and are used to access physical or conceptual devices attached to the system. NFS device files always refer to a device attached to the local system and can generally be used where a local device file would be used. Like named pipes, device files are operated on through normal file system operations. For example, to write to the system console, you can write to the file */dev/console*.

EXAMPLE:

To illustrate the use of device files, consider the following:

System C is an NFS client of the server System S, and has mounted file system / from host S on */mnt* (Super-user on system C executed the command *mount S:/ /mnt*). If a process on System C attempts to write to */mnt/dev/console*, a device file representing the system console, the output will go to the system console on System C, not on System S. If a process on System S attempts to write to */dev/console*, which is the same "file" that System C wrote to, it will actually write to the console on System S.

NFS Mounts with Device Files

NFS device files are not secure. Therefore, the system administrator has the option of turning off device file access on a per-NFS mount basis. The administrator uses the *-o nodevs* option to the *mount(1m)* command.

EXAMPLE:

```
mount -o nodevs nfserver:/servermountpoint /clientmountpoint
```

Note

The *nodevs* option does not turn off support of named pipes.

Mounting From NFS Device Files

You may mount a local disk that is represented by a remote NFS device file.

EXAMPLE:

```
mount /mnt/nfs/dev/dsk/0s0 /localmntpt
```

Access to the newly mounted file system will proceed as if the disk had been mounted from a local device file.

Note

Access to the local disk's mounted file system will not be affected even if the NFS file system is unmounted.

Normally when unmounting a file system, you can give either the name of the device file or the name of the mount point. However, if the NFS server is down or the NFS file system is down, you must give the mount point to unmount the local disk.

EXAMPLE:

You would enter the following to unmount a local disk:

```
umount /localmntpt
```

instead of:

```
umount /mnt/nfs/dev/dsk/0s0
```

The latter case will not fail if the NFS server is down, but it will hang until the server comes back up as any other NFS access does.

Remote Execution Facility (REX)

The Remote Execution Facility allows you to execute commands on a remote host. REX is similar to the Berkeley service remote shell (*remsh(1)*) with two major differences:

- Your environment is simulated on the remote host
- You can execute interactive commands on the remote host

Remote Procedure Call (RPC)

NFS Services consists of remote programs composed of remote procedures called from the client nodes on the network. Optimally, a remote procedure computes results based entirely on its own parameters. Thus,

the procedure (and therefore, the network service) is not tied to any particular operating system or hardware.

NFS clients access server information and processes by making a remote procedure call. RPC allows a client process to execute functions on a server via a server process. Though these processes can reside on different network hosts, the client process does not need to know about the networking implementations.

The client first calls an RPC function to initiate the RPC transaction. The client system then sends an encoded message to the server. This message includes all the data needed to identify the service and user authentication information. If the message is valid (i.e., calls an existing service and the authentication passes) the server performs the requested service and sends a result message back to the client.

Remote Procedure Call Protocol Compiler (RPCGEN)

RPCGEN is a Remote Procedure Call compiler. You use it to convert applications running on a single computer to ones that run over a network. It is also used to assist in writing Remote Procedure Call applications simply and directly. With RPCGEN, your development time will be reduced and you will spend less time coding and debugging network interface code.

You produce three of the files required to convert an application to run on a network. These files are:

- **protocol description file**
- **client side file**
- **server side function file**

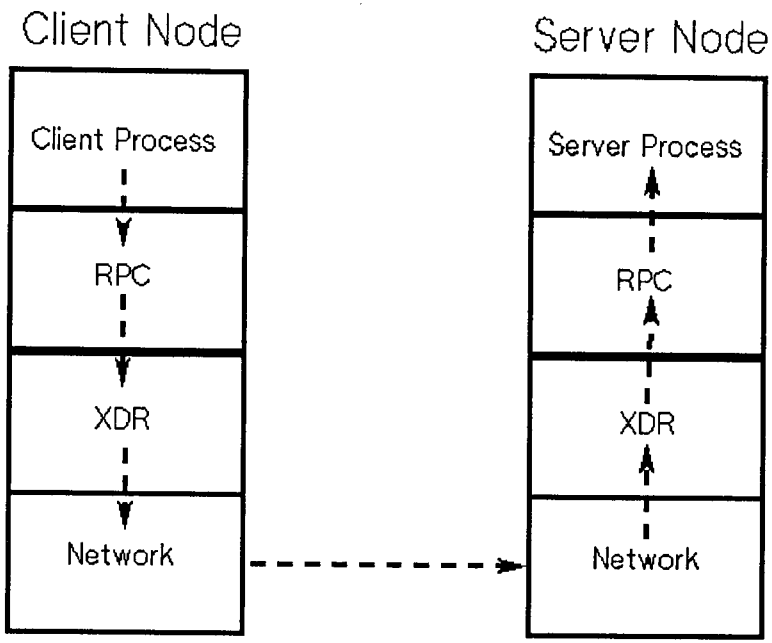
RPCGEN accepts remote program interface definitions (the protocol description file) written in RPC and produces the following C output files, which you may use as a starting point, rewriting as necessary:

- **header file**
- **client side subroutine file**
- **server side skeleton file**
- **XDR (External Data Representation) routine file**

If you wish to use the RPCGEN compiler to write RPC applications, refer to the "RPCGEN Programming Guide" chapter in the *Programming and Protocols for NFS Services* manual.

External Data Representation (XDR)

RPC uses the eXternal Data Representation functionality to translate machine dependent data formats (i.e., internal representations) to a universal format used by all network hosts using RPC/XDR. Thus, XDR enables heterogeneous nodes and operating systems to communicate with each other over the network.



RPC and XDR Data Transfer

Note: This figure does not correspond to the ISO Model.

Network Lock Manager

NFS Services includes the Network Lock Manager and the Network Status Monitor. The Network Lock Manager supports file locking and synchronized access to shared files via *lockf* and *fcntl* for NFS. The Network Status Monitor is used by the Network Lock Manager to maintain the stateful locking service within the stateless NFS environment. It allows applications to monitor the status of other computers and systems.

Yellow Pages (YP)

The Yellow Pages (YP) is an **optional** service containing a collection of cooperating YP server processes that provide YP clients access to data. You can administer all the databases from one YP **master server** since it propagates data across the network to other YP servers. YP includes the following features.

- YP manages unlimited databases. Typically these include files in */etc*: *group*, *hosts*, *netgroup*, *networks*, *passwd*, *protocols*, *rpc*, and *services*.

  For example, programs previously read */etc/hosts* to find an Internet address that corresponds to a host name. When you added a new node to the network, you had to add a new entry to every node's */etc/hosts* file, Now programs can use YP to obtain information from other YP servers.

- Since the YP master server propagates all **maps** (databases) to the **slave servers**, a YP client receives consistent information regardless of which YP server it accesses.

- If a remote node running a YP server process crashes, YP client processes can obtain YP services from another YP server.

- Since the YP interface uses RPC and XDR, the service is available to other vendors.

YP Advantages

YP has several advantages.

- YP enables you to automatically keep user IDs and group IDs consistent among all the nodes participating in NFS file sharing.

  Without YP, you have to manually keep these IDs consistent for NFS.

- YP provides the convenience of centrally administering the */etc* files: *password*, *group*, *hosts*, *netgroup*, *networks*, *rpc*, *services*, and *protocols*.

  Without YP, you must individually administer these files on each node.

YP Disadvantages

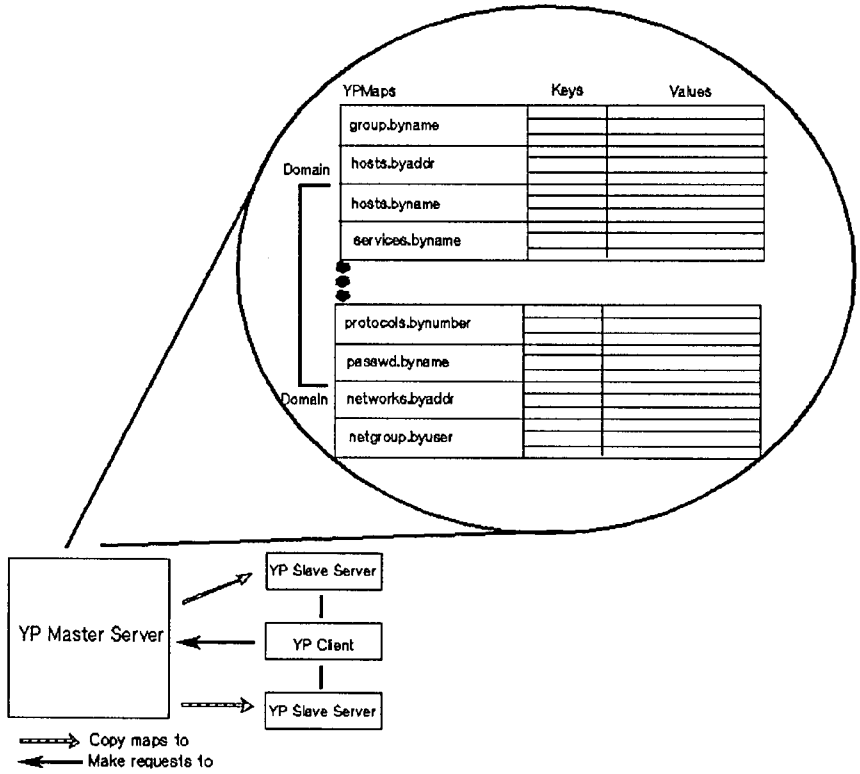
YP has the following disadvantages.

- Since YP provides YP clients access to data via the network, YP clients may observe slower performance than if the data were accessed from local files. For example, with YP, logging in may take more time if the YP server is busy.

- If any of the YP servers are unstable, remote access to files may be slower since the YP client may have to rebind to another YP server. If no other YP server is available, users may not be able to login to their nodes without access to the YP's *passwd* map.

- YP does not make changes visible to all users unless the changes are made on the YP master server.

- The YP slave servers do not immediately see the changes made to the YP master server maps. The updated maps become consistent among all YP servers only after each slave server successfully copies the maps via *ypxfr(1M)*.



Yellow Pages Structure

Refer to the following figure and subsections for a summary of how components within Yellow Pages work together: maps, YP domains, YP servers (masters and slaves), and YP clients.

YP Maps

The YP system stores information in YP **maps** (databases). Each map contains a set of keys and associated values: one key per value and one value per key. (A value may be a string of characters with imbedded blanks or tabs). For example, in the *passwd.byname* map, all the login names are the keys and their matching lines from */etc/passwd* are the values.

Each map has a unique **map name** that programs use to access the map. Programs must know the format of the data in the map. Many of the maps are derived from ASCII files such as */etc/hosts*, */etc/group*, and */etc/passwd*. The map format is usually identical to the ASCII file format.

Note

If using YP to provide the information stored in the standard maps' ASCII files, you **must** recompile any applications that read data from those files using standard C library routines.

This recompilation ensures the files can obtain data from the YP maps. If you do not recompile the applications, they will access only the local files. If the local files are not as current as the YP maps, the applications may not work correctly.

YP Servers and YP Clients

YP servers are nodes that provide access to YP maps via the network. These maps are in */usr/etc/yp* subdirectories named after the appropriate YP domains. (See the next section, "YP Domains.")

YP clients are nodes that request access to YP maps from a YP server.

1. A YP client that is not bound sends a broadcast to all YP servers on the network.

2. The YP client **binds** to the first YP server that responds. (Each YP client binds to one YP server per YP domain.)

3. If the request is the YP client's first attempt to access data, the YP client remembers which YP server responded to the request. Subsequent requests by this YP client go directly to this YP server.

4. If the bound YP server is down or unavailable, the YP client automatically rebinds to the first YP server that responds to another broadcast.

Note

A YP client can also be configured as any combination of a YP server, NFS client, or NFS server.

A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

YP Domains

A **YP domain** is a logical grouping of the set of maps contained on YP servers. You can have different YP domains for multiple sets of nodes on the LAN without worrying about the maps interfering with each other.

- Each one of the nodes within the same YP domain must have the same domain name.

- Maps using the same name in different YP domains can have different contents.

You implement a YP domain as a subdirectory of */usr/etc/yp* on each YP server; the name of this subdirectory is the name of the YP domain. For example, maps in the *research* YP domain would be in */usr/etc/yp/research*. (Note, YP domain names are case sensitive.)

The */etc/netnfsrc* file usually contains the default YP domain name. You can change the default by executing the *domainname(1)* command or by editing */etc/netnfsrc* and then rebooting the system.

YP Masters and YP Slaves

Only two types of nodes have YP databases: master and slave servers.

The **YP master server** is the node on which YP maps are built from ASCII files; it therefore, contains the master databases (maps) which other YP servers (slaves) copy. Note, the YP master server may also provide YP clients access to YP maps.

Note

You should **create and modify YP databases only on the YP master server**; otherwise, all YP databases will not be consistent across the YP servers.

The **YP slave servers** are the nodes that receive the propagated maps from the YP master server. In turn, they provide YP clients access to YP maps.

Though a YP server may be master for one map and slave for another, random assignment of maps to YP master servers may cause confusion. Therefore, only one YP server should be the master for all maps within a YP domain.

Virtual Home Environment (VHE)

Virtual Home Environment (VHE) is an HP-developed service that allows you to configure your login environment on remote nodes to mirror the login environment on your home node. (Home node refers to the node on which your home directory physically resides.) VHE is an optional service that is available to any HP-UX system that has the NFS product. It may also be used with other UNIX systems that support symbolic links and NFS.

If you find that you never need to work from a remote node, you may want to skip this section.

VHE Advantages

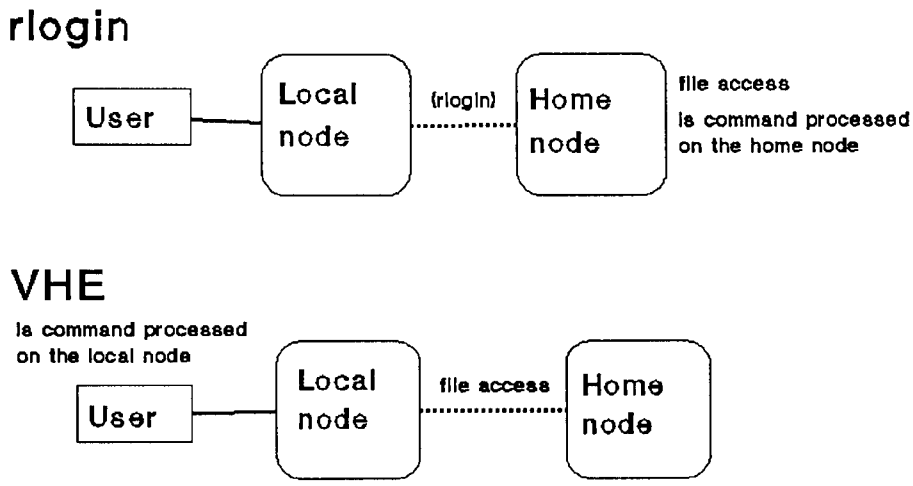
VHE's major advantage is that you can sit down at any remote node (assuming you have login permission), login, and enter into the work environment that is associated with the login on your home node (your home directory as specified in */etc/passwd*). This includes:

- home shell configuration (i.e., whichever shell you are configured to use on your home node appears when you login to a remote node).
- access to files on the file systems exported for VHE on any computers connected with VHE on the network to which you have a login and file access permission.
- use of previously defined aliases (only for C or K shells) and shell variables.
- use of customized shell scripts (assuming shells operate similarly on your home node and the node you are currently using).
- use of compiled files under your home directory from your home node (assuming your home node and the node you are logged into are of the same architecture and operating system).

Thus, VHE allows you to minimize the number of computer interfaces you must learn to be productive on the various computers that are running NFS on your network and **you are no longer tied to a particular computer to complete your work tasks.**

Another advantage of VHE is that it distributes computational work more efficiently between nodes than ARPA/Berkeley terminal emulation services such as *telnet* or *rlogin*. Unlike *telnet* or *rlogin*, VHE does not return to your home node, that contains your home environment login, to execute tasks.

Instead, VHE takes advantage of the computing capacity of the machine you are currently using. For example, if you use VHE on a node other than the home node and perform an *ls* command of a directory on the home node, the *ls* command is executed from the **local** */bin* directory. VHE does not return to your home node's */bin* directory to execute the *ls* command. The following figure illustrates this concept.



Comparison: VHE and rlogin Performing an ls command

VHE Disadvantages

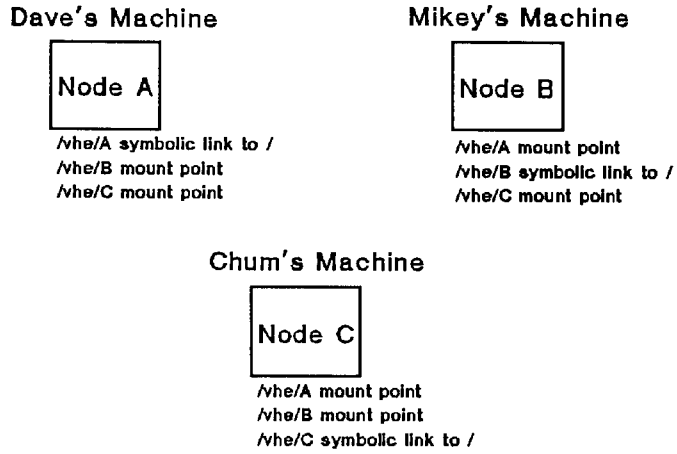
VHE has the following disadvantages.

- Though you can edit source code files originating from different types of computers on the network, you will not be able to **execute** object code files from a computer of a different architecture using VHE. For example, consider the following: you are currently working on an HP 9000 Series 300 and running VHE, and your home node is an HP 9000 Series 800 machine. If you try to execute an object code file on the HP 9000 Series 300 machine from the Series 800 machine it will not succeed. However, you can execute a script from the Series 800 machine.

- If you specify pathnames or hardware attributes in your host's *.profile* or *.login* files, you may have to modify these files to use VHE effectively. For example, the *.login* file needs to prompt for the terminal type if you plan to use VHE from more than one terminal or display type. If you do not already have this capability, then look in the sample */etc/d.login* or */etc/d.profile* files for samples of how to do this.

How VHE Works

The following diagram illustrates the directory structure of nodes in a network using VHE.



Directory Structures of Nodes Using VHE

Each node is connected to the others via NFS Services. In the picture, each node is a home node for a different user (Dave, Mikey and Chum). Each user has a customized work environment set up by the login process. Directories on each home node correspond to each of the remote nodes. For example, on node A there is a directory */vhe/B* that corresponds to node B. Using these directories as mount points, a mount is done by each node to each remote node. (The definitions of mounts and mount points are included in the "Glossary." More detailed information is contained in the "NFS Configuration and Maintenance" chapter).

Using VHE gives each node access to file systems located on the remote nodes. To maintain consistency when an individual is logged in to his or her home node, a symbolic link (a pointer) points to the host's root directory.

In a single node HP-UX configuration, the */etc/passwd* file contains the directory that becomes the home directory for the user upon logging in. For use with VHE, */etc/passwd* is edited such that all of the home direc-

tories are prefixed with a mount point or a symbolic link. When the login program performs a *cd* to the user's home directory, the *cd* and subsequent requests are made to the users home node via NFS Services unless logging in on your home node.

Example Grouping

In the */etc/passwd* file, the appropriate mount point or symbolic link is added to the beginning of the pathname of the home directory for each user. The example below shows how the lines in */etc/passwd* would look for the users Dave, Mikey and Chum as shown above.

```
dave::117:100:Dave: /vhe/A/users/dave:/bin/csh
mikey::118:100:mikey  Pom  :/vhe/B/users/mikey:/bin/sh
chum::119:200:chum  Pom:/vhe/C/users/chum:/bin/ksh
```

No matter which node Dave logs in on, his home directory is */users/dave* on node A. When scripts such as *.login* or *.cshrc* are executed, they define the execution environment as customized by Dave. His files, shell variables and aliases are available just as if he had physically logged in on node A.

Because VHE is not a virtual terminal program, when Dave executes processes, they are executed on the node he is logged into. If he is on node B, processes are executed on node B, not his native host A. For example, consider the following. Dave is working at node B and his system administrator has configured VHE to be running. Dave does the following command on node B:

```
cc testfile.c
```

The *cc* from node B's */bin* directory is executed, but *testfile.c* is used from Dave's current working directory on node A.

3

Common Commands

This chapter describes how to access files using NFS. It also explains how to use common NFS and Yellow Pages (YP) commands.

Note

All references to **servers** and **clients** apply to NFS servers and clients unless preceded by **YP**.

Key Terms

Client

- A node that requests data or services from other nodes (servers).
- A process that requests other processes to perform operations.

Note: An NFS client can also be configured as any combination of an NFS server, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Export To make a file system available to remote nodes via NFS.

File System An entire unit (disk) that has a fixed size.

Host A node that has primary functions other than switching data for the network.

Internet Address A four-byte quantity that is distinct from a link-level address and is the network address of a computer node. This address identifies both the specific network and the specific host on the network.

Key (YP) A string of characters (no imbedded blanks or tabs) that indexes the values within a map so the system can easily retrieve information. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.

Map (YP) A file consisting of logical records; a search key and related value form each record. YP clients can request the value associated with any key within a map.

YP map is synonymous with **YP database.**

Map Nickname (YP) — A synonym for the YP map name when using certain YP commands.

Master Server (YP) — The node on which one or more YP maps are constructed from ASCII files. These maps are then copied to the YP slave servers for the YP clients to access.

Mount — To obtain access to a remote or local file system or directory (import).

Mount Point — The name of the directory on which a file system or part of a file system is mounted.

Node — A computer system that is attached to or is part of a computer network.

Server

- A node that provides data or services to other nodes (clients) on the network.
- A process that performs operations as requested by other processes.

Note: An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Value (YP) A unit of information stored in YP maps; each value has a corresponding key (index) so the system can easily retrieve it. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.

Yellow Pages (YP) An optional network service composed of databases (maps) and processes that provide YP clients access to the maps. The YP service enables you to administer these databases from one node.

YP may or may not be active; check with your system administrator.

YP Client

- A node that requests data or services from YP servers.
- A YP process that requests other YP processes to perform operations.

Note: A YP client can also be configured as any combination of a YP server, NFS client, or NFS server. (A YP server **must** also be configured as a YP client.)

YP Database See "Map (YP)."

YP Domain A logical grouping of YP maps (databases) stored in one location. YP domains are specific to the YP network service and are not associated with other network domains.

YP Map See "Map (YP)."

YP Password The password for a user's login ID that exists in the YP *passwd* map. The password is the same one as the user password, but is administered through the YP.

You do not have to have a password to access the YP databases.

YP Server

- A node that provides data (maps) or services to other nodes (YP clients) on the network using YP.
- A YP process that performs operations as requested by other YP processes.

Note: A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

NFS Commands

Use this section to understand how to access files via NFS and how to use the following NFS commands. Refer to the *NFS Services Reference Pages* for detailed explanations of the commands and all their options.

- *on* command
- *rpcinfo(1M)*
- *rup(1)*
- *rusers(1)*
- *showmount(1M)*

NFS Remote File Access

NFS allows many users to share the same files. Since access techniques are transparent, remote file access remains similar to local file access.

The super-user must perform two actions before you can access remote files via NFS.

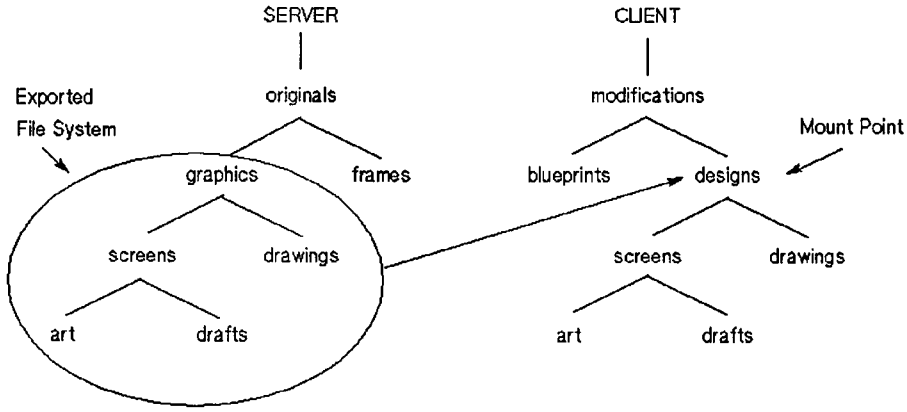
- On the server, export the file system (i.e., make it available) to the client
- On the client, mount (import) the file system

Access to remote files is the same as for local files. You need to include either the complete path name starting with / (slash) or the path name relative to the current directory.

Note

If operating in an HP-UX cluster environment and accessing a *CDF* (context dependent file) via an NFS mount, the *CDF* member is chosen based on the context of the NFS server, not the accessing node. Since this access method may return unexpected results, HP recommends you do **not** use ***CDF*s with NFS.**

EXAMPLE:



Example NFS Remote File Access	Example Entry
Edit the *drafts* file on the **server.**	`server% vi /originals/graphics/screens/drafts`
Edit the *drafts* file on the **client.**	`client% vi /modifications/designs/screens/drafts`
While in the *frames* directory, the **server** copies the *art* file from the *screens* directory.	`server% cp /originals/graphics/screens/art art`
While in the *blueprints* directory, the **client** copies the *art* file from the *screens* directory.	`client% cp /modifications/designs/screens/art art`
While in the *screens* directory, the **server** copies the *art* file to the *frames* directory.	`server% cp art /originals/frames`
While in the *screens* directory, the **client** copies the *art* file to the *blueprints* directory.	`client% cp art /modifications/blueprints`

rpcinfo

Execute *rpcinfo(1M)* to determine which remote programs are registered with a system's *portmap(1M)* daemon.

By providing a host name, you can list the registered RPC programs on a specific host. If you do not specify a host name, *rpcinfo(1M)* defaults to the local host.

To list the program, version, and protocol numbers, use the *-p* option.

EXAMPLE: **Execute:** `rpcinfo -p` *node_7*

System Response:

```
program vers proto  port
   100003    2   udp  2049  nfs
   100004    2   udp  1028  ypserv
   100004    2   tcp  1027  ypserv
   100004    1   tcp  1027  ypserv
   100007    2   tcp  1028  ypbind
   100007    1   udp  1037  ypbind
   100001    3   udp  1069  rstatd
   100002    1   udp  1073  rusersd
   100002    2   udp  1073  rusersd
   100005    1   udp  1076  mountd
   100008    1   udp  1078  walld
   100012    1   udp  1080  sprayd
```

To see if a particular remote program and version is available using UDP, use the *-u* option.

EXAMPLE: **Execute:** `rpcinfo -u` *node_2 mountd 1*

System Response: This response indicates that the *portmap(1M)* daemon that the system knows about program #100005 and that it is available.

```
program 100005 version 1 ready and waiting
```

rup

Execute *rup(1)* to list host information, including how long they have been running, how many users are logged on to them, and their load average. By providing a host name, you can list information about a specific host.

Executing *rup(1)* without providing a host name causes an RPC broadcast. The local node collects responses until the RPC times out (quits). This process generally takes about two minutes.

EXAMPLE: **Execute:** `rup` *node_1 node_2 node_3 node_4*

System Response: The last three columns of this response show the load averages for 1, 5, and 15 minute intervals.

```
node_1  up          15:53,  loadaverage:0.11,0.17,0.15
node_2  up  2 days,  19:42,  load average: 0.00, 0.01, 0.01
node_3  up  21 days, 11:34,  load average: 1.66, 1.68, 1.60
node_4  up          19:24,  loadaverage:0.14,0.18,0.14
```

To sort the display by "up time," use the *-t* option.

EXAMPLE: **Execute:** `rup -t`

System Response:

```
collecting responses...
    node_7  up  21days,  11:28,  load average:1.16,1.42,1.52
11.2.33.44  up  12 days,  22:15,  load average: 1.08, 0.82, 0.57
    node_8  up  7days,   18:27,  load average:0.12,0.09,0.09
   node_12  up  6days,   21:20,  load average:0.10,0.08,0.09
55.6.77.88  up  3 days,  3mins,  load average: 0.00, 0.01, 0.01
    node_6  up  2days,   22:49,  load average:0.00,0.00,0.02
99.0.11.22  up           18:14,  load average:0.00,0.00,0.05
33.4.55.66  up           0 min,  load average: 014, 004, 002
```

rusers

Execute *rusers(1)* to list the host names and users logged in for all remote nodes. By providing a host name, you can list information about a specific remote node.

Executing *rusers(1)* without providing a host name causes an RPC broadcast. The local node collects responses until the RPC times out (quits). This process generally takes about two minutes.

EXAMPLE: **Execute:** `rusers`

System Response: This response displays the host name or internet address in the first column and the users in the second column.

```
77.8.99.00    root
node_6        user_4 user_3 user_8 user_11
node_3         u_2
node_1         u_7
11.2.33.44    root root
node_2        root u_5 root
node_16      test_user
node_9        rootx root u_7 root
node_7         root
```

You can list more extensive information by using the *-l* command: user, host, tty (terminal), login date and time, and idle time (in minutes and seconds).

EXAMPLE: **Execute:** `rusers -l` *node_8 node_4*

System Response: The last two columns in this response show the login date and time followed by the idle time.

```
rootx   node_8:console        Apr 07 14:00   20:29
user_3  node_4:tty03          Apr 12 08:09    :23
```

showmount

Execute *showmount(1M)* to list all the clients that have remotely mounted a file system. By providing a host name, you can specify the host. If you do not specify a host name, *showmount(1M)* defaults to the local host. For example, you might want to determine which nodes have your file systems mounted.

To print all remote mounts in a *client:directory* format, use the *-a* option. The directory listed is the root of the file system that was mounted.

EXAMPLE: **Execute:** `showmount -a`

System Response: This response displays the client followed by the directory.

```
node_4:/tmp
node_7:/
node_2:/tmp
node_12:/usr/tmp/sys_rick
node_6:/tmp/y
node_8:/
```

To print a list of exported file systems, use the *-e* option.

EXAMPLE: **Execute:** `showmount -e` *node_7*

System Response:

```
export list for  node_7:
/                node_31 node_32 node_11 node_6
/users/proj      node_8 node_12
```

Remote Execution Facility (REX)

REX permits you to execute commands on a remote host in an environment similar to your own.

on command

Execute the *on* command to provide the user interface for remote execution of commands. The *on* command simulates your current environment on the server. For information on REX and the *on* command, refer to the "*Remote Execution Facility*" chapter in this manual.

Yellow Pages Overview

The Yellow Pages (YP) is an **optional** network database service that enables YP clients to access information from any correctly configured YP server on the network. One YP master server can automatically propagate modifications across the network.

YP Maps

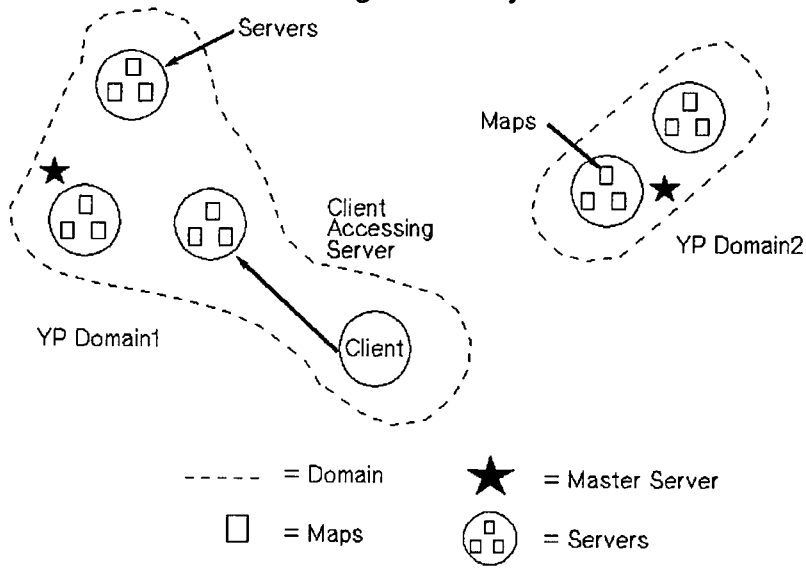
The YP system stores information in YP **maps** (databases) that are consistent across the nodes. Each map has a unique, case-sensitive map name that is used for accessing maps.

Each map consists of **keys** (for indexing) and **values** (data). You can use YP commands for querying for values associated with a particular key within a map and for retrieving key-value pairs within a map.

YP Servers and YP Clients

YP servers store and provide access to the YP maps (databases); **YP clients** request data from the maps residing on YP servers. Since different YP servers have consistent YP maps, responses are identical no matter which YP server answers a request.

- A YP client can also be configured as any combination of a YP server,



Yellow Pages Overview

NFS client, or NFS server.

- A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

YP Domains

A **YP domain** is a logical grouping of YP maps; each YP server contains a set of maps for at least one YP domain.

YP domains enable maps with the same names to exist on one LAN; the maps are made unique by belonging to separate YP domains. With

YP domains, you need not worry about the maps interfering with each other.

- Each of the nodes within a YP domain has the same YP domain name.
- Maps using the same name in different YP domains may have different contents.

YP domains are implemented as subdirectories of */usr/etc/yp* on the YP servers only; the name of each subdirectory is the name of a YP domain. For example, maps in the *research* YP domain would be in */usr/etc/yp/research*. (Note, YP domain names are case sensitive.)

YP Master and Slave Servers

Only two types of nodes have YP databases: master and slave servers.

The YP **master server** is the node on which all YP maps within a particular YP domain are created and modified. As modifications occur, the YP **slave servers** copy the maps to ensure all YP databases are alike; in turn, they provide resources to the YP clients.

Note, YP clients can bind to both YP master and slave servers.

YP Commands

Since YP hides the details of how and where data is stored, you do not need to know all the configuration details to access information. You can, however, use the following commands to determine the location and content of YP information.

- *domainname(1)*
- *ypcat(1)*
- *ypmatch(1)*
- *yppasswd(1)*
- *ypwhich(1)*

domainname

Execute *domainname(1)* to display the current YP domain name.

```
domainname
```

For example, you might need to determine the current YP domain name to define a netgroup in */etc/netgroup*. (Netgroups are network-wide groups of nodes and users defined in */etc/netgroup* on the master server.)

ypcat

Execute *ypcat(1)* to list the contents of a specified YP map. You can use either the map name or map nickname to specify the desired map.

EXAMPLE: **Execute:** `ypcat group.byname` **or** `ypcat group`

System Response: This response displays the group name, the group ID (GID), and the members of the group.

```
daemon::5:notes,anon,uucp
users::23:window,nowindow
other::1:root,daemon,uucp,who,date,dooley,sync
root::0:root
mail::6:root
sys::3:root,bin,sys,adm
rje::8:rje,shqer
bin::2:root,bin,daemon,lp
adm::4:root,adm,daemon
```

To list the map nicknames applicable to the *ypcat(1)* command, use the *-x* option.

EXAMPLE: **Execute:** `ypcat -x`

System Response:

```
Use  "passwd" for  map "passwd.byname"
Use  "group" for map "group.byname"
Use "networks" for map "networks.byaddr"
Use "hosts" for map "hosts.byaddr"
Use "protocols" for map "protocols.bynumber"
Use "services" for map "services.byname"
Use "aliases" for map "mail.aliases"
Use  "ethers" for map "ethers.byname"
```

ypmatch

Execute *ypmatch(1)* to print the data (values) associated with one or more keys in a specified YP map. You can use either the map name or map nickname to specify the desired map.

To list the map nicknames applicable to the *ypmatch(1)* command, use the
-x option.

EXAMPLE: **Execute:** `ypmatch` *my_node* hosts.byname

System Response: This response displays the internet address (value) associated with the *hosts.byname* map for the node *my_node*.

```
11.2.33.44  my_node
```

yppasswd

The YP password is the password for a user's login ID that exists in the YP *passwd* map. It is used as the user password, but is administered through YP. Note, you do not have to have a password to access the YP databases.

If you change your password with the *passwd(1)* command, you will change only the entry in your local */etc/passwd* file if the entry exists. If your password is not in the file, the following error message occurs when using *passwd(1)*.

```
Permission denied.
```

If this error occurs, execute *yppasswd(1)*.

YP Password Guidelines

Execute *yppasswd(1)* to change or install a password associated with a specified login name in the YP *passwd* map.

The following list provides the requirements for creating and changing YP passwords. Note, these guidelines are different from those of *passwd(1)*. (Refer to the "HP NFS Services vs. Local HP-UX" appendix.)

- Only the owner or super-user can change a YP password. The super-user must know the current YP password to change another user's YP password.
- Only the first eight characters of the YP password are significant; the rest are truncated.
- A YP password must contain at least five characters if it includes a combination of either
  - uppercase and lowercase letters or
  - alphanumeric characters
- A YP password must contain at least four characters if it includes a combination of uppercase letters, lowercase letters, and numeric characters.
- A YP password must contain at least six characters if it includes only monocase letters.

YP Password

Use the following steps to create or change your YP password in the YP *passwd* map.

1. Execute the *yppasswd(1M)* command.

   ```
   yppasswd user_login_name
   ```

2. The system prompts you for the old YP password even if one does not exist. If it exists, enter the old YP password; otherwise, press **RETURN**.

   Note, the YP password may be different from the one in your local */etc/passwd* file.

3. The system prompts you for the new YP password twice to ensure you enter the correct response. Enter your new YP password twice, pressing **RETURN** after each entry.

   The system now updates the master server *passwd* map.

EXAMPLE: **Execute:** `yppasswd`

System Response:

```
Old YP password:
New password:
Retype new password:
The YP passwd has been changed on host_name, the master YP passwd
server.
```

ypwhich

Execute *ypwhich(1)* to print the host name of the YP server supplying YP services to a YP client.

To list all available maps and their YP master server host names, use the *-m* option. You can use either the map name or map nickname to determine which YP server is the master server for a specified YP map.

To list the map nicknames applicable to the *ypwhich(1)* command, use the *-x* option.

EXAMPLE: **Execute:** `ypwhich -m`

System Response: This response displays the available maps and their YP master server host names.

```
services.byname     node_1
rpc.bynumber        node_1
protocols.bynumber  node_1
protocols.byname    node_1
passwd.byuid        node_1
passwd.byname       node_1
networks.byname     node_1
networks.byaddr     node_1
netgroup.byuser     node_1
netgroup.byhost     node_1
netgroup            node_1
hosts.byname        node_1
hosts.byaddr        node_1
group.byname        node_1
group.bygid         node_1
vhe_list            node_1
ypservers           node_1
```

4

Installation

The following steps are a checklist of NFS installation procedures. You may have already completed several of these steps. You will most likely start with Step 4. Steps 4 through 6 are explained in detail in this chapter.

1. Prepare your HP 9000 Series 300 system for operation. Refer to *Installing and Maintaining NS-ARPA Services* documentation.
   - Inspect hardware
   - Create and maintain a network map

2. Ensure your computer is running the HP-UX 6.0 operating system or a later version. If you do not know which system your computer is running, execute the *uname -r* command. Call your HP representative if you do not have HP-UX 6.0 or a later version.

   If you are installing HP-UX for the first time, refer to the *HP-UX Installation Manual*. If you have HP-UX and are upgrading to a later version, refer to the *HP-UX Series 300 System Administrator Manual, Vol. I, or to the AXE User's Manual* for information on the *update* procedure.

3. Ensure your computer is running the NS-ARPA Services product. Refer to *Installing and Maintaining NS-ARPA Services* documentation.

4. Install the NFS Series 300 software. You will need to use the *update* procedure to install the NFS Series 300 software. If you do not already understand the *update* procedure, refer to the *HP-UX Series 300 System Administrator Manual, Vol. I,* or to the *AXE User's Manual* for detailed *update* information. Refer to the "Install Software" section of this chapter for brief instructions.
   - Use the */etc/update* command
   - Configure the new kernel

5. Add your HP 9000 Series 300 computer to the network. Refer to *Installing and Maintaining NS-ARPA Services* documentation.
   - Assign an internet address
   - Create device files for the node

6. Relink programs that access remote directories. If you will be using the YP (Yellow Pages) Service, relink programs that call C library routines which access YP files.

Key Terms

CDF Context Dependent File.

A hidden directory that contains all the versions of a file needed by the different cnodes.

Cluster One or more workstations linked together with a local area network (LAN), but consisting of only one root file system.

Cnode Any node operating in an HP-UX cluster environment, including diskless nodes and the root server.

Diskless Cnode A node in an HP-UX cluster that uses networking capabilities to share file systems, but does not have a file system physically attached.

Internet Address A four-byte quantity that is distinct from a link-level address and is the network address of a computer node. This address identifies both the specific network and the specific host on the network.

LAN Local Area Network.

NFS Network File System.

Node A computer system that is attached to or is part of a computer network.

Root Server The only node in an HP-UX cluster that has file systems physically attached to it.

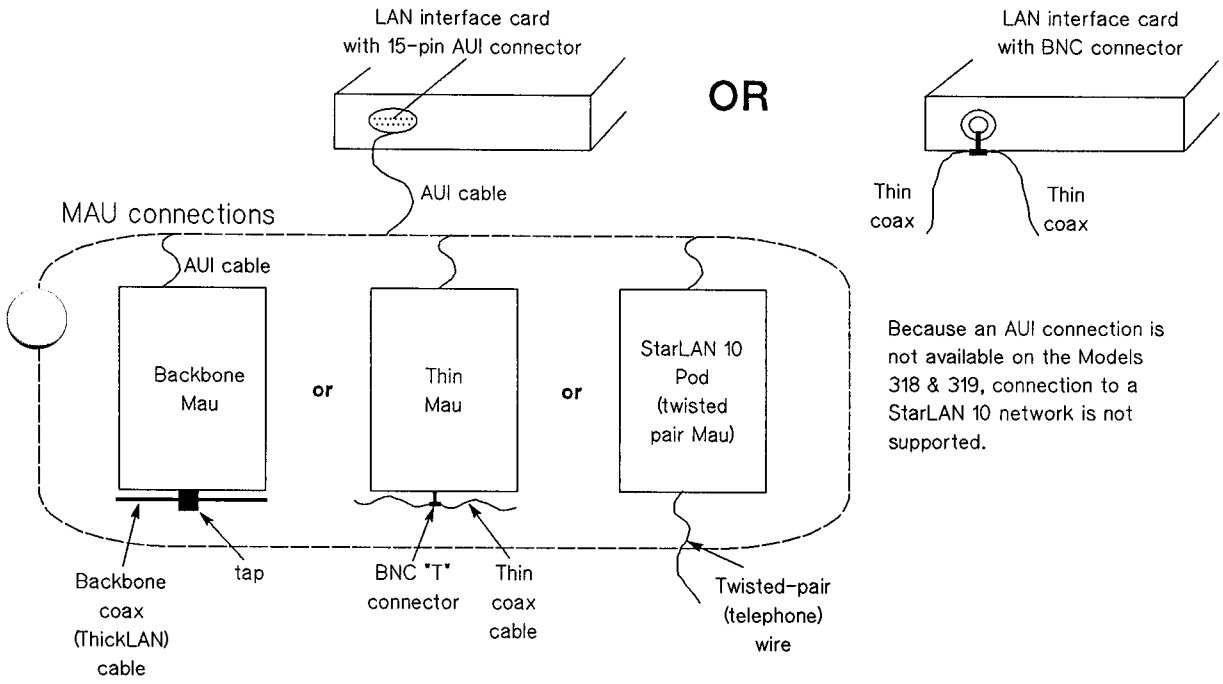
Update The HP-UX command that installs software onto the system.

Yellow Pages (YP) An optional network service composed of databases (maps) and processes that provide YP clients access to the maps. The YP service enables you to administer these databases from one node.

Prepare the System

To prepare your HP 9000 Series 300 computer for operation on the LAN, you must ensure your LAN hardware is installed correctly. For LAN hardware installation instructions for the Series 300 computer, refer to the following documentation.

- *HP 98643A LAN/300 Link LANIC Installation Manual*
- *LAN Cable and Accessories Installation Manual*



Series 300 LAN Interfaces and Cable Connections

HP 9000 Series 300 Models 318, 330, and 350 are shipped with an installed interface card that is functionally equivalent to an HP 98643A LAN/300 Link Interface card and 28641A Thin MAU. A connector is

available on the backplane of these computers for attaching a T-connector. No MAU or AUI cable is required for these Series 300 models.

Another step in preparing your system is to update your network map with all new installation information (e.g., new computers, cable changes). If you do not have a network map, HP strongly recommends you create one. Refer to *Installing and Maintaining NS-ARPA Services* documentation for guidelines.

Install Software

Before installing NFS Services software, you should ensure the following items are true.

- Your computer is running the HP-UX 6.0 operating system or a later version. To check which version of HP-UX you are currently running, execute the *uname -r* command.
- The NS-ARPA Services software is installed. To verify whether the NS-ARPA Services have been installed check with your systems administrator. If you are the systems administrator, and you have not already installed the NS-ARPA Services, refer the *Installing and Maintaining NS-ARPA Services* documentation for installation and configuration instructions.

Use Update

Before installing NFS Services, refer to the *HP-UX Series 300 System Administrator Manual* to familiarize yourself with the menu operations and device file information.

After you are certain the required HP-UX filesets and NS-ARPA Services software are installed, use the */etc/update* program to install NFS Services software. The */etc/update* program takes you through the installation procedure step by step.

Note

The */etc/update* command reports a fatal error if too many processes are running on the system. Therefore, execute */etc/shutdown* before beginning */etc/update*.

---

1. From the / (root) directory, execute */etc/shutdown.*

   This command causes all processes to be killed and all file systems but the root file system to be unmounted. This takes a few minutes to complete.

2. Load the tape (or the first 3 1/2-inch disk) containing NFS Services software. Labels on the tapes and disks indicate which software products are on the media.

3. Execute the */etc/update* command.

   This causes the system to automatically re-boot.

4. *If you are installing the NFS Services stand-alone product, skip to the next step.* If you are installing NFS Services as part of a bundled system,

   Select HP_NFS from the partition menu.

5. Mark and load the NFS file sets: NFS_CMDS and NFS_INCL.

   The */etc/update* program lists each file set as it is loaded. **Note:** if you are installing NFS Services as part of a bundled system, you may notice that other file sets are loaded at this time. This occurs automatically as part of a bundled system installation.

6. *If you are using only the AXE product, skip to the next step.*
   If you are using the Programming Environment,

   Mark and load the NFS_MAN file set.

   This causes the unformatted manual pages, for use in the Programming Environment (PE) to be loaded. **Note:** if you are installing NFS Services as part of a bundled system, you may notice that other file sets are

loaded at this time. This occurs automatically as part of a bundled system installation.

7. *If you are using the Programming Environment (PE), skip to the next step.*
   If you are using only the AXE product, then.
   Mark and load the NFS_MANC file set.

   This causes the formatted manual pages for the AXE product to load.

8. After installing the manual pages, select QUIT until you exit */etc/update* (i.e., you may need to choose QUIT twice).

9. When you exit */etc/update,* the system automatically reboots. Wait for the system to reboot.

10. After the system reboots, login as super-user and display the */etc/newconfig* directory. The following files were added to the */etc/newconfig* directory as a result of the installation. You will use these files when you configure NFS Services and Yellow Pages, which is described later in this manual.

    - */etc/newconfig/netgroup*
    - */etc/newconfig/netnfsrc*
    - */etc/newconfig/rpc*
    - */etc/newconfig/vhe_mounter*
    - */etc/newconfig/vhe_u_mnt*
    - */etc/newconfig/ypinit*
    - */etc/newconfig/yp_Makefile*
    - */etc/newconfig/ypmake*
    - */etc/newconfig/ypxfr_1perday*
    - */etc/newconfig/ypxfr_1perhour*
    - */etc/newconfig/ypxr_2perday*

For descriptions of the files listed above and other files in */etc/newconfig,* refer to */etc/newconfig/README.*

Note

You have completed installing the NFS Services product. To prepare to configure the services, you will need to configure a new kernel. To configure a new kernel, refer to the next section.

Configure a New Kernel

To prepare the NFS Services product for use, you must configure a new HP-UX operating system kernel either by executing */etc/reconfig* or by editing */etc/conf/dfile* and executing */etc/config*.

- Execute the */etc/reconfig* command if you created the existing kernel (*/hp-ux*) either via */etc/reconfig* or via */etc/update* using */etc/conf/dfile* .

  The */etc/reconfig* program prompts you through all the necessary steps. Note, executing *reconfig* moves the existing */etc/conf/dfile* to */etc/conf/dfile.BCKUP*.

  Refer either to the *HP-UX Series 300 System Administrator Manual* or to the *HP-UX Reference* manual, *config(1M)* section. These manuals describe how to configure a custom kernel with special parameters.

- If you are adding NFS Services to a */etc/conf/dfile* that you changed (customized) to suit to your needs, edit the */etc/conf/dfile* you are using to build the new kernel and add the *nfs* entry. To avoid losing information, HP recommends that you first copy the */etc/conf/dfile* before making changes to it.

Note

If operating in an HP-UX cluster environment, edit the */etc/conf/dfile* for every cnode, including the root server, and reconfigure a kernel for each cnode that is based on this */etc/conf/dfile*. In an HP-UX cluster environment, */etc/conf/dfile* is a *CDF* (context dependent file).

Edit the dfile

Refer to the following table if you are editing a customized */etc/conf/dfile*, rather than executing */etc/reconfig*, to configure NFS into an HP-UX operating system kernel.

Configure Kernel by editing the dfile	Comments
`cd /etc/conf`	Change the directory to the configuration directory.
`cp` *dfile dfile.save*	Save your customized *dfile* by copying it to another file.
`cp dfile.full.lan` *dfile*	Copy the *dfile.full.lan* into the customized *dfile*.
`vi` *dfile*	Edit this *dfile* to include your custom information. You must also remove the * (asterisk) from the **nfs line.*
`/etc/config` *dfile*	This *config* command creates two files: *conf.c* and *config.mk*.
`make -f config.mk`	This *make* command creates the */etc/conf/hp-ux* file, your new kernel.
`cp /hp-ux /SYSBCKUP`	Save your old kernel. **Note: If operating in an HP-UX cluster environment, do not save your old kernel on diskless cnodes.**
`cp hp-ux /hp-ux`	Copy the new kernel to the root directory.
`/etc/reboot`	Reboot the new kernel.

Add a Computer to the Network

If you have not already done so, refer to *Installing and Maintaining NS-ARPA Services* documentation for instructions on adding your HP 9000 Series 300 computer to the network. You will need to perform the following steps.

1. Determine and assign an internet address.

2. Edit netlinkrc.

3. Create device files for the node.

After rebooting the system, log in as super-user and refer to the "NFS Configuration and Maintenance," "YP Configuration and Maintenance," and "VHE Configuration and Maintenance" chapters to configure your system with NFS, YP (if applicable), and VHE.

Relink Programs

You **must** relink programs that access remote directories. Otherwise, these programs will not be able to access remote directories mounted through NFS since the old directory routines use a *read* call instead of a *getdirentries* call to access those directories.

If using YP to provide the information stored in the standard maps' ASCII files, you **must** relink any programs that read data from those files using standard C library routines. If you do not relink the programs, they will access only the local files. If the local files are not as current as the YP maps, the programs may not work correctly.

- Programs compiled with previous ARPA/Berkeley libraries *get*ent* (*gethostent*, *getnetent*, *getservent*, *getprotoent* and related routines) will not have access to the YP databases where information in files like */etc/hosts* can be stored.

- Programs compiled with the previous *getpwent* and *getgrent* routines will not have access to information (passwords and groups) stored in the global YP password and group databases.

5

NFS Configuration and Maintenance

This chapter describes a basic NFS configuration without Yellow Pages. The latter portion describes how to administer and maintain the NFS service once you have it configured. For specific NFS information, refer to the following sections.

- Key Terms
- Guidelines
- NFS Configuration
- NFS Maintenance

Refer to the *NFS Services Reference Pages* for detailed NFS information.

Note

All references to **servers** and **clients** in this chapter apply to NFS servers and NFS clients unless otherwise specified.

Key Terms

Alias A term for referencing alternate networks, hosts, and protocols names.

Client

- A node that requests data or services from other nodes (servers).
- A process that requests other processes to perform operations.

Note: An NFS client can also be configured as any combination of an NFS server, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Clock Skew A difference in clock times between systems.

Cluster One or more workstations linked together with a local area network (LAN), but consisting of only one root file system.

Cnode Any node operating in an HP-UX cluster environment, including diskless nodes and the root server.

Daemon Background programs that are always running, waiting for a request to perform a task.

Diskless Cnode A node in an HP-UX cluster that uses networking capabilities to share file systems, but does not have a file system physically attached.

Export To make a file system available to remote nodes via NFS.

File System An entire unit (disk) that has a fixed size.

GID A value that identifies a group in HP-UX.

Hard Mount A mount that causes NFS to retry a remote file system request until it succeeds, you interrupt it (default option), or you reboot the system.

Host A node that has primary functions other than switching data for the network.

Import To obtain access to a remote file system from an outside source; to mount.

Internet Address A four-byte quantity that is distinct from a link-level address and is the network address of a computer node. This address identifies both the specific network and the specific host on the network.

Interruptable Mount A mount that allows you to interrupt an NFS request by pressing an interrupt key. (Though the interrupt key is not standardized, common ones include **CTRL-C** and **BREAK**.)

Mount To obtain access to a remote or local file system or directory (import).

Mount Point — The name of the directory on which a file system is mounted.

Netgroup — A network-wide group of nodes and users defined in */etc/netgroup*.

NFS — Network File System.

Node — A computer system that is attached to or is part of a computer network.

Root Server — The only node in an HP-UX cluster that has file systems physically attached to it.

Server

- A node that provides data or services to other nodes (clients) on the network.
- A process that performs operations as requested by other processes.

Note: An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Soft Mount — An optional mount that causes access to remote file systems to abort requests after one NFS attempt.

UID — A value that identifies a user in HP-UX.

Unmount — To remove access rights to a file system or disk that was mounted via the *mount(1M)* command.

Update The HP-UX command that installs software onto the system.

Yellow Pages (YP) An optional network service composed of databases (maps) and processes that provide YP clients access to the maps. The YP service enables you to administer these databases from one node.

YP may or may not be active; check with your system administrator.

YP Domain A logical grouping of YP maps (databases) stored in one location. YP domains are specific to the YP network service and are not associated with other network domains.

Guidelines

Refer to the following guidelines for information regarding

- network memory,
- configuration files,
- daemons, and
- servers.

Network Memory

Network memory is configurable using three parameters: *netmeminit*, *netmemmax* and *netmemthresh*. The default values are generally sufficient for most NFS configurations. However, if you change these parameters, do **not** set *netmemmax* equal to or less than *netmemthresh*.

For more information, refer to the *HP-UX Series 300 System Administrator Manual*.

Configuration Files

The following table lists the files that must be configured for your system to operate correctly. (Refer to the *NFS Services Reference Pages* for detailed information.)

Configuration File	Description
/etc/checklist	Contains a list of file systems that are automatically mounted at boot time.
/etc/exports	Contains a list of file systems that clients may import. Note, create this file only on servers.
/etc/inetd.conf	Contains information about servers started by *inetd(1M)*, including RPC services.
/etc/netgroup	Contains a mapping of network group names (netgroups) to a set of node, user, and YP domain names; both */etc/exports* and */etc/passwd* can use the netgroups defined in */etc/netgroup*. Classifies the nodes for remote mounts. **For ARPA Services**, classifies the users for remote logins and remote shells. You can specify netgroups in */etc/hosts.equiv* and *$HOME/.rhosts*.
/etc/netnfsrc	Automatically executed at boot time to start the NFS networking (e.g., starts daemons and servers, defines servers and clients).

Configuration File	Description
/etc/rpc	Maps the RPC program names to the RPC program numbers and vice versa. This file is static; it is already correctly configured.
/usr/adm/inetd.sec	Checks the internet address of the host requesting a service against the list of hosts allowed to use the service. Specifies how many remote users can simultaneously start remote services in the local system and which remote hosts (or networks) can use the system.

Daemons

The following table lists the networking daemons (background programs) that are always running, waiting for a request to perform a task.

Daemon	Description
biod(1M)	Asynchronous block I/O daemons for NFS clients. **If operating in an HP-UX cluster environment,** *biod(1M)* must be running on all cnodes in the cluster.
inetd(1M)	Internet daemon that listens on service ports. • Reads */etc/inetd.conf* to determine the appropriate server for handling the incoming request • Listens for and accepts network requests • Invokes the appropriate server **Note:** Since *inetd(1M)* contacts *portmap(1M)* on behalf of the servers it starts, you **must** start *portmap(1M)* before starting *inetd(1M)*.
nfsd(1M)	NFS server daemon that responds to client file system requests. When a client program needs to read or write in a remote file system, it sends a request to that system's *nfsd(1M)* process. **If operating in an HP-UX cluster environment,** *nfsd(1M)* should be running on the root server if it is servicing NFS requests. Any *nfsd(1M)* daemons running on client cnodes are ignored.

Daemon	Description
pcnfsd(1M)	Daemon that authenticates a PC user's access to files. It takes the user name and password, and then either • succeeds (returns a valid UID and GID), or • fails (indicates the name and password are unacceptable). **Note:** Though *pcnfsd* enables PC users to use printer spooling facilities on HP-UX systems, they **must** have the appropriate PC networking software product for it to work.
portmap(1M)	Daemon that converts RPC program numbers into port numbers. When *inetd(1M)* starts, it tells *portmap(1M)* • which RPC servers it is listening for, • on which ports it is listening, and • the RPC program numbers and versions it serves. When a client makes an RPC call to a given program number, it first contacts *portmap(1M)* on the server node to determine the port number where RPC requests should be sent. **Note:** Since *inetd(1M)* contacts *portmap(1M)* on behalf of the servers it starts, you **must** start *portmap(1M)* before starting *inetd(1M)*.

Servers

The following table lists the networking servers (processes that perform operations as requested by other processes).

Server	Description
mountd(1M)	Answers file system mount requests by reading */etc/exports* to determine which file systems are available to nodes and users; invoked by *inetd(1M)*. The *showmount(1M)* command calls *rpc.mountd* to list the clients with local file systems mounted. **If operating in an HP-UX cluster environment,** *mountd(1M)* should be running on the root server if servicing NFS requests. Any *mountd(1M)* server running on a client cnode is ignored.
rstatd(1M)	Returns statistics obtained from the kernel; invoked by *inetd(1M)*. The *rup(1)* program uses *rpc.rstatd*.
rusersd(1M)	Lists the users on the local host; invoked by *inetd(1M)*. The *rpc.rusersd* server provides the *rusers(1)* program information about the local users. The *rusers(1)* program then sums and displays the information.

Server	Description
rwalld(1M)	Handles all *rwall(1M)* requests; invoked by *inetd(1M)*. The RPC program *rwall(1M)* sends a message to *rpc.rwalld* on a given host. Each *rpc.rwalld* accepts this message and writes it to all users on the host it is serving using *wall(1M)*.
sprayd(1M)	Records the packets sent by *spray(1M)*; invoked by *inetd(1M)*.

NFS Configuration

Configuring your system is the process of setting up your software so it operates correctly and according to your specifications. The following list is an overview of the steps you must complete to configure the nodes on your network with NFS Services. The steps are described in more detail after the overview list.

1. Compare the files in the */etc/newconfig* directory to their their corresponding existing files.

2. Set UIDs and GIDs

3. Create an NFS server
   1. Edit */etc/netnfsrc*
   2. Edit */etc/inetd.conf*
   3. Edit */usr/adm/inetd.sec* (if necessary)
   4. Edit */etc/netgroup*
   5. Edit */etc/hosts*
   6. Create and Edit */etc/exports*
   7. Reboot the system (if necessary)

4. Create an NFS client
   1. Edit */etc/netnfsrc*
   2. Mount file systems
   3. Reboot the system (if necessary)

5. If applicable, configure Yellow Pages (YP). (Refer to the "YP Configuration and Maintenance" chapter.)

6. If applicable, configure the Virtual Home Environment (VHE) service. (Refer to "VHE Configuration and Maintenance" chapter.)

7. Execute*/etc/netnfsrc* (or reboot) when you are finished with all of the configuration, including setting up YP and VHE.

1. Compare /etc/newconfig Files to Existing Files

When your installed the NFS services software, several new files were copied into the */etc/newconfig* directory. Perform the following steps to prepare to configure the NFS service.

1. Compare each */etc/newconfig* file listed below with its counterpart shown in the following list.

File in /etc/newconfig directory	Counterpart in /etc directory
netgroup	*netgroup*
netnfsrc	*netnfsrc*
rpc	*rpc*

2. If the files are the same, then skip to the next section, "2. Set UIDs and GIDs."

3. If you have previously customized the files that exist in the */etc* directory, or if the files are from an older version of the software they will differ from those in */etc/newconfig*. If there are differences, copy the current files in */etc* to a safe location and do **one** of the following:

   - change the versions in */etc* to reflect the differences in the files in */etc/newconfig*.

OR

   - copy the files in */etc/newconfig* to */etc*. Then re-customize the files in */etc* if necessary.

2. Set UIDs and GIDs

The UID field from an */etc/passwd* entry and the GID field from an */etc/group* entry authenticate NFS users. The client passes this UID and GID to a server for use when checking file ownership and permission.

To ensure only the users in the correct group receive the privileges set by the file's owner, edit */etc/passwd* and */etc/group* so that each user has one unique UID and one unique GID that is the same on all servers and clients.

If using Yellow Pages (YP) Service, you can configure YP so you can centrally administer */etc/passwd* and */etc/group*. Note, local UIDs and GIDs are not required if using YP.

If not using YP, you can use one of the following two methods to either create new */etc/passwd* and */etc/group* files or you can modify the existing ones.

- Create one */etc/passwd* and one */etc/group* file to ensure UIDs and GIDs are consistent for each NFS user across the network. Copy these files to all NFS network nodes.

  When updating UIDs or GIDs, you will need to recopy the files to each node. You can automate this process by using shell scripts and the NS-ARPA Services.

  A disadvantage of this method is that it gives exactly the same access to all users across the network. A user with a valid password for a super-user account would have super-user privileges on all nodes configured in this fashion.

- Edit */etc/passwd* and */etc/group* on each node to ensure UIDs and GIDs are consistent for each user across the network.

  If you modify UIDs or GIDs affecting more than one node, you will have to modify each node affected by the change. For example, if adding a new user you will need to update the */etc/passwd* and */etc/group* files residing on each system to which the new user will have access.

Though more time consuming and error prone, this method allows each system to have a different set of users.

3. Create an NFS Server

You must be super-user to create an NFS server.

To create an NFS server, complete the following steps.

1. _____ Edit */etc/netnfsrc*
2. _____ Edit */etc/inetd.conf*
3. _____ Edit */usr/adm/inetd.sec* (if necessary)
4. _____ Edit */etc/netgroup*
5. _____ Edit */etc/hosts*
6. _____ Create and Edit */etc/exports*
7. _____ Reboot the system (if necessary)

An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Note

If operating in an HP-UX cluster environment and configuring NFS on the root server, you must also configure NFS on all clients in the cluster. If the root server does not have NFS configured, then none of the clients can.

---

1. Edit /etc/netnfsrc

The */etc/netnfsrc* file activates the NFS daemons and servers.

- To define the node as an NFS server, set the *NFS_SERVER* variable to any digit other than zero.
- If the node is also a client, you may want to set the *NFS_CLIENT* variable to any digit other than zero now. (Refer to the "4. Create an NFS Client" section to complete client configuration procedures.)
- If the node is also a server for PC-NFS requests, set the *PCNFS_SERVER* variable to any digit other than zero.

Client Only	`NFS_CLIENT=1` `NFS_SERVER=0`
Server Only	`NFS_CLIENT=0` `NFS_SERVER=1`
Both Client and Server	`NFS_CLIENT=1` `NFS_SERVER=1`
Neither Client nor Server	`NFS_CLIENT=0` `NFS_SERVER=0`
PC-NFS Server	`PCNFS_SERVER=1`

You can refer directly to the comments (lines beginning with # (pound) signs) for editing instructions and for descriptions of each activity executed by */etc/netnfsrc*.

Note

If you edit this file other than specified in this document, HP recommends you incorporate personal comments for future system administration.

```
#!/bin/sh
#       netnfsrc  --      NFS startup file
##
#       Depending on the configuration parameters you set within,
#       this script sets up some or all of the following:
#*      YP specific:
#           domainname  --      the YP domain name
#
#       and starts up some or all of the following programs:
#           portmap     --      RPC (program_#,version) -> port_# mapper
#           nfsd        --      NFS daemons
#           biod        --      async BIO daemons
#           pcnfsd      --      PC-NFS daemon
#*      YP specific:
#           ypbind      --      YP client process (all YP nodes)
#           ypserv      --      YP server process (YP server only)
#           yppasswdd   --      YP password daemon (YP master server only)
##
#       NFS_CLIENT      --      1 if this node is an NFS client, 0 if not
#       NFS_SERVER      --      1 if this node is an NFS server, 0 if not
#       Note:   it is possible for one host to be a client, a server, both
#               or neither!  This system is an NFS client if you will be
#               NFS mounting remote file systems; this system is a server
#               if you will be exporting file systems to remote hosts.
#       See Also: nfsd(1M), mount(1M)
##
#       Note:   this has nothing to do with whether or not the system is
#               a rootserver or diskless client workstation.  There is a
#               test for this later (to set the variable LDISK).
##
NFS_CLIENT=0
NFS_SERVER=0
.
.
.
.
PCNFS_SERVER=0
```

2. Edit /etc/inetd.conf

To activate the RPC services, remove all # comment marks (pound signs) from */etc/inetd.conf* lines beginning with *#rpc*. If you want one of these services activated but the line was removed, you may need to obtain a new version of */etc/inetd.conf* from */etc/newconfig*.

Note

After editing */etc/inetd.conf*, you must reconfigure *inetd(1M)* as follows.

```
inetd -c
```

RPC Services Security

The *inetd(1M)* security facility works only when the *inetd(1M)* executes a server. For the RPC services that do not exit after each service request, *inetd(1M)* provides a security check only for the first request. Successive requests bypass the *inetd(1M)* and are subject only to the security checking performed by the individual RPC services. However, you can make the *inetd(1M)* perform a security check for every RPC request by

- adding the *-e* option to the */etc/inetd.conf* entry for the RPC service and
- then specifying the RPC service in the first field of */usr/adm/inetd.sec*. (Refer to the next section, "3. Edit /usr/adm/inetd.sec.")

Note, adding the *-e* option makes the RPC server respond slower since it has to restart for each request.

RPC Entries

Refer to the following list for a brief description of each RPC service line present in */etc/inetd.conf*.

```
rpc dgram udp wait root /usr/etc/rpc.mountd 100005 1 rpc.mountd -e
```

The *rpc.mountd* program reads */etc/exports* to see what the available file systems are and to whom they are exported. It also keeps a list of all mounted file systems. The program supports version 1.

The *-e* option forces *inetd(1M)* to perform a security check for *rpc.mountd* on every request.

```
rpc stream tcp nowait root /usr/etc/rpcrexd 100017 1 rpc.rexd
```

The *rpc.rexd* program is the server for the *on(1)* program. The program supports version 1.

```
rpc dgram udp wait root /usr/etc/rpc.rstatd 100001 1-3 rpc.rstatd
```

The *rpc.rstatd* program provides kernel statistics. The program supports versions 1 through 3.

```
rpc dgram udp wait root /usr/etc/rpc.rusersd 100002 1-2 rpc.rusersd
```

The *rpc.rusersd* program provides information about active users on remote nodes and the amount of time they have been idle. The program supports versions 1 and 2.

```
rpc dgram udp wait root /usr/etc/rpc.rwalld 100008 1 rpc.rwalld
```

The *rpc.rwalld* program writes a message sent by *rwall(1M)* to all users logged on to the system. The program supports version 1.

```
rpc dgram udp wait root /usr/etc/rpc.sprayd 100012 1 rpc.sprayd
```

The *rpc.sprayd* program accepts RPC requests, reads UDP packets, and then tells how fast it read them; you can use the results to gauge performance. The program supports version 1.

3. Edit /usr/adm/inetd.sec (if necessary)

NFS operates under the assumption you have a "friendly" network; meaning, you can trust all users attached to your network. Since this assumption may not apply to everyone, refer to the following sections to improve your file security.

The */usr/adm/inetd.sec* configuration file is provided in the **NS-ARPA Services** product. It is **not** solely for NFS access.

This file allows you to determine

- how many remote services can run simultaneously on the local host and
- which hosts are allowed to remotely use the local host.

Note

If *inetd(1M)* is running, it rereads */usr/adm/inetd.sec* after you make changes to it. Your changes apply only to services started after the file is reread, but not to any currently running services.

Set Maximum Number of Remote Connections

On the first line in */usr/adm/inetd.sec*, enter the maximum number of simultaneous remote services to be started by *inetd(1M)*.

```
MAXNUM number
```

If you do not specify a MAXNUM value, the default is 1000.

Specify Accesses to Services

Each entry in */usr/adm/inetd.sec* has the following format.

service_name allow/deny host_specifier(s)

(**Note:** Enter only *allow* or *deny*)

/usr/adm/inetd.sec Entry Fields	Description
service_name	Name of a valid service (including RPC services) with an entry in */etc/inetd.conf*. • For RPC services, *service_name* is the name of the service that matches its program number in */etc/rpc*. This entry **must** have a corresponding entry in */etc/inetd.conf* which contains the *-e* option. For other services, *service_name* is the one listed in */etc/services*. • Specify only one service per entry. • If a remote service is not specified in */usr/adm/inetd.sec*, then it is not restricted by *inetd(1M)*. • If an entry in */usr/adm/inetd.sec* specifies the service name and nothing else, *inetd(1M)* allows all hosts to attempt access.
allow/deny	The *allow* entry instructs *inetd(1M)* to approve the host or network for access to the specified service. The *deny* entry instructs *inetd(1M)* to disapprove the host or network for access to the specified service.

host_specifier(s) Name of a host or a network listed in */etc/hosts* or */etc/networks*, or an internet address in the standard internet notation.

- You can specify more than one host or network by separating each *host_specifier* with a blank or tab.
- You can use the * (wild card character) or - (range character) in any field of a network or host address.
- You cannot use aliases.

RPC Services Security

You can make *inetd(1M)* perform its *inetd.sec(4)* security check for every RPC request by following these two steps.

1. Add the *-e* option to the RPC service line in */etc/inetd.conf*. (Refer to the "2. Edit /etc/inetd.conf" section or *inetd.conf(4)*.)

   EXAMPLE: `rpc dgram udp wait root /usr/etc/rpc.mountd 100005 1 rpc.mountd -e`

2. Specify the RPC service in the first field in */usr/adm/inetd.sec*.

/usr/adm/inetd.sec Example RPC Entry	**Effect on System Security**
`mountd allow hostA`	Allows only *hostA* to access *rpc.mountd*
`walld deny 111.56.78.9 10.*`	Denies access to *rpc.rwalld* from the following hosts • 111.56.78.9 (internet address) • all hosts that are part of network 10.*

4. Edit /etc/hosts

Note

If YP is running, do not edit */etc/hosts* on any node except the YP master server; otherwise, local changes will not be propagated.

Each file system in */etc/exports* may be followed by a netgroup name or a host name. If you want to export local file systems to a specific host, the host name **must** have an entry in */etc/hosts* regardless of whether it is specified as part of a netgroup in */etc/netgroup*.

Ensure that a line with the following format exists in */etc/hosts* for each host to which you will be exporting file systems.

internet_address host_name

- The *internet_address* uniquely identifies the node and must be in decimal dot notation. (Refer to *hosts(4)* for internet address information.)

  EXAMPLE: `192.45.36.5 node_7`

- The *host_name* is the node name.

The root user should own */etc/hosts* with the permission 0444 (`r--r--r--`).

5. Edit /etc/netgroup

Note

If YP is running, do not edit */etc/netgroup* on any node except the YP master server; otherwise, local changes will not be propagated.

The */etc/netgroup* file enables you to define a specific network-wide group of nodes as a **netgroup.** You can then limit file system access by exporting file systems (via */etc/exports*) to the netgroups defined.

The system uses */etc/netgroup* to verify host names whenever clients perform remote mounts. (Refer to *netgroup(4)*.)

For ARPA Services, the system uses */etc/netgroup* to verify users when clients perform remote logins or remote shells. (Refer to *hosts.equiv(4)*.)

Add a line with the following format for each netgroup you wish to define.

The entry may contain any number of netgroup names,

netgroup_name1 netgroup_name2 netgroup_name3 ...
netgroup_name1 member1 member2 ...

though you must then define these netgroups within /etc/netgroup

The *member n* is equal to the triple (*host_name, user_name, yp_domain_name*)

- The entry may contain any number of netgroup names, though you **must** then define these netgroups within */etc/netgroup*.

- You can assign more than one triple to a netgroup by enclosing each separate set within parentheses *(host_name, user_name, yp_domain_name)*.

- Leave any of these three fields empty to signify a wild card (i.e., blank fields match anything). For example, *(,,research)* matches all hosts and users in the *research* YP domain.

- A - (dash) in any of these three fields means **match nothing.** For example, *(-,mike,graphs)* does not match any hosts, but it does match the user *mike* in the *graphs* YP domain.

- Each *host_name* must have an entry in */etc/hosts*.

- The *yp_domain_name* is the name of the YP domain to which you currently belong. To determine your current YP domain name, execute the *domainname(1)* command.

  The commands using */etc/netgroup* assume you are not looking for any YP domain other than the one assigned on your node.

EXAMPLES:

/etc/netgroup Example Entry	The Netgroup Includes
`netgroup1 (,,)`	Everyone on the network
`netgroup2 (,darren,graphic)`	The user *darren* on any host in the *graphic* YP domain
`netgroup3 (node_7,,graphic)`	Any user on the *node_7* host in the *graphic* YP domain
`netgroup4 (node_2,john,)`	The user *john* on the *node_2* host in any YP domain
`netgroup5 (,andy,graphic) (node_1,mike,)`	The user *andy* on any host in the *graphic* YP domain and the user *mike* on the *node_1* host in any YP domain
`netgroup6 (-,annette,graphic)`	The user *annette* in the *graphic* YP domain, no host included

6. Create and Edit /etc/exports

You control the available file systems by your entries in the server's */etc/exports* file. Each time a server receives a mount request, *mountd(1M)* accesses */etc/exports* to see if the file system is exported and which systems can access it.

- The server must have the file system mounted locally before it can be exported.
- You must export the entire file system; you cannot export specific directories (though clients can mount specific directories).
- The path name in */etc/exports* must be the same path name as the directory on which the local file system is mounted.
- If the *-async* option is set (see *exports(4)*) for a file system, asynchronous writes on the NFS server occur.

Caution

The *-async* option increases write performance on the NFS server by allowing asynchronous writes on the NFS server's file system.

However, use caution in deciding whether to use the *-async* option. An unreported data loss **may** occur if the option is set **and** the NFS server hardware experiences a power loss, system panic or other failure.

Do not use the *-async* option with file systems that contain:

- files which are accessed by the O_SYNCIO flag (which is set by the *fnctl(2)* or *open(2)* calls),
- data that cannot be reconstructed (e.g., a file system containing database files),
- files synchronized with *fsync(2)*, or

- critical applications requiring absolute data integrity.

If you are unsure whether any of the previous conditions apply, do not use the *-async* option.

You control the file system's availability by specifying a netgroup or host name; otherwise, the file system becomes available to everyone on the network running NFS. After accessing */etc/exports*, the system checks */etc/netgroup* for the netgroup definition; if it is not present, the system checks */etc/hosts* for the host name. (For more information, refer to the previous sections, "4. Edit /etc/hosts" and "5. Edit /etc/netgroup.")

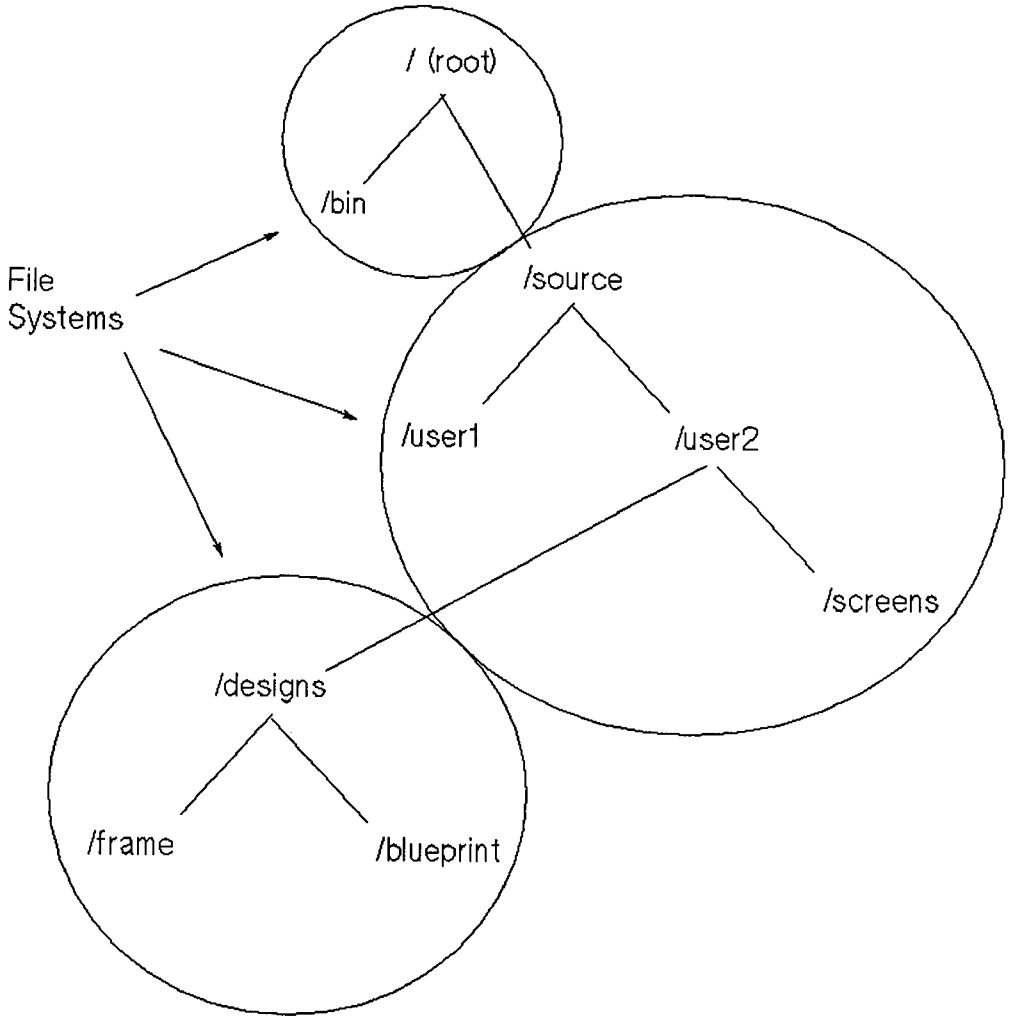
Note

If importing a file system containing a user's home directory, the user may not be able to login if the remote file system is not accessible.

If a client has a file system mounted and you edit */etc/exports* to change availability of that file system, the client's access will not change. To prevent the client from accessing the server's files, on the client you must either unmount the file systems or reboot.

/etc/exports Entry Formats	System Response
/complete_filesystem_pathname	Exports the file system to everyone on the network and defaults to **synchronous** writes on the NFS server.
/complete_filesystem_pathname netgroup_1 netgroup_2	Exports the file system only to specified netgroups
/complete_filesystem_pathname client_1 client_2	Exports the file system only to specified clients
/complete_filesystem_pathname client_2 netgroup_1	Exports the file system only to the specified client and netgroup.
/complete_filesystem_pathname `-async` *client_1*	Exports the file system to the specified client and causes **asynchronous** writes on the NFS server.

EXAMPLE:


/ (root)
/bin
File
Systems
/source
/user1
/user2
/screens
/designs
/frame
/blueprint

/etc/exports Example Entry *	Example System Response
/	Export / (root) to all clients. Clients will not receive the file system *source* since / is the exported file system. The *source* and *designs* file systems will not be seen by this mount. Clients will receive the */bin* directory since is it part of the / file system.
/source	Export *source* to all clients. Clients will not receive the file system *designs* or / since *source* is the exported file system. The *designs* and / file systems will not be seen by this mount.
/source/user2/designs	Export *designs* to all clients.
/source/user2/designs -async system1	Export *designs* to the client *system1* and allow **asynchronous** writes on the NFS server.
/source/user2/designs lab	Export *designs* to the netgroup *lab*.

```
/source/user2/designs system1 lab
```

Export *designs* to the client *system1* and the netgroup *lab*.

***Note:** You must define all hosts in */etc/hosts* and all netgroups in */etc/netgroup* or, if you are using YP, ensure that all hosts and netgroups are defined on the master YP server.

7. Reboot System

After you finish the configuration procedure, execute */etc/netnfsrc* or reboot the system to activate the daemons and servers.

The rebooting process does not unmount any of the server's file systems that were remotely mounted by other network nodes. However, these nodes will not be able to access any of the server's files until the server is operating again.

4. Create an NFS Client

You must be super-user to create an NFS client.

To create an NFS client, complete the following steps.

1. _____ Edit */etc/netnfsrc*
2. _____ Mount file systems
3. _____ Reboot the system (if necessary).

An NFS client can also be configured as any combination of an NFS server, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

1. Edit /etc/netnfsrc

The */etc/netnfsrc* file activates the NFS daemons and servers.

- To define the node as an NFS client, set the *NFS_CLIENT* variable to any digit other than zero.
- If the node is also a server, you may want to set the *NFS_SERVER* variable to any digit other than zero now. (Refer to the "Create an NFS Server" section to complete server configuration procedures.)
- If the node is also a server for PC-NFS requests, set the *PCNFS_SERVER* variable to any digit other than zero.

Client Only	`NFS_CLIENT=1` `NFS_SERVER=0`
Server Only	`NFS_CLIENT=0` `NFS_SERVER=1`
Both Client and Server	`NFS_CLIENT=1` `NFS_SERVER=1`
Neither Client nor Server	`NFS_CLIENT=0` `NFS_SERVER=0`
PC-NFS Server	`PCNFS_SERVER=1`

You can refer directly to the comments (lines beginning with # (pound) signs) for editing instructions and for descriptions of each activity executed by */etc/netnfsrc*.

Note

If you edit this file other than specified in this document, HP recommends you incorporate personal comments for future system administration.

```
#!/bin/sh
#       netnfsrc  --      NFS startup file
##
#       Depending on the configuration parameters you set within,
#       this script sets up some or all of the following:
#*      YP specific:
#           domainname  --      the YP domain name
#
#       and starts up some or all of the following programs:
#           portmap     --      RPC (program_#,version) -> port_# mapper
#           nfsd        --      NFS daemons
#           biod        --      async BIO daemons
#           pcnfsd      --      PC-NFS daemon
#*      YP specific:
#           ypbind      --      YP client process (all YP nodes)
#           ypserv      --      YP server process (YP server only)
#           yppasswdd   --      YP password daemon (YP master server only)
##
#       NFS_CLIENT      --      1 if this node is an NFS client, 0 if not
#       NFS_SERVER      --      1 if this node is an NFS server, 0 if not
#       Note:   it is possible for one host to be a client, a server, both
#               or neither!  This system is an NFS client if you will be
#               NFS mounting remote file systems; this system is a server
#               if you will be exporting file systems to remote hosts.
#       See Also: nfsd(1M), mount(1M)
##
#       Note:   this has nothing to do with whether or not the system is
#               a rootserver or diskless client workstation.  There is a
#               test for this later (to set the variable LDISK).
##
NFS_CLIENT=0
NFS_SERVER=0
.
.
.
.
PCNFS_SERVER=0
```

2. Mount File Systems

Review the servers' */etc/exports* file on your LAN to determine the file systems to which you want the client to have access. You will need to mount each of these file systems on the clients.

For each file system you should determine if you want it

- mounted automatically at boot time via */etc/checklist* or
- mounted only when manually specified via the *mount(1M)* command.

Since an attempt to mount a remote file system requires using another node and the network, the mount may not succeed the first time. You can vary the number of times NFS attempts to mount a file system by using the *retry* option.

After the mount is successful, the manner in which NFS handles requests depends on whether the mount is hard (default) or soft.

NFS Hard Mount

Hard mounted file systems with the default *int* (interrupt) cause NFS to retry a request until it succeeds, you interrupt it, or you reboot the system. If the *noint* option is activated and an NFS server goes down, the system retries the request until the server comes up again or you reboot the system.

If the server does not respond to a hard mount request, NFS writes the following message in the network error log file.

```
NFS: server host_name not responding, still trying
```

Refer to *Installing and Maintaining NS-ARPA Services* documentation for more error log information.

Note, if a server goes down from which you previously performed a hard mount, you may not be able to access mounted file systems on other nodes unless you reboot the problem server or interrupt all its requests.

NFS Soft Mount

Soft mounted file systems abort requests after one attempt. NFS writes an error to the log file if the server does not respond to a request. The message varies depending on what type of request is made.

```
NFS server host_name not responding, giving up

NFS function_name failed for server server_name: TIMED OUT
```

Note

If a user's home directory is in a remote file system, the user will not be able to login if the remote file system is not accessible (e.g., the server goes down, the network fails).

Mount Guidelines

Refer to the following guidelines whether mounting file systems automatically via the */etc/checklist* file or manually via the *mount(1M)* command. For more specific information, refer to *checklist(4)* and *mount(1M)*.

- You cannot mount a remote file system unless the server has an entry for your node in */etc/exports* or unless */etc/exports* makes the file system available to everyone on the network. (Execute *showmount(1M)* to list mounted file systems.)
- Though you can export only file systems, you can mount file systems or directories.
- When you mount a new file system on top of a directory already containing files, the directory's files will no longer be accessible unless you execute *umount(1M)* to unmount the mounted file system.

  To avoid masking a directory, HP recommends you mount the file system on top of an **empty** directory.

- You cannot mount or unmount an open directory (a directory in which someone is currently operating).
- You must specify a **mount point** (name of a local directory on which the file system will be mounted).
- **If operating in an HP-UX cluster environment**
  - If a cnode mounts a remote file system, all cnodes in the cluster can access the remote file system.
  - If using NFS to mount a file system attached to a cluster, you must use the root server host's name as the node name specified in the *mount(1M)* command.
  - If a cnode mounts a remote file system, any cnode in that cluster can unmount the remote file system.
- Before mounting a file system, refer to the following table and determine the options you want the mount to have.
  - You must specify an option if mounting via */etc/checklist*; you do not have to specify an option if mounting via *mount(1M)*.
  - You do not have to list options in a specific order; however, you must separate the options with commas (not spaces).

NFS Mount Options	Description
bg	Background: If the first request to a remote node's *mountd(1M)* fails, the mount process continues retrying the request in the background.
defaults	Defaults: The mount takes all the default options without you having to individually specify them. The defaults are noted within this table by asterisks (*). You only need to specify defaults when mounting via */etc/checklist*; the *mount(1M)* command automatically provides the defaults.
*fg**	Foreground: If the first request to a remote node's *mountd(1M)* fails, the *mountd(1M)* daemon retries the requests in the foreground.
*hard**	Hard Mount: NFS retries until the request succeeds or you reboot the system. If using the *int* default option, you can interrupt the file system request.
*int**	Interruptable Mount: You can press an interrupt key to abort an NFS request. (Though the interrupt key is not defined, common ones include **CTRL - C** and **BREAK**.)
noauto	No Automatic Mount: Prevents the file system from being mounted when the *mount -a* option is executed. You only need to specify *noauto* when mounting via */etc/checklist*.
nointr	No Interruptable Mount: You cannot interrupt processes waiting for NFS requests to complete.
nosuid	No *setuid*: You cannot execute files on the remote file system with either the *setuid* or *setgid* bits set.

*** = Default**

NFS Mount Options	Description
port = *n*	Port = *n* Default *n* = 2049 (the NFS server port) Specifies the UDP port at which the NFS server is contacted. You should not have to reset this value.
retrans = *n*	Retransmit = *n* Default *n* = 4 When NFS sends a request to a remote system, RPC attempts to transmit the request *n* times. If RPC does not receive a response after *n* attempts, soft mounts return an error and hard mounts retry the request.
retry = *n*	Retry = *n* Default *n* = 1 The *mount(1M)* command retries mounting the file system *n* times; the default is one. For example, if a *mount(1M)* attempt fails once and the default is one, *mount(1M)* tries once more before quitting.
ro	Read Only: Access rights are **Read Only**.
rsize = *n*	Read requests size = *n* Default *n* = 8192 (8K) Specifies the maximum read request size used in communicating with the server.
*rw**	Read/Write: Access rights are **Read and Write**.

NFS Mount Options	**Description**
soft	Soft Mount: NFS aborts the request after RPC attempts to transmit the request *n* times (as specified by the *retrans* option).
*suid**	*setuid*: You can execute programs on the remote file system that have *setuid* as one of their permissions.
timeo = *n*	Timeout = *n* Default *n* = 7 Specifies the initial timeout (in tenths of seconds) for NFS requests. When an NFS request occurs, RPC sends the request, waits 0.7 seconds for a response, and then retries the request. After the initial timeout, the timeout increases by multiples of two each time no response is received. When a specified number of *retrans* retransmissions have been sent with no reply, soft mounts return an error and hard mounts retry the request. **Note:** If performing NFS mounts through a gateway and you see several *server not responding* messages within a few minutes, change the timeout default value (7) to a value of 10 or greater until you stop seeing the message.
wsize = *n*	Write size = *n* Default *n* = 8192 bytes (8K) Specifies the maximum write request size used in communicating with the server.

*** = Default**

Edit /etc/checklist for Automatic Mounts

If you want the file system mounted automatically, add an entry for it in the */etc/checklist* file. At boot time, */etc/netnfsrc* executes *mount -at nfs* to mount all NFS file systems listed in */etc/checklist*.

Edit */etc/checklist* to append the hosts and file systems you wish to import using the following format. All of the default options are activated when you specify *defaults*. You must specify either *defaults* or at least one option.

NFS Hard Mounts via /etc/checklist

server_name:*/imported_filesystem* */mount_point* `nfs defaults 0 0`

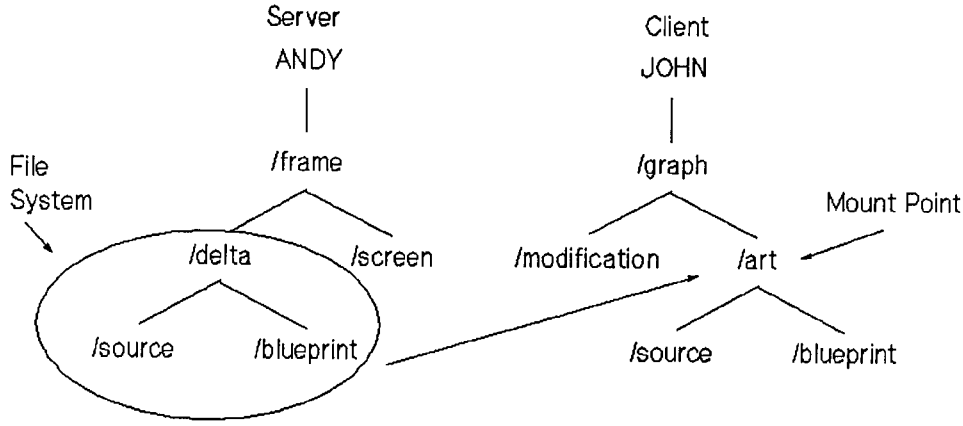
OR

server_name:*/imported_filesystem* */mount_point* `nfs [options] 0 0`

NFS Soft Mounts via /etc/checklist

server_name:*/imported_filesystem* */mount_point* `nfs soft[,,other options] 0 0`

The *nfs* stands for NFS mounts. NFS ignores the two zeros (0 0), though they must be present.



EXAMPLES: */etc/checklist* Automatic Mounts

/etc/checklist Example Entry on the Client JOHN	**Resulting Mount Options***
`ANDY:/frame/delta /graph/art nfs defaults 0 0`	Foreground Hard Mount Interruptable Port = 2049 Read and Write Read Size = 8192 Retransmit = 4 Retry = 1 *setuid* Timeout = 0.7 Write Size = 8192 **Note:** All of these options are by default.
`ANDY:/frame/delta /graph/art nfs ro,retry=6,timeo=3 0 0`	Read Only Retry = 6 Timeout =0.3
`ANDY:/frame/delta /graph/art nfs bg,retrans=8,soft 0 0`	Background Retransmit = 8 Soft Mount
`ANDY:/frame/delta /graph/art nfs noauto,noint,nosuid 0 0`	No Automatic Mount No Interruptable Mount No *setuid*
`ANDY:/frame/delta /graph/art nfs rsize=1024,wsize=1024 0 0`	Read Size = 1024 bytes Write Size = 1024 bytes

* The default options are activated when you specify *defaults*. They are also active with other options unless you specify otherwise. The default options are listed only once for this example.

Execute mount(1M) for Manual Mounts

Execute *mount(1M)* to mount an NFS file system manually. NFS file systems mounted via *mount(1M)* are only mounted as long as the client is running or until they are unmounted via *umount(1M)*. If the client goes down, you will have to re-mount the file system.

Do not use *mount(1M)* if you listed the file system in */etc/checklist* since it will have already been mounted.

Use the following *mount(1M)* format for NFS mounts. All of the default options are activated unless you specify otherwise.

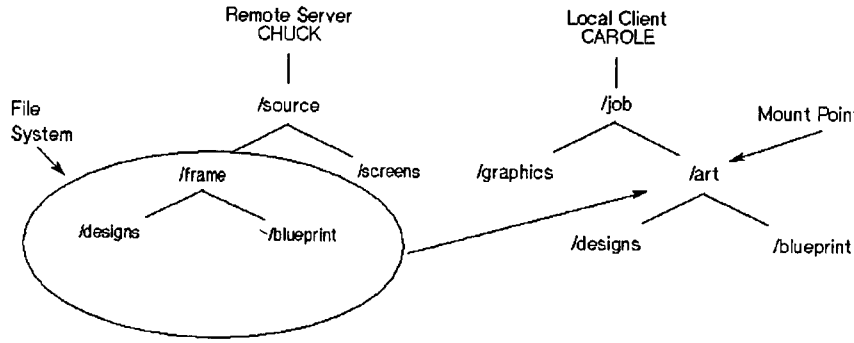
NFS Hard Mounts via mount(1M)

`mount [-o` *options*`]` *server_name*`:`*/filesystem* */mount_point*

NFS Soft Mounts via mount(1M)

`mount -o soft[,`*other_options*`]` *server_name*`:`*/filesystem* */mount_point*

EXAMPLES: *mount(1M)* Manual NFS Mounts



mount(1M) Example Command	Resulting Mount Options*
`mount CHUCK:/source/frame /job/art`	Foreground Hard Mount Interruptable Port = 2049 Read and Write Read Size = 8192 Retransmit = 4 Retry = 1 *setuid* Timeout = 0.7 Write Size = 8192 **Note:** All of these options are by default.
`mount -o ro,retrans=8,timeo=3 CHUCK:/source/frame /job/art`	Read Only Retransmit = 8 Timeout = 0.3
`mount -o bg,retry=6,rw,soft CHUCK:/source/frame /job/art`	Background Read and Write Retry = 6 Soft Mount
`mount -o noauto,noint,nosuid CHUCK:/source/frame /job/art`	No Automatic Mount No Interruptable Mount No *setuid*
`mount -o rsize=1024,wsize=1024 CHUCK:/source/frame /job/art`	Read Size = 1024 bytes Write Size = 1024 bytes

* All of the default options are activated unless you specify otherwise. The default options are listed only once for this example.

3. Reboot System

After you finish the configuration procedures, execute */etc/netnfsrc* or reboot the servers and clients to activate the daemons and servers.

The rebooting process unmounts all local file systems and directories that were manually mounted by the client (i.e., were not automatically mounted by */etc/checklist*).

5. Configure YP (optional)

If you plan to use the optional Yellow Pages (YP) service, refer to the "YP Configuration and Maintenance" chapter for detailed configuration procedures.

6. Configure VHE (optional)

If you plan to use the optional Virtual Home Environment (VHE) service, refer the the VHE Configuration and Maintenance" chapter for detailed configuration procedures.

7. Execute /etc/netnfsrc

To complete the configuration procedure, execute */etc/netnfsrc* (or reboot) your system.

Note

You have completed configuring the base NFS service. Refer to the remaining part of the chapter for maintenance information.

Maintenance

To keep NFS running correctly and efficiently, refer to the following sections to ensure it stays configured to meet your changing needs.

- Remove NFS File Access
- Update Software
- Clock Skew

Remove NFS File Access

You may need to remove file access to NFS either from the

- client to keep local users from accessing mounted remote file systems or
- server to keep all clients from accessing file systems via NFS.

Unmount File Systems from Client

If needed, you can unmount file systems on a client. Unmounting file systems prevents further access to the server's files until you remount the file system.

- Executing the *umount(1M)* command unmounts file systems mounted either via *mount(1M)* or */etc/checklist*.
- You cannot unmount an open directory or a parent of an open directory (i.e., a directory in which someone is currently operating).

- **If operating in an HP-UX cluster environment**
  - If a cnode mounts a remote file system, any cnode in that cluster can unmount the remote file system.
  - If a cnode unmounts a file system, all cnodes in the cluster will have that file system unmounted.

Unmount File Systems on Clients	Action
One NFS file system on a client	On the client, execute *umount.* `umount` *mount_point_name*
All file systems on one client	On the client, execute *umount -a.* `umount -a` **Note:** This command unmounts **all** file systems, not just NFS file systems. **If operating in an HP-UX cluster environment,** clients should **not** execute *umount -a.*
All NFS file systems on all clients	On all clients, execute *umount -at.* `umount -at nfs`
All file systems listed in */etc/mnttab* that were remotely mounted from a specified server	On all clients, execute *umount -h.* `umount -h` *server_name*

Prevent Access to Server File Systems

If needed, you can prevent clients from accessing file systems on the network servers.

Prevent Access to Server File Systems	Action
One NFS file system from a client	1. You have two options for step#1. • If a netgroup is specified for that file system in */etc/exports*, remove the host name from the netgroup entry in the server's */etc/netgroup* file. • If a host name is specified for that file system in */etc/exports*, remove the host name from the server's */etc/exports* file. 2. On the client, execute *umount.* `umount` *mount_point_name*
One NFS file system from a netgroup	1. On the server, remove the netgroup name (associated with that file system) from either the */etc/exports* file or from */etc/netgroup*. 2. On all members in the netgroup, execute*umount.* `umount` *mount_point_name*

Prevent Access to Server File Systems	Action
All NFS file systems from all clients	1. On all clients, execute *umount*. `umount` *mount_point_name* 2. On the server, you have two options for step #2. • Kill the *nfsd(1M)* daemon or daemons (usually four); the system prohibits NFS accesses only until you restart the *nfsd(1M)* daemons or you reboot the system. • Edit */etc/netnfsrc* to change the *NFS_SERVER* = value to zero, and reboot the system. `NFS_SERVER=0`

Update Software

To install a new system release to a server, use the */etc/update* program to install software. (Refer to the *HP-UX Series 300 System Administrator Manual* for detailed instructions.)

The following list includes configuration files loaded during the */etc/update* process. Some of these files contain example entries to help you configure them correctly.

- */etc/checklist*
- */etc/inetd.conf*
- */etc/netgroup*
- */etc/netnfsrc*
- */etc/rpc*
- */usr/adm/inetd.sec*

Note

If you are mounting file systems, then load **only** those file sets that reside on the local file systems.

When using */etc/update*, the system creates new configuration files in the */etc/newconfig* directory. These files correspond to the original configuration files which the system leaves in */etc*.

- Compare each file in */etc/newconfig* with its existing counterpart in */etc* to determine if you need to update or replace the file.
- If needed, edit the */etc/newconfig* files to meet your specific needs.
- Once the */etc/newconfig* file suits your configuration needs, replace the existing file in */etc* with the new one in */etc/newconfig*.
- You might want to save the old configuration file for later reference.

Clock Skew

The NFS client and server clocks may not be synchronized since each workstation keeps its own time. Problems may occur because of these time differences.

If your application depends on the local time or file system timestamps, then it may have to handle clock skew problems if it uses remote files. For example, when giving utime(2) a NULL pointer for the times value, the following process occurs.

1. The system sets the access time and modification time according to the client node clock.

2. It then sends these times over to the server, which then changes the inode to reflect the new access and modification times.

3. The server node identifies the change in the inode and thus, modifies the inode's status change time according to its own clock.

   The result is a high probability of differing times between the file or directory's access and modification times versus its status change time.

Note

HP corrected the clock skew problems that existed with the *ls* command and the source code control command *SCCS*.

If operating in an HP-UX cluster environment, all nodes in the cluster have the same time as the root server's clock. Therefore, clock skew problems exist only if the root server's clock is different from other NFS servers.

EXAMPLE: This example shows how a command could be affected by the clock skew.

Problem Most programs logically assume an existing file could not be created in the future; one example is *ls*. (Note, this example shows how HP corrected this problem.)

The *ls -l* has two basic forms of output, depending on how old the file is.

```
$ date
April   7  15:27:31 PST 1987                    Form One

$ ls -l file*
-rw-r--r--  1 root  other     Aug 26  1981 file
-rw-r--r--  1 root  other     Apr 07  15:26 file2
                                                Form Two
```

Form One of *ls* prints the month, day, and year of the last file modification if the file is **more** than six months old. Form 2 prints the month, day, hour, and minute of the last file modification if the file is **less** than six months old.

The *ls* command calculates the age of a file by subtracting the modification time of the file from the current time. If the results are greater than six months, the file is "old."

Now assume that the time on the server is three minutes ahead of the local node's time (April 7, 15:30:31). The following commands demonstrate the effect of this clock skew prior to HP's correction of the problem.

```
$ date
April 7  15:27:31 PST 1987
$ touch file3
$ ls -l file*
-rw-r--r--  1 root  other   0 Aug 26  1981 file
-rw-r--r--  1 root  other   0 Apr 07 15:26 file2
-rw-r--r--  1 root  other   0 Apr 07  1987 file3
```

The problem is that the difference of the two times is negative, but the variable in the computation is unsigned. A signed negative number has the same representation (bit pattern) as a very large unsigned number.

local node time = 15:27:31
modification time = local node time plus 180 seconds

local node time	15:27:31
- modification time	- (15:27:31 + 180)
large unsigned number which is greater than six months	- 180 seconds

Problem Correction

HP corrected the problem so that *ls* now prints the month, day, and minute for files between six months old and one hour ahead of time. Other applications may also require such modification.

```
$ date
April 07  15:27:31 PST 1987
$ touch file3
$ ls -l file*
-rw-r--r--  1 root  other   0 Aug 26  1981 file
-rw-r--r--  1 root  other   0 Apr 07 15:26 file2
-rw-r--r--  1 root  other   0 Apr 07 15:30 file3
```

6

Remote Execution Facility (REX)

Introduction

This chapter describes how to configure and execute commands on a remote host using the Remote Execution Facility (REX).

REX consists of:

- The *on(1)* command
- The *rexd(1M)* (remote execution daemon)

The *on* command provides the REX user interface on the client. It also communicates with *rexd* to execute commands remotely. *rexd* runs on the server and facilitates the execution of the remote commands.

The functionality of REX is similar to that of remote shell (*remsh(1)*) with two important differences:

1. REX executes commands in an environment similar to that of the invoking user. Your environment is simulated by:
   - Copying all of your environment variables to the remote computer.
   - Mounting the file system containing your current working directory on the remote computer via NFS (if it is not already mounted on the remote computer).

   Your command is then executed on the remote computer in the remote version of your working directory, using your (the invoking user's) environment variables.

2. REX allows you to execute interactive commands such as *vi*.
   - In this case your current tty settings (e.g. your current "break" character) are also copied to the remote system.

The on Command

The *on* command provides the user interface for remote execution of commands. When executing the *on* command, you specify:

- a host on which to run the remote command
- the command to run
- arguments for the command

The *on* command then simulates your current environment on the server by passing your environment variables and information about your current working directory to the remote host. The *rexd* daemon on the server mounts the file system that contains your current working directory if it is not already mounted on the server. After the environment is simulated, the command executes in the simulated environment on the remote host.

Note

Your environment is simulated on the remote host but not completely recreated. Execution of a given command on a remote host will not always produce the same results as the executing the command on your local computer. The simulated environment and the environment's limitations are discussed below in "Environment Simulation."

The syntax of the *on* command is as follows:

```
on [-i | -n] [-d] host [ command [argument ] ....]
```

Host specifies the name of the host on which to execute *command.* There must be an entry for *host* in the local computers's host data base.

Command specifies the command to execute on *host.* If *command* is not specified, *on* will start a shell on *host.*

You may specify three options (*-i, -n, -d*). The *-i* option must be used when invoking interactive commands, the *-n* option must be used when running commands in the backgound with job control, and the *-d* option is used when you wish to receive diagnostic messages.

Use of the *-d* option with either *-i* or *-n* is permitted.

EXAMPLE: `on -i -d host`

or

```
on -n -d host
```

You cannot use the *-i* and *-n* options at the same time.

EXAMPLE:

```
on -i -n host
```

is not permitted.

The -i Option (Interactive Mode)

The *-i* option invokes the interactive mode. This option must be specified for all interactive commands (commands which expect to be communicating with a terminal). Examples of interactive commands are *vi(1)*, *csh(1)*, and *more(1)*. If this option is specified with a non-interactive command such as *sort(1)*, it will be executed as an interactive command, but there may be no difference in behavior.

EXAMPLE: `on -i node_7 vi`

The -n Option (No Input Mode)

The *-n* option sends the remote program an end-of-file when the program reads from standard input instead of connecting the standard input (*stdin*) of the *on* command to the standard input (*stdin*) of the remote command. The *-n* option is necessary when running commands in the background with job control.

The -d Option (Debug Mode)

The *-d* option allows you to receive diagnostic messages during the start up of the *on* command. The messages may be useful in detecting configuration problems if the *on* command is failing while connecting to a given host.

Configuration Requirements

The following list details the configuration requirements that must be met for you to execute the *on* command from node A to node B:

- You must be logged into a user account (other than root) on node A.
- You must have an account on node B, and the UIDs for the accounts on node A and node B must be the same. If this is not the case, one of two things will happen:
  - If the UID associated with the user on node A is not associated with any user on node B, the *on* command will fail with the error:

    ```
    on hostname: rexd: User id xxxx is not valid.
    ```

  - If your UID on node A is associated with another user on node B, then the command will be executed on node B as the user associated with the UID. (The second case is a serious security limitation. More details are given in the "Security Considerations" section of this chapter).
- The file system that contains your current working directory must be exported in a manner that allows computer B to mount it. Note that the current working directory may be a directory on another remote computer C, which is being accessed via NFS. If your current working

directory is being accessed via RFA, the *on* command will fail with the following error message:

```
on: current directory (<current_dir>) is remote via RFA
      RFA directories not supported by on.
```

current_dir is the path name of your current working directory.

- Node B must have *rexd* configured to execute.

Environment Simulation

As mentioned above, your environment is simulated on the remote computer, not mirrored. Therefore, certain limitations exist that may cause the execution of a given command to produce different results when executed on the local computer and a remote computer via *on*. These limitations are as follows:

- If the file system is not already mounted on the remote computer, the file system containing your current working directory will be mounted on the remote computer in a subdirectory of */usr/spool/rexd*. If the file system is already mounted on the remote computer, the mount point is the current mount point for the file system. Therefore, the use of absolute path names can cause problems.

EXAMPLE:

User mjk on node A is in his home directory (*/users/mark/mjk*) and executes the *on* command to start a shell on a remote system. When the shell is started, the current directory will be */usr/spool/rexd/rexdAXXXX/users/mark/mjk* (where *A* is a letter and *XXXX* is a 4 digit number). If mjk now types the command *cd*, one of two events will occur, depending on the configuration of the file system on the remote computer:

- If the path */users/mark/mjk* exists on the remote system, the current directory will be */users/mark/mjk* on the remote system, which is not equivalent to */users/mark/mjk* on the local system.

- If the path */users/mark/mjk* does not exist on the remote system then executing *cd* will return an error.

This type of behavior could cause a script that executes *cd* or uses absolute file names to produce different results when executed remotely.

- Another example where the use of absolute path names may occur, without being obvious, is the use of $PATH. Implicit use of $PATH may cause a different version of a command (or a different command) to be executed in the remote case.

- Relative path names will work if they are within the same file system as your current working directory. If a relative path name crosses a file system boundary it will encounter problems similar to those presented by use of absolute path names.

- Finally, the *on* command will fail if your current working directory is being accessed by RFA. This occurs because REX is unable to simulate your environment. In this case, you will receive the following error message:

```
on: current directory (<current_dir>) is remote via RFA
      RFA directories not supported by on.
```

Configuring rexd

Configuring *rexd* on a system allows the system to act as a server, executing commands for clients that execute an *on* command. Before configuring *rexd* to run on a system, you should read the "Security Considerations" section in this chapter.

When *rexd* is configured, it is started by *inetd(1M)* when a request for remote execution is made by a client. *inetd* obtains the information it needs to start *rexd* from the file */etc/inetd.conf*. The following entry must be in the file */etc/inetd.conf* in order for *inetd* to start *rexd*:

```
rpc stream tcp nowait root path 100017 1 \
rpc.rexd [ options ]
```

Where:

path is the path name of the *rexd* executable in the file system. The *rexd* shipped with the HP NFS product is located in */usr/etc/rpc.rexd*.

options are the options that change the behavior of *rexd*. Each of the possible options is described below:

The -l option

You can log any errors reported by *rexd* to a file by adding *-l log_file* at the end of the configuration entry in */etc/inetd.conf*, where *log_file* is the name of the file where errors are logged. If *log_file* exists, *rexd* appends messages to the file. If *log_file* does not exist, *rexd* creates it. Messages are not logged if the *-l* option is not specified.

The information logged to the file includes the date and time of the error, the host name, process ID and name of the function generating the error, and the error message. Note that different RPC services can share a single log file since enough information is included to uniquely identify each error.

EXAMPLE:

Thus, the entry in */etc/inetd.conf* to log errors to the file */usr/adm/rexd.log* is:

```
rpc stream tcp nowait root /usr/etc/rpc.rexd 100017 1 \
rpc.rexd -l /usr/adm/rexd.log
```

The -m option

Specifying *-m mountpoint* changes the default directory containing mount points. This directory is used for mounting client file systems. The following entry in */etc/inetd.conf* causes client file systems to be mounted as */client/mnt/rexdAXXXX* instead of */usr/spool/rexd/rexdAXXXX* (where *A* is a letter and *XXXX* is a 4 digit number):

```
rpc stream tcp nowait root /usr/etc/rpc.rexd 100017 1 \
rpc.rexd -m /client/mnt
```

The owner, group, and all other users must have read and execute permission for *mountpoint* or an *on* command may fail for a user that does not have the proper permission to *mountpoint*.

The -r option

The *-r* option causes the *rexd* to use stronger security checking than it uses by default (see "Security Considerations"). When started with the *-r* option, *rexd* denies access to a client unless one of the following conditions is met:

- The name of the client is in the */etc/hosts.equiv* file on the server.
- The user on the server, associated with the UID sent by the client, has an entry in *$HOME/.rhosts* that specifies the client name followed by:

white space and an end of line

or

the user's name and an end of line.

EXAMPLE:

If a user assigned to UID 7 on NODE1 executes the following *on(1)* command,

```
on NODE2 pwd
```

then user *mjk* (assuming user *mjk* on NODE2 is assigned UID 7) on NODE2 must have one of the following entries in *$HOME/.rhosts*.

```
NODE1
```

```
NODE1 mjk
```

Security Considerations

The design and implementation of REX incorporates several security limitations that you should consider before configuring *rexd*.

REX restricts access to a system by use of UIDs. That is, the client (*on*) passes the invoking user's UID to the server (*rexd*) to determine if the invoking user is a valid user. This creates several security limitations:

- If the client and the server do not have the same mapping of user to UIDs, a user on a client may be able to access the server as some other user.
- A malicious user can set the desired UID in the outgoing packets and access the server as any of the server's valid users other than root. An individual with their own workstation can set up a user account with the desired UID.

The impact on system security can be reduced by using the file */usr/adm/inetd.sec*. The entries in this file specify a set of networks and hosts that are allowed or denied access to a service that is started by *inetd*. For more details on the use of */usr/adm/inetd.sec* see the *inetd.sec(4)* reference page.

The consequences can also be reduced by use of the *-r* option when starting *rexd*. See the previous section, "Configuring rexd," for more details about the *-r* option.

Under normal NFS use, only root is allowed to mount remote file systems. However, when *rexd* is in use, you can mount a file system on the server by executing the following instructions:

1. *cd* to a directory in the file system you wish to mount.
2. Execute the *on* command to start a shell on the computer on which you wish to mount the file system.
3. From another window, shell layer, or system, log into the server and *cd* to a directory in the file system that *rexd* mounted.

4. Switch back to the previous window, shell layer, or system and exit the shell created by the *on* command.

Since another user is busy in the mounted file system, *rexd* will be unable to unmount the file system. Hence, the user has mounted the file system.

Diagnostics

The on Command Error Messages

on: unknown host <*host*>

The host name <*host*> was not found in the *hosts* database.

on: cannot connect to server on <*host*>

The host <*host*> is down, unreachable on the network or not running *rexd(1M)*.

on: can't find <*current_dir*>

A problem occurred trying to find your current working directory (<*current_dir*>).

on: can't locate mount point for <*current_dir*>

A problem occured trying to determine the mount point of your current working directory (<current_dir>).

on: standard input (*stdin*) is not a tty

The standard input (*stdin*) of the *on* command with the *-i* option is not a tty.

on: current directory (<*current_dir*>) is remote via RFA
RFA directories not supported by on.

Your current working directory (<*current_dir*>) is being accessed from a remote host via RFA, an HP networking product. Use of *on* from a directory accessed in this manner is not supported.

on <*server*>: rexd: <*message*>

Errors which occur on the server <*server*> and are propagated back to the client. These messages are documented in the "DIAGNOSTICS" section of *rexd(1M)* found in the *NFS Services Reference Pages*.

rexd Error Messages

The following is a subset of the messages that may appear in the log file if the *-l* option is used. Some of these messages are also returned to the client.

rexd: could not umount <*dir*>

rexd was unable to *umount(2)* your current working file system. See *rexd(1M)* in the *NFS Services Reference Pages* for more details.

rexd: mountdir (<*mountdir*>) is not a directory

The path name <mountdir>, under which temporary mount points are created, is not a directory or does not exist.

rexd: <*command*>: Command not found

rexd could not find <command>.

rexd: <*command*>: Permission denied

rexd was denied permission to execute <command>.

rexd: <*command*>: Text file busy

The executable file is currently open for writing.

rexd: <*command*>: Can't execute

rexd was unable to execute <*command*>.

rexd: root execution not allowed

Root execution is not allowed by *rexd*.

rexd: User ID <*UID*> not valid

The UID <*UID*> is not assigned to a user on the server.

rexd: User id <*UID*> denied access

rexd was started with the *-r* option, and the remote execution request did not meet either of the conditions required by the *-r* option.

rexd: <*host*> is not running mountd

The host <*host*> on which the user's current working directory is located is not running *mountd(1M)*. Therefore, *rexd* is unable to mount the required file system.

rexd: not in export list for <*file system*>

The host on which the client's current working directory is located does not have the server on the export list for the file system <*file_system*> containing the client's current working directory. Therefore, *rexd* is unable to mount the required file system.

7

The Network Lock Manager

Introduction

This chapter explains file and record locking using the Network Lock Manager and the Network Status Monitor. It also explains how file locking is used to synchronize access to shared files.

File and record locking allows cooperating processes to synchronize access to shared files. You interface with the networking service by way of the standard *lockf()* system call interface, and rarely require any detailed knowledge of how it works. The operating system maps user calls to *lockf()* and *fcntl()* into Remote Procedure Call (RPC)-based messages to the local lock manager. The fact that the file system may be located on a different node is not really a complication—until a failure occurs.

All computers fail or simply shut down from time-to-time, and in an NFS environment, where multiple computers can have access to the same file at the same time, the process of recovering from a failure is necessarily more complex than in a non-network environment. Furthermore, locking is inherently stateful. If a server fails, clients with locked files must be able to recover their locks. If a client fails, the locks will be released when the client comes back up. To preserve the overall transparency of NFS, the recovery of lost locks must not require the intervention of the applications themselves. This is accomplished as follows:

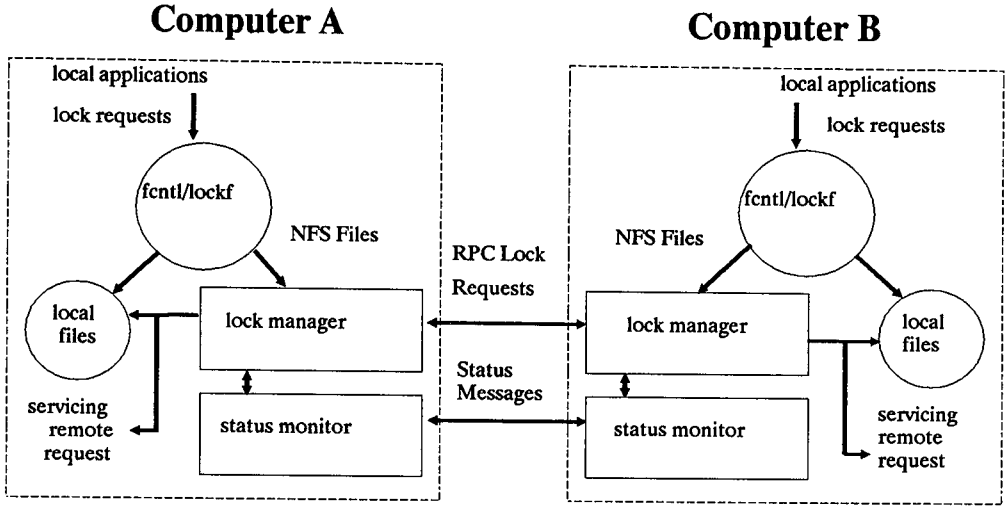
- Basic file access operations, such as read and write, use a stateless protocol (the NFS protocol). All interactions between NFS servers and clients are atomic—the server doesn't remember anything about its clients from one interaction to the next. In the case of a server

failure, client applications will sleep until the server recovers and NFS operations can complete.

- Stateful services (those that require the server to maintain client information from one transaction to the next) such as the locking service, are not part of NFS. They are separate services that use the status monitor (see "The Network Status Monitor" section at the end of this chapter) to ensure that their implicit network state information remains consistent with the real state of the network. There are two specific state-related problems involved in providing locking in a network context:
  - If the client has failed, the lock can be held forever by the server.
  - If the server has failed, it loses its state (including all its lock information) when it recovers.

The Network Lock Manager solves both of these problems by cooperating with the Network Status Monitor to ensure that it is notified of relevant computer failures. The Lock Manager protocol then allows it to recover the lock information it needs when a computer recovers from a failure.

The following illustration depicts the overall structure of the network locking service.



Architecture of the Network Locking Service

At each server site, a lock manager process accepts lock requests, made on behalf of client processes by a remote lock manager, or on behalf of local processes by the kernel. The client and server lock managers communicate with RPC calls. When the lock manager receives a remote lock request for a computer that it doesn't hold a lock for, the lock manager registers interest in that computer with the local status monitor. The lock manager then waits for notification from the local status monitor that the computer is up. The local status monitor continues to watch the status of registered computers, and notifies the lock manager when one of them is rebooted (after a failure). If the lock request is for a local file, the lock manager tries to satisfy it, and communicates back to the application along the appropriate RPC path.

If the failure of a client is detected, the server releases the failed client's locks, on the assumption that the client application will request locks again as needed. If the recovery (and, by implication, the failure) of a server is detected, the client lock manager retransmits all lock requests previously granted by the recovered server. This retransmitted

information is used by the server to reconstruct its locking state. See the "Locking Protocol" section for more detail.

The locking service, then, is essentially stateless. Or to be more precise, its state information is carefully circumscribed within a pair of system daemons that are set up for automatic application-transparent failure recovery. If a server fails, and thus loses its state, it expects that its clients will be notified of the failure and send it the information that it needs to reconstruct its state. The key in this approach is the status monitor, which the lock manager uses to detect both client and server failures.

Note

Recovery cannot occur until the remote system is rebooted.

The Locking Protocol

The lock style implemented by the HP-UX network lock manager supports deadlock detection on a per-server basis only (see the *lockf(2)* and *fcntl(2)* reference pages for details).

Despite network lock manager's adherence to the *lockf() / fcntl()* semantics, a few subtle points about its behavior need to be mentioned. They are:

- When an NFS client goes down and comes back up, the lock managers on all servers are notified by their status monitors, and the lockds release their locks, on the assumption that the lock managers will request locks again when they want them. When a server fails, however, the clients wait for the server to come back up. When it does, the server's lock manager gives the client lock managers a grace period to submit lock reclaim requests. During this period the server's lock manager accepts only reclaim requests from remote lock managers. The client status monitors notify their respective lock

managers when the server recovers. The default grace period is 50 seconds.

- It is possible that, after a server failure, a client may not be able to recover a lock that it had on a file on that server. This can happen because another process may have accessed the lock before the recovering application process. In this case, the *SIGLOST* signal will be sent to the process (the default action for this signal is to kill the application).

- The local lock manager does not reply to the operating system's lock request until the server lock manager has acknowledged the local lock manager's request. Further, if the lock request is on a server new to the local lock manager, the lock manager registers its interest in that server with the local status monitor and waits for its reply. If either the status monitor or the server's lock manager are unavailable, the reply to a lock request for remote data is delayed until the server becomes available.

- Only advisory mode locking is supported. Enforcement mode is not supported for NFS files.

The Network Status Monitor

The Network Lock Manager relies heavily on the Network Status Monitor to maintain the inherently stateful locking service within the stateless NFS environment. However the status monitor can also be used to support other kinds of stateful network services and applications. Normally, failure recovery is one of the most difficult aspects of network application development, and requires a major design and installation effort. The status monitor simplifies this task.

The status monitor works by providing a general framework for collecting network status information. Implemented as a daemon that runs on all network computers, it uses a simple protocol that allows applications to monitor the status of other computers. Its use improves overall robustness, and avoids situations in which applications running on different computers (or even on the same computer) disagree about the

status of a site—a potentially dangerous situation that can lead to inconsistencies in many applications.

Applications that use the status monitor do so by registering the computers they are interested in. The status monitor then tracks the status of those computers, and when one of them fails it notifies the interested applications of the failure, and the applications may take whatever actions are necessary to reestablish a consistent state.

A few advantages of this approach are:

- Only applications that use stateful services must pay the overhead—in time and in size—of dealing with the status monitor.
- The implementation of stateful network applications is simplified, since the status monitor shields application developers from the complexity of the network.

8

YP Configuration and Maintenance

The Yellow Pages (YP) is an **optional**, distributed network lookup service that allows you to administer databases from one node on the network. With YP you can maintain a single set of user and group IDs for all nodes within a specified set (YP domain). For specific YP information, refer to the following sections.

- Key Terms
- YP Databases
- YP Commands
- YP Configuration
- Verify YP
- Disable YP
- YP Maintenance

Refer to *ypfiles(4)* for a complete explanation of the YP database and directory structure.

If you do not have YP administrative responsibilities, refer to the"Common Commands" chapter for general YP usage information.

Note

For this chapter only, all references to servers and clients are YP specific.

Key Terms

Bind

- Process by which a client locates and directs all requests for data to a specific server.
- Process of establishing the address of a socket that allows other sockets to connect to it or to send data to it.

Cluster

One or more workstations linked together with a local area network (LAN), but consisting of only one file system.

Cnode

One or more workstations linked together with a local area network (LAN), but consisting of only one file system.

Diskless Cnode

A node in an HP-UX cluster that uses networking capabilities to share file systems, but does not have a file system physically attached.

Escape Sequence (YP)

Characters used within files to force inclusion and exclusion of data from YP databases. The escape sequences are as follows.

- + (plus)
- - (minus)
- + *@netgroup_name*
- -*@netgroup_name*

Export

To make a file system available to remote nodes via NFS.

File System An entire unit (disk) that has a fixed size.

GID A value that identifies a group in HP-UX.

Global (YP) A means of access in which the system always reads YP maps rather than the local ASCII files.

Host A node that has primary functions other than switching data for the network.

Internet Address A four-byte quantity that is distinct from a link-level address and is the network address of a computer node. This address identifies both the specific network and the specific host on the network.

Key (YP) A string of characters (no imbedded blanks or tabs) that indexes the values within a map so the system can easily retrieve information. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.

Local (YP) A means of access in which the system first reads the local ASCII file. If it encounters an escape sequence, it then accesses the YP databases.

Map (YP) A file consisting of logical records; a search key and related value form each record. YP clients can request the value associated with any key within a map.

YP map is synonymous with **YP database.**

Master Server (YP) The node on which one or more YP maps are constructed from ASCII files. These maps are then copied to the YP slave servers for the YP clients to access.

Netgroup A network-wide group of nodes and users defined in */etc/netgroup*.

Node A computer system that is attached to or is part of a computer network.

Propagate To copy maps (data) from one YP server to another.

Root Server The only node in an HP-UX cluster that has file systems physically attached to it.

Slave Server (YP) A node that copies YP maps from the YP master server and then provides YP clients access to these maps.

UID A value that identifies a user in HP-UX.

Value (YP) A unit of information stored in YP maps; each value has a corresponding key (index) so the system can easily retrieve it. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.

Yellow Pages (YP) An optional network service composed of databases (maps) and processes that provide YP clients access to the maps. The YP service enables you to administer these databases from one node.

YP may or may not be active; check with your system administrator.

YP Client

- A node that requests data or services from YP servers.
- A YP process that requests other YP processes to performoperations.

Note: A YP client can also be configured as any combination of a YP server, NFS client, or NFS server. (Note, a YP server **must** also be configured as a YP client.)

YP Database See "Map (YP)."

YP Domain A logical grouping of YP maps (databases) stored in one location. YP domains are specific to the YP network service and are not associated with other network domains.

YP Map See "Map (YP)."

YP Password The password for a user's login ID that exists in the YP *passwd* map. The YP password is the same one as the user password, but is administered through the YP.

You do not have to have a YP password to access the YP databases.

YP Server

- A node that provides data (maps) or services to other nodes (YP clients) on the network using YP.
- A YP process that performs operations as requested by other YP processes.

Note: A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

YP Databases

The *ypmake(1M)* script creates the standard YP databases from the following ASCII files. You can also create additional YP databases. (Refer to *ypfiles(4)*.)

- */etc/group*
- */etc/hosts*
- */etc/netgroup*
- */etc/networks*
- */etc/passwd*
- */etc/protocols*
- */etc/rpc*
- */etc/services*

Other maps may be present, like *ethers* and *mail.aliases*, that may be used by other vendors or applications.

Note

If the */usr/etc/yp* directory is part of a file system that supports only short file names (14 characters maximum), then any maps you create can have only 10 characters. This restriction exists because the *makedbm(1M)* command automatically adds the *.dir* and *.pag* suffixes to each map name. Refer to the *HP-UX System Administrator Manual* for more information.

Local and Global Maps

Clients access the above ASCII files and their corresponding YP maps in one of two ways, depending on whether the YP maps are local or global.

- A map is **local** if the system first accesses the local ASCII file. If the file contains an escape sequence, the system then accesses the YP database.
- A map is **global** if the system accesses only the YP database (never accesses the local ASCII file).

If a node is not a client, the system accesses only the local ASCII files for information.

YP Maps	Type	Access
/etc/group */etc/passwd*	Local	If a + (plus) entry exists at the beginning of a line, the system retrieves data from the corresponding YP map; otherwise, the YP maps are unused. Occurrences of +*@netgroup_name* and -*@netgroup_name* at the beginning of a line cause the system to reference YP. (Refer to *group(4)* and *passwd(4)* for complete information regarding these escape sequences.)
/etc/hosts */etc/netgroup/* */etc/networks* */etc/protocols* */etc/rpc* */etc/services*	Global	The system consults only the YP for data. If YP is not running, it looks at the local ASCII files.

Note

If using YP to provide the information stored in the standard maps' ASCII files, you **must** relink any applications that read data from those files.

This relinking ensures the files can obtain data from the YP maps. If you do not relink the applications, they will access only the local files. If the local files are not as current as the YP maps, the applications may not work correctly.

Escape Sequences

Escape sequences are characters used within a file at the beginning of a line to force inclusion and exclusion of data from YP databases. (Refer to the following reference pages: *passwd(4)*, *hosts(4)*, *netgroup(4)*, *host.equiv(4)*, and *group(4)*.) The escape sequences are as follows.

- + (plus)
- - (minus)
- + *@netgroup_name*
- -*@netgroup_name*

Escape Sequence	Description
+ (plus)	Use + (plus) in */etc/passwd* and */etc/group* to retrieve one or more entries from the YP *passwd* and *group* maps, respectively. The plus designates specific entries to be retrieved from YP.
- (minus)	Use - (minus) in */etc/passwd* and */etc/group* to ignore any subsequent entries with the same name. This process hides the matching names occurring in the YP *passwd* and *group* maps, respectively. Therefore, it disallows access to particular entries.
+ *@netgroup_name*	Use + *@netgroup_name* in */etc/passwd* to insert the matching entries from the YP *passwd* map for all members of a network group. **For ARPA Services** Use + *@netgroup_name* in */etc/hosts.equiv* and *$HOME/.rhosts* to include a network group's entries in their lists of allowed users.
- *@netgroup_name*	Use -*@netgroup_name* in */etc/passwd* to disallow the matching entries from the YP *passwd* map for all members of a network group. **For ARPA Services** Use -*@netgroup_name* in */etc/hosts.equiv* and *$HOME/.rhosts* to exclude a network group's entries from their lists of allowed users.

Netgroups

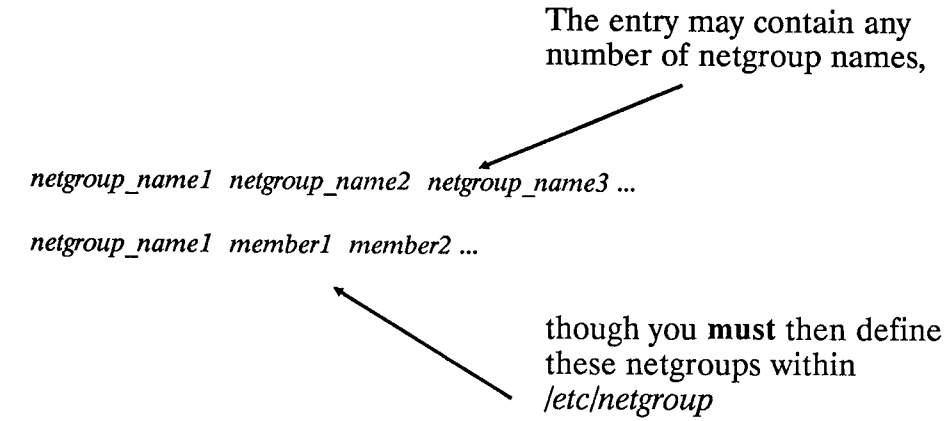
Netgroups are network-wide groups of nodes and users defined in */etc/netgroup* on the master server. Use these groups for permission checking during login and remote mount. **For ARPA Services**, you can also use these groups for permission checking during remote login (*rlogin(1)*) and remote shell execution (*remsh(1)*).

The master server uses */etc/netgroup* to generate three YP maps in the */usr/etc/yp/domain_name* directory: *netgroup*, *netgroup.byuser*, and *netgroup.byhost*. The *netgroup* map contains basic information found in */etc/netgroup*. The other two maps contain more specific information to accelerate the lookup of netgroups given the user or host.

The programs consulting the YP *netgroup* maps include *login(1)*, *mountd(1M)*, *rlogin(1)*, and *remsh(1)*.

Program	Description
login(1)	Consults the maps to resolve netgroup names in */etc/passwd*
mountd(1M)	Consults the maps to resolve netgroup names in */etc/exports*
rlogin(1) *remsh(1)*	**For ARPA Services** Consults the *netgroup* map if netgroup names are in */etc/hosts.equiv* or *$HOME/.rhosts*

To limit access to file systems, edit */etc/exports* to include the appropriate netgroup names. Then define the netgroup in */etc/netgroup* using the following format. (Refer to *exports(4)* and *netgroup(4)*.)



The *member n* is equal to the triple *(host_name, user_name, yp_domain_name)*

- The entry may contain any number of netgroup names, though you **must** then define these netgroups within */etc/netgroup*.
- You can assign more than one triple to a netgroup by enclosing each separate set within parentheses *(host_name, user_name, yp_domain_name)*.
- Leave any of these three fields empty to signify a wild card (i.e., blank fields match anything). For example, *(,,research)* matches all hosts and users in the *research* YP domain.
- A - (dash) in any of these three fields means **match nothing**. For example, *(-,mike,graphs)* does not match any hosts, but it does match the user *mike* in the *graphs* YP domain.

- Each *host_name* must have an entry in */etc/hosts*. (See *hosts(4)*.)
- The *yp_domain_name* is the name of the YP domain to which you currently belong. The commands using */etc/netgroup* assume you are not looking for any YP domain other than the one assigned on your node. (To list your current YP domain name, execute the *domainname(1)* command.)

EXAMPLE: The following example is a sample */etc/netgroup* file. (Refer to *netgroup(4)* for a complete file format description and a definition of lines and fields.)

```
#
# Engineering: Everyone, but mike, has a node.
# The node 'testing' does not have any users associated with it.
#
engineering hardware software
hardware (mercury,jeff,mickie) (venus,dave,mickie) (testing,-,mickie)
software (earth,carole,mickie) (mars,darren,mickie) (-,mike,mickie)
#
# Marketing: Time-sharing on pluto
#
marketing (pluto,andy,mickie) (pluto,cristina,mickie)
(pluto,chuck,mickie)
#
# Others
#
allusers (-,,mickie)
allhosts (,-,mickie)
```

The YP domain name for all the example netgroups is *mickie*. The users and hosts are classified into netgroups as follows.

Netgroup	Users	Hosts
hardware	jeff, dave	mercury, venus, testing
software	carole, darren, mike	earth, mars
engineering	jeff, dave, carole, darren, mike	mercury, venus, earth, mars, testing
marketing	andy, cristina, chuck	pluto
allusers	every user in the YP *passwd* map	no hosts
allhosts	no users	all hosts in the YP *hosts* map

Files Related to YP

For ARPA Services

The files */etc/hosts.equiv* and *$HOME/.rhosts* are not in the YP system; however, they are related to YP. If these files contain a plus (+) or minus (-) entry with the argument *@netgroup*, the system consults the YP *netgroup* map for data. (Refer to *netgroup(4)* and *hosts.equiv(4)*). For example, a line consisting of

```
+@engineering
```

in */etc/hosts.equiv* will include all members of engineering as defined in the local file */etc/netgroup* or in the YP database. A line consisting only of a plus (+) allows access to all hosts.

The same holds true for *$HOME/.rhosts*. Also, in */$HOME/.rhosts*, a host name followed by a plus (+) means any user coming from that host name will be allowed to access this account through *rlogin(1)* or *remsh(1)*. (See *hosts.equiv(4)*.)

YP Commands

Refer to the following table for a brief description of all YP commands. Refer to the "Common Commands" chapter for a more detailed description of the YP commands you might want to use on a daily basis (i.e., those YP commands that do not require super-user access).

YP Commands	Description
domainname(1)	Use *domainname(1)* to determine or change a YP domain name.
makedbm(1M)	**Note:** Use this version of *makedbm(1M)* only with YP. A tool for building YP maps. Use *makedbm(1M)* to build or rebuild databases not built by */usr/etc/yp/ypmake*. Use *makedbm(1M)* to disassemble a map so that you can see the key-value pairs comprising it.
ypbind(1M)	Used by each client to determine to which server it should bind. **Note:** This entry exists in the *NFS Services Reference Pages* as *ypserv(1M)*; it exists online as *ypbind(1M)*.
ypcat(1)	Lists the contents of a YP map.

YP Commands	Description
ypinit(1M)	On YP master servers, *ypinit(1M)* constructs maps from */etc* files. On YP slave servers, *ypinit(1M)* copies the initial map versions from the master server.
ypmake(1M)	A script, initially called by *ypinit(1M)*, that builds standard YP maps from ASCII files. These files are usually in */etc*: *passwd*, *hosts*, *group*, *netgroup*, *networks*, *protocols*, *rpc*, and *services*.
ypmatch(1)	Prints the value for one or more specified keys in a YP map.
yppasswd(1)	Changes the password for your current login ID in the YP *passwd* map. (You do not have to have a YP password to access the YP databases.)
yppasswdd(1M)	A server, running only on the master server, that permits users to change their password in the YP *password* map.
yppoll(1M)	Asks any *ypserv(1M)* for the information it holds about a single map.

YP Commands	Description
yppush(1M)	Used by the master server to administer a running YP system. The *yppush(1M)* command causes a YP map to be copied (using *ypxfr(1M)*) from the maps' master server to each slave server in the YP domain.
ypserv(1M)	Provides access to data stored in YP maps on servers. **If operating in an HP-UX cluster environment,** *ypserv(1M)* should be running on the root server.
ypset(1M)	Tells the local *ypbind(1M)* process to obtain YP services for a YP domain from a specific server.
ypwhich(1)	Tells you which server a node is currently using or which server is master of a specified map.
ypxfr(1M)	Transfers a YP map from one server to another. Run *ypxfr(1M)* one of three ways: • *yppush(1M)* periodically, • *ypxfr(1M)* interactively, or • *via cron(1M)* periodically.

YP Configuration

Refer to the following checklist for YP configuration steps.

1. Create a YP Master Server
2. Create a YP Client
3. Create a YP Slave Server
4. Propagate the YP Maps

Before Configuring YP

Compare /etc/newconfig Files to Existing Files

When you installed the NFS services software, several new files were copied into the */etc/newconfig* directory. Perform the following steps to prepare to configure YP.

1. Compare each */etc/newconfig* file listed below with its counterpart shown in the following list.

File in /etc/newconfig directory	Counterpart in /usr/etc/yp directory
ypinit	*ypinit*
yp_Makefile	*Makefile*
ypmake	*ypmake*
ypxfr_1perday	*ypxfr_1perday*
ypxfr_1perhour	*ypxfr_1perhour*
ypxfr_2perday	*ypxfr_2perday*

2. If the files are the same, skip to the next section, "1. Create a YP Master Server."

3. If you have previously customized the files that exist in the */usr/etc/yp* directory, or if the files are from an older release of the software, they will differ from files in */etc/newconfig*. If there are differences, copy the current files in */usr/etc/yp* to a safe location and do **one** of the following:

   - change the versions in */usr/etc/yp* to reflect the differences in the files in */etc/newconfig*.

OR

   - copy the files in */etc/newconfig* to */usr/etc/yp*. Then re-customize the files in */usr/etc/yp* if necessary.

1. Create a YP Master Server

You must be super-user to create a YP master server (i.e., to build the YP master databases). You should also be in a single user state of operation.

A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

Before Creating a YP Master Server

Perform the following steps before creating your master server.

1. Ensure */etc* files are complete and current: *passwd*, *hosts*, *group*, *networks*, *protocols*, *rpc*, and *services*. (Refer to the *NFS Services Reference Pages*.)

2. If you know the correct configuration, create the */etc/netgroup* file. (Refer to *netgroup(4)*.)

Note

The YP maps store only the first occurrence if
- a duplicate user name or duplicate user ID exists in */etc/passwd* or
- a duplicate internet address or duplicate host name exists in */etc/hosts*.

Security

If you want to restrict access to the master server to a smaller set of users than defined by the complete */etc/passwd* file, perform the following steps.

1. Copy the entire */etc/passwd* file to a different file (e.g., */etc/passwd.yp*).

2. Delete undesired users from the original */etc/passwd* file. To prevent all entries in the YP *passwd* map from being able to log in, this smaller file should **not** include the following line.

```
+::0:0:::
```

3. Edit */usr/etc/yp/ypinit* as follows.

   Change: `PWFILE=/etc/passwd`

   To: `PWFILE=/etc/passwd.yp`

4. Edit */etc/netnfsrc* as follows.

   Change: `/usr/etc/rpc.yppasswdd /etc/passwd -m passwd PWFILE=/etc/passwd`

   To: `/usr/etc/rpc.yppasswdd /etc/passwd.yp -m passwd PWFILE=/etc/passwd.yp`

5. If you have *rpc.yppasswdd* running, kill and restart it.

```
/usr/etc/rpc.yppasswdd /etc/passwd.yp -m passwd PWFILE=/etc/passwd.yp
```

If in the future you need to run *ypmake(1M)* and you have restricted access to the master server as just described, enter the following line.

```
/usr/etc/yp/ypmake passwd PWFILE=/etc/passwd.yp
```

Creating a YP Master Server

Refer to the following steps to create your master server. **If operating in an HP-UX cluster environment**, HP recommends that the root server be the master server.

1. Set the YP domain name using the *domainname(1)* command. This YP domain name must be the same one used for all clients and servers within this YP domain.

   `domainname` *YP_domain_name*

2. Execute *ypinit(1M)* with the *-m* parameter.

```
/usr/etc/yp/ypinit -m
```

3. The system asks whether you want the procedure to quit at the first non-fatal error. You may want to answer "no" since you can later correct the errors.

   a. Respond *no* or *n* for *ypinit(1M)* to continue regardless of the errors. After the procedure finishes, correct all errors that occurred.

   b. Respond *yes* or *y* for *ypinit(1M)* to quit at the first error. Correct each error as it occurs. This procedure takes longer since you have to correct the errors one by one and run *ypinit(1M)* until no more errors occur.

4. The *ypinit(1M)* script prompts you for a list of hosts that will become servers. Note: if you want this node to serve a YP domain that is different from the one set by the *domainname(1)* command, use the *DOM=* parameter in *ypinit(1M)*. (For details, refer to *ypinit(1M)*.)

Note

You may save time and work later by adding other hosts now that you expect to have as slave servers. Do not, however, add every host on the network because when the master server updates the databases, it would try to copy its databases to every host.

Starting the YP Master Server

You should edit */etc/netnfsrc* to automatically start the master server at boot time. You can also manually start it now.

Manually Starting YP Master Server

1. If you have not already done so, set the YP domain name using the *domainname(1)* command. This YP domain name must be the same one used for all clients and servers within this YP domain.

```
domainname YP_domain_name
```

2. Execute *ypserv(1M)*.

```
/usr/etc/ypserv
```

Note: If operating in an HP-UX cluster environment, start *ypserv(1M)* only on the root server, and start *ypbind(1M)* on every cnode.

3. Execute *ypbind(1M)*.

```
/etc/ypbind
```

Automatically Starting YP Master Server (at Boot Time)

Edit */etc/netnfsrc* to perform the following actions.

Note: A zero in the *YP_CLIENT, YP_MASTER_SERVER*, or *YP_SLAVE_SERVER* field disables the node from working as a client, master server, or slave server respectively.

- Set YPDOMAINto the YP domain name.

```
YPDOMAIN=YP_domain_name
```

You will need to use this same YP domain name for all clients and servers within this YP domain.

- Set YP_MASTER_SERVER to a value other than zero. Changing this variable permits users to change their YP password.

```
YP_MASTER_SERVER=1
```

- Set the YP_SLAVE_SERVER to zero to disable the node as a slave server.

```
YP_SLAVE_SERVER=0
```

- Set YP_CLIENT to a value other than zero.

```
YP_CLIENT=1
```

2. Create a YP Client

You must be super-user to create a YP client.

A YP client can also be configured as an NFS client, NFS server or both. All YP servers **must** also be configured as YP clients.

Before Creating a YP Client

1. For the client you intend to create, determine a YP domain on your network.
2. Ensure that a server is available in the YP domain in which the client will exist (i.e., YP databases exist and ypserv(1M) is running). (Refer to the "1. Create a YP Master Server" section.) If a server is not availabe in the same YP domain as the client, users will be unable to log into the client.

Creating a YP Client

To ensure nodes access the YP databases, abbreviate or eliminate the following files which traditionally store the information. (For suggested modifications, refer to the following "Altering a Client's Files" section.)

Note

Do **not** abbreviate or eliminate these files if the client is also the master server.

- */etc/group*
- */etc/hosts*
- */etc/netgroup*
- */etc/networks*
- */etc/passwd*
- */etc/protocols*
- */etc/rpc*
- */etc/services*

Altering a Client's Files

The following table provides suggestions for altering the client files.

Client File	Suggested Modification
/etc/group	You may want to reduce */etc/group* to a single line containing a plus (+) followed by a colon (:). This line forces all translations of group names and group IDs to occur via the YP service. `+:`
/etc/hosts	Ensure */etc/hosts* contains an entry for the local host name. The system accesses these entries when the YP service is not yet available. After the *ypbind(1M)* process is running, the system never accesses */etc/hosts*. EXAMPLE: Sample YP client's */etc/hosts* entry `192.9.1.87 local_host # Byron W. Donnell`
/etc/netgroup */etc/networks* */etc/protocols* */etc/services*	Move these files to backup names.

Client File	Suggested Modification
/etc/hosts.equiv **For ARPA Services**	The system first accesses */etc/hosts.equiv* directly. If a +*@netgroup* or -*@netgroup* entry exists, the system accesses the YP *netgroup* map. Note, using *netgroup* reduces *rlogin(1)* and *remsh(1)* problems that occur because different */etc/hosts.equiv* files exist on different nodes. For more control over logins, edit */etc/hosts.equiv* as follows. 1.Enter either a plus (+) or (-) to enable or disable *login*, respectively. 2.Enter the at (@) character. 3.Enter the *netgroup_name* as defined in the global netgroup database. EXAMPLE: +@*netgroup1_name* (trusted) +@*netgroup2_name* (trusted) -@*netgroup3_name* (distrusted)
$HOME/.rhosts **For ARPA Services**	The system first accesses *$HOME/.rhosts* directly. If a +*@netgroup* or -*@netgroup* entry exists, the system accesses the YP *netgroup* map. (Refer to the above */etc/hosts.equiv* example.) Since the super-user's *$HOME/.rhosts* controls remote super-user access to the local node, HP recommends restricted access. To restrict access, either make the list of trusted hosts explicit or use netgroup names.

Client File	Suggested Modification

/etc/passwd	Ensure */etc/passwd* contains
	• entries for the root user,
	• entries for the primary users, and
	• an escape entry to use the YP service.
	
	Entries in the local */etc/passwd* file mask identical name entries in the YP *passwd* maps. Delete all other names and enter *+::0:0:::* as the last line. This line causes library routines looking for a particular entry to search the YP database.
	
	`+::0:0:::`
	
	EXAMPLES: Sample entries in */etc/passwd*
	
	`+ap::::Dave Hamil:/usr2/ap:/bin/csh`
	
	The system pulls an entry for *ap* from the YP *passwd* map because of the + (plus) escape character.
	
	It obtains the *UID*, *GID*, and *password* from the YP and obtains the comment field, home directory, and default shell from the */etc/passwd* entry.
	
	If no entry for *ap* exists in the YP, the system reacts as though no entry for *ap* exists anywhere.
	
	`ap::140:100:Mike Donn:/usr2/ap:/bin/csh`
	
	Since the plus (+) escape character is not present, the system does not access YP. User *ap has no password.*

Client File	Suggested Modification

/etc/passwd	EXAMPLES: Sample entries in */etc/passwd*
	
	`+ap:`
	
	The system obtains all information from the YP *passwd* map for user *ap*.
	
	`+::0:0:::`
	
	The system obtains all information from the YP *passwd* map for all users not already encountered.

Starting the YP Client

You should edit */etc/netnfsrc* to automatically start the client at boot time. You can also manually start it now.

Manually Starting YP Client

1. If you have not already done so, set the YP domain name using the *domainname(1)* command. This YP domain name must be the same one used for all clients and servers within this YP domain.

```
domainname YP_domain_name
```

2.Execute *ypbind(1M).*

```
/etc/ypbind
```

Automatically Starting YP Client (at Boot Time)

Edit */etc/netnfsrc* to perform the following actions.

- Set *YPDOMAIN* to the same YP domain name used on all clients and servers within this YP domain.

```
YPDOMAIN=YP_domain_name
```

- Set YP_CLIENT to a value other than zero.

```
YP_CLIENT=1
```

Note: A zero in the *YP_CLIENT* field disables the node from working as a YP client.

Note

If you want the node to be a server also, refer to either the "1. Create a YP Master Server" or "3. Create a YP Slave Server" section for complete instructions.

---

3. Create a YP Slave Server

You must be super-user to create a YP slave server.

You may want to create slave servers to improve the reliability of your system.

A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

Before Creating a YP Slave Server

Before creating a slave server, ensure the

- master server exists (see "1. Create a YP Master Server" section) and
- *ypserv(1M)* is running on the master server.

Creating a YP Slave Server

Refer to the following steps to create a slave server.

1. Set the YP domain name using the *domainname(1)* command. This YP domain name must be the same one used for all clients and servers within this YP domain.

   ```
   domainname YP_domain_name
   ```

2. Execute *ypinit(1M)* with the *-s* parameter.

   ```
   /usr/etc/yp/ypinit -s master_server_name
   ```

3. The system asks whether you want the procedure to quit at the first non-fatal error. You may want to answer "no" since you can later correct the errors.

   Note: if you want this node to serve a YP domain that is different from the one set by the *domainname(1)* command, use the *DOM=* parameter to *ypinit(1M)*. (For details, refer to *ypinti(1M)*.) If you use the *DOM=* parameter, ensure that the master server serves the domain that you specify.

   a. Respond *no* or *n* for *ypinit(1M)* to continue regardless of the errors. After the procedure finishes, correct all errors that occurred.

   b. Respond *yes* or *y* for *ypinit(1M)* to quit at the first error. Correct each error as it occurs. This procedure takes longer since you have to correct the errors one by one and run *ypinit(1M)* until no more errors occur.

4. Since the slave server is also a client, abbreviate or eliminate the files which traditionally implement the database. Refer to the previous table "Altering a YP Client's Files" in the "2. Create a YP Client" section.

Starting the YP Slave Server

You should edit */etc/netnfsrc* to automatically start the slave server at boot time. You can also manually start it now.

Manually Starting YP Slave Server

1. If you have not already done so, set the YP domain name using the *domainname(1)* command. This YP *domain name* must be the same one used for all clients and servers within this YP domain.

```
domainname YP_domain_name
```

2.Execute *ypserv(1M)*.

```
/usr/etc/ypserv
```

Note: If operating in an HP-UX cluster environment, start *ypserv(1M)* only on the root server, and start *ypbind(1M)* on every cnode.

3. Execute *ypbind(1M)*.

```
/etc/ypbind
```

Automatically Starting YP Slave Server (at Boot Time)

Edit */etc/netnfsrc* to perform the following actions.

Note: A zero in the *YP_CLIENT*, *YP_MASTER_SERVER*, or *YP_SLAVE_SERVER* field disables the node from working as a client, master server, or slave server respectively.

- Set YPDOMAIN to the same YP domain name used on all clients and servers within this YP domain.

```
YPDOMAIN=YP_domain_name
```

- Set the YP_MASTER_SERVER to zero to disable the node as a master server.

```
YP_MASTER_SERVER= 0
```

- Set YP_SLAVE_SERVER to a value other than zero.

```
YP_SLAVE_SERVER=1
```

- Set YP_CLIENT to a value other than zero.

```
YP_CLIENT=1
```

4. Propagate YP Maps

"Propagate YP maps" means to copy a map from one server to another. Initially, *ypinit(1M)* copies the maps when you create slave servers.

After the slave servers are initialized, you will use *ypxfr(1M)* to transfer updated maps from the master server to the slaves. You can run *ypxfr(1M)* three ways:

- periodically from *cron(1M)* on each slave server,
- periodically by executing *yppush(1M)* on the master server, or
- interactively executing *ypxfr(1M)* on a slave server.

crontab(1)

Maps have different change rates. For example, *protocols.byname* may not change for months, but *passwd.byname* may change several times a day.

Create *crontab(1)* entries to periodically run *ypxfr(1M)* at a rate appropriate for each map in the YP database. The *ypxfr(1M)* command will contact the master server and transfer the map only if the master's copy is more recent than the local copy.

To avoid a *crontab(1)* entry for each map, group the maps with approximately the same change characteristics. Place these maps in a shell script you can run via *cron(1M)*.
Suggested groupings, mnemonically named, are in */usr/etc/yp: ypxfr_1perhour*, *ypxfr_1perday*, and *ypxfr_2perday*. If the rates of change are inappropriate for your needs, either modify or replace these shell scripts.

Execute these shell scripts on each slave server in the YP domain. Alter the exact time of execution from one server to another to prevent this process from slowing down the master.

EXAMPLE: *crontab(1)* entries for using these scripts

```
# At 9:00 PM daily, transfer the group, networks, protocols,
# rpc, services, and ypservers maps.

    0 21 * * * /usr/etc/yp/ypxfr_1perday

# At 45 minutes past the hour, transfer the passwd maps.

45 * * * * /usr/etc/yp/ypxfr_1perhour

# At 11:30 AM and 11:30 PM daily, transfer the ethers,
# hosts,mail.aliases and netgroup maps.

        30 11,23 * * * * /usr/etc/yp/ypxfr_2perday
```

You can check and transfer maps with unique change characteristics by explicitly invoking *ypxfr(1M)* from within your *crontab(1)* file.

EXAMPLE: `25,55 * * * * /usr/etc/yp/ypxfr passwd.byname`

yppush(1M) Execute *yppush(1M)* only on the master server to copy a map to each server in the YP domain (retrieved from the *ypservers* map).

1. The *yppush(1M)* command sends a "transfer map" request to each of the servers.

2. In turn, *ypserv(1M)* on each server executes *ypxfr -C*.

3. The *ypserv(1M)* daemon then passes *ypxfr(1M)* the information needed to identify and transfer the map.

EXAMPLE: `/usr/etc/yp/yppush passwd.byname`

ypxfr(1M) Execute *ypxfr(1M)* interactively only in exceptional situations. For example, execute it when creating a temporary server to make a test environment, or when trying to quickly propagate maps to make a server consistent with the other servers.

EXAMPLE: `/usr/etc/yp/ypxfr` *map_name*

If you want the map transferred from a server other than the master, specify it using the *-h* option with *ypxfr(1M)*.

EXAMPLE:

`/usr/etc/yp/ypxfr -h` *server_name* `passwd.byname`

Verify YP

To verify a client is bound to a server, login to that client and execute *ypwhich(1)*.

- If the client is bound, the response will be the host name of that server.
- If the client is not bound, you will receive the following message.

```
YP domain domain_name not bound.
```

If you try *ypwhich(1)* several times and continue to receive the *not bound* response, the node is unable to locate a server for that YP domain on the network. Review your YP configuration process. If you did not make errors, refer to the "Troubleshooting" chapter.

To verify the YP is being accessed, login to a client node as a user whose password entry must be served by the YP. If the login does not work, review your YP configuration process. If you did not make errors, refer to the "Troubleshooting" chapter.

Note

You have now completed configuring YP. If you are configuring YP for the first time (with NFS Services), and you plan to use the Virtual Home Environment (VHE), you can now skip to the "VHE Configuration and Maintenance" chapter. If you do not plan to use VHE, return to the section, "7. Execute /etc/netnfsrc" in the "NFS Configuration and Maintenance" chapter.

YP Maintenance

To keep YP running correctly and efficiently, ensure it stays configured to meet your changing needs. Refer to the following sections to help you meet these needs.

- Diable YP
- Modify YP Maps
- Add New YP Servers
- Add New Users to a Node
- Make a Different Node the YP Master
- Change YP Password
- Log Files
- Create Non-standard YP Maps

Disable YP

If you choose to disable the YP service, use the following steps.

1. Set the YP domain name to null (no spaces).

```
domainname ""
```

2. If the YP service is currently running, kill the *ypbind(1M)* and *ypserv(1M)* processes.

3. Edit */etc/netnfsrc* to change the YP values.

   a. Change the *YP_MASTER_SERVER*, *YP_SLAVE_SERVER*, and *YP_CLIENT* values to zero.

   ```
   YP_MASTER_SERVER=0
   YP_SLAVE_SERVER=0
   YP_CLIENT=0
   ```

   b. Remove the *YPDOMAIN* variable if one exists.

   ```
   YPDOMAIN=
   ```

   If this YP domain is specified in */etc/netgroup*, remove the YP domain name throughout this file.

4. If the above YP domain is specified in */etc/netgroup*, remove the YP domain name throughout this file.

5. Restore any files that you altered for YP use. For example, you may need to add users back to the */etc/passwd* file.

6. Reboot the system.

Modify YP Maps

You must be super-user to modify YP maps.

Note

Modify maps **only on the master server**; otherwise, the changes will not be propagated correctly to the slave servers.

You may change most of the standard YP maps, like */etc/hosts*, by first editing the ASCII file and then running *ypmake(1M)*. Refer to the following "Manual Modifications to YP Maps" section if you are

- adding non-standard maps,
- editing maps for which no ASCII file exists, or
- changing the set of servers after the system is running.

Whether using *ypmake(1M)* in */usr/etc/yp* or one of the following manual procedures, the goal is the same: a new, well-formed database must reside in the YP domain directory on the master server. (Refer also to *makedbm(1M)*).

Note

Never modify a map directly; **always** use *makedbm(1M)* to create the map.

Manual Modifications to YP Maps

You may want to change the following maps manually.

- Non-standard maps (i.e., those that are specific to the applications of a particular vendor or site, but are not part of HP's release)
- Maps that rarely change
- Maps for which no ASCII file exists (e.g., *ypservers* map)

1. Move to the directory in which the maps you want to modify exist.

   ```
   cd /user/etc/yp/YP_domain_name
   ```

2. Execute *makedbm -u* to disassemble the map into a form which is modifiable using HP-UX tools.

   a. Redirect the *makedbm -u* output to a temporary file and modify it. Execute *makedbm(1M)* using the temporary file as input to create the new versions.

   EXAMPLE:

   ```
   ../makedbm -u mapname > tmpfile
   vi tmpfile  #  (make the required changes)
   ../makedbm tmpfile mapname
   rm tmpfile
   ```

   b. Use a pipe to modify the *makedbm(1M)* output which you can then direct as input to *makedbm(1M)*. Note, you can use this method only if the disassembled map is updated via *awk*, *sed*, or a *cat* append.

   EXAMPLE: Add a new key-value pair to the *map_name* map

   ```
   ( ../makedbm -u map_name; echo  newkey newvalue  ) | \
   ../makedbm - map_name
   ```

EXAMPLE: Suppose you want to create a non-standard YP map. You want it to consist of key-value pairs in which the keys are strings like *al*, *bl*, *cl*, and *dl*, and the values are *ar*, *br*, *cr*, and *dr*. After creating the map, you notice it is missing *dl* and *dr*.

You could use one of two procedures to create the new map: one using an existing ASCII file, the other using standard input.

Example: Existing ASCII File

Assume the following situation.

- An ASCII file exists named */usr/etc/yp/john_map.asc*
- The file was created with an editor or shell script on the master server
- *john_map* is the name of the map you want to recreate
- *graphs_domain* is the YP domain subdirectory where the map is located
- The YP map was created from this file by entering

```
cd /usr/etc/yp
./makedbm john_map.asc graphs_domain/john_map
```

Now you notice the map is missing *dl* and *dr*. To correct the error, **modify the map by first modifying the ASCII file** as follows.

```
cd /usr/etc/yp
<make editorial change to john_map.asc to add
the dl and dr line>
./makedbm john_map.asc graphs_domain/john_map
```

To verify the new map has the changes you made, enter the following command.

```
./makedbm -u graphs_domain/john_map | more
```

Example: Using Standard Input

Assume the following situation.

- *wes_map* is the name of the map you want to create (no ASCII file exists from which the map was built)
- *reports_domain* is the YP domain subdirectory in which you will create the map

First, create the YP map from the keyboard by entering input on the master server as follows.

```
cd /usr/etc/yp
./makedbm - reports_domain/wes_map
al ar
bl br
cl cr
CTRL-D
```

To modify the map, use *makedbm(1M)* to create a temporary ASCII intermediate file that can be edited.

```
cd /usr/etc/yp
./makedbm -u reports_domain/wes_map > wes_map.temp
```

Now edit *wes_map.temp* to add the *dl* and *dr* line. Create a new version of the database with the following commands.

```
./makedbm wes_map.temp reports_domain/wes_map
rm wes_map.temp
```

Add New YP Servers

You must be super-user to add new YP servers.

If a new slave server is not in the original set, recreate the *ypservers* map on the master server. If needed, rebuild the *hosts* map also. (Refer to *ypmake(1M)*.)

1. If the server's address is not in */etc/hosts*, edit */etc/hosts* to include the new server's address and then execute *ypmake(1M)*.

   ```
   <Edit /etc/hosts>
   /usr/etc/yp/ypmake hosts
   ```

2. Add the host's name to the *ypservers map in the YP domain as follows.*

   ```
   cd /usr/etc/yp
   (./makedbm -u home_domain/ypservers;\
   echo new_ypslave_name) | ./makedbm - home_domain/ypservers
   yppush ypservers
   ```

3. On the new slave server, complete the steps in the "3. Create a YP Slave Server" section.

Add New Users to a Node

You must be super-user to add new users to a node.

Refer to the *HP-UX Series 300 System Administrator Manual* to add new users to a node. The procedure consists of (1) editing the master server's */etc/passwd* and */etc/group* files, (2) making a home directory, and (3) defining the new user's environment.

Remember to update the YP *passwd* and *group* databases by running */usr/etc/yp/ypmake*. If you are using an alternate file to build the YP *password* databases, use its full path name instead of */etc/passwd.*

```
/usr/etc/yp/ypmake group passwd PWFILE=alternate_passwd_file
```

Make a Different Node the YP Master

You must be super-user to change the YP master server to a different node.

1. Copy the following files from your current master server to the node that will be the new master server.

   - */etc/hosts*
   - */etc/netgroup*
   - */etc/networks*
   - */etc/protocols*
   - */etc/rpc*
   - */etc/services*

2. Merge */etc/group* and */etc/passwd* on the current master server with those on the node that will be the new master server. (If using an alternate password file, you need only copy it.) This merging creates files suitable for building maps for all clients.

   Merging ensures machine-specific password and group entries are kept intact. Either save or delete entries taken from the old master server files. For example, in */etc/passwd* save user entries and remove the other node's root entry.

3. If */usr/etc/yp/ypmake*, */usr/etc/yp/ypinit*, or */usr/etc/yp/Makefile* was modified on the old master server to build non-standard maps, copy them and other files from which the non-standard maps are built.

4. On the new master server, complete all steps in the "1. Create a YP Master Server" section.

5. To prevent starting *yppasswdd(1M)* on the old master server, edit its */etc/netnfsrc* file to change the *YP_MASTER_SERVER* value to zero.

```
YP_MASTER_SERVER=0
```

6. If the old master server is to be a slave server, complete the steps in the "3. Create a YP Slave Server" section and the steps in the "2. Create a YP Client" section.

7. Reboot the new master server.

8. Reboot the old master server.

9. To ensure maps are consistent on all servers, execute *ypinit(1M)* on each slave server using the new master server's host name.

```
ypinit -s new_master_hostname
```

Create or Change YP Password

The YP password is the password for a user's login ID that exists in the YP *passwd* map. The YP password is used as the user password, but is administered through YP. Note, you do not have to have a YP password to access the YP database.

If you change your password with the *passwd(1)* command, you will change only the entry in your local */etc/passwd* file if the entry exists. If your password is not in the file, the following error message occurs.

```
Permission denied.
```

If this error occurs, execute *yppasswd(1)*.

YP Password Guidelines

The following list provides the requirements for creating and changing YP passwords.

- Only the owner or super-user can change a YP password. The super-user must know the current YP password to change another user's YP password.
- Only the first eight characters of the YP password are significant; the rest are truncated.
- A YP password must contain at least five characters if it includes a combination of either
  - uppercase and lowercase letters or
  - alpha-numeric characters.
- A YP password must contain at least four characters if it includes a combination of uppercase letters, lowercase letters, and numeric characters.
- A YP password must contain at least six characters if it includes only monocase letters.

Note

You can change a YP password in the YP *passwd* map using *yppasswd(1)* only if *rpc.yppasswdd* is running on the master server. (See *yppasswdd(1M)*.)

YP Password

Refer to the following steps to create or change your YP password in the YP *passwd* map.

1. Execute the *yppasswd* command.

   ```
   yppasswd user_login_name
   ```

2. The system prompts you for the old YP password even if one does not exist. If it does exist, enter the old YP password; otherwise, press **RETURN.**

   Note, the YP password may be different from the one on your local node.

3. The system prompts you for the new YP password twice to ensure you enter the correct response. Enter your new YP password twice, pressing **RETURN** after each entry.

   The system now updates the master server *passwd* map.

Log Files

Using the *-l* option, you can execute *ypbind(1M)*, *ypserv(1M)*, and *yppasswdd(1M)* so that diagnostic and error messages are written to log files.

```
/etc/ypbind -l ypbind_log_file
/usr/etc/ypserv -l ypserv_log_file
/usr/etc/rpc.yppasswdd -l yppasswdd_log_file
```

Preceding each message logged to the file are the date, time, host name, process ID, and daemon name generating the message. Since the messages are uniquely identified by this information, these daemons can share a single log file.

If you execute the daemons without the *-l* option, the following responses occur.

- The *ypbind(1M)* daemon writes its messages directly to the system console, */dev/console*.
- The *ypserv(1M)* daemon writes its messages to the */usr/etc/yp/ypserv.log* file if it exists when *ypserv(1M)* is started.
- The *yppasswdd(1M)* daemon provides no messages.

The *ypxfr(1M)* command appends transfer information (which map from which server and how many entries it has) to the file */usr/etc/yp/ypxfr.log* if it exists. The logging occurs only if *ypxfr(1M)* is not being run directly by someone at a terminal.

EXAMPLE: Logging occurs if the log file exists and *cron(1M)* is running *ypxfr(1M)* directly, using a *crontab(1)* entry like the following one.

```
30 * * * * /usr/etc/yp/ypxfr yp_map
```

All log files could potentially grow without limit until they use up the available file system space. To avoid this occurrence, periodically check the file sizes. One method of preventing this problem is to create a *crontab(1)* entry for each log file as follows.

```
0 1 * * 1,3,5 cat /dev/null > log_file
```

This line truncates *log_file* at 1:00 A.M. every Monday, Wednesday, and Friday.

Create Non-standard YP Maps

You must be super-user to create and propagate non-standard YP maps.

The */usr/etc/yp/ypmake* file supports all of the standard maps shipped by HP. Non-standard maps are those maps which you create that are not originally supported by the */usr/etc/yp/ypmake* file.

1. Modify */usr/etc/yp/ypmake* on the master server so the map can be rebuilt. Modification requirements vary extensively. Generally, though, you need to filter a human-readable ASCII file through HP-UX utilities.

   If the file system in which /usr/etc/yp exists supports only short file names (14 characters maximum), limit the new map name lengths to 10 characters maximum. Note, however, the system automatically handles the longer standard YP map names.

2. If using *Makefile* in */usr/etc/yp* on the master server to build the maps, modify it so the new map can be rebuilt. (See *ypmake(1M)*.)

3. Modify */usr/etc/yp/ypinit* on the master server to include the name of your new map in the list of *MASTER_MAPS*. Copy this modified script to all server nodes. This process ensures that any re-initialized or new slave servers will serve the new map.

4. For a client to access the data in the new map, it must exist on each of the servers. Execute the newly modified *ypmake(1M)* on the master server to build and copy the map to the current slave servers.

   ```
   /usr/etc/yp/ypmake
   ```

   Slave server support for the propagation of new maps consists of adding *crontab(1)* entries or adding new entries to one of the *ypxfr(1M)* shell scripts described in the "Propagate YP Maps" section.

EXAMPLE: This example spans several pages. Refer to the bold face headings for different sections of the example.

Initial Example Environment

Keep a list of the login names and the host names of all nodes on which each user is allowed to login.

- The information is stored in */usr/etc/access_list*.
- The custom YP map you wish to build from this file is *access*.

The general form of the ASCII file */usr/etc/access_list* is as follows.

```
login_name1 [ host_name1 [ host_name2 ... ] ]

login_name2 [ host_name1 [ host_name2 ... ] ]
.
.
.
login_namen
[ # comments ]
```

- Each user has only one line.
- After each login name are zero or more host names. The user can log into any of these hosts.
- You can use both comments with a # (pound sign) in column one and blank lines.

The following samples could be in */usr/etc/acc_list*.

```
carole      alpha       catfish      handel
gerbil      catfish

# bigmak is a new hire who has not yet arrived

bigmak
mr_jad      axesys      handel
daveysan    satie       yogurt
chum        handel
speedy      handel      satie        catfish
fielding    alpha       beta         catfish
```

All of the users except for *bigmak* can login on one or more systems.

You may want to use the login name as the key for storing this data in the *access* map so you can search the map with commands like *ypmatch*.

```
%ypmatch chum gerbil bigmak carole access
```

The above *ypmatch* command would provide output like the following.

```
chum        handel
gerbil      catfish
bigmak
carole      alpha       catfish    handel
```

Modify *ypmake*

Modify */usr/etc/yp/ypmake* on the master server as follows.

1. Insert a new function called *access()* after the *services()* function.

```
access(){
        grep -v ^[   ]*# $1 | grep -v ^[   ]*$ | \
       awk 'BEGIN { OFS= \t ; } { print $1, $0 }' | \
       $MAKEDBM - $MAPDIR/access
}
```

This function creates a map that has a key as the first field of each input record, creates a value that is the entire record, and skips over comment lines.

2. Add a new pattern to the case statement that is preceded by "`for ARG in $*; do.`"

You **must** place this information before the pattern `*) in the case statement.`

```
access)
    if [ 'expr  $MAPS  :  .* access.* ' -eq 0 ]; then
         MAPS= $MAPS access
    fi;;
```

3. Add the new map name to the default list of MAPS to build. This addition ensures all maps are built (including the access map) if ypmake(1M) is called with no maps specified.

```
MAPS=${MAPS:-'passwd group hosts networks rpc \
            services protocols netgroup access'}
```

4. Add a new pattern to the case statement that is preceded by "`for MAP in $MAPS; do.`"

```
access)     build /usr/etc/access_list access;;
```

Modify *Makefile*

If using the *Makefile* in */usr/etc/yp* on the master server to build the maps, modify it as follows.

1. Insert a new variable called *ACCESS* after the *SERVICES* variable.

```
SERVICES = services services.byname
ACCESS = access
```

2. Add the new *ACCESS* variable to the definition of the *ALL_MAPS* variable.

```
ALL_MAPS= ${PASSWD} ${GROUP} ${HOSTS} \
    ${NETWORKS} ${RPC} ${SERVICES} \
    ${PROTOCOLS} ${NETGROUP} ${ACCESS}
```

Modify *ypinit*

1. Modify the */usr/etc/yp/ypinit* shell script on the master server to include the new map in list of all maps built on the master server.

```
MASTER_MAPS= group.bygid group.byname \
        hosts.byaddr hosts.byname netgroup netgroup.byhost \
        netgroup.byuser networks.byaddr networks.byname\
        passwd.byname passwd.byuid protocols.byname \
        protocols.bynumber rpc.bynumber services.byname
        access
```

2. Copy this modified script to all current and future YP servers.

Maintain a Current Access Map on Each Slave Server

1. Execute the newly modified *ypmake(1M)* on the master server to build and copy the *access* map to the current slave servers.

```
/usr/etc/yp/ypmake
```

2.On each slave server, modify the appropriate *ypxfr(1M)* script to periodically copy the *access* map from the master server.

```
# ypxfr_1perday - Perform daily YP map check and updates
/usr/etc/yp/ypxfr group.bygid
/usr/etc/yp/ypxfr group.byname
.
.
.
/usr/etc/yp/ypxfr access
```

Check the Map's Contents

Execute a few YP commands to verify the success of your work.

```
%ypwhich -m
services.byname     host1
.
.
.
access              host1
```

This *ypwhich -m* command shows that the server you are bound to now serves the *access* map.

Note, the order of the *ypcat(1)* listing does not match the order of your file contents.

```
% ypcat access
fielding      alpha        beta        catfish
daveysan      satie        yogurt
speedy        handel       satie       catfish
mr_jad        axesys       handel
gerbil        catfish
carole        alpha        catfish     handel
bigmak
chum          handel
```

The following *ypmatch(1)* command shows how you can selectively retrieve information from your new *access* map.

```
% ypmatch speedy daveysan fielding mr_jad access
speedy        handel       satie       catfish
daveysan      satie        yogurt
fielding      alpha        beta        catfish
mr_jad        axesys       handel
```

9

VHE Configuration and Maintenance

Virtual Home Environment (VHE) is an HP-developed service that allows you to configure login environments on remote nodes to mirror the login environment on the users' home nodes. VHE is available to any HP-UX system on a network running the NFS Services product.

You can choose whether to configure and use the service, although when you install NFS Services, VHE is also installed. For an overview of how VHE works, refer to the "NFS Services Overview" chapter.

Note

The Yellow Pages (YP) service is not mandatory for using VHE, but this chapter shows how to use VHE assuming YP is configured and used.

If you do not plan to use YP, you must have an alternate process for maintaining consistency of the */etc/passwd* and */etc/vhe_list* files for all nodes in the VHE group.

Configuration Overview

The following list is an **overview** of the steps you must complete to configure the nodes on your network with VHE. The steps are described in more detail after the overview list.

1. Prepare for configuring nodes with VHE by obtaining node names for the hosts in your network that will use VHE, installing and configuring NFS Services, and installing and configuring YP (or instituting an alternate mechanism for maintaining consistent user and group IDs, internet address to host name mappings, password entries and *vhe_list*).

2. Compare VHE files in */etc/newconfig* directory with existing files in the */usr/etc/vhe* directory.

3. For each node, decide which file systems are to be mounted and determine the names of mount point directories.

4. Create */etc/vhe_list* on the YP master server using the information from step 3.

5. Edit the */etc/passwd* file on the YP master server node to contain users' home directories which, in turn, contain the appropriate mount point directories.

6. Distribute the new */etc/vhe_list* and */etc/passwd* files by executing *ypmake* on the master YP server.

7. On each node, edit */etc/exports*.

8. On each node using VHE, execute */usr/etc/vhe/vhe_mounter*.

9. Verify that VHE is running correctly.

Configuration

The following sections describe the configuration steps in detail.

Note

You must be super-user to configure VHE.

---

1. Complete Preparation Steps

For each node that will use VHE, perform the following steps:

- obtain a host name.
- install and configure NFS Services.
- install and configure YP (or institute your own mechanism for maintaining consistent host names, group and password entries).

To obtain the host names for the nodes on your network that will use VHE, check your */etc/hosts* file. If YP is running, you can use the *ypcat hosts* command to look at the host information.

To install and configure NFS Services, refer to the "NFS Configuration and Maintenance" chapter.

To install and configure YP, refer to the "YP Configuration and Maintenance" chapter. VHE can use YP for file administration. For VHE to function, it needs all of the nodes using VHE to have a consistent view of the */etc/passwd* and /etc/vhe_list files. YP provides this; if not using YP, you must ensure consistency by some other method.

The */etc/vhe_list* file contains a list of all of the nodes that are using NFS to do the same remote mounts. (This is explained in detail in "4. Create /etc/vhe_list.")

YP maintains single versions of the */etc/passwd* and */etc/vhe_list* files on the YP master server. From the YP master server, you can add or delete users, change users' home nodes and directories, and add or delete nodes from the VHE group. Once changes are made to */etc/passwd* and */etc/vhe_list*, the changes are made in the YP maps and propagated to the YP slave servers through the *ypmake* program.

2. Compare /etc/newconfig Files to Existing Files

When you installed the NFS services software, several new files were copied into the */etc/newconfig* directory. Perform the following steps to prepare to configure VHE.

1. Compare each */etc/newconfig* file listed below with its counterpart shown in the following list.

File in /etc/newconfig directory	**Counterpart in /usr/etc/vhe directory**
vhe_mounter	*vhe_mounter*
vhe_script	*vhe_script*

2. If the files are the same, skip to the next section, "Determine File Systems and Mount Point Directories."

3. If you have previously customized the files that exist in the */usr/etc/vhe* directory, they will differ from those in */etc/newconfig*. If there are differences, copy the current files in */usr/etc/vhe* to a safe location and do **one** of the following:

   - change the versions in */usr/etc/vhe* to reflect the differences in the files in */etc/newconfig*.

OR

   - copy the files in */etc/newconfig* to */usr/etc/vhe*. Then re-customize the newly copied files in */usr/etc/vhe* if necessary.

3. Determine File Systems and Mount Point Directories

For each node that is using VHE, determine and write down the file systems you want to mount and the directories you want to use as mount points. Use the following conventions when completing this step.

- Begin each mount point pathname with a common path component. (In the examples for this manual, */vhe* is used.)

- Attach to the above pathname the host name of the machine you plan to mount. For example, for a machine named *vic*, the mount point pathname is */vhe/vic*. The machine name must match **exactly** the name returned by the *hostname* command (e.g., letters that are in lower case must be typed as lower case and letters that are upper case must be typed as upper case).

- For each file system that will be mounted from each machine to be connected with VHE, attach the file system name to the mount point name. To continue with the above example, if the machine *vic* has two file systems to be mounted: / and */users*, this would result in the pathnames for the two mount points to be */vhe/vic/* and */vhe/vic/users*. In the case of */vhe/vic/*, you should delete the / at the end of the pathname, resulting in the mount point */vhe/vic*.

4. Create /etc/vhe_list

The */etc/vhe_list* file contains a list of all directories that are mount points for your VHE environment. Each node accesses this list for the most current mount point information via NFS mounts. File systems of the remote node are mounted on the appropriate mount point using NFS.

To create the */etc/vhe_list* file, complete the following items.

- As super-user, edit a file named *vhe_list* in the */etc* directory of the YP master server. The *vhe_list* file is installed at the time the NFS product is installed.
- For each mount point on each node create a one-line entry with the following form:

  hostname file_system mount_point [*mount_options*]

  where

  - *hostname* is the name of the node whose file system is mounted.
  - *file_system* is the name of the remote file system on the node to be mounted.
  - *mount_point* is the name of the local directory that acts as the mount point for the NFS mount.
  - *mount_options* is an optional field in *vhe_list* that contains options that are passed to the *mount* command. There should be no spaces between items in the *mount_options* field, and the items should be separated by commas. For example, to set the read and write size to 1024 bytes this field would look like:

    rsize=1024,wsize=1024

Later, the */usr/etc/vhe/vhe_mounter* script uses these fields to perform the appropriate NFS mounts. This script also creates the directories that will be the mount points, so it is not necessary for you to create these directories. If a file exists with the same name as one of the mount point directories, the script produces an error message. In this case, you need to either change the name of the existing file or change the name of the mount point directory.

If you are not using YP, after you create the */etc/vhe_list* file you need to distribute the */etc/vhe_list* file to all the nodes in the VHE group.

Example: Simple Configuration with Single File System per Node

In the simplest case, each node has only one file system which is the root file system. Every node needs to have a set of directories for all members of the group. For example, consider a group consisting of the nodes A, B, C and D. A list of mount points for this group is /vhe/A, /vhe/B, /vhe/C and /vhe/D. Now taking these two lists, an */etc/vhe_list* file with the following contents is created:

```
A  /  /vhe/A
B  /  /vhe/B
C  /  /vhe/C
D  /  /vhe/D
```

Example: Node with Multiple File Systems

Note

If you do not have multiple file systems on each node, you can go to "5. Update /etc/passwd."

Doing mounts of several file systems from one node requires some care in creating the */etc/vhe_list* file. For example, if */usr* is a separate file system on node C, and you execute the following on node A:

```
mount C:/  /vhe/C
```

An *ls* of */vhe/C/usr* on node A shows it as an empty directory because NFS allows access to separate file systems only if they are explicitly mounted.

This directory can be used to do a *mount* of the */usr* file system of node C by executing the following on node A:

```
mount C:/usr  /vhe/C/usr
```

Now an *ls* of */vhe/C/usr* on node A shows the contents of the */usr* file system on node C.

The example group is changed to show this complication with additional file systems:

A	1 file system under "/"
B	2 file systems one under "/" and one under "/users"
C	2 file systems one under "/" and one under "/usr"
D	1 file system under "/"

When a node has multiple file systems, you may choose to have all the file systems mounted (as with C) or to have only some of the file systems mounted (as with B). When */usr/etc/vhe/vhe_mounter* is run, the mount point directories are created, if necessary, and the NFS mounts are made.

Using the rules outlined in "4. Create /etc/vhe_list," for the above group of nodes, you would create the following */etc/vhe_list* file:

```
A  /       /vhe/A
B  /users  /vhe/B/users
C  /       /vhe/C
C  /usr    /vhe/C/usr
D  /       /vhe/D
```

5. Update /etc/passwd

Update the */etc/passwd* file on the YP master server to force home directory access through the mount points. The entries in */etc/passwd* should have the following form:

login_name:encrypted_password:UID:GID:comment:/vhe/hostname/home_dir:shell

The following example shows */etc/passwd* file entries before and after the VHE configuration.

Note

If you are not using YP, after updating the */etc/passwd* file, you must distribute the changes to all nodes in the VHE group.

Example

In this example, the first user 's home directory is on node A; the second user's home directory is on node B; and the third user's home directory is on node C. All of the */users* directories are in the root file systems on their respective nodes.

Before VHE configuration:

```
andy::117:100:andy:/users/andy:/bin/csh
speedy::118:100:darren:/users/speedy:/bin/ksh
chum::119:200:Cris:/users/chum:/bin/sh
```

After VHE configuration:

```
andy::117:100:andy: /vhe/A/users/andy:/bin/csh
speedy::118:100:darren: /vhe/B/users/speedy:/bin/ksh
chum::119:200:Cris: /vhe/C/users/chum:/bin/sh
```

Example: Nodes with Multiple File Systems

Nodes with multiple file systems do not change how the home directories are updated for VHE. For example, consider the following two entries in */etc/passwd*. Fielding's home node is node B, which has two file systems; Jeff's home node is node C, which has two file systems. The nodes are from the example shown above.

Before VHE configuration

```
Fielding::120:200:fielding:/users/fm:/bin/csh
Jeff::121:100:Jeff:/users/jbrl:bin/csh
```

After VHE configuration

```
Fielding::120:200:fielding: /vhe/B/users/fm:/bin/csh
Jeff::121:100:Jeff: /vhe/C/users/jbrl:/bin/csh
```

For node B, */users* is its own file system and is mounted on the directory */vhe/B/users*. This causes no change in the naming convention for the home directory. For node C, */users* is on the root (/) file system. Node C also has another file system: */usr*. If Jeff wants to be able to change the default pathname to his mail file from */usr/mail/jbrl* to */vhe/C/usr/mail/jbrl* (to read mail via VHE), the */usr* file system must be mounted on */vhe/C/usr*.

6. Update /etc/exports

On each node that needs to export file systems, edit the */etc/exports* file to reflect all of the file systems that are available for NFS mounting from each node. Details on this can be found in the "NFS Configuration and Maintenance" chapter.

7. Distribute /etc/vhe_list and /etc/passwd

To distribute the */etc/vhe_list* and */etc/passwd* files (i.e., make them accessible to all the nodes using YP that are part of the same YP domain), execute the following command on the YP master server.

```
/usr/etc/yp/ypmake
```

This builds the YP maps and propagates the maps to the YP slave servers.

8. Execute /usr/etc/vhe/vhe_mounter

Note

The */usr/etc/vhe/vhe_mounter* script should be run when all nodes in the VHE group are powered up and ready for NFS mounting. If they are not ready for NFS mounting, then error messages are printed. These are not fatal errors; to recover from them you should retry *vhe_mounter* when the nodes are available for mounting.

The */usr/etc/vhe/vhe_mounter* script uses the information in */etc/vhe_list* to create the appropriate mount point directories on each node. When *vhe_mounter* notices that it is about to make a directory with the same name as the node from which *vhe_mounter* is executed, it makes a symbolic link with the same pathname and links it to the node's root directory. When the *vhe_mounter* process completes running on each node, the proper mount points and symbolic links are created for each node.

The */usr/etc/vhe/vhe_mounter* script also does NFS mounts using the appropriate directories to the remote machines on each node. When the mounts are complete, a node is ready for VHE.

To execute */usr/etc/vhe/vhe_mounter* for each node separately, execute the following script on each node:

```
/usr/etc/vhe/vhe_mounter
```

To run */usr/etc/vhe/vhe_mounter* for all nodes using VHE from a single node, execute the following as a batch file.

```
for i in ` ypcat vhe_list | awk '{ print $1 } ' | sort -u `
do
 remsh $i /usr/etc/vhe/vhe_mounter
done
```

Note

For this script to execute correctly, all nodes must be running ARPA/Berkeley Services with super-user capability allowed between the nodes when using *remsh.*

Example

This example shows the mount points and symbolic links resulting from the following */etc/vhe_list* file:

```
A  /  /vhe/A
B  /  /vhe/B
C  /  /vhe/C
D  /  /vhe/D
```

The listing below shows the mount points and symbolic links for each node after the */usr/etc/vhe/vhe_mounter* script completes running on each node (symlink = / denotes a symbolic link to the root (/) directory):

Node				
A	/vhe/A symlink = /	/vhe/B Directory	vhe/C Directory	/vhe/D Directory
B	/vhe/A Directory	/vhe/B symlink = /	/vhe/C Directory	/vhe/D Directory
C	/vhe/A Directory	/vhe/B Directory	/vhe/C symlink = /	vhe/D Directory
D	/vhe/A Directory	/vhe/B Directory	/vhe/C Directory	/vhe/D symlink = /

9. Verify that VHE is Correctly Configured

To check if VHE is configured correctly, pick a login name that had a mount point added to its home directory. After */usr/etc/vhe/vhe_mounter* has been run on each node, go to each node and log in using that selected login name (with the appropriate password). If VHE is correctly configured, the logins are successfully completed, and you are always placed in the execution environment associated with the selected login name.

Note

You have now completed configuring the VHE service. The following sections describe advanced usage or set-up problems you may encounter when using VHE.

If you are configuring VHE as part of the NFS Services configuration, return to the "7. Execute /etc/netnfsrc" section in the "NFS Configuration and Maintenance" chapter.

Configuration Refinements

The configuration procedure presented in the previous sections addresses most configuration cases. However, you may wish to refine your VHE configuration. This section explains how to refine your VHE configuration to allow NFS mounts to be done in the background.

NFS mounts in the Background

You can alter the */usr/etc/vhe/vhe_mounter* script to allow mounts to be done in the background. This eases the situation where all nodes are not ready to respond when a node tries to mount them. To mount nodes in the background, you need to edit the */usr/etc/vhe/vhe_mounter* script.

The *vhe_mounter* file has a shell variable called BACKGROUND_MOUNT whose initial value is set to 0. To allow nodes to be mounted in the background:

- Use an editor to set the value to something other than 0.
- Save the file and execute the */usr/etc/vhe/vhe_mounter* script.

These changes cause NFS mounts to occur in the background. If the mounts are not successful on the first try, the NFS mounts continue to execute in the background.

Note

Because each mount executes as a separate process until it completes or until the *retries* option for the NFS mount is exceeded, there may be a problem if there are many nodes (more than 30) in the VHE group.

VHE Maintenance

To keep VHE running correctly and efficiently, refer to the following sections.

Unmounting file systems

If needed, you can unmount all of the remotely mounted file systems. The easiest method of doing this is to execute the following:

```
umount -a -t nfs
```

This command can only be used when there are no VHE users logged on. If VHE is currently being used, the mount point directories will be busy and *umount* will not unmount a directory that is busy.

Just as having multiple file systems available for remote mounting required mounting to be done in a specific order, unmounting file systems must be done in the proper order. The order is just the reverse from the order that the mounts were done. The *umount* command with the "-a -t" options does this automatically.

For example:

```
mount A:/ /vhe/A
mount A:/usr /vhe/A/usr

umount /vhe/A/usr
umount /vhe/A
```

Adding or Deleting VHE Nodes

You may need to add or delete nodes from the VHE configuration. To do this, you need to perform the following steps.

1. Update the */etc/vhe_list* on the YP master server by either removing file systems that are no longer available (if a node is being deleted) or adding file systems that you want to become available (if a node is being added). Refer to the section in this chapter called "3. Create /etc/vhe_list" for more information about how to do this.

2. Edit the */etc/passwd* file to show the addition of mount points to the home directory pathname. Refer to the section in this chapter called "5. Update /etc/passwd" for more information on how to do this. If you are removing file systems, you need to edit this file to delete mount points from the home directory pathname.

3. To distribute the */etc/vhe_list* and */etc/passwd* files to the YP servers, execute the following command on the YP master server:

   ```
   /usr/etc/yp/ypmake
   ```

4. Then execute the following:

   ```
   /usr/etc/vhe/vhe_mounter
   ```

   The script uses the information found in */etc/vhe_list* to decide which new file systems to mount. The */usr/etc/vhe/vhe_mounter* script does not attempt to unmount a node deleted from the group. *vhe_mounter* needs to be executed on all of the nodes in the group for all of the nodes to be updated.

Advanced Usage

Adding altlogin and mounter Logins

The two logins of altlogin and mounter can be added to */etc/passwd* by the super-user. This allows the user to:

- log in using the mounter ID to complete NFS mounts to a node, if for some reason a node was not mounted when *vhe_mounter* was executed.
- log in using altlogin to access the node where they currently are. This is useful if their home node is down.

These logins are similar to *who* and *date* because they execute a program. Mounter executes *vhe_u_mnt*, and altlogin executes *vhe_altlog*.

- The *vhe_u_mnt* program executed by the *mounter* login only attempts to mount a file system of a node that is found in the */etc/vhe_list* file. This prevents users from performing mounts to arbitrary nodes. Users can only perform mounts that could have been done by */usr/etc/vhe/vhe_mounter*. If the node name entered at the prompt is not found in */etc/vhe_list*, then an error message is printed and the mounts are not completed.
- The *vhe_altlog* program executed by altlogin prompts for a login ID and then attempts to do a *su* using the provided login ID. The user is then prompted for a password by *su*. If the proper password is given, the user is logged in with the home directory of */tmp*. (If a proper password is not given, the user is not allowed access to the system.) Once logged in, none of the user's execution environment is available, but he or she can use the system.

To make these logins valid, you need to add them to the */etc/passwd* file. Do this by adding an entry for each login to the */etc/passwd* file. These entries should be similar to the following:

```
mounter::6:1::/:/usr/etc/vhe/vhe_u_mnt
altlogin::6:1::/tmp:/usr/etc/vhe/vhe_altlog
```

The values shown in the above lines in UID, GID and home directory can be replaced with other values. Also note no password is provided in the above lines, but passwords can be entered if desired. If passwords are entered, tell the users allowed to use those logins what the associated passwords are because they **must** provide them when logging in.

Mounter Example

In this example, *dave* attempts to log in from node B when his home node, node A, is not mounted on node B. The following sequence would occur:

```
login: dave
Password:
Unable to change directory to /vhe/A/users/dave

login: mounter
Password:
Enter the name of the node to mount:
A

login: dave
Password:
<Dave gets logged in>
```

Altlogin Example

This section shows an example of using altlogin. *Julia* is currently working at node B. Her home node A is not up, but Julia can gain access to node B in the following way:

Altlogin Example

This section shows an example of using altlogin. *Julia* is currently working at node B. Her home node A is not up, but Julia can gain access to node B in the following way:

```
login: altlogin
Enter your login name: Julia
Password:
%
```

Julia is now logged in at node B.

$HOME

If you are writing scripts that make reference to files in a home directory, those file names should be prefixed with $HOME (for Sh or Ksh). For Csh, file names should be prefixed with a ~ character.This allows a file to be accessed in a consistent manner even if the home directory pathname changes.

$ROOT

To make a distinction between system files (like the password file) for the local and the home nodes, the following can be added to the *.profile* or *.login* file (*home_node* should be replaced with the name of the node):

```
ROOT=/vhe/home_node
export ROOT
```

This allows easier access to system files on a user's home node. For example, instead of typing:

```
more /vhe/home_node/etc/passwd
```

The user types:

```
more $ROOT/etc/passwd
```

Alternate Mount Points

The mount examples in this chapter are prefixed with */vhe*. In addition to */vhe* mount points, there may be other file systems users in a VHE group want to regularly access.

For example, in a given VHE group, node A has file system */Design*. To have a consistent view of this file system among all users in the VHE group, the */Design* file system can be mounted on a pathname */Design*. To do this, the following line would be added to the *vhe_list* file:

```
A /Design /Design
```

Using VHE for Mail

To extend VHE to handle mail tasks:

- change your default mailbox pathname to have a mount point added to the beginning of it (just as the home directories are changed in */etc/passwd*).
- specify the above pathname as the file to be used by the mail handler of your choice. If that mail file is on a separate file system, it must also be mounted to be available.

For example, if user *fm*'s home node is A, this shows how the mailx program can be invoked to read mail over NFS:

```
mailx -f /vhe/A/usr/mail/fm
```

In this example, if */usr* was a separate file system on A, then the following would be added to */etc/vhe_list*:

```
A /usr /vhe/A/usr
```

10

Troubleshooting

If a node on the network is not operating correctly, use this chapter to identify and correct the problem. Most problems occur when

- installing the network
- changing the network (e.g., adding a node or extending the coaxial cable), or
- another system on the LAN fails.

Before troubleshooting the problem, get or create your network map as described in the *Installing and Maintaining NS-ARPA Services* documentation. Use this map when checking configuration and network layout information. Remember to update it any time you make a change to the network.

Note

All references to **servers** and **clients** apply to NFS servers and clients unless preceded by **YP**.

Key Terms

Client

- A node that requests data or services from other nodes (servers).
- A process that requests other processes to perform operations.

Note: An NFS client can also be configured as any combination of an NFS server, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Cluster One or more workstations linked together with a local area network (LAN), but consisting of only one file system.

Cnode Any node operating in an HP-UX cluster environment, including diskless nodes and the root server.

Daemon Background programs that are always running, waiting for a request to perform a task.

Diskless Cnode A node in an HP-UX cluster that uses networking capabilities to share file systems, but does not have a file system physically attached.

Export To make a file system available to remote nodes via NFS.

Hard Mount A mount that causes NFS to retry a remote file system request until it succeeds, you interrupt it (default option), or you reboot the system.

Host A node that has primary functions other than switching data for the network.

Map (YP) A file consisting of logical records; a search key and related value form each record. YP clients can request the value associated with any key within a map.

YP map is synonymous with **YP database.**

Master Server (YP) The node on which one or more YP maps are constructed from ASCII files. These maps are then copied to the YP slave servers for the YP clients to access.

Mount To obtain access to a remote or local file system or directory (import).

Mount Point The name of the directory on which a file system is mounted.

Netgroup A network-wide group of nodes and users defined in */etc/netgroup*.

Node A computer system that is attached to or is part of a computer network.

Root Server The only node in an HP-UX cluster that has file systems physically attached to it.

Server

- A node that provides data or services to other nodes (clients) on the network.
- A process that performs operations as requested by other processes.

Note: An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Slave Server (YP)

A node that copies YP maps from the YP master server and then provides YP clients access to these maps.

Soft Mount

An optional mount that causes access to remote file systems to abort requests after one NFS attempt.

Yellow Pages (YP)

An optional network service composed of databases (maps) and processes that provide YP clients access to the maps. The YP service enables you to administer these databases from one node.

YP may or may not be active; check with your system administrator.

YP Client

- A node that requests data or services from YP servers.
- A YP process that requests other YP processes to perform operations.

Note: A YP client can also be configured as any combination of a YP server, NFS client, or NFS server. (A YP server **must** also be configured as a YP client.)

YP Database See "Map (YP)."

YP Domain A logical grouping of YP maps (databases) stored in one location. YP domains are specific to the YP network service and are not associated with other network domains.

YP Map See "Map (YP)."

YP Password The password for a user's login ID that exists in the YP *passwd* map. The YP password is the same one as the user password, but is administered through the YP.

You do not have to have a password to access the YP databases.

YP Server

- A node that provides data (maps) or services to other nodes (YP clients) on the network using YP.
- A YP process that performs operations as requested by other YP processes.

Note: A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

Troubleshooting References

Troubleshooting the NFS Services primarily concerns the areas: power up and connectivity, NFS Services, YP Services and VHE. This chapter only addresses NFS, YP and VHE problems. Link diagnostics and troubleshooting are in the *Installing and Maintaining NS-ARPA Services* documentation.

If your system is having problems communicating with or through a non-HP system, refer also to the appropriate user and system administration documentation for that system.

Power Up and Connectivity Testing

Refer to the following documentation if your system cannot communicate with other systems on the network.

- *HP 98643A LAN/300 Link LANIC Installation Manual*
- *HP Repeater Installation Manual* (**only** if you are using a HP 92223A repeater)
- *HP-UX Installation Manual*
- *HP-UX Reference* manuals
- *HP-UX Series 300 System Administrator Manual*
- *LAN Cable and Accessories Installation Manual*
- *Installing and Maintaining NS-ARPA Services*

Troubleshooting Sections

Refer to the "Troubleshoot NFS" section or the *NFS Services Reference Pages* if you cannot mount a remote file system, access a remotely mounted file system, or experience other problems with the NFS service.

Refer to the "Troubleshoot Yellow Pages" section or the *NFS Services Reference Pages* if you configured the system to use YP, but cannot access files serviced by it.

Refer to the "Troubleshoot VHE" section if you configured the system to use VHE, but it doesn't function as described in the "VHE Configuration and Maintenace" chapter.

Guidelines

Troubleshooting is an elimination process that narrows a problem. If a process worked before but does not work now, first consider what has changed. For example, have you moved hardware or modified configuration files?

Start with the minimum number of variables, then gradually and selectively add other variables such as the following.

- If you cannot communicate with one system, try another one. If the second system works, the problem may be with the first remote system and not your system.
- If one system cannot communicate with yours, try another one. If neither system can communicate with yours but they can communicate with each other, the problem may be with your system.
- If one service does not work, try another one. The problem may be with a particular service to a particular system and not a problem with the system itself.

Common Network Problems

Network problems generally occur under the following circumstances.

- File permissions on the client or server restrict the operation.
- Network services on the client or server are misconfigured or malfunctioning.
- Network LAN software or hardware is misconfigured or malfunctioning.

Initial Troubleshooting

You should first check the following situations to ensure they are not the cause. If they are not, refer to the flowcharts in this chapter.

Configuration

1. Is your host running HP-UX 6.0 or later? Execute *uname -a* or *uname -r* to check the HP-UX version number.
2. Does your system have the recommended 256K additional memory for networking software?
3. Is your Series 300 a supported configuration? If you are unsure, contact your HP support representative.
4. Does the error occur on a node other than a Series 300? If so, refer to the appropriate system documentation.

Hardware

The *Installing and Maintaining NS-ARPA Services* documentation contains details about troubleshooting hardware problems.

1. Are all connections along the network cabling tight?
2. Is each cable segment less than 500 meters for ThickLAN and less than 100 meters for ThinLAN?
3. Are there no more than two repeaters between you and the node with which you want to communicate?

4. Are you mixing Ethernet[1] hardware with IEEE 802.3[2] hardware? This is **not** an acceptable combination since they do not have the same electrical characteristics.

5. Is there a 50 ohm terminator at the end of each cable?

6. Is the MAU tapped correctly into the cable?

7. Is the cable grounded in only one place?

8. Is the AUI solidly connected to the interface card?

9. Is the host hardware working correctly?

Network Communication

1. Is the remote node HP certified? If you are unsure, contact your HP support representative.

2. Can any other two nodes on the network communicate? If not, the problem may be global. Refer to the *LAN Cable and Accessories Installation Manual* and the *Installing and Maintaining NS-ARPA Services* documentation.

3. Have you performed the corrective action supplied with the error message you received? Consult the appropriate entry in the network reference pages.

---

(1) Ethernet is a local area network system developed by Digital Equipment Corporation, Intel Corporation, and Xerox Corporation.

(2) IEEE 802.3 is a networking standard that is accepted by the Institute of Electrical and Electronic Engineers.

4. If using gateways, do both hosts have routing information to each other? Refer to the *ARPA/Berkeley Services Reference Pages, route(1M)* section.

5. **If operating in an HP-UX cluster environment** and trying to mount an NFS file system, ensure you are using the root server's host name as the node specified in the *mount(1M)* command. Note, you can troubleshoot NFS specific problems from the root server.

6. **If operating in an HP-UX cluster environment** and having Link problems, cnodes will not be able to boot. Since Link diagnostics reside on the root disk, first test the Link from the root server. (Refer to *Installing and Maintaining NS-ARPA* documentation.)

NFS and Yellow Pages

1. Is the client system trying to perform tasks as super-user on the remote system? Executing *setuid* root programs cannot access files or directories unless the permission *other* allows it.

2. Was network communication established between the client and server using the procedures outlined in the *HP-UX Series 300 System Administrator Manual* and in the "NFS Configuration and Maintenance" and "YP Configuration and Maintenance" chapters?

3. Is the problem associated with remote file locking? The *lockf(2)* call fails when attempting to lock a remote file. Prior to HP-UX release 6.5, NFS Services did not support file locking on remote file systems.

4. Is the problem associated with attempts to access remote device files? Prior to release 6.5, HP-UX did **not** support remote access to device files.

5. Does the *inetd* security file (*/usr/adm/inetd.sec*) on the remote system limit access to the remote system for the RPC service you are trying to access?

6. Is the file system listed in the server's */etc/exports*?

7. Does */etc/exports* restrict file system access to a **specific** netgroup or host?

   a. The */etc/netgroup* file must list the netgroup if it is specified in */etc/exports*.

   b. The */etc/hosts* file must contain the host if it is specified either in */etc/exports* or in */etc/netgroup*.

8. Is the file system or directory mounted? To check, execute the *mount(1M)* command.

9. If the file system is suppose to be automatically mounted, is it listed in */etc/checklist*?

10. If programs accessing remote files hang, is the NFS or YP server down?

11. Is data on remote nodes corrupted? Ensure only one system is writing to the file at a time; NFS allows more than one client to write to a file simultaneously.

Error Messages

The problem can exist on the server even though the error message may not occur on it.

Since most of the error messages are self-explanatory, you can determine the necessary corrective action when simple errors occur. For the other error messages, follow the corrective action supplied in the *NFS Services Reference Pages* for that service. (These error messages are preceded by the name of the service.)

Errnos

NFS provides two *errno* values: *ESTALE* and *EREMOTE*.

ESTALE You cannot reference the file because it no longer exists. This situation can occur since NFS allows a file opened by a client to be removed by a user on another node.

EREMOTE You cannot mount file systems from a server that the server has remotely mounted (i.e., you cannot use NFS servers as NFS gateways).

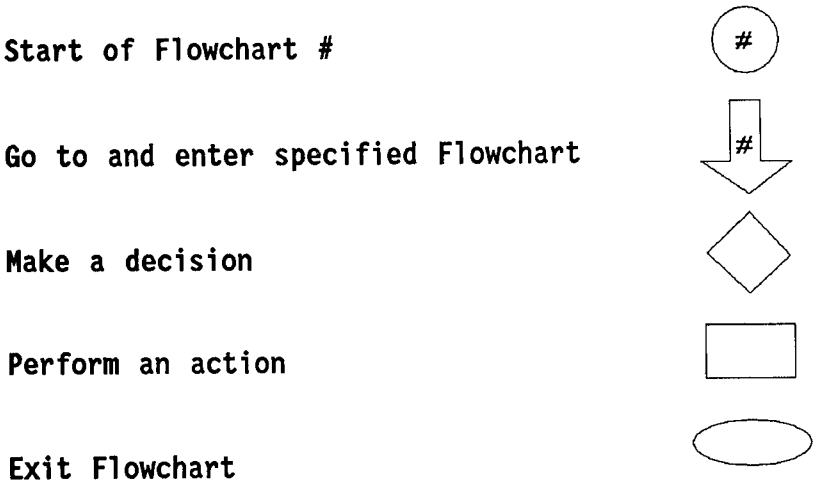
Unsolved Problems

If you do not solve the problem after working through the previous troubleshooting steps and following flowcharts, call your HP support representative for assistance. Provide as much information about the problem as possible, including information from your network map and the following items.

- The activity you were attempting when the error occurred. Describe the HP-UX commands, job streams, result codes, and events leading to and including the problem.
- The version or update information for all software you are running. You should be able to find this information on your *Install* or *Update* media.
- The error messages you received. Record all error messages and numbers that appeared both on all nodes.
- The troubleshooting steps you tried.
- The problems you ruled out and why.

Flowchart Format

Each of the following flowcharts have a corresponding set of labelled explanations. You can use the flowcharts alone or with the explanatory text for more detail.

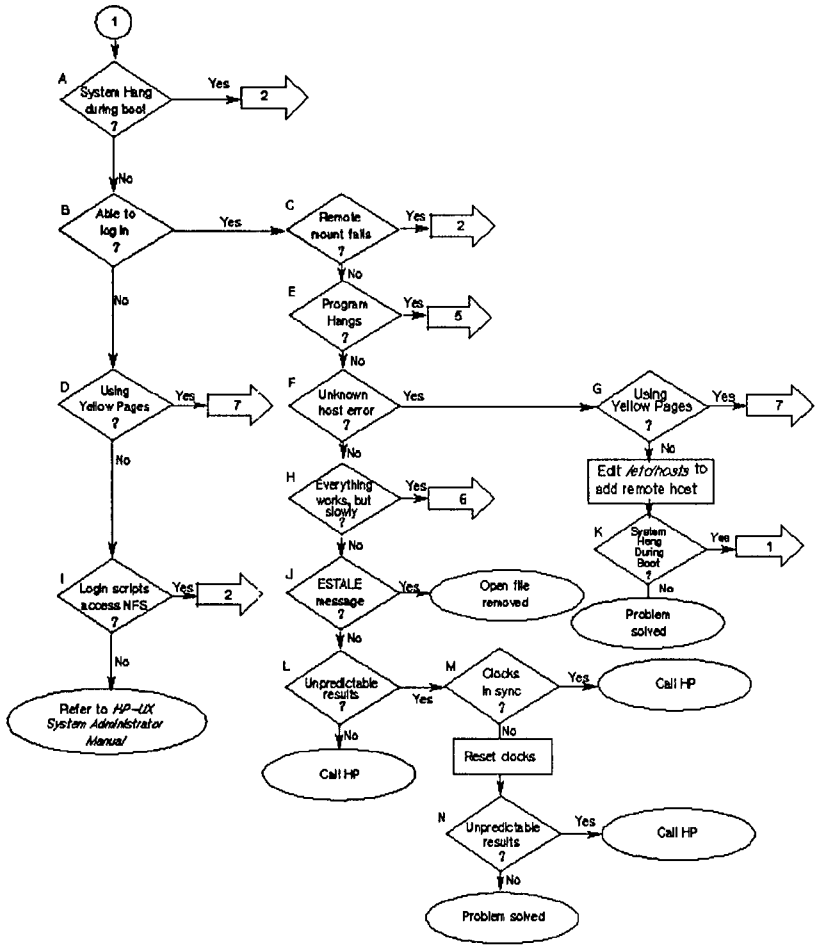


Flowchart Symbols

Note

These flowcharts are for HP systems. Processes referenced in the flowcharts may not be part of NFS products from other vendors (e.g., *portmap(1M)*).

Troubleshoot NFS



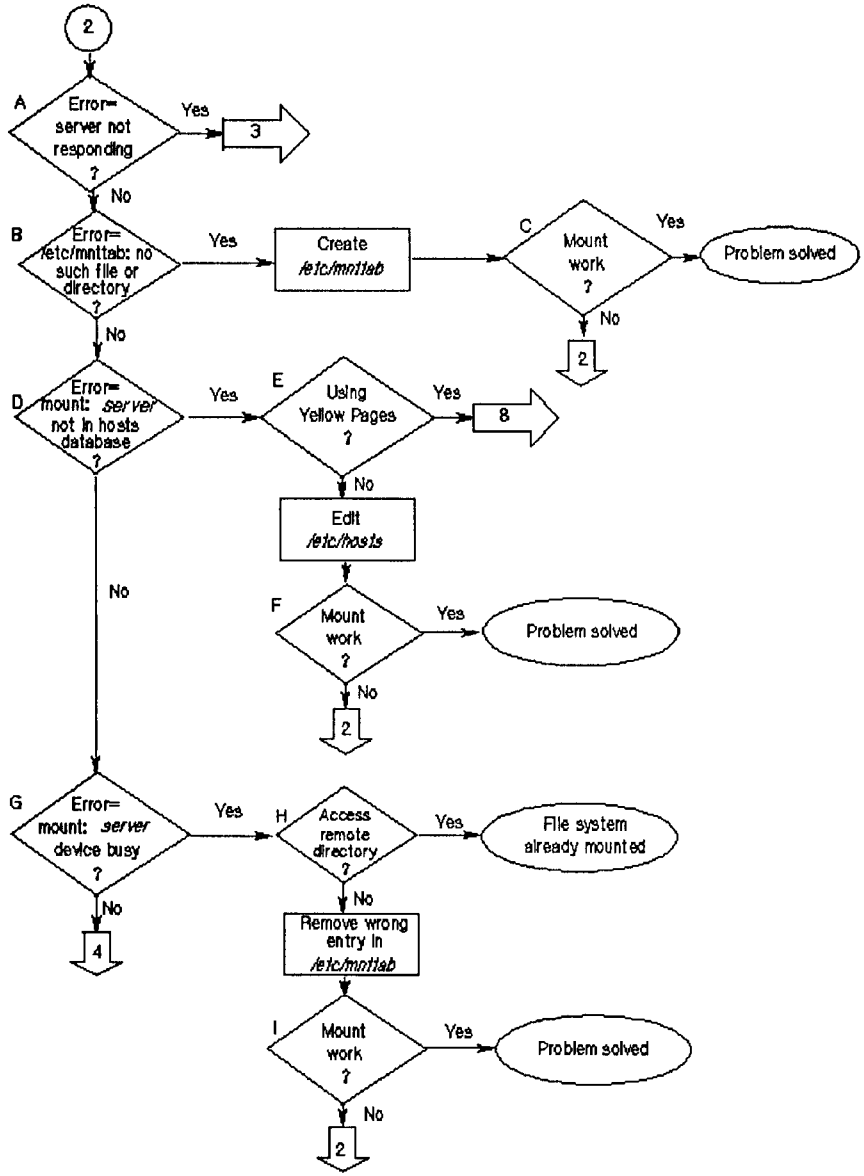
Flowchart 1: Initial Steps to Narrowing the Problem

Initial Steps to Narrowing the Problem (Flowchart 1)

Begin your troubleshooting efforts with Flowchart 1 since it helps you determine the best troubleshooting path based on the problem's symptoms.

Question	Yes: Action	No: Action
A. Does the system hang during boot when mounting remote files? Systems hanging during boot where remote mounts generally occur may indicate one or more servers are down or the network connection to one or more servers is faulty.	See Flowchart 2.	See **B.**
B. Are you able to login?	See **C.**	You will receive error messages or the system will fail to respond if you cannot log in to it. See **D.**
C. When trying to mount a remote file system, do error messages indicate the attempt failed?	See Flowchart 2.	See **E.**
D. Are you using YP?	See Flowchart 7.	See **I.**
E. Do programs performing remote file accesses hang?	See Flowchart 5.	See **F.**

F. Does the system report unknown host errors during execution of commands or programs?	See **G.**	See **H.**
G. Are you using YP?	See Flowchart 7.	Edit */etc/hosts* to add remote host, and then see **K.**
H. Does everything work, but slowly?	See Flowchart 6.	See **J.**
I. Do your login scripts perform NFS remote file accesses?	See Flowchart 2.	The problem is probably unassociated with the network services. Refer to the system login information in the *HP-UX Series 300 System Administrator Manual.*
M Does the following message occur? ESTALE	The file was removed by another user. NFS allows file removal at any time.	See **L.**
K. Does the system hang during boot?	Restart Flowchart 1.	Problem solved.
L. Are you receiving unpredictable results when executing programs or commands?	See **M.**	Call your HP support representative.
M. Are the server and client clocks synchronized?	Call your HP support representative.	Reset the clocks using the *date(1)* command, and then see **N.**
N. Do you receive unpredictable results to commands or programs?	Call your HP support representative.	Problem solved.



Flowchart 2: Mount Fails

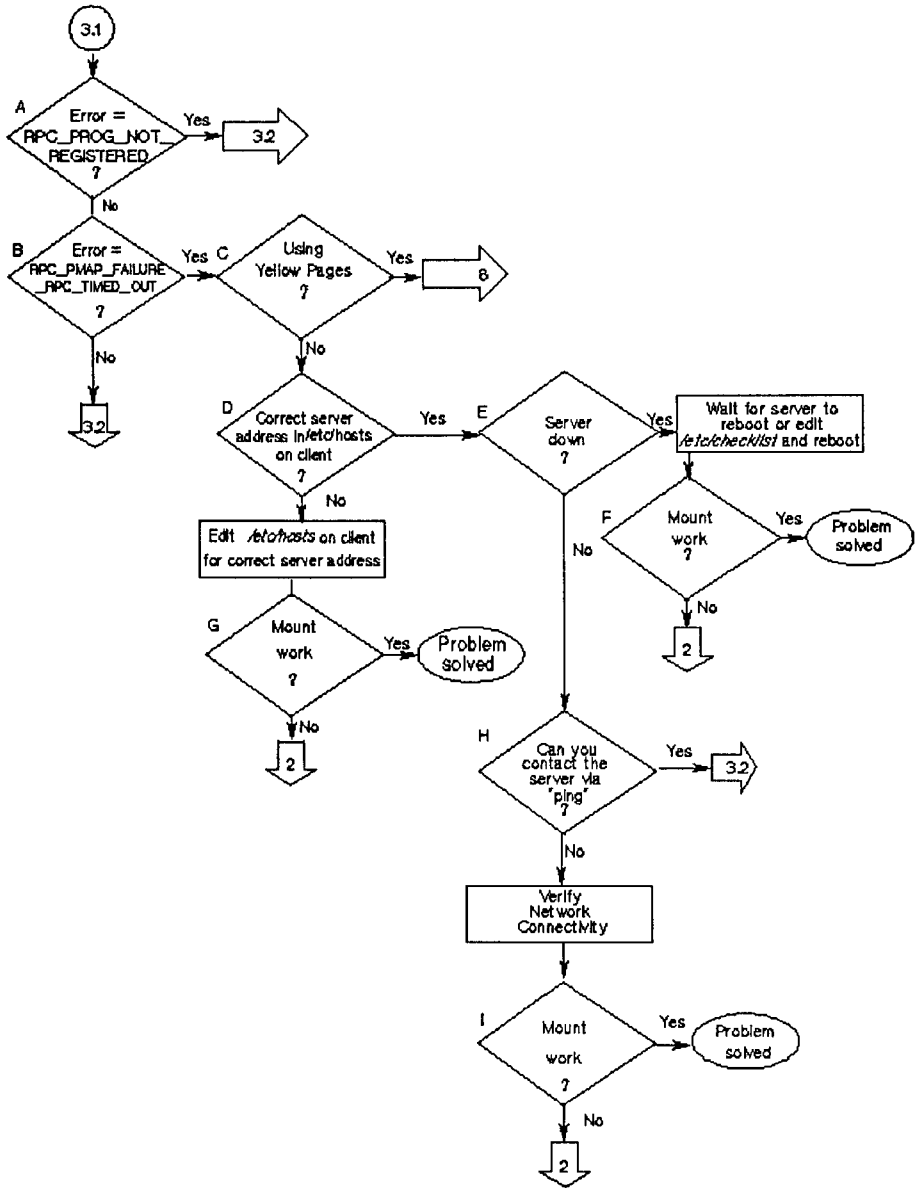
Mount Fails (Flowchart 2)

Use Flowchart 2 if your system hangs during the booting process when remote file systems are mounted or if your remote mount attempts are unsuccessful.

Before using Flowchart 2, remember to check the *mount(1M)* command syntax and correct errors according to the error messages.

Question	Yes: Action	No: Action
A. Does the following error message occur on the client? `server not responding`	See Flowchart 3.	See **B.**
B. Does the following error message occur on the client? `/etc/mnttab: no such file or directory`	Create */etc/mnttab* on the client, and then see **C.** The system uses */etc/mnttab* to log all mounted file systems. **Note:** Generally, at boot time */etc/rc* creates */etc/mnttab*.	See **D.**
C. Can you mount the remote system?	Problem solved.	Restart Flowchart 2.
D. Does the following error message occur on the client? `mount: server not in hosts database`	See **E.**	See **G.**
E. Are you using YP?	See Flowchart 8.	Edit */etc/hosts* on the client to include the desired remote host, and then see **F.**

F. Can you mount the remote system?	Problem solved.	Restart Flowchart 2.
G. Does the following error message occur on the client? `mount: server device busy`	See **H.**	See Flowchart 4.
H. Can you access a remote directory in the desired remote file system?	You do not need to mount the file system since it is already mounted; problem solved.	On the client, remove the incorrect entry in */etc/mnttab* for the remote file system you are trying to mount, and then see **I.**
I. Can you mount the remote system?	Problem solved.	Restart Flowchart 2.



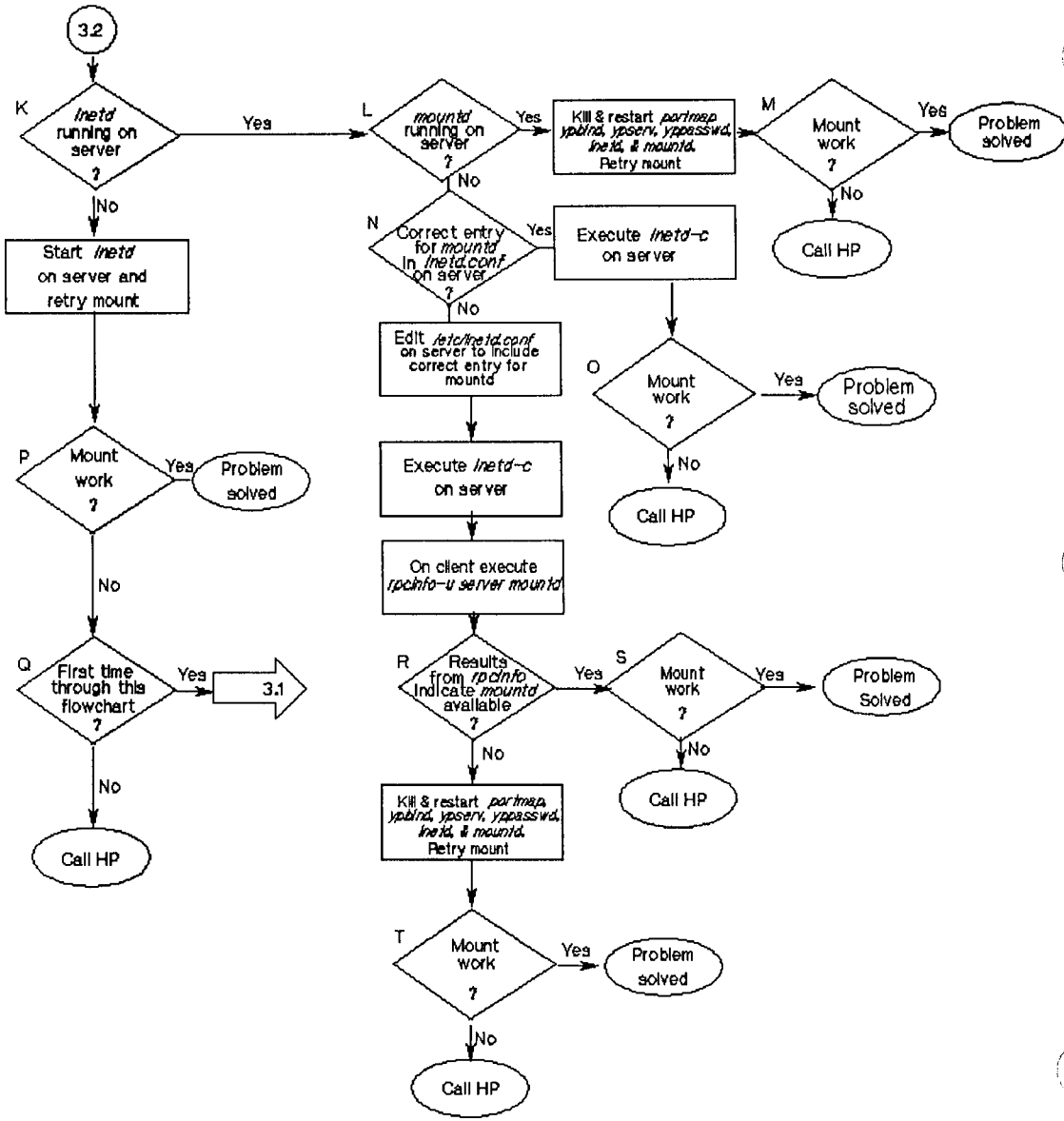
Flowchart 3.1: Server Not Responding

Server Not Responding (Flowchart 3.1)

This flowchart and corresponding instructions consist of two parts: Flowchart 3.1 and 3.2.

Question	Yes: Action	No: Action
A. Does the following error message occur? `RPC_PROG_NOT_REGISTERED`	See Flowchart 3.2.	See **B.**
B. Does the following error message occur? `RPC_PMAP_FAILURE: RPC_TIMED_OUT`	See **C.**	See Flowchart 3.2.
C. Are you using YP?	See Flowchart 8.	See **D.**
D. Is the server's address correct in the client's */etc/hosts*?	See **E.**	Edit the client's */etc/hosts* to include the correct address for the server you are trying to mount. See **G.**
E. Is the server you are trying to mount down? To check, ask your system administrator or try other network services to that system.	You have two options: • Do nothing on the system until the server reboots. • Edit the client's */etc/checklist* to remove the NFS entry for that server; reboot the system. See F.	See **H.**
F. Can you mount the remote system?	Problem solved.	See Flowchart 2.
G. Can you mount the remote system?	Problem solved.	See Flowchart 2.

H. Can you contact the server using the ping diagnostic? Refer to the *Installing and Maintaining NS-ARPA Services* manual for ping diagnostic information	See Flowchart 3.2.	Refer to the *Installing and Maintaining NS-ARPA Services* manual to verify Link connectivity, and then see **I.**
I. Can you mount the remote system?	Problem solved.	See Flowchart 2.

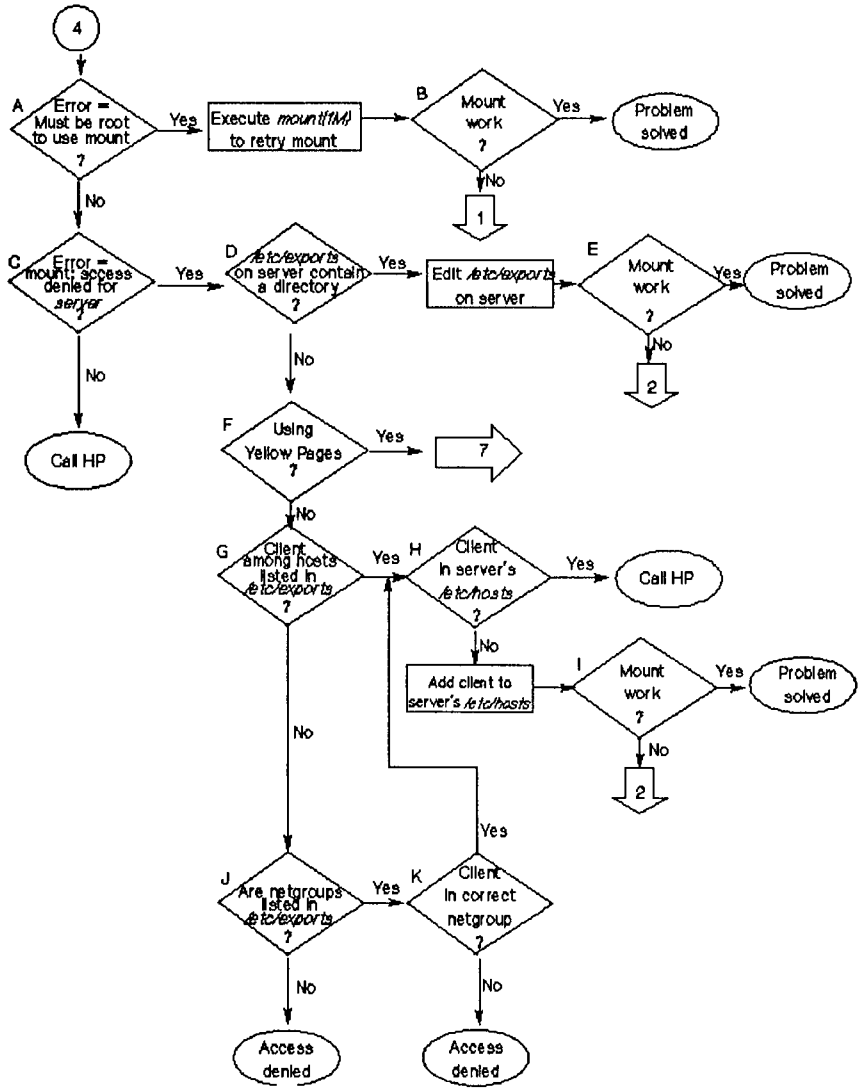


Flowchart 3.2: Server Not Responding

Server Not Responding (Flowchart 3.2)

Question	Yes: Action	No: Action
K. Is *inetd(1M)* running on the server?	See **L.**	Start */etc/inetd* on the server, retry the mount, and then see **P.**
L. Is *mountd(1M)* running on the server?	Kill and restart the following daemons on the server in the order specified. • *portmap(1M)* • *ypbind(1M)* * • *ypserv(1M)* * • *yppasswdd(1M)* * • *inetd(1M)* • *mountd(1M)* * **only** if using YP Retry the mount, and then see **M.**	See **N.**
M. Can you mount the remote system?	Problem solved.	Call your HP support representative.
N. Is the correct *mountd(1M)* entry in *inet d.conf* on the server? Ensure the entry is not commented out with a # (pound sign).	Execute *inetd -c* on the server, and then see **O.**	1. Edit the server's */etc/inetd.conf* file to include the correct *mountd(1M)* entry. 2. Execute *inetd -c* on the server to read changes in */etc/inetd.conf.* 3. Execute *rpcinfo -u* on the client. `rpcinfo -u` *server* `mountd` 4. See **R.**
O. Can you mount the remote system?	Problem solved.	Call your HP support representative.
P. Can you mount the remote system?	Problem solved.	See **Q.**

Q. Is this the first time you used this flowchart for this problem?	Restart Flowchart 3.1.	Call your HP support representative.
R. Do the results from *rpcinfo -u* indicate a *mount d (1M)* process is available on the server?	See **S.**	Kill and restart the following daemons on the server in the order specified. • *portmap(1M)* • *ypbind(1M)* * • *ypserv(1M)* * • *yppasswdd(1M)* * • *inetd(1M)* • *mountd(1M)* * **only** if using YP Retry the mount, and then see **T.**
S. Can you mount the remote system?	Problem solved.	Call your HP support representative.
T. Can you mount the remote system?	Problem solved.	Call your HP support representative.

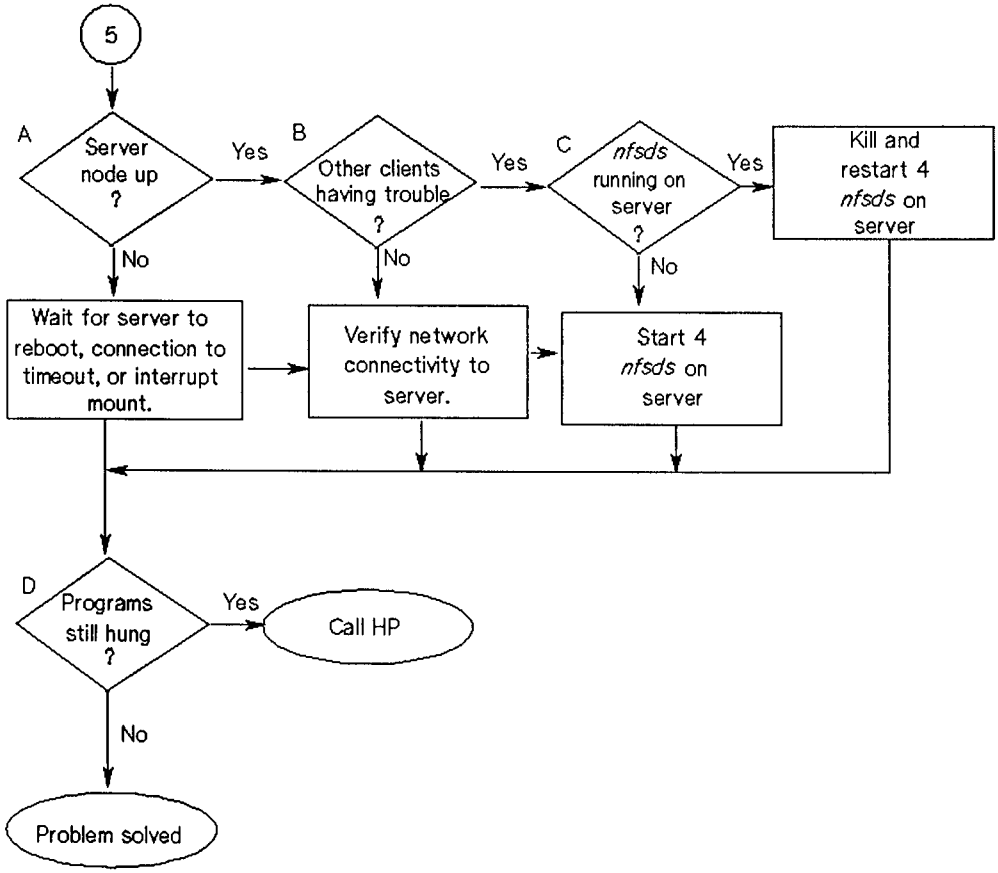


Flowchart 4: Restricted Access

Restricted Access (Flowchart 4)

Question	Yes: Action	No: Action
A. Does the following error message occur on the client? `Must be root to use mount`	Login as super-user, execute *mount(1M)*, and then see **B.**	See **C.**
B. Can you mount the remote system?	Problem solved.	See Flowchart 1.
C. Does the following error message occur on the client? `mount: access denied for` *server*	See **D.**	Call your HP support representative.
D. Does the server's */etc/exports* file list a directory rather than a file system?	Edit the server's */etc/exports* to contain the file system rather than a directory, and then see **E.**	See **F.**
E. Can you mount the remote system?	Problem solved.	See Flowchart 2.
F. Are you using YP?	See Flowchart 7.	See **G.**
G. If hosts are listed in */etc/exports,* is the client among the hosts listed for the desired file system?	See **H.**	See **J.**
H. Is the client listed in the server's */etc/hosts?*	Call your HP support representative.	Add client to server's */etc/hosts,* and then see **I.**
I. Can you mount the remote system?	Problem solved.	See Flowchart 2.

J. Are netgroups listed for this file system in server's */etc/exports*?	See **K.**	Access for this client is deliberately denied.
K. Is the client listed in the appropriate net-group for this file system in */etc/netgroup*?	See **H.**	Access for this client is deliberately denied.

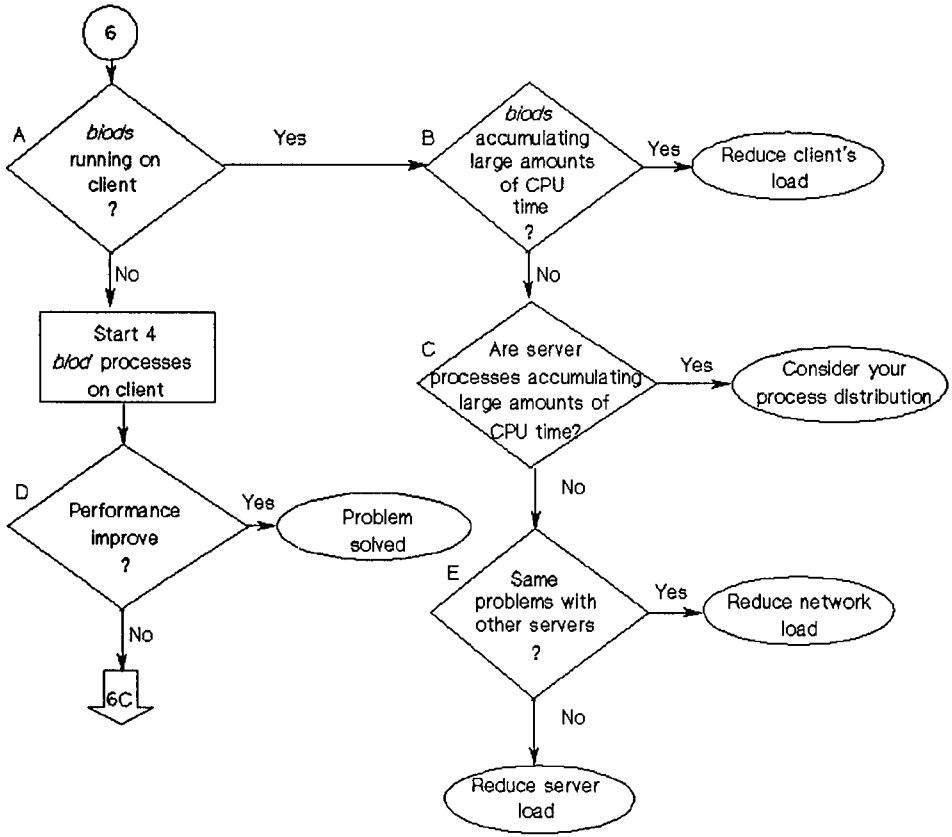


Flowchart 5: Programs Hang

Programs Hang (Flowchart 5)

Programs are most likely to hang if network communication is lost to the server, if the server is down, or if daemons are hung.

Question	Yes: Action	No: Action
A. Is the server node running?	See **B.**	For hard mounts, either • wait for the server to reboot or • interrupt the mount. For soft mounts, wait for the mount to time out. See **D.**
B. Are other client nodes having trouble?	See **C.**	Verify the network connectivity. Refer to the *Installing and Maintaining NS-ARPA Services* manual. See **D.**
C. Are *nfsd(1M)* daemons running on the server?	Kill and restart four *nfsd(1M)* daemons on the server, and then see **D.**	Start four *nfsd(1M)* daemons on the server, and then see **D.**
D. Do the programs hang?	Call your HP support representative.	Problem solved.

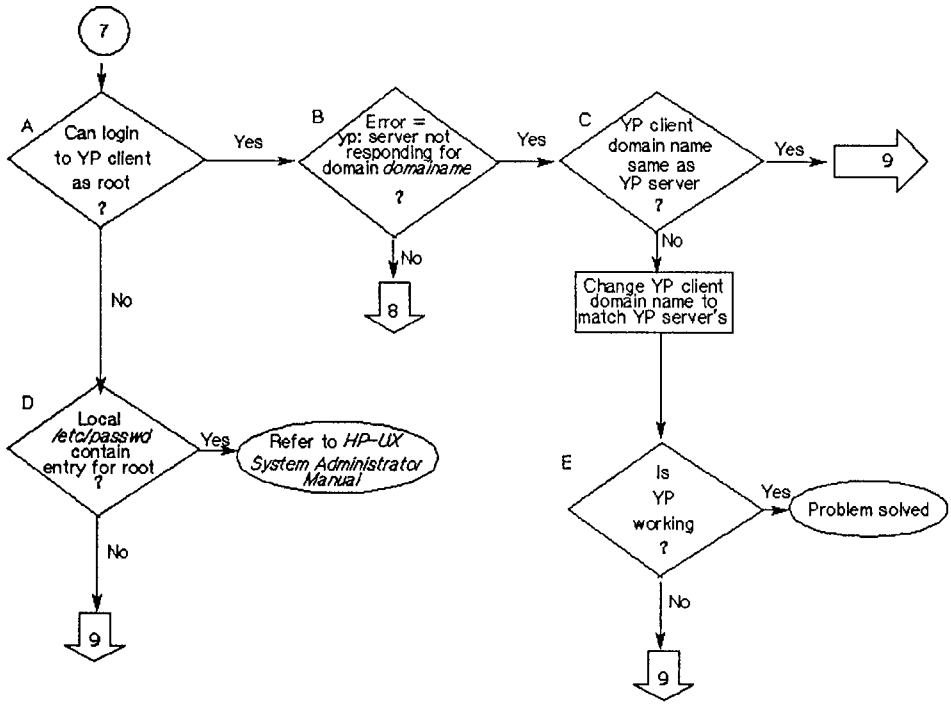


Flowchart 6: Performance Problems

Performance Problems (Flowchart 6)

Question	Yes: Action	No: Action
A. Are the *biod(1M)* daemons running on the client?	See **B.**	Start four *biod(1M)* processes on the client, and then see **D.**
B. Are the client biod(1M) daemons accumulating large amounts of CPU time? 1. List the client processes using *ps*. 2. Copy a large file to the server system, and list the client *biod* processes again. 3. Compare the CPU time for the biod(1M) processes before and after the file copy.	Reduce the client's load to fewer NFS transactions by reducing the number of users or storing more files locally.	See **C.**
C. Are processes on the server accumulating large amounts of CPU time (especially *nfsd(1M)*, *inetd(1M)*, and *portmap(1M)*?	Consider whether you need to distribute your processing by adding additional systems.	See **E.**
D. Has performance improved?	Problem solved.	See **C.**
E. Are the same performance problems evident with other servers?	Reduce the network load.	Reduce the server's load by adding more servers.

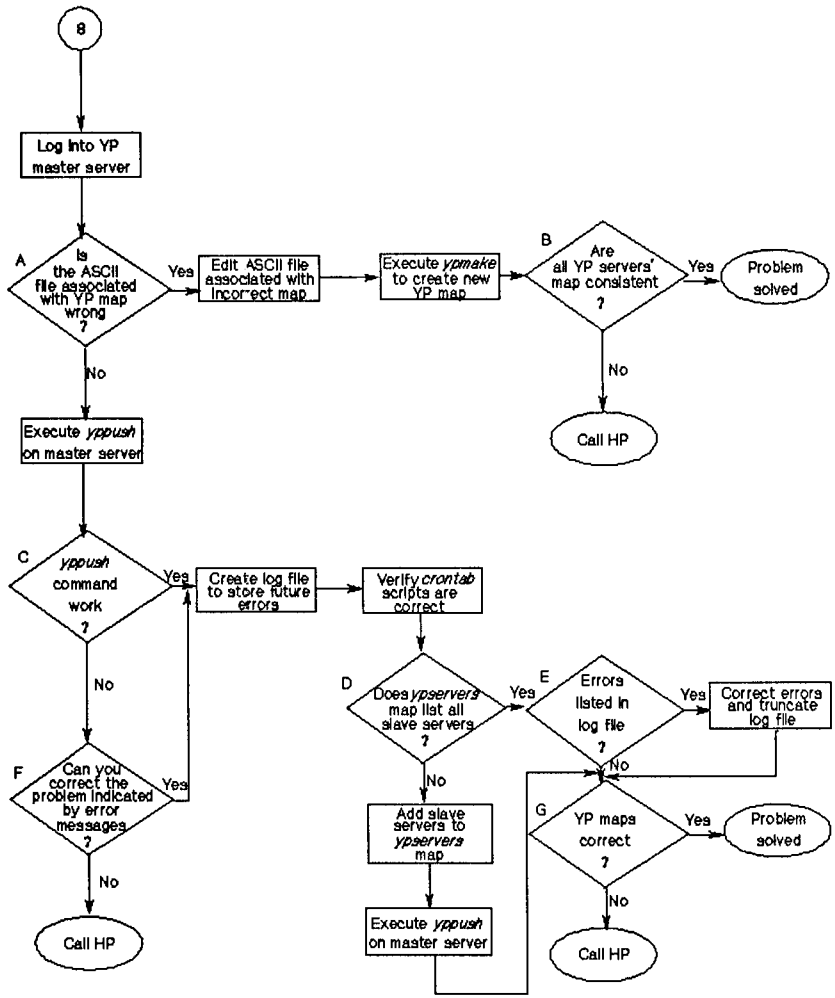
Troubleshoot Yellow Pages



Flowchart 7: Initial Steps to Troubleshooting YP

Initial Steps to Troubleshooting YP (Flowchart 7)

Question	Yes: Action	No: Action
A. Can you login as root on the YP client?	See **B.**	See **D.**
B. Does the following error message occur on the console or in the *ypbind* log file? `yp: server not responding for domain` *domain_name*	See **C.**	See Flowchart 8.
C. Is the YP client's YP domain name the same as the YP server's?	See Flowchart 9.	Change the YP client's YP domain name to be the same as the YP server's, and then see **E.** `domainname` *domain_name*
D. Does the local */etc/passwd* file contain an entry for root?	The problem is not associated with YP or NFS. Refer to the *HP-UX Series 300 System Administrator Manual.*	You cannot login to the YP client until YP is functioning unless you have an entry for a user in the local */etc/passwd* file. See Flowchart 9.
E. Is YP working? If you can access the YP server's maps using **ypcat** or *ypmatch,* YP is probably functioning correctly.	Problem solved.	See Flowchart 9.



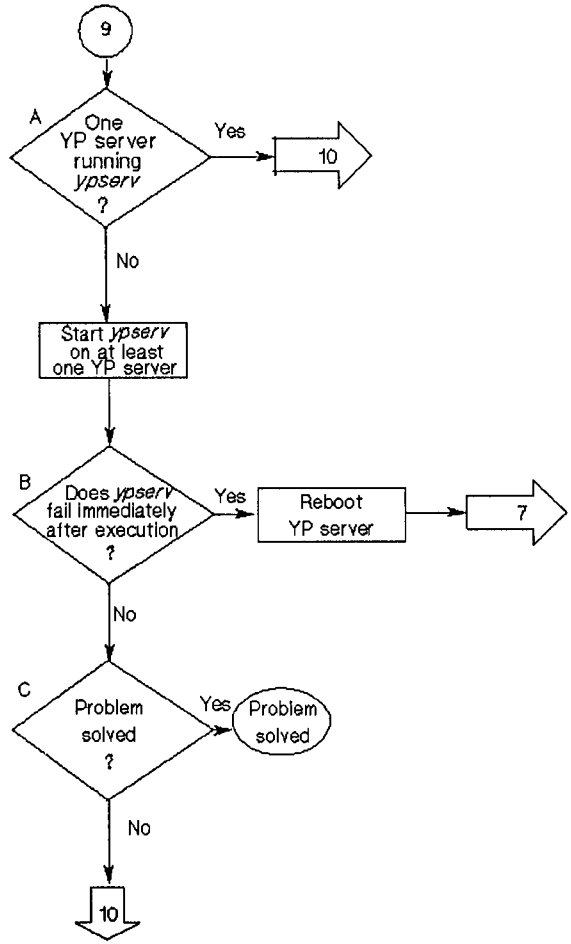
Flowchart 8: Incorrect YP Maps

Incorrect YP Maps (Flowchart 8)

Login to the YP master server as root before starting Flowchart 8.

Question	Yes: Action	No: Action
A. On the YP master server, does the ASCII file associated with the YP map need to be updated (e.g., update */etc/hosts*)?	1. Edit the ASCII file associated with the incorrect YP map. 2. Execute *ypmake(1M)* to create and distribute a new map to the YP slave servers. 3. See **B.**	Execute *yppush(1M)* on the YP master server, and then see **C.** yppush *map_name*
B. Are all YP server's maps consistent? You can determine this by executing *yppoll* and then comparing order numbers.	Problem solved.	Call your HP support representative.
C. Does *yppush(1M)* work correctly? If you do not receive error messages associated with the command, it probably executed successfully.	1. Create the log file */usr/etc/yp/ypxfr.log* to trap future errors associated with *yppush(1M)* on each YP slave server. 2. Verify that *crontab(1M)* scripts (on each slave server) copying the maps are correct. 3. See **D.**	See **F.**

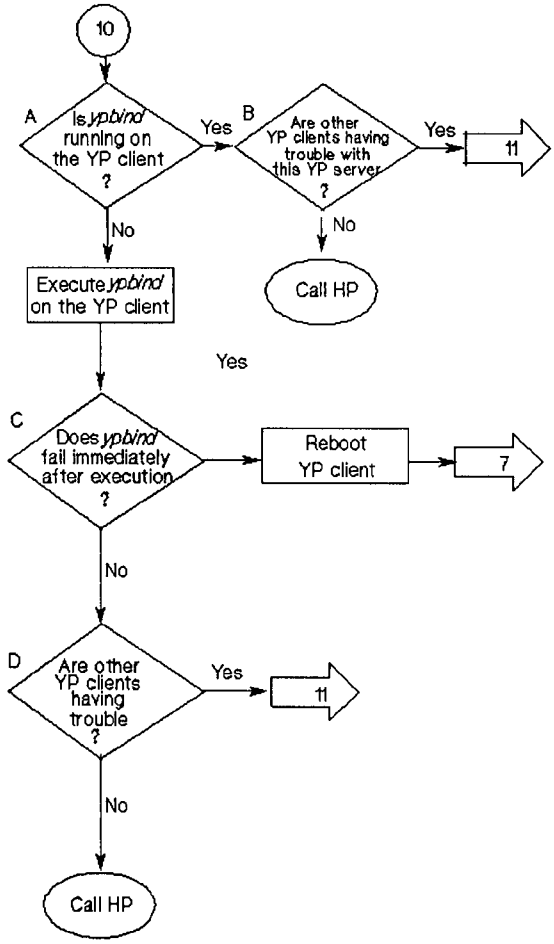
D. Does the *ypservers* map list all YP slave servers? `ypcat -k ypservers`	See **E.**	1. Add any missing YP slave server to the *ypservers* map. 2. Execute *yppush(1M)* on the YP master server to update all YP slave servers. 3. See **G.**
E. Does */usr/etc/yp/ypxfr.log* on the slave server list errors?	Correct the errors, truncate the log file, and then see **G.**	See **G.**
F. Can you correct the problem indicated by the error message?	1. Create the log file */usr/etc/yp/ypxfr.log* to trap future errors associated with *yppush(1M)* on each YP slave server. 2. Verify that *crontab(1M)* scripts (on each slave server) distributing the maps are correct. 3. See **D.**	Call your HP support representative.
G. Are the YP maps correct?	Problem solved.	Call your HP support representative.



Flowchart 9: ypserv(1M) Problems

ypserv(1M) Problems (Flowchart 9)

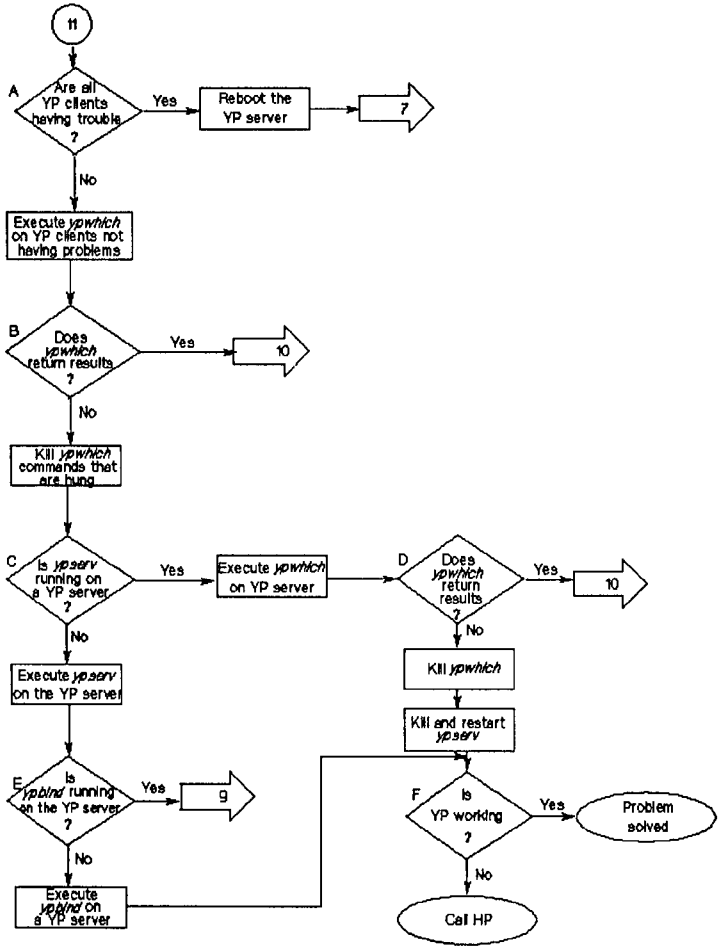
Question	Yes: Action	No: Action
A. Is at least one YP server in the YP domain running *ypserv(1M)*?	See Flowchart 10.	Start *ypserv(1M)* on at least one YP server in the YP domain, and then see **B.**
B. Does *ypserv(1M)* fail immediately after starting it?	Reboot the YP server, and then see flowchart 7.	See **C.**
C. Is the problem solved?	Problem solved.	See Flowchart 10.



Flowchart 10: ypbind(1M) Problems

ypbind(1M) Problems (Flowchart 10)

Question	Yes: Action	No: Action
A. Is *ypbind(1M)* running on the YP client?	See **B.**	Execute *ypbind(1M)* on the YP client, and then see **C.**
B. Are other YP clients having trouble with this YP server?	See Flowchart 11.	Call your HP support representative.
C. Does *ypbind(1M)* crash immediately after starting it?	Reboot the YP client, and then see Flowchart 7.	See **D.**
D. Are other YP clients having trouble with this YP server?	See Flowchart 11.	Call your HP support representative.

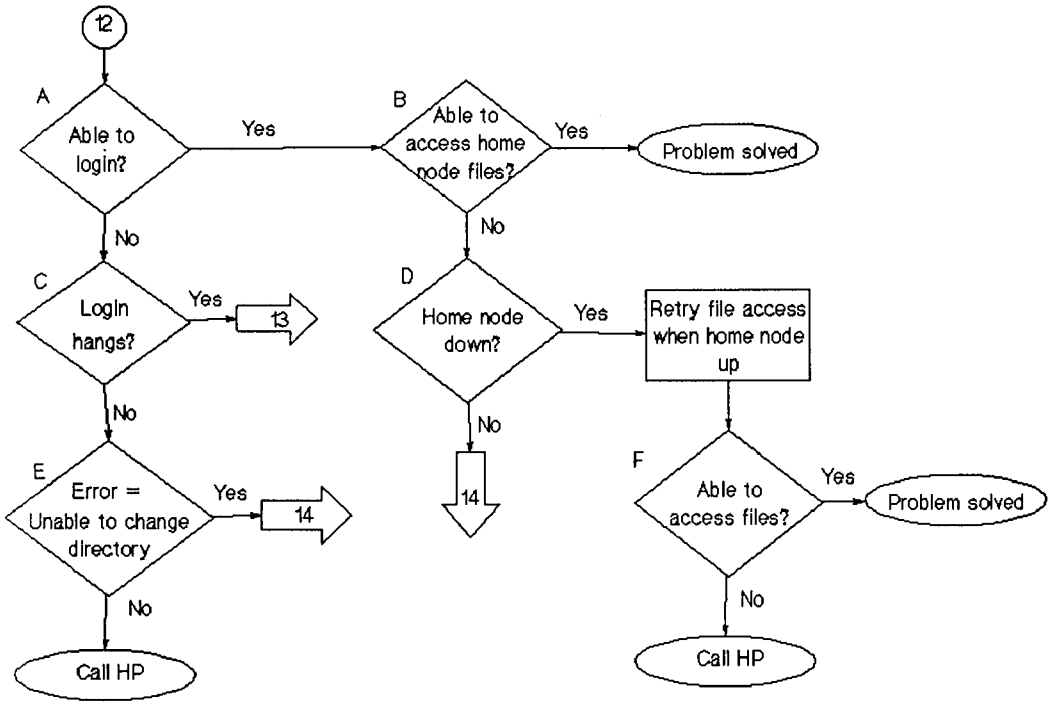


Flowchart 11: Multiple YP Client Problems

Multiple YP Client Problems (Flowchart 11)

Question	Yes: Action	No: Action
A. Are all YP clients having trouble with this YP server?	Reboot the YP server, and then see Flowchart 7.	Execute *ypwhich(1)* on the YP client nodes not having problems, and then see **B.**
B. Does the *ypwhich(1)* command return results on the YP client?	See Flowchart 10.	Kill *ypwhich(1)* commands that are hung on YP clients, and then see **C.**
C. Is *ypserv(1M)* running on the YP server?	Execute *ypwhich(1)* on the YP server, and then see **D.**	Execute *ypserv(1M)* on the YP server, and then see **E.**
D. Does *ypwhich(1)* return results on the YP server?	See Flowchart 10.	1. Kill *ypwhich(1)* on the YP server. 2. Kill and restart *ypserv(1M)*. 3. See **F.**
E. Is *ypbind(1M)* running on the YP server?	See Flowchart 9.	Execute *ypbind(1M)* on the YP server, and then see **F.**
F. Is YP functioning correctly on all YP clients?	Problem solved.	Call your HP support representative.

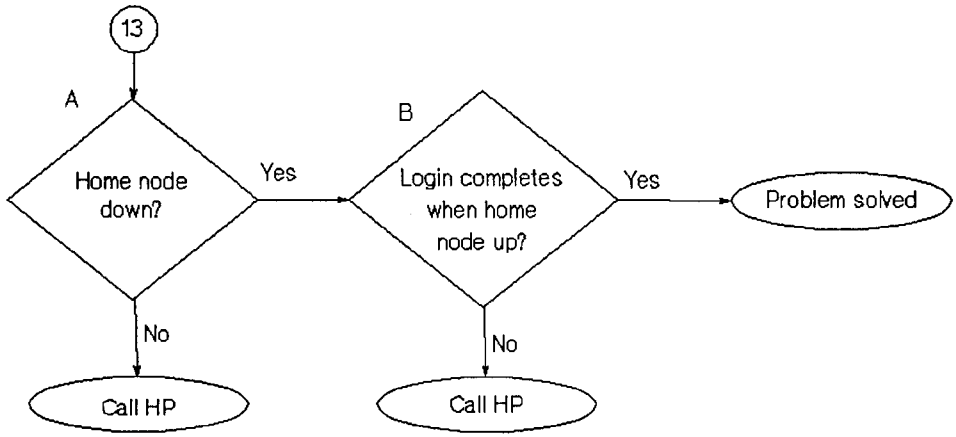
Troubleshoot VHE



Flowchart 12: Initial Steps to Troubleshooting VHE

Initial Steps to Troubleshooting VHE (Flowchart 12)

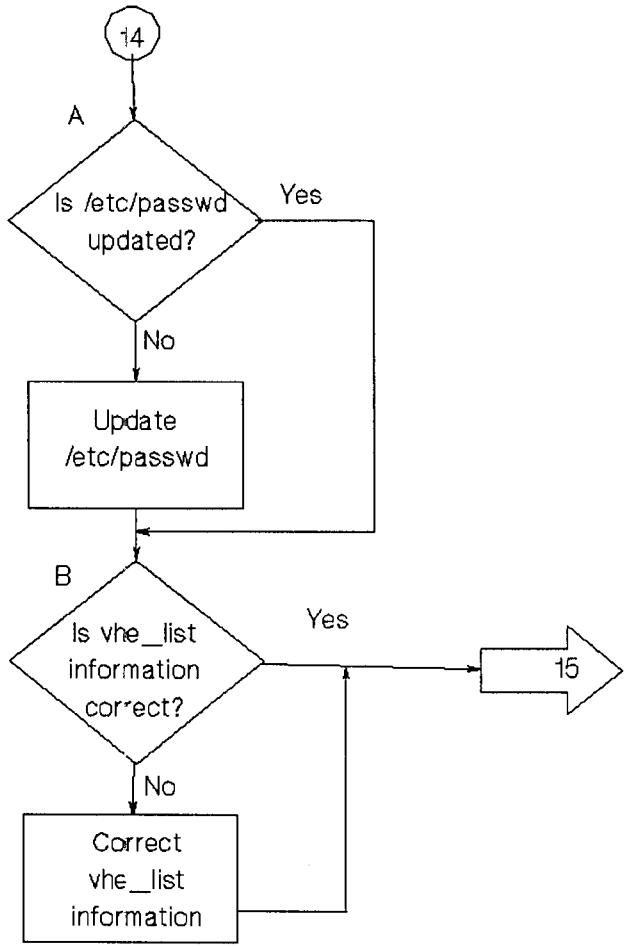
Question	Yes: Action	No: Action
A. Are you able to log in?	See **B.**	See **C.**
B. Are you able to access files on the home node?	No problem.	See **D.**
C. Does the machine hang during login?	See Flowchart 13.	See **E.**
D. Is the home node down?	Retry accessing files when the home node is up; then see **F.**	See Flowchart 14.
E. Do you receive the following error message?		
`Unable to change directory to home directory`	See Flowchart 14.	Call your HP support representative.
F. Are you able to access files on the home node?	Problem solved.	Call your HP support representative.



Flowchart 13: Home Node Goes Down After Mount Done

Home Node Goes Down After Mount Done (Flowchart 13)

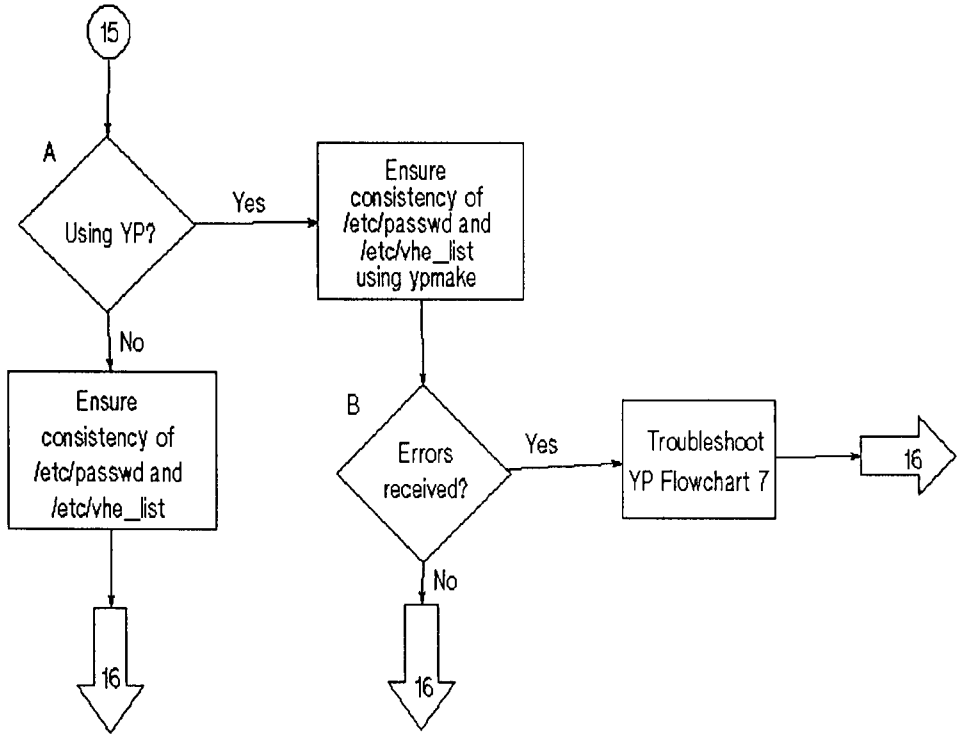
Question	Yes: Action	No: Action
A. Is the home node down?	Try logging in again once the home node comes up; then see **B.**	Call your HP support representative.
B. Does the login complete once the home node comes up?	Problem solved.	Call your HP support representative.



Flowchart 14: Checking /etc/passwd and /etc/vhe_list Files

Checking /etc/passwd and /etc/vhe_list Files (Flowchart 14)

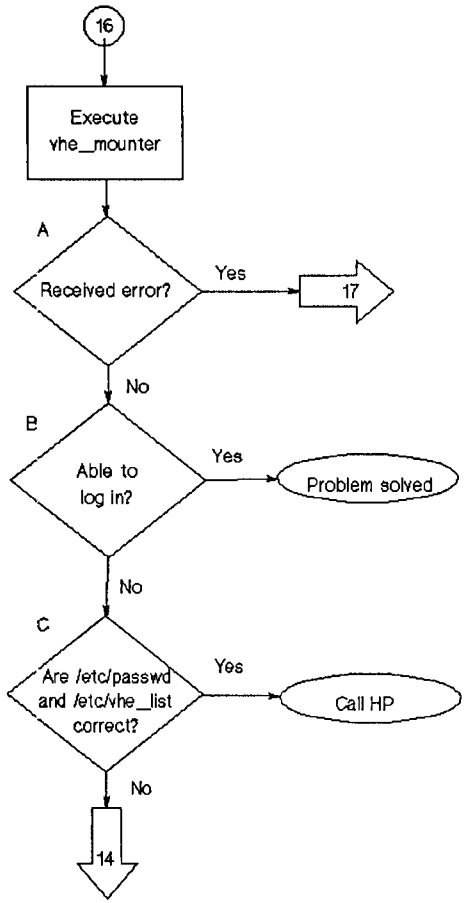
Question	Yes: Action	No: Action
A. Is the */etc/passwd* file updated to prefix the home directory with the NFS mount points?	See **B.**	Update the */etc/passwd* file as described in the "VHE Configuration and Maintenance" chapter; go to **B.**
B. Is the information in the */etc/vhe_list* file correct?	See Flowchart 15.	Correct the */etc/vhe_list* file information; see Flowchart 15.



Flowchart 15: Consistency of /etc/passwd and /etc/vhe_list

Consistency of /etc/passwd and /etc/vhe_list (Flowchart 15)

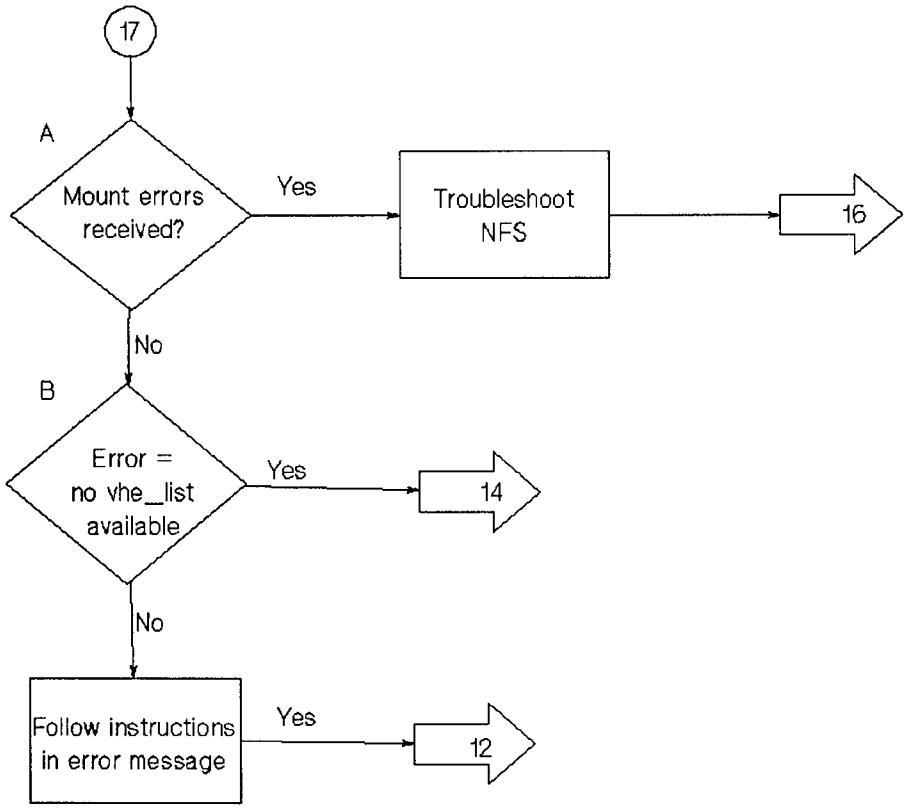
Question	Yes: Action	No: Action
A. Are you using Yellow Pages (YP) to ensure consistency of */etc/passwd* and */etc/vhe_list* information?	Ensure consistency of the */etc/passwd* and */etc/vhe_list* files on all nodes in the VHE group by executing the following command: `/usr/etc/yp/ypmake passwd vhe_list` See **B.**	Ensure consistency of the */etc/passwd* and */etc/vhe_list* files on all nodes in the VHE group
B. Did you receive any errors when executing *ypmake (1M)*?	Go to the YP Flowchart 7 and complete troubleshooting steps; then return to VHE Flowchart 16.	Go to Flowchart 16.



Flowchart 16: Execution of vhe_mounter

Execution of vhe_mounter (Flowchart 16)

Question	Yes: Action	No: Action
A. Did you receive any errors while executing *vhe_mounter (1M)*?	See Flowchart 17.	See **B.**
B. Are you able to log in?	Problem Solved.	See **C.**
C. Is the information for the home node entered into the */etc/passwd* and */etc/vhe_list* files?	Call your HP support representative.	See Flowchart 14.



Flowchart 17: Error Message from vhe_mounter

Error Message from vhe_mounter (Flowchart 17)

Question	Yes: Action	No: Action
A. Were any mount errors encountered (mount errors begin with `mount:`)?	Troubleshoot NFS (Flowchart 1); then see Flowchart 16.	See **B.**
B. Does the following error message occur? `no vhe_list available`	See Flowchart 14.	If an error message other than those mentioned is printed, follow the instructions in that error message; then re-enter Flowchart 12 to see if problem is solved.

A

HP NFS Services vs. Local HP-UX

If you have applications running on HP-UX, they may behave differently over NFS Services. Use this appendix to understand the basic differences between NFS Services and local HP-UX operations.

HP NFS Services Networking Operation	Local HP-UX Operation
Append Mode	**Append Mode**
If two processes operating on different clients open the same file using *O_APPEND*, the *write* operation may not append data to the file.	If two processes open the same file using *O_APPEND*, the *write* operation should append information to the file.
chacl(1)	**chacl(1)**
You can only use the *-F* option. The other options of *chacl(1)* are not supported over NFS.	You can use all options locally.
Device Files	**Device Files**
NFS does not support remote access to device files, but does support local access to device files via NFS.	HP-UX supports local access to device files.

HP NFS Services Networking Operation	Local HP-UX Operation
File Locking	**File Locking**
NFS supports remote file locking for NFS reads and writes in advisory mode only.	HP-UX supports local file locking in advisory and enforcement modes.
getacl(2) system call	**getacl(2) system call**
Is not supported over NFS.	Is supported locally.
Group Membership	**Group Membership**
A user may be a member of eight groups. If a user who is a member of more than eight groups attempts to access a file, the system accesses only the first eight groups for permission checking.	A user may be a member of up to 20 groups.
Long File Names	**Long File Names**
NFS can access file names up to 255 characters in length **if** the remote NFS file system supports them.	The local HP-UX file system both short and long file name file systems. On short file name systems, HP-UX silently truncates file names that are longer than 14 characters in length.
Note: Since the local file system truncates long file names to 14 characters, conflicting file names might occur when accessing a file system that supports long file names.	

HP NFS Services Networking Operation	Local HP-UX Operation
mknod(1M) Command	**mknod(1M) Command**
The *mknod* command will work only with named pipes over NFS.	You can use the *mknod* command locally for all file types.
Named Pipes	**Named Pipes**
NFS named pipes cannot be used to communicate between machines in the same discless cluster.	Named pipes can be used to communicate among clients in a discless cluster.
Reading Directories	**Reading Directories**
You cannot use the *read(2)* call to read a remote directory, rather you should use *readdir(3)*.	You can use the *read(2)* call to read a local directory. However, to do so can restrict migration of programs to future HP-UX versions.
RFA	**RFA**
Using a network special file on an NFS file system is not supported.	Is supported locally.
setacl(2) system call	**setacl(2) system call**
Is not supported over NFS.	Is supported locally.
setaclentry(3) library routine	**setaclentry(3) library routine**
Is not supported over NFS.	Is supported locally.

HP NFS Services Networking Operation	Local HP-UX Operation
Super-user Permission	**Super-user Permission**
The super-user UID *0* is mapped to *-2* by default.	Super-user has permission to perform any operation locally (by definition).
Anything requiring super-user permission may not work over NFS. For example, a super-user may not be able to perform the following tasks. • Link and unlink directories • Alter directories such as /, */etc*, and */bin* • Use *chmod* to set sticky or setuid bits • *mknod* of device files	

HP NFS Services Networking Operation	Local HP-UX Operation
System Time	**System Time**
Commands that access clocks on different systems may not provide consistent times since system clocks differ For example, if you give the *utime(2)* command a *NULL* pointer for the *times* value, the following process occurs. 1. The system sets the access time and modification time according to the client node clock. 2. It then sends these times over to the server which changes the inode to reflect the new access and modification times. 3. The server node identifies the change in the inode and thus, modifies the inode's status change time according to its own clock. The result is a high probability of differing times between the server's access and modification times versus its status change time.	Commands that access clocks on the local system provide consistent times.

HP NFS Services Networking Operation	Local HP-UX Operation
System Time	**System Time**
Note: If operating in an HP-UX cluster environment, all nodes in the cluster have the same time as the root server's clock. Therefore, clock skew problems exist only if the root server's clock is different from other NFS servers.	
Unlinking	**Unlinking**
The server does not keep state information and does not know if a process has a file open.	If you open a local file and unlink it before you close the file, the file descriptor for the open file will still be valid to access the file.
• The server will unlink a file if it receives a request to do so; thus, subsequent requests for the file will result in an error. • If a process opens a file and then unlinks it, the client renames the file so it appears to be gone. When the process quits, the client then unlinks the renamed file. • If the unlink request comes from a different node than from where the open request came from, the file is deleted.	

HP NFS Services Networking Operation	Local HP-UX Operation
yppasswd(1) Command	**passwd(1) Command**
This command does not have a **password aging** feature.	This command has a **password aging** feature.
The super-user must know the current password to change another user's password.	Super-user does not have to know the password to change another user's password.
• A password must contain at least five characters if it includes a combination of either: uppercase and lowercase letters, or alphanumeric characters.	• Each password must have six or more characters: at least two alpha characters and at least one numeric or special character.
• A password must contain at least four characters if it includes a combination of uppercase letters, lowercase letters, and numeric characters.	• Each password must differ from the user's login name and any reverse or circular shift of that name.
• A password must contain at least six characters if it includes only monocase letters.	• New passwords must differ from the old by at least three characters.

HP NFS Services Networking Operation	Local HP-UX Operation
pathconf/fpathconf	**pathconf/fpathconf**
The following variables for the *pathconf/fpathconf* system calls are not supported over NFS:	All variables are supported locally for the *pathconf/fpathconf* system calls:
_PC_CHOWN_RESTRICTED variable	**_PC_CHOWN_RESTRICTED variable**
_PC_LINK_MAX variable	**_PC_LINK_MAX variable**
_PC_NAME_MAX variable	**_PC_NAME_MAX variable**
_PC_NO_TRUNC variable	**_PC_NO_TRUNC variable**
_PC_PATH_MAX variable	**_PC_PATH_MAX variable**
The following variables for the pathconf/fpathconf system calls return local information over NFS.	
_PC_MAX_CANON variable	**_PC_MAX_CANON variable**
_PC_MAX_INPUT variable	**_PC_MAX_INPUT variable**
_PC_VDISABLE variable	**_PC_VDISABLE variable**
The following variable for the *pathconf/fpathconf* systems calls is supported over NFS:	
_PC_PIPE_BUF variable	**_PC_PIPE_BUF variable**

B

Migrating from RFA to NFS

When using networks consisting of all HP systems, the Remote File Access (RFA) service provides distributed file access among Series 300 and 800 computers.

Use this appendix if you wish to translate your RFA applications to NFS applications.

Why Migrate to NFS Services?

Using NFS Services has several advantages.

- NFS works with other vendors' equipment and other operating systems.
- NFS is a defacto industry standard.
- NFS allows transparent file access.
- NFS with YP provides a centrally administered database.

Similarities

HP NFS and RFA have the following similarities.

- No remote device access
- No remote command execution
- Not all UNIX[1] semantics are fully supported

Differences

Refer to the following table for a list of differences between HP NFS and RFA.

NFS Services	RFA
You can run *setuid* programs accessing data on remote file systems.	You cannot run *setuid* programs accessing data on remote file systems.
NFS operates in a heterogeneous operating system environment.	RFA operates on HP-UX operating systems only.
Only the super-user can perform remote NFS mounts.	All users can establish access to remote file systems.
You can centrally administer your database using YP service.	You have no centrally administered database.

(1) UNIX (R) is a U.S. registered trademark of AT&T in the U.S.A. and other countries.

NFS Services	RFA
All users with read access to the mount point can read the remote file system.	Only users performing *netunam* can access the remote file systems.
Read and write file caching occurs on the clients; read caching occurs on the servers.	Read and write file caching occurs on the servers; caching does not occur on the clients.
The servers are stateless (do not remember client activities) and therefore, can be rebooted without interfering with client activities. (The client can resume access to the server when it is rebooted.)	The servers have state and therefore, remember the activities in which the client is involved.
One mount gives you access to only one file system.	One *netunam* gives you access to all file systems under the root directory.

Changing Scripts from RFA to NFS

Changing RFA scripts to NFS requires only minor changes. You can change both shell scripts that accept different path names and those that use hard-coded path names.

Shell Scripts that Accept Different Paths

Shell scripts that accept different paths require only minor modifications.

- You must perform a remote mount of a file system or directory either
  - as part of the script or
  - before executing the script.

  Since super-user must execute mounts, the script must be *setuid root* if the mount is performed as part of the script. If the script's owner does not have super-user permissions, the super-user can configure */etc/checklist* to automatically mount the remote file systems at boot time. This process allows users to execute scripts without checking to see if the remote file system is accessible.

- If RFA is **not** being used, remove any calls to *netunam* from the script. Removing these calls prevents *netunam* failures from causing the scripts to fail.

Shell Scripts with Hard-coded Paths

You can handle shell scripts with hard-coded path names in two ways.

- Change the path name in the script to correspond to the NFS mount point.
- Create a path name for the NFS mount point which corresponds to the path name in the script.

To mount the remote file system either as part of the script or automatically via */etc/checklist*, you must modify the shell scripts as described above in "Shell Scripts that Accept Different Paths."

Change Pathnames

Change the path name in the script to correspond to the NFS mount point.

EXAMPLE: The script has a hard-coded path name of */net/systemB/project*, and you want to mount the remote directory */project* on */user/project* as follows.

```
mount systemB:/project /user/project
```

Now change the script to use the path name */user/project* in place of */net/systemB/project*.

Create New Pathnames

Create a path name for the NFS mount point that corresponds to the path name in the script.

EXAMPLE: The script has a hard-coded path name of */net/systemB/project* which accesses the remote directory */project*.

1. Remove the network special file */net/systemB*.

2. Create the directories */net/systemB* and */net/systemB/project*.

   mount systemB:/project /net/systemB/project

The path name, therefore, remains the same.

Note

For RFA, access to the remote system is via a network special file. Creating an NFS mount point with the same name as the network special file for the remote system could cause confusion. Problems will not occur if

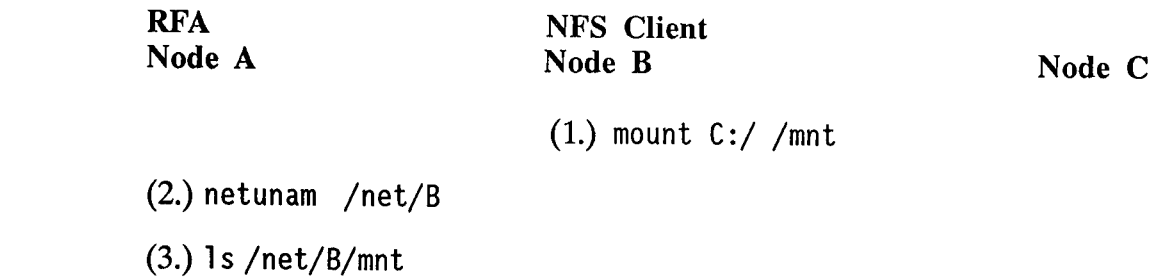
- the system does not use RFA and if
- you remove the network special file.

All remote access will then be via mount points that have the same names as the network special files that were removed.

RFA through NFS

RFA functions are operational through NFS. For example, an RFA node can access a remote directory by going through an NFS client. Note, however, **NFS functions cannot operate through RFA** because NFS parses the path names differently than RFA. Therefore, an NFS node cannot access a remote node by going through RFA.

EXAMPLE: RFA remote mount through an NFS client



1. Node B performs an NFS mount to Node C.
2. Node A then performs a netunam to Node B.
3. Node A lists (ls) the contents of Node C's root directory.

C

NFS in an HP-UX Cluster Environment

Reference this appendix for interactions between NFS Services and HP-UX cluster environments using diskless capabilities.

HP-UX Cluster Terms

CDF Context Dependent File: a hidden directory which contains all the versions of a file needed by the different cnodes.

Cluster One or more workstations linked together with a local area network (LAN), but consisting of only one file system.

Cnode Any node operating in an HP-UX cluster environment, including diskless nodes and the root server.

Diskless Cnode A node in an HP-UX cluster that uses networking capabilities to share file systems, but does not have a file system physically attached.

Root Server The only node in an HP-UX cluster that has file systems physically attached to it.

NFS Configuration and Maintenance

Configure

If you configure NFS on the root server, you must also configure NFS on all clients in the cluster. If the root server does not have NFS configured, then none of the clients can use NFS.

Daemons

- The *nfsd(1M)* daemon should be running on the root server if it is servicing NFS requests. Any *nfsd(1M)* daemons running on client cnodes are ignored.
- The *mountd(1M)* daemon should be running on the root server if servicing NFS requests. Any *mountd(1M)* daemons running on client cnodes are ignored.
- The *biod(1M)* daemon must be running on all cnodes in the cluster.

Mount Unmount

- If a cnode mounts a remote file system, all cnodes in the cluster can access the remote file system.
- If using NFS to mount a file system attached to a cluster, you must use the root server host's name as the node name specified in the *mount(1M)* command.
- If a cnode mounts a remote file system, any cnode in that cluster can unmount the remote file system.
- If a cnode unmounts a file system, all cnodes in the cluster will have that file system unmounted.
- Clients should **not** execute *umount -a.*

Context Dependent Files (CDF)

When accessing a *CDF* via an NFS mount, the *CDF* member is chosen based on the context of the NFS server, not the accessing node. Since this access method may return unexpected results, HP recommends you do not mix *CDF*s with NFS.

Clock Skew

All nodes in the HP-UX cluster have the same time as the root server's clock. Therefore, clock skew problems exists only if the root server's clock is different from other NFS servers.

YP Configuration and Maintenance

HP recommends that you execute *ypserv(1M)* only on the root server

- for better performance and
- to ensure the root server is the only YP server for that cluster.

Troubleshooting

You can troubleshoot NFS specific problems from the root sever.

- If trying to mount an NFS file system, ensure you are using the root server's host name as the node specified in the *mount(1M)* command.
- If problems exist in the Link, cnodes will not be able to boot. Since Link diagnostics reside on the root disc, first test the Link from the root server. (Refer to the *Installing and Maintaining NS-ARPA Services* manual.)

D

Password Security

This appendix explains the restrictions and limitations on the use of encrypted passwords and the secure password file with Yellow Pages. If you wish to review the normal use of passwords with Yellow Pages, see the "YP Configuration and Maintenance" section in this manual. If you require additional information on the secure password file, see *passwd(4)* in the *HP-UX Reference* manual.

HP-UX S300 now supports a secure password file (*/.secure/etc/passwd*) used to hide your encrypted passwords from non-privileged users. Therefore, it is probable that if you use the secure password file, your */etc/passwd* file will probably contain (in the password field) a character that is not part of the set of characters used in an encrypted password (.e.g. *). The YP database will not contain encrypted passwords if you use this */etc/passwd* file to build your Yellow Pages (YP) password database. This prevents non-privileged users from reading your passwords, as anyone with access to YP commands such as *ypcat* or YP library routines such as *yp_first* and *yp_next* can read the YP database.

If you are using the secure password file only to use the auditing subsystem and you do not need to hide your encrypted passwords, you can maintain an */etc/passwd* file that contains encrypted passwords that match those in your secure password file. You can then use this */etc/passwd* file to build your Yellow Pages (YP) database.

Note

A password in the */.secure/etc/passwd* file takes precedence over the password stored in Yellow Pages (YP).

If you wish to hide the encrypted passwords in your HP systems and wish to continue to use the YP password database to maintain other information kept on the password file, you can do the following:

- Build your YP password database on the HP YP master server using a password file that does not contain encrypted passwords (e.g. uses "*" in the password field).
- On an HP YP client, maintain a copy of the secure password file so the passwords in that file will be used at login.

and/or

- On an HP or non-HP Yellow Pages client, maintain the encrypted password in the /etc/passwd file through a YP escape.

EXAMPLE:

```
+username:encrypted_passwd:::::
```

Glossary

Alias

A term for referencing alternate networks, hosts, and protocols names.

ARPA

Advanced Research Projects Agency

A U.S. government agency that was instrumental in developing and using the original ARPA Services networking standards.

Bind

- Process by which a client locates and directs all requests for data to a specific server.

- Process of establishing the address of a socket that allows other sockets to connect to it or to send data to it.

CDF

Context Dependent File.

A hidden directory that contains all the versions of a file needed by the different cnodes.

Client

- A node that requests data or services from other nodes (servers).

- A process that requests other processes to perform operations.

Note: An NFS client can also be configured as any combination of an NFS server, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Clock Skew — A difference in clock times between systems.

Cluster — One or more workstations linked together with a local area network (LAN), but consisting of only one file system.

Cnode — Any node operating in an HP-UX cluster environment, including diskless nodes and the root server.

Daemon — Background programs that are always running, waiting for a request to perform a task.

Diskless Cnode — A node in an HP-UX cluster that uses networking capabilities to share file systems, but does not have a file system physically attached.

Escape Sequence (YP) — Characters used within files to force inclusion and exclusion of data from YP databases. The escape sequences are as follows.

- + *(plus)*
- - (minus)
- +*@netgroup_name*
- -*@netgroup_name*

Export — To make a file system available to remote nodes via NFS.

File System — An entire unit (disk) that has a fixed size.

GID — A value that identifies a group in HP-UX.

Global (YP) — A means of access in which the system always reads YP maps rather than the local ASCII files.

Hard Mount — A mount that causes NFS to retry a remote file system request until it succeeds, you interrupt it (default option), or you reboot the system.

Home Node

A term used in Virtual Home Environment (VHE) to refer to the machine on which a user's home directory physically resides.

Host

A node that has primary functions other than switching data for the network.

Host Node

A term used in Virtual Home Environment (VHE) to refer to the node a user is logged in to. This node environment is set up from the configuration files found on the user's home node.

Import

To obtain access to a remote file system from an outside source; to mount.

Internet Address

A four-byte quantity that is distinct from a link-level address and is the network address of a computer node. This address identifies both the specific network and the specific host on the network.

Interruptable Mount

A mount that allows you to interrupt an NFS request by pressing an interrupt key. (Though the interrupt key is not standardized, common ones include **CTRL - C** and **BREAK**.)

Key (YP)

A string of characters (no imbedded blanks or tabs) that indexes the values within a YP map so the system can easily retrieve information. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.

Local (YP)

A means of access in which the system first reads the local ASCII file. If it encounters an escape sequence, it then accesses the YP databases.

Map (YP)	A file consisting of logical records; a search key and related value form each record. YP clients can request the value associated with any key within a map. **YP map** is synonymous with **YP database.**
Map Nickname (YP)	A synonym for the YP map name when using certain YP commands.
Master Server (YP)	The node on which one or more YP maps are constructed from ASCII files. These maps are then copied to the YP slave servers for the YP clients to access.
Mount	To obtain access to a remote or local file system or directory (import).
Mount Point	The name of the directory on which a file system is mounted.
Netgroup	A network-wide group of nodes and users defined in */etc/netgroup*.
NFS	Network File System.
Network Lock Manager	A facility for locking files and synchronizing access to shared files.
Network Status Monitor	A daemon running on all network computers to maintain stateful locking service within NFS. It also allows applications to monitor the status of other computers.
Node	A computer system that is attached to or is part of a computer network.
Propagate	To copy maps (data) from one YP server to another.

Protocol

The rules and steps by which servers and clients exchange data and control information.

Remote Execution Facility (REX)

A facility which allows a user to execute commands on a remote node.

Remote Procedure Call (RPC)

A call made by clients either to access server information or to request action from servers.

Remote Procedure Call Protocol Compiler (RPCGEN)

A remote procedure call compiler used to help programmers write RPC applications by automatically generating necessary programs and code fragments.

Root Server

The only node in an HP-UX cluster that has file systems physically attached to it.

Server

- A node that provides data or services to other nodes (clients) on the network.
- A process that performs operations as requested by other processes.

Note: An NFS server can also be configured as any combination of an NFS client, YP client, or YP server. (A YP server **must** also be configured as a YP client.)

Slave Server (YP)

A node that copies YP maps from the YP master server and then provides YP clients access to these maps.

Soft Mount

An optional mount that causes access to remote file systems to abort requests after one NFS attempt.

Stateless

Servers do not maintain (preserve) information relating to each file being served. Each file request moves across the network with the parameters attached to it locally (e.g., read and write privileges).

Steady State

Servers maintain (preserve) information relating to each file being served.

For YP, the information contained in a YP map is consistent among all YP servers within a given YP domain (i.e., is not in the process of being updated).

UID

A value that identifies a user in HP-UX.

Unmount

To remove access rights to a file system or disk that was mounted via the *mount(1M)* command.

Update

The HP-UX command that installs software onto the system.

Value (YP)

A unit of information stored in YP maps; each value has a corresponding key (index) so the system can easily retrieve it. For example, in the *passwd.byname* map, the users' login names are the keys and the matching lines from */etc/passwd* are the values.

VHE

See "Virtual Home Environment."

Virtual Home Environment (VHE)

A network service that allows users to log in at host nodes and utilize their home nodes' execution environments.

External Data Representation (XDR)

A protocol that translates machine-dependent data formats (i.e., internal representations) to a universal format used by other network hosts using XDR.

Yellow Pages (YP)

An optional network service composed of databases (maps) and processes that provide YP clients access to the maps. The YP service enables you to administer these databases from one node.

YP may or may not be active; check with your system administrator.

YP Client

- A node that requests data or services from YP servers.
- A YP process that requests other YP processes to perform operations.

Note: A YP client can also be configured as any combination of a YP server, NFS client, or NFS server. (A YP server **must** also be configured as a YP client.)

YP Database

See "Map (YP)."

YP Domain

A logical grouping of YP maps (databases) stored in one location. YP domains are specific to the YP network service and are not associated with other network domains.

YP Map

See "Map (YP)."

YP Password

The password for a user's login ID that exists in the YP *passwd* map. The YP password is the same one as the user password, but is administered through the YP.

You do not have to have a YP password to access the YP databases.

YP Server

- A node that provides data (maps) or services to other nodes (YP clients) on the network using YP.

- A YP process that performs operations as requested by other YP processes.

Note: A YP server **must** also be configured as a YP client. It can also be configured as an NFS server, NFS client, or both.

Index

A

Adding Computers 4-13
Append Mode A-1
Automatic Mounts 5-43

B

bg Mount Options 5-40
Binding 2-2
biod(1M) 5-9

C

chacl(1) A-1
Clients
  NFS 2-1
  Unmount File Systems 5-48
  YP 2-15, 3-16
Clock Skews 5-53
Clusters C-1
  CDFs C-1, C-3
  Clock Skews C-3
  cnodes C-1
  Daemons C-2
  Diskless Cnodes C-1
  Mounts C-2
  NFS Configuration C-2
  NFS Maintenance C-2
  Root Servers C-1
  Troubleshooting C-3
  Unmounts C-2
  YP Configuration C-3
  YP Maintenance C-3
Cnodes C-1
Commands
  Common NFS 3-6
  Common YP 3-19
  domainname(1) 3-19, 8-17
  makedbm(1M) 8-17
  on command 6-3
  on() 6-1
  rpcinfo(1M) 3-9
  rup(1M) 3-11
  rusers(1M) 3-12
  showmount(1M) 3-13
  YP 8-17
  ypbind(1M) 8-17
  ypcat(1) 3-20, 8-17
  ypinit(1M) 8-19
  ypmake(1M) 8-19
  ypmatch(1) 3-21, 8-19
  yppasswd(1) 3-21, 8-19
  yppasswdd(1M) 8-19
  yppoll(1M) 8-19
  yppush(1M) 8-20
  ypserv(1M) 8-20
  ypset1M) 8-20

  ypwhich(1) 3-24, 8-20
  ypxfr(1M) 8-20
Configuration 5-16

NFS 5-1, 5-13
YP 8-21
Configuration Files
/etc/checklist 5-7, 5-43
/etc/exports 5-7, 5-28
/etc/hosts 5-24
/etc/inetd.conf 5-7, 5-19
/etc/netgroup 5-7, 5-25
/etc/netnfsrc 5-7, 5-17, 5-34
/etc/rpc 5-8
/usr/adm/inetd.sec 5-8, 5-21
NFS 5-7
Configure Kernel 4-11
crontab(1) 8-36

D

Daemons
biod(1M) 5-9
Clusters C-2
inetd(1M) 5-9, 5-19
NFS 5-9
nfsd(1M) 5-9
pcnfsd(1M) 5-10
portmap(1M) 5-10
rexd(1M) 6-1
Device Files A-1, 2-6
dfile 4-12
Diagnostics 6-11, 6-13
Diskless Cnodes C-1
Documentation
Contents 1-2
Conventions 1-5
Guide 1-6
Overview 1-1 - 1-2
domainname(1) 3-19, 8-17

E

Editing
/etc/checklist 5-43
/etc/exports 5-28
/etc/hosts 5-24
/etc/inetd.conf 5-19
/etc/netfsrc 5-34
/etc/netgroup 5-25
/etc/netgroups 8-12
/etc/netnfsrc 5-17
/usr/adm/inetd.sec 5-21
Environment Simulation 6-5
EREMOTE 10-13
Errnos
EREMOTE 10-13
ESTALE 10-13
Error Messages 10-12
on command 6-5, 6-11
rexd 6-12
Escape Sequences 8-10
ESTALE 10-13
/etc/checklist 5-7, 5-43
/etc/exports 5-7, 5-28
/etc/group 8-28
/etc/hosts 5-24, 8-28
/etc/hosts.eqiv 6-8
/etc/hosts.equiv 8-16, 8-29
/etc/inetd.conf 5-7, 5-19, 6-6 - 6-7
/etc/netgroup 5-7, 5-25, 8-28
/etc/netgroups 8-12
/etc/netnfsrc 5-7, 5-17, 5-34
/etc/networks 8-28
/etc/passwd 8-30
/etc/protocols 8-28
/etc/rpc 5-8
/etc/services 8-28
Export File Systems 5-28
External Data Representation 2-11

F

fcntl() 7-1
fg Mount Defaults 5-40

File Access A-2, 2-4, 3-6
Limiting 5-24 - 5-25, 5-28
Removing 5-48
File Locking A-2
File Names A-2
File Systems
Automatic Mounts 5-43
Availability of 5-24 - 5-25
Manual Mounts 5-45
Mounting of 5-37
Preventing Access 5-50
Removing Access 5-48
Unmount 5-48
Files
NFS Configuration 5-7
Flowchart Formats 10-14
fpathconf A-8

G

getacl(2) A-2
GIDs, Setting of 5-15
Global Maps 8-8
Group Membership A-2

H

Hard Mounts 5-37, 5-40, 5-43, 5-45
$HOME/.rhosts 6-8, 8-16, 8-29
HP-UX Clusters C-1

I

inetd(1M) 5-9, 5-19
Installation 4-1
Adding Computers 4-13
Configure Kernel 4-11
dfile 4-12
Preparing the System 4-5
Recompile Programs 4-14
Software 4-7
update 4-7
int Mount Defaults 5-40

L

Local Maps 8-8
lockf() 7-1
Locking Protocol 7-4
Log Files 8-49

M

Maintenance
NFS 5-48
YP 8-40
makedbm(1M) 8-17
Manual Mounts 5-45
Maps 2-15, 3-15
Global 8-8
Local 8-8
Maintenance 8-41
Manual Modifications 8-42
Modification 8-41
Non-standard 8-42, 8-51
Propagation 8-36
Master Server
Changing of 8-46
Master Servers
Automatic Start 8-25
Configuration 8-23
Manual Start 8-25
Security 8-23
_MAX_INPUT A-8
Memory 5-6
mkdnod A-3
Mount Defaults
fg 5-40
hard 5-40

int 5-40
port 5-41
retrans 5-41
retry 5-41
rsize 5-41
rw 5-41
setuid 5-42
timeo 5-42
wsize 5-42
Mount Options
bg 5-40
noauto 5-40
noint 5-40
nosuid 5-40
ro 5-41
soft 5-42
mount(1M) 5-45
mountd(1M) 5-11
Mounts
/etc/checklist 5-43
Automatic 5-43
Guidelines 5-38
Hard 5-37, 5-40, 5-43, 5-45
Manual 5-45
Options 5-40
Soft 5-38, 5-42 - 5-43, 5-45
Wide Area Networks 5-42

N

_NAME_MAX A-8
Named Pipes A-3, 2-5
Netgroups 5-25, 8-12
Network Lock Manager 2-12, 7-1 - 7-2, 7-4, 7-6
Network Memory 5-6
Network Status Monitor 7-5
NFS
Common Commands 3-6
Daemons 5-9
Servers 5-11
NFS Client
Configuration 5-34
NFS Clients 2-1
NFS Configuration 5-1, 5-13
Clients 5-34
Clusters C-2
Edit */etc/checklist* 5-43
Edit */etc/exports* 5-28
Edit */etc/hosts* 5-24
Edit */etc/inetd.conf* 5-19
Edit */etc/netfsrc* 5-34
Edit */etc/netgroup* 5-25
Edit */etc/netnfsrc* 5-17
Edit */usr/adm/inetd.sec* 5-21
Files 5-7
Guidelines 5-6
Memory 5-6
Reboot System 5-33, 5-47
Security 5-19, 5-21
Servers 5-17
Set UIDs and GIDs 5-15
NFS Maintenance 5-48
Clock Skews 5-53
Clusters C-2
Remove NFS File Access 5-48
Update Software 5-52
NFS Servers 2-1
Configuration 5-17
Security 5-19, 5-21
NFS Services
Overview 2-1, 2-3
NFS to RFA
Changing Scripts B-4
NFS Troubleshooting 10-11, 10-17
nfsd(1M) 5-9
noauto Mount Options 5-40
noint Mount Options 5-40
Non-standard Maps 8-42
nosuid Mount Options 5-40
_NO_TRUNC A-8

P

pathconf A-8
_PC_CHOWN_RESTRICTED A-8
_PC_LINK_MAX A-8
_PC_MAX_CANON A-8
_PC_PATH_MAX A-8
_PC_PIPE_BUF A-8
_PC_VDISABLE A-8
PC NFS Servers 5-17, 5-34
pcnfsd(1M) 5-10
port Mount Defaults 5-41
portmap(1M) 5-10
Program Recompiling 4-14
Propagate Maps 8-36

R

Reading Directories A-3
Reboot System 5-33, 5-47
Recompile Programs 4-14
Remote Execution Facility 2-8, 6-1 - 6-2, 6-4, 6-6, 6-8, 6-10, 6-12
Remote File Access A-2, 2-4, 3-6
  Limiting 5-24 - 5-25, 5-28
  Removing 5-48
Remote Procedure Call 2-8, 7-1
Remote Services
  Setting Access to 5-21
  Setting Maximum Number of 5-21
Remove NFS File Access 5-48
retrans Mount Defaults 5-41
retry Mount Defaults 5-41
RFA A-3
RFA through NFS B-7
ro Mount Options 5-41
Root Servers C-1
RPC Services
  /etc/inetd.conf Entries 5-20
  /usr/adm/inetd.sec Entries 5-23
  Activation of 5-19
  Security 5-19, 5-23
RPCGEN 2-9
rpcinfo(1M) 3-9
rsize Mount Defaults 5-41
rstatd(1M) 5-11
rup(1M) 3-11
rusers(1M) 3-12
rusersd(1M) 5-11
rw Mount Defaults 5-41
rwalld(1M) 5-12

S

Security 5-19, 5-21, 5-23
  Considerations 6-9
  Limitations 6-9
  Master Servers 8-23
Servers
  mountd(1M) 5-11
  NFS 2-1, 5-11
  PC NFS 5-17, 5-34
  Preventing Access 5-50
  rstatd(1M) 5-11
  rusersd(1M) 5-11
  rwalld(1M) 5-12
  sprayd(1M) 5-12
  Stateless 2-2
  YP 2-15, 3-16
  YP Masters 2-17, 3-18
  YP Slaves 2-17, 3-18
setacl(2) A-3
settuple(3) A-3
setuid Mount Defaults 5-42
showmount(1M) 3-13

Slave Servers
  Automatic Start 8-35
  Configuration 8-33
  Manual Start 8-35
Soft Mounts 5-38, 5-42 - 5-43, 5-45

Software Installation 4-7
Software Update 5-52
sprayd(1M) 5-12
Stateless Servers 2-2
Super-user Permission A-4
System Time A-5

T

timeo Mount Defaults 5-42
Troubleshooting 10-1
  Clusters C-3
  Configuration 10-9
  Errnos 10-13
  Error Messages 10-12
  Hardware 10-9
  Incorrect YP Maps 10-43
  Initial Steps 10-9, 10-17
  Initial YP Steps 10-41
  Mount Fails 10-21
  Network Communication 10-10
  Network Problems 10-8
  NFS 10-11, 10-17
  Performance Problems 10-39
  Programs Hang 10-37
  References 10-6
  Restricted Access 10-33
  Server Not Responding 10-25
  Unsolved Problems 10-13
  VHE 10-52, 10-54, 10-56, 10-58, 10-60, 10-62, 10-64
  YP 10-11, 10-41
  YP Clients 10-51
  ypbind(1M) 10-49
  ypserv(1M) 10-47

U

UID, Setting of 5-15
umount(1M) 5-48
Unlinking A-6
Unmount File Systems 5-48
update 4-7, 5-52
/usr/adm/inetd.sec 5-8, 5-21, 6-9
/usr/etc/rpc.rexd 6-7
/usr/spool/rexd 6-5

V

VHE Advanced Usage
  Alternate Mount Points 9-21
  altlogin Login 9-18
  mounter Login 9-18
  Using VHE for Mail 9-21
VHE Configuration 9-3
  /etc/exports 9-10
  /etc/passwd 9-3, 9-9
  /etc/vhe_list 9-3, 9-6
  Mounts in Background 9-15
  Overview 9-2
  Refinements 9-15
  Using YP 9-4
  vhe_mounter 9-11
VHE Maintenance 9-16
  Adding or Deleting Nodes 9-17
  Unmounting File Systems 9-16
Virtual Home Environment
  Advanced Usage 9-18 - 9-19, 9-21
  Advantages 2-18
  Concepts 2-21 - 2-22
  Configuration 9-3
  Disadvantages 2-20
  Maintenance 9-16
  Overview 2-18, 9-1
  Troubleshooting 10-52, 10-54, 10-56, 10-58, 10-60, 10-62, 10-64

W

Wide Area Network Mounts 5-42
wsize Mount Defaults 5-42

Y

Yellow Pages 2-12
  Advantages 2-13
  Clients 3-16
  Commands 3-19
  Concepts 2-14
  Configuration 8-21
  Databases 8-7
  Disabling of 8-40
  Disadvantages 2-13
  Domains 2-16, 3-16
  Escape Sequences 8-10
  Log Files 8-49
  Maps 2-15, 3-15, 8-8
  Masters 2-17, 3-18
  Overview 3-15
  Servers 3-16
  Slaves 2-17, 3-18
  Structure 2-14
  Troubleshooting 10-11, 10-41
  Verification of 8-39
YP Clients 2-15
  Alteration of 8-28
  Automatic Start 8-32
  Configuration 8-27
  Manual Start 8-32
  Troubleshooting 10-51
YP Configuration 8-21
  $HOME/.rhosts 8-29
  /etc/group 8-28
  /etc/hosts 8-28
  /etc/hosts.equiv 8-29
  /etc/netgroup 8-28
  /etc/networks 8-28
  /etc/passwd 8-30
  /etc/protocols 8-28
  /etc/services 8-28
  Clients 8-27
  Clusters C-3
  Master Servers 8-23
  Propagate Maps 8-36
  Slave Server 8-33
YP Domains 2-16, 3-16
YP Maintenance 8-40
  Add New Users 8-45
  Adding Servers 8-45
  Changing Master Servers 8-46
  Log Files 8-49
  Manual Modification
    to Maps 8-42
  Master Servers 8-46
  Modify YP Maps 8-41
  Non-standard Maps 8-51
  Password 8-47
YP Password 3-21 - 3-23, 8-47
YP Servers 2-15
  Adding of 8-45
  Maintenance 8-45
ypbind(1M) 8-17
ypcat(1) 3-20, 8-17
ypinit(1M) 8-19
ypmake(1M) 8-19
ypmatch(1) 3-21, 8-19
yppasswd(1) A-7, 3-21 - 3-23,
  8-19, 8-47
yppasswdd(1M) 8-19
yppoll(1M) 8-19
yppush(1M) 8-20, 8-38
ypserv(1M) 8-20
ypset(1M) 8-20
ypwhich(1) 3-24
ypxfr(1M) 8-20, 8-36, 8-38

Reader Comment Card

HP 9000 Series 300
Using and Administering NFS Services
50969-90001, E0189

We welcome your evaluation of this manual. Your comments and suggestions will help us improve our publications. Please tear this card out and mail it in. Use and attach additional pages if necessary.

Please circle the following Yes or No:

• Is this manual well organized?	Yes	No
• Is the information technically accurate?	Yes	No
• Are instructions complete?	Yes	No
• Are concepts and wording easy to understand?	Yes	No
• Are examples and pictures helpful?	Yes	No
• Are there enough examples and pictures?	Yes	No

Comments:_____

_____

_____

_____

_____

Name:_____

Title:_____

Company:_____

Address:_____

City & State:_____

Zip:_____

HEWLETT PACKARD

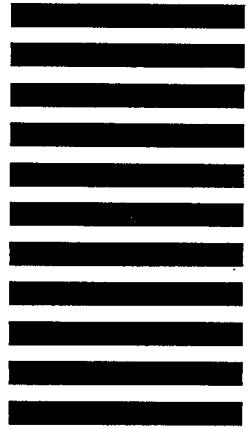
Printed in USA

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 37 LOVELAND,CO

POSTAGE WILL BE PAID BY ADDRESSEE



Hewlett-Packard Company
Colorado Networks Division
3404 East Harmony Road
Fort Collins, CO 80525

ATTN: User Information Development Department

Fold Here

Tape Please do not staple Tape

HP Part Number
50969-90001
Printed in U.S.A. E0189