

**HP 9000 Networking**  
**Managing Highly Available NFS®**

**HP Part No. B5125-90001**  
**Printed in U.S.A.**  
**E0496**

**Edition 1**

© Copyright 1996, Hewlett-Packard Company.



## Legal Notices

The information in this document is subject to change without notice.

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

**Warranty.** A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

**Restricted Rights Legend.** Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Hewlett-Packard Company  
19420 Homestead Road  
Cupertino, California 95014 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

**Copyright Notices**

©copyright 1983-96 Hewlett-Packard Company, all rights reserved.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

©copyright 1986-1996 Sun Microsystems, Inc.

**Trademark Notices**

MC/ServiceGuard<sup>®</sup> is a registered trademark of Hewlett-Packard Company.

NFS<sup>®</sup> is a registered trademark of Sun Microsystems, Inc.

NIS<sup>™</sup> is a trademark of Sun Microsystems, Inc.

UNIX<sup>®</sup> is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.



---

## Contents

### **1 Overview of Highly Available NFS**

Limitations of Highly Available NFS 9

Supported Configurations 10

Simple Failover to an Idle NFS Server 11

Failover from One Active NFS Server to Another 12

A Host Configured as Adoptive Node for Multiple Packages 13

Cascading Failover, with Three Adoptive Nodes 14

Server-to-Server Cross Mounting 15

How the Control and Monitor Scripts Work 17

Starting the NFS Services 17

Halting the NFS Services 17

Monitoring the NFS Services 18

On the Client Side 19

### **2 Installing and Configuring Highly Available NFS**

Installing Highly Available NFS 23

Before Creating a Highly Available NFS Package 24

Configuring a Highly Available NFS Package 28

Copying the Template Files 28

Editing the Package Configuration File (nfs.conf) 29

Editing the Control Script (nfs.ctl) 31

Configuring Server-to-Server Cross-Mounts (Optional) 34

Creating the MC/ServiceGuard Binary Configuration File 37

---

## Contents

### 3 Sample Configurations

#### Three-Server Mutual Takeover 41

Cluster Configuration File for Three-Server Mutual Takeover 43

Package Configuration File for pkg01 44

NFS Control Script for pkg01 45

Package Configuration File for pkg02 46

NFS Control Script for pkg02 47

Package Configuration File for pkg03 48

NFS Control Script for pkg03 49

#### One Adoptive Node for Two Packages 50

Cluster Configuration File for Adoptive Node for Two Packages 52

Package Configuration File for pkg01 53

NFS Control Script for pkg01 54

Package Configuration File for pkg02 55

NFS Control Script for pkg02 56

#### Three-Server Cascading Failover 57

Cluster Configuration File for Three-Server Cascading Failover 59

Package Configuration File for pkg01 60

NFS Control Script for pkg01 61

Package Configuration File for pkg02 62

NFS Control Script for pkg02 63

#### Two Servers with NFS Cross-Mounts 64

Cluster Configuration File for Two-Server NFS Cross-Mount 66

Package Configuration File for pkg01 67

NFS Control Script for pkg01 68

Package Configuration File for pkg02 70

NFS Control Script for pkg02 71

---

**Overview of Highly Available NFS**

## Overview of Highly Available NFS

Highly Available NFS<sup>®</sup> is a tool kit that allows you to use MC/ServiceGuard to set up highly available NFS servers.

You must set up an MC/ServiceGuard cluster before you can set up Highly Available NFS. For instructions on setting up an MC/ServiceGuard cluster, see the *Managing MC/ServiceGuard* manual.

Highly Available NFS is a separately purchased set of configuration files and control scripts, which you customize for your specific needs. These files, once installed, are located in `/opt/cmcluster/nfs`.

MC/ServiceGuard allows you to create high availability clusters of HP 9000 Series 800 computers. A high availability computer system allows applications to continue in spite of a hardware or software failure. Highly available systems protect users from software failures as well as from failure of a system processing unit (SPU) or local area network (LAN) component. In the event that one component fails, the redundant component takes over, and MC/ServiceGuard coordinates the transfer between components.

An NFS server is a host that “exports” its local directories (makes them available for client hosts to mount using NFS). On the NFS client, these mounted directories look to users like part of the client’s local file system.

With highly available NFS, the NFS server package containing the exported file systems can move to a different node in the cluster in the event of failure. After MC/ServiceGuard starts the NFS package on the adoptive node, the NFS file systems are re-exported from the adoptive node with minimum disruption of service to users. The client side “hangs” until the NFS server package comes up on the adoptive node. When the service returns, the user can continue access to the file. You do not need to restart the client.

## Limitations of Highly Available NFS

The following limitations apply to the use of Highly Available NFS:

- File locks are not maintained when an NFS server package moves to a new node. Any applications that use file locking must reclaim their locks after an NFS server package fails over. An application that loses its file lock as a result of an NFS package failover will not be notified. If the server is also an NFS client, any file locks it holds will be lost.
- The Highly Available NFS tool kit available on HP-UX release 10.10 may be used *only* on release 10.10 or later.
- You cannot use Highly Available NFS for an NFS diskless cluster server.
- You can use Highly Available NFS only in an MC/ServiceGuard cluster with 4 nodes or fewer.

## Supported Configurations

Highly Available NFS supports the following configurations:

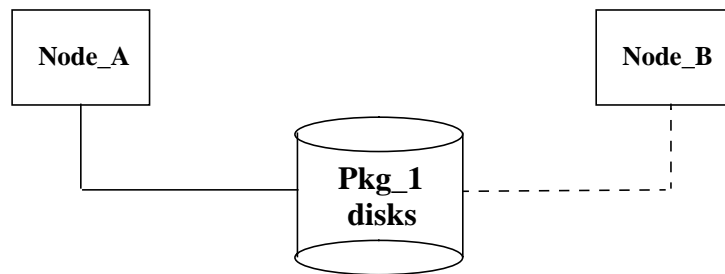
- Simple failover from an active NFS server node to an idle NFS server node.
- Failover from one active NFS server node to another active NFS server node, where the adoptive node supports more than one NFS package after the failover.
- A host configured as an adoptive node for more than one NFS package. The host may also be prevented from adopting more than one failed package at a time.
- Cascading failover, where a package may have up to three adoptive nodes.
- Server-to-server cross mounting, where one server may mount another server's file systems, and the mounts are not interrupted when one server fails.

These configurations are illustrated in the following sections.

### Simple Failover to an Idle NFS Server

Figure 1 shows a simple failover from an active NFS server node to an idle NFS server node.

#### Before Failover:



#### After Failover:

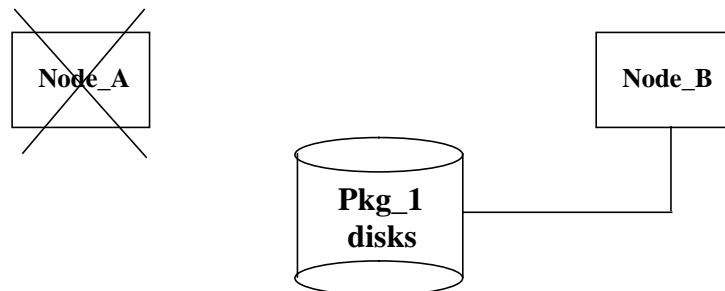


Figure 1

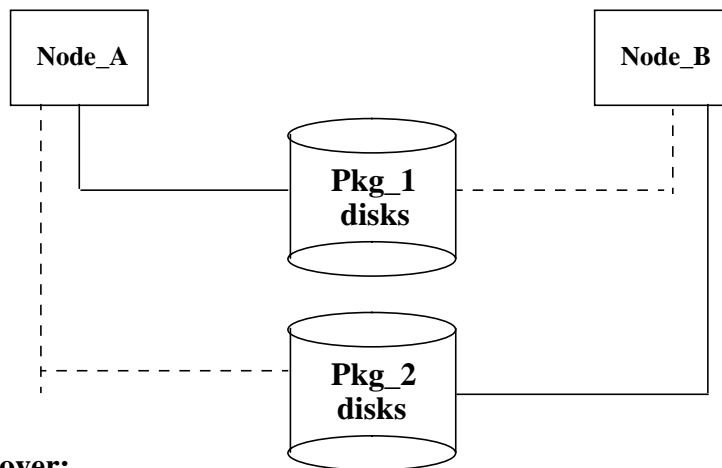
#### Simple Failover to an Idle NFS Server

**Node\_A** is the primary node for NFS server package **Pkg\_1**. When **Node\_A** fails, **Node\_B** adopts **Pkg\_1**. This means that **Node\_B** locally mounts the file systems associated with **Pkg\_1** and exports them. Both **Node\_A** and **Node\_B** must have access to the disks that hold the file systems for **Pkg\_1**.

### Failover from One Active NFS Server to Another

Figure 2 shows a failover from one active NFS server node to another active NFS server node.

#### Before Failover:



#### After Failover:

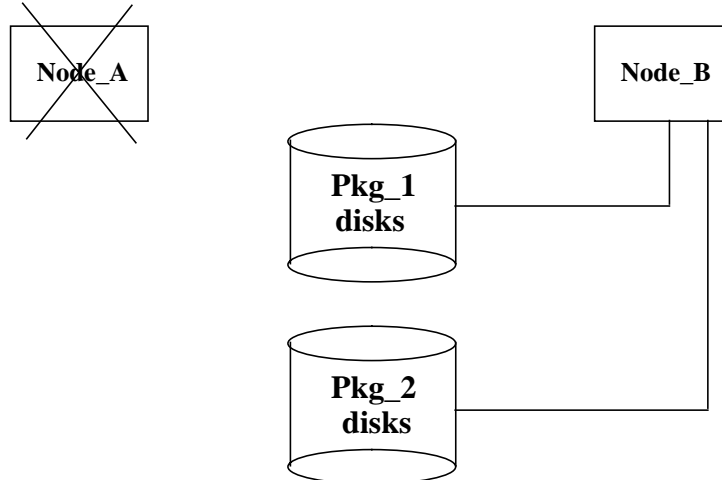


Figure 2

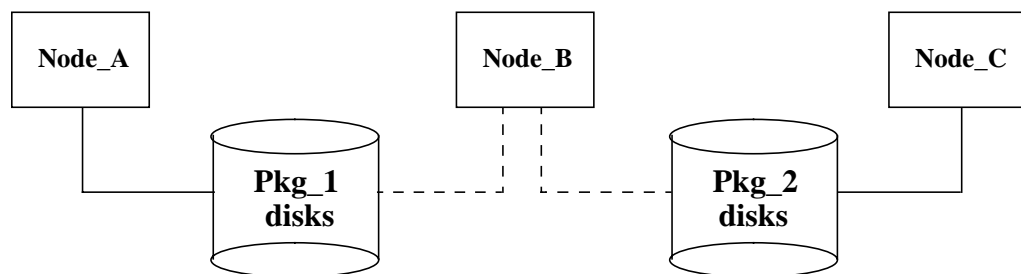
### Failover from One Active NFS Server to Another

In Figure 2, **Node\_A** is the primary node for **Pkg\_1**, and **Node\_B** is the primary node for **Pkg\_2**. When **Node\_A** fails, **Node\_B** adopts **Pkg\_1** and becomes the server for both **Pkg\_1** and **Pkg\_2**.

### A Host Configured as Adoptive Node for Multiple Packages

Figure 3 shows a three-node configuration where one node is the adoptive node for packages on both of the other nodes. If either **Node\_A** or **Node\_C** fails, **Node\_B** adopts the NFS server package from that node.

#### Before Failover:



#### After Failover:

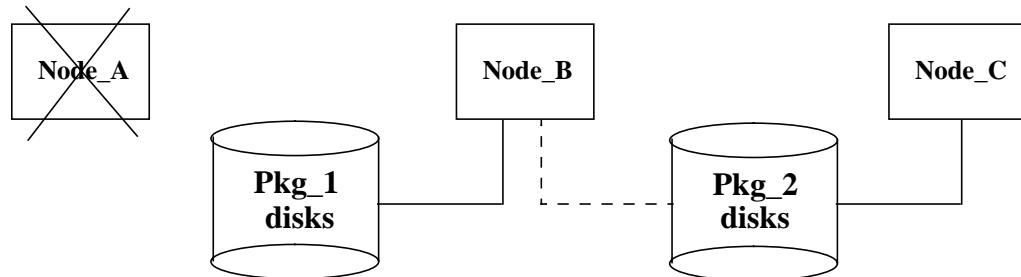


Figure 3

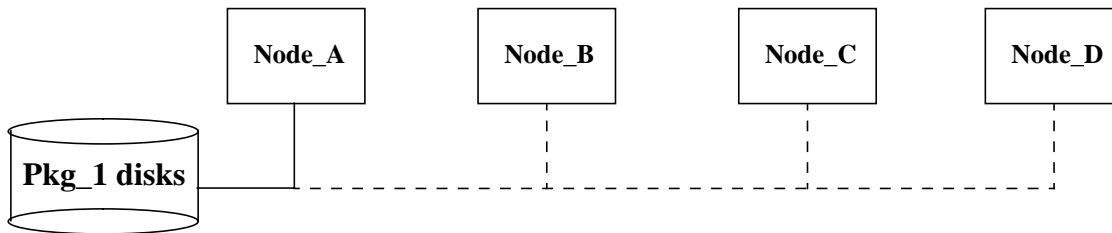
### A Host Configured as Adoptive Node for Multiple Packages

When **Node\_A** fails, **Node\_B** becomes the server for **Pkg\_1**. If **Node\_C** fails, **Node\_B** will become the server for **Pkg\_2**. Alternatively, you can set the package control option in the NFS control script to prevent **Node\_B** from adopting more than one package at a time. With the package control option, **Node\_B** may adopt the package of the first node that fails, but if the second node fails, **Node\_B** will not adopt its package. The package control option prevents a node from becoming overloaded by adopting too many packages. If an adoptive node becomes overloaded, it can fail.

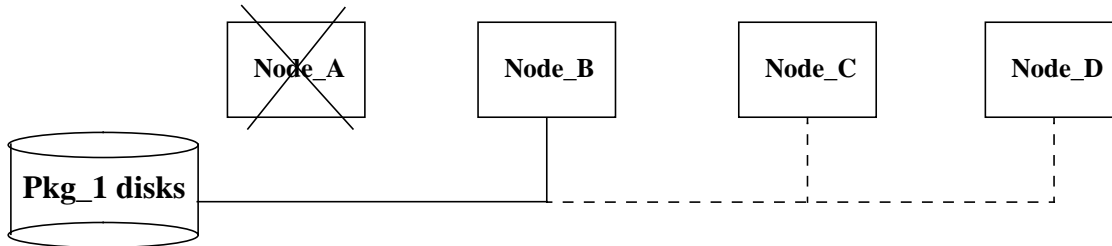
### Cascading Failover, with Three Adoptive Nodes

A package may be configured with up to three adoptive nodes. Figure 4 shows this configuration. If **Node\_A** fails, **Pkg\_1** is adopted by **Node\_B**. However, if **Node\_B** is down, **Pkg\_1** is adopted by **Node\_C**, and if **Node\_C** is down, **Pkg\_1** is adopted by **Node\_D**. The adoptive nodes are listed in the package configuration file, `/etc/cmcluster/nfs/nfs.conf`, in the order in which they will be tried. Note that all four nodes must have access to the disks for the **Pkg\_1** file systems.

#### Before Failover:



#### After Failover of Node\_A:



#### After Failover of Node\_B:

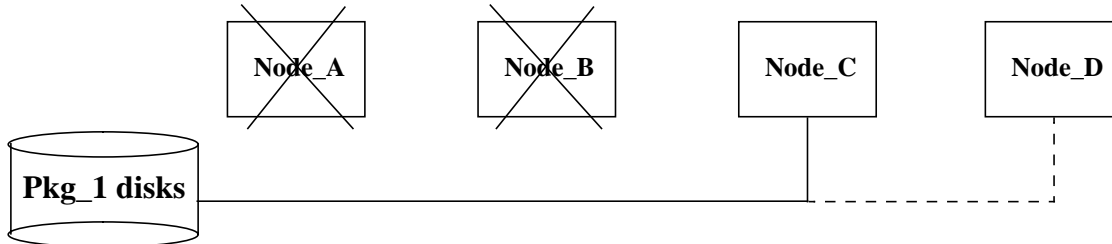


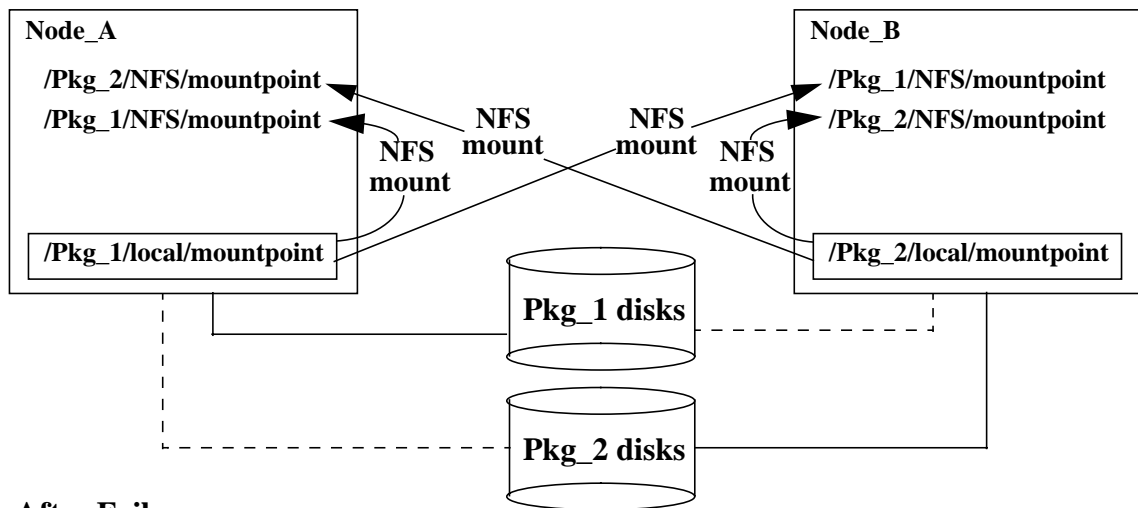
Figure 4

Cascading Failover, with Three Adoptive Nodes

### Server-to-Server Cross Mounting

Two NFS server nodes may NFS-mount each other's file systems and still act as adoptive nodes for each other's NFS server packages. Figure 5 illustrates this configuration.

#### Before Failover:



#### After Failover:

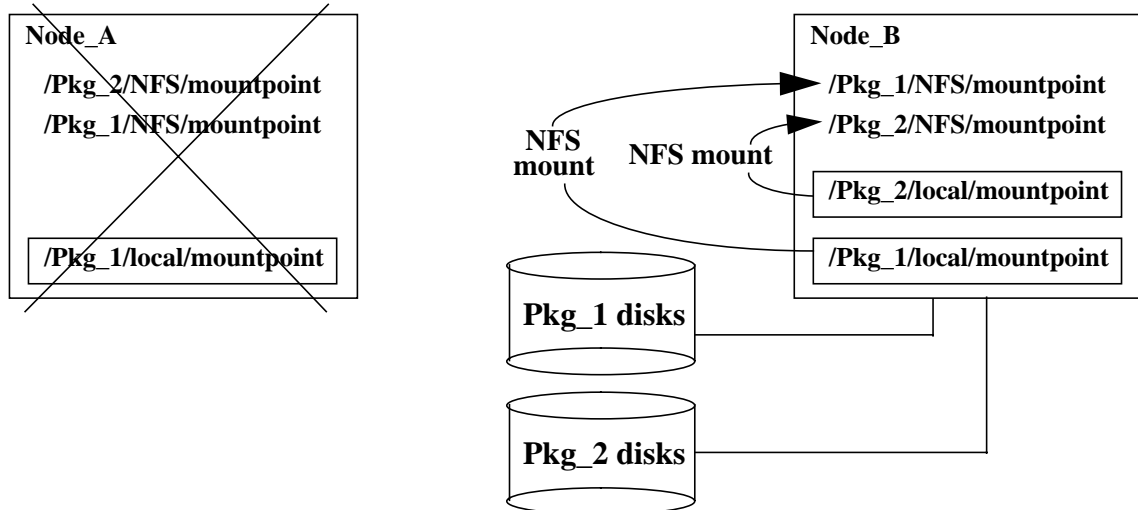


Figure 5

Server-to-Server Cross Mounting

## Overview of Highly Available NFS Supported Configurations

The advantage of server-to-server cross-mounting is that every server has an identical view of the file systems. The disadvantage is that, on the node where a file system is locally mounted, the file system is accessed through an NFS mount, which has poorer performance than a local mount.

Each node NFS-mounts the file systems for both packages. If **Node\_A** fails, **Node\_B** mounts the filesystem for **Pkg\_1**, and the NFS mounts are not interrupted.

## How the Control and Monitor Scripts Work

As with all ServiceGuard packages, the control script starts and stops the NFS package and determines how the package will operate once it becomes available on a particular node. The control script contains two sets of code that operate depending on whether the script is called with the **start** parameter or the **stop** parameter. A template control script is provided in `/opt/cmcluster/nfs/nfs.cntl`. “Editing the Control Script (nfs.cntl)” on page 31 tells you how to modify this template file for your own NFS packages.

### Starting the NFS Services

When called with the **start** parameter, the control script does the following:

- Activates the volume group or volume groups associated with the package.
- Mounts each file system associated with the package.
- Initiates the NFS monitor script to check periodically on the health of NFS services, if you have configured your NFS package to use the monitor script.
- Exports each file system associated with the package so that it can later be NFS-mounted by clients.
- Assigns a package IP address to the LAN card on the current node.

After this sequence, the NFS server is active, and clients can NFS-mount the exported file systems associated with the package.

### Halting the NFS Services

When called with the **stop** parameter, the control script does the following:

- Removes the package IP address from the LAN card on the current node.
- Un-exports all file systems associated with the package so that they can no longer be NFS-mounted by clients.
- Halts the monitor process.
- Halts the **rpc.lockd** and **rpc.statd** daemons to release file locks so that

## Overview of Highly Available NFS

### How the Control and Monitor Scripts Work

file systems can be unmounted. If the server is also an NFS client, any file locks it holds will be lost.

- Restarts the **rpc.lockd** and **rpc.statd** daemons so that they can manage the file locks for other NFS packages still running on the server.
- Unmounts each file system associated with the package.
- Deactivates each volume group associated with the package.

After this sequence, the NFS package is inactive on the current node and may start up on an alternate node or be restarted later on the same node.

### Monitoring the NFS Services

The monitor script `/etc/cmcluster/nfs/nfs.mon` works by periodically checking the status of NFS services using the **rpcinfo** command. If any service fails to respond, the script exits, causing a switch to an adoptive node.

The monitor script monitors only those NFS services configured in the `/etc/rc.config.d/nfsconf` file to be started at system boot. If you set the **NFS\_SERVER** variable to 1, the monitor script monitors the **nfsd**, **rpc.lockd**, and **rpc.statd** processes. (If one **nfsd** process dies or is killed, the package fails over, even if other **nfsd** processes are still running.) If you set the **PCNFS\_SERVER** variable to 1, the monitor script monitors the **pcnfsd** process, and if you set the **START\_MOUNTD** variable to 1, it monitors the **rpc.mountd** process.

The default NFS control script does not invoke the monitor script. You do not have to run the NFS monitor script to use Highly Available NFS. If your NFS package configuration file specifies **PKG\_SWITCHING\_ENABLED YES** and **NET\_SWITCHING\_ENABLED YES** (the defaults), the package will switch to the next adoptive node or to a standby network interface in the event of a node or network failure. However, if one of the NFS services goes down while the node and network remain up, you need the NFS monitor script to detect the problem and to switch the package to an adoptive node.

Whenever the monitor script detects an event, it logs it to a file with the same name as your NFS control script but with a **.log** extension. Each NFS package has its own log file. For example, if your control script is called `/etc/cmcluster/nfs/nfs1.cnt1`, the log file is called `/etc/cmcluster/nfs/nfs1.cnt1.log`.

## On the Client Side

The client should NFS-mount a file system using the package name in the **mount** command. The package name is associated with the package's relocatable IP address. On client systems, be sure to use a hard mount and set the proper retry values for the mount. Alternatively, set the proper timeout for automounter. The timeout should be greater than the total end-to-end recovery time for the highly available NFS package—that is, running **fsck**, mounting file systems, and exporting file systems on the new node. (With journalled file systems, this time should be between one and two minutes.) Setting the timeout to a value greater than the recovery time allows clients to reconnect to the file system after it returns to the cluster on the new node.



---

**Installing and Configuring Highly Available NFS**

## Installing and Configuring Highly Available NFS

This chapter explains how to configure Highly Available NFS. You must set up your MC/ServiceGuard cluster before you can configure Highly Available NFS. For instructions on setting up an MC/ServiceGuard cluster, see the *Managing MC/ServiceGuard* manual.

This chapter contains the following sections:

- Installing Highly Available NFS
- Before Creating a Highly Available NFS Package
- Configuring a Highly Available NFS Package

---

## Installing Highly Available NFS

- 1 Use the HP-UX Software Distributor (SD) to install the Highly Available NFS file set. Issue the following command to start the SD **swinstall** utility:

```
/usr/sbin/swinstall
```

The Software Distributor is documented in *Managing HP-UX Software with SD-UX*.

The files will be installed in the `/opt/cmcluster/nfs` directory. The following files are part of the tool kit:

- **README**. Description of the tool kit contents.
- **nfs.cntl**. The control script that runs and halts the package.
- **nfs.conf**. The package configuration file.
- **nfs.mon**. The monitor script.
- **nfs\_xmnt**. A script for handling cross-mounted NFS server packages.

---

### **NOTE:**

If the Highly Available NFS package has previously been installed, the files are in `/opt/cmcluster/toolkit/nfs`. Use **swremove** to remove these files before installing the latest version of Highly Available NFS.

- 2 Create a directory called `/etc/cmcluster/nfs`.
- 3 Issue the following command to copy the Highly Available NFS template files to the newly created `/etc/cmcluster/nfs` directory:

```
cp /opt/cmcluster/nfs/* /etc/cmcluster/nfs
```

## Before Creating a Highly Available NFS Package

Before creating a Highly Available NFS package, perform the following tasks:

- 1 Set up your MC/ServiceGuard cluster according to the instructions in the *Managing MC/ServiceGuard* manual.
- 2 On the primary node and all adoptive nodes for the NFS package, set the **NFS\_SERVER** variable to 1 in the `/etc/rc.config.d/nfsconf` file:

```
NFS_SERVER=1
```

Do not configure the exported directories in the `/etc/exports` file. When an NFS server boots up, it attempts to export all file systems in its `/etc/exports` file. If those file systems are not currently present on the NFS server node, the node cannot boot properly. This happens if the server is an adoptive node for a file system, and the file system is available on the server only after failover of the primary node.

- 3 If your NFS servers must serve PC clients, set the **PCNFS\_SERVER** variable to 1 in the `/etc/rc.config.d/nfsconf` file on the primary node and each adoptive node:

```
PCNFS_SERVER=1
```

If you run the NFS monitor script, setting the **PCNFS\_SERVER** variable to 1 will cause the monitor script to monitor the **pcnfsd** daemon. Then, if the **pcnfsd** daemon fails, your NFS package will fail over to an adoptive node. If you do not want to monitor **pcnfsd**, do not run the NFS monitor script, or set the **PCNFS\_SERVER** variable to 0 and run **pcnfsd** manually from the command line.

- 4 If your NFS servers will also be NFS clients, set the **START\_MOUNTD** variable to 1 in the `/etc/rc.config.d/nfsconf` file on the primary node and each adoptive node:

```
START_MOUNTD=1
```

If you configure `rpc.mountd` in the `/etc/inetd.conf` file, set the `START_MOUNTD` variable to 0. If the `START_MOUNTD` variable is set to 0, the NFS monitor script will not monitor the `rpc.mountd` process. If the `START_MOUNTD` variable is set to 1, and you run the NFS monitor script, your NFS package will fail over to an adoptive node if `rpc.mountd` fails.

- 5 On the primary node and all adoptive nodes for the NFS package, set the `NUM_NFSD` variable in the `/etc/rc.config.d/nfsconf` file to the number of `nfstd` daemons required to support all the NFS packages that could run on that node at once. It is better to run too many `nfstd` processes than too few. In general, you should configure a minimum of four `nfstd` processes and at least two `nfstd` processes for each exported file system. So, for example, if a node is the primary node for a package containing two exported file systems, and it is an adoptive node for another package containing three exported file systems, you should configure it to run at least 10 `nfstd` processes:

```
NUM_NFSD=10
```

- 6 Issue the following command on the primary node and all adoptive nodes to start the NFS server processes:

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

- 7 Configure the disk hardware for high availability. Disks must be protected using HP's MirrorDisk/UX product or an HP High Availability Disk Array with PV links. Data disks associated with highly available NFS must be external disks. All the nodes that support the Highly Available NFS package must have access to the external disks. For most disks, this means that the disks must be attached to a shared bus that is connected to all nodes that support the package. For information on configuring disks, see the *Managing MC/ServiceGuard* manual.
- 8 Use SAM or LVM commands to set up volume groups, logical volumes, and file systems as needed for the data that will be exported to clients.

The names of the volume groups must be unique within the cluster, and the major and minor numbers associated with the volume groups must be the same on all nodes. In addition, the mounting points and exported file system names must be the same on all nodes.

The preceding requirements exist because NFS uses the major number, minor number, inode number, and exported directory as part of a file handle to uniquely identify each NFS file. If differences exist between the primary and adoptive nodes, the client's file handle would no longer point to the correct file

## Installing and Configuring Highly Available NFS

### Before Creating a Highly Available NFS Package

location after movement of the package to a different node.

It is recommended that filesystems used for NFS be created as journalled file systems (FStype vxfs). This ensures the fastest recovery time in the event of a package switch to another node.

- 9 Make sure the user IDs and group IDs of those who access the highly available NFS file system are the same on all nodes that can run the package. Make sure the `/etc/passwd` and `/etc/group` files are the same on the primary node and all adoptive nodes, or use NIS to manage the passwd and group databases. For information on configuring NIS, see the *Installing and Administering NFS Services* manual.
- 10 Create an entry for the name of the package in the DNS or NIS name resolution files, or in `/etc/hosts`, so that users will mount the exported file systems from the correct node. This entry maps the package name to the package's relocatable IP address.
- 11 Decide whether to place executables locally on each client or on the NFS server. There are a number of trade-offs to be aware of regarding the location of executables with Highly Available NFS.

The advantages of keeping executables local to each client are as follows:

- No failover time. If the executables are local to the client, there is no delay if the NFS server fails.
- Faster access to the executables than accessing them through the network.

The advantage of putting the executables on the NFS server is as follows:

- Executable management. If the executables are located in one centralized location, the administrator must update only one copy when changes are made.

If executables are placed on the NFS server, you need to ensure that interrupts are handled correctly in a highly available environment. The client must mount the filesystem using the `nointr` option. This mount option will ensure that the executable continues running correctly on the client after a failover of the server occurs. For example, enter the following command on the NFS client:

```
mount -o nointr relocatable_ip:/usr/src /usr/src
```

where `relocatable_ip` is the IP address of the package, and `/usr/src` represents the mount points of the server and the client, respectively.

## Installing and Configuring Highly Available NFS Before Creating a Highly Available NFS Package

Without the **nointr** option, if an interrupt (or a **SIGKILL**, **SIGHUP**, **SIGINT**, **SIGQUIT**, **SIGTERM**, or **SIGALRM** signal) is sent to an executable while the NFS server is failing over, NFS will terminate the executable. This is a standard feature of NFS that allows interrupts such as ^C to kill a “hung” client executable if the NFS server is down. Specifying the **nointr** option resolves this problem. See the **mount\_nfs**(1M) man page for more information.

## Configuring a Highly Available NFS Package

To configure a highly available NFS package, complete the following tasks, described in this section:

- Copying the Template Files
- Editing the Package Configuration File (`nfs.conf`)
- Editing the Control Script (`nfs.cnt1`)
- Configuring Server-to-Server Cross-Mounts (Optional)
- Creating the MC/ServiceGuard Binary Configuration File

### Copying the Template Files

If you will run only one highly available NFS package in your MC/ServiceGuard cluster, you do not have to copy the template files. However, if you will run multiple highly available NFS packages, each package must have its own package configuration file and control script.

For each highly available NFS package you will run, make a copy of the package configuration file (`nfs.conf`) and the control script (`nfs.cnt1`) with a unique name, as in the following example:

```
cd /etc/cmcluster/nfs
cp nfs.conf nfs1.conf
cp nfs.conf nfs2.conf
cp nfs.cnt1 nfs1.cnt1
cp nfs.cnt1 nfs2.cnt1
```

## Editing the Package Configuration File (nfs.conf)

- 1 Set the **PACKAGE\_NAME** variable.

```
PACKAGE_NAME pkg01
```

You can use the default package name if you will run only one Highly Available NFS package on your MC/ServiceGuard cluster. Each package must have a unique name.

- 2 Create a **NODE\_NAME** variable for each node that will run the package. The first **NODE\_NAME** should specify the primary node. All the **NODE\_NAME** variables following the primary node should specify the adoptive nodes, in the order in which they will be tried.

```
NODE_NAME thyme  
NODE_NAME basil  
NODE_NAME sage
```

- 3 Set the **RUN\_SCRIPT** and **HALT\_SCRIPT** variables to the full path name of the control script (`/etc/cmcluster/nfs/nfs.cnt1` or whatever you have renamed it). You do not have to specify a timeout for either script.

```
RUN_SCRIPT /etc/cmcluster/nfs/nfs1.cnt1  
RUN_SCRIPT_TIMEOUT NO_TIMEOUT  
HALT_SCRIPT /etc/cmcluster/nfs/nfs1.cnt1  
HALT_SCRIPT_TIMEOUT NO_TIMEOUT
```

- 4 If you want to run the NFS monitor script, set the **SERVICE\_NAME** variable to indicate the NFS monitor script:

```
SERVICE_NAME nfs1.monitor
```

Each package must have a unique service name. The **SERVICE\_NAME** variable in the package configuration file must match the **NFS\_SERVICE\_NAME** variable in the control script.

If you do not wish to run the NFS monitor script, comment out the **SERVICE\_NAME** variable:

```
# SERVICE_NAME nfs.monitor
```

## Installing and Configuring Highly Available NFS

### Configuring a Highly Available NFS Package

You do not have to run the NFS monitor script. If your NFS package configuration file specifies **PKG\_SWITCHING\_ENABLED YES** and **NET\_SWITCHING\_ENABLED YES** (the defaults), the package will switch to the next adoptive node or to a standby network interface in the event of a node or network failure. The NFS monitor script causes the package to fail over if any of the monitored NFS services fails. NFS services are monitored if they are configured in the `/etc/rc.config.d/nfsconf` file to be started at system boot. If you set the **NFS\_SERVER** variable to 1, the `nfsd`, `rpc.lockd`, and `rpc.statd` processes are monitored. If you set the **PCNFS\_SERVER** variable to 1, the `pcnfsd` process is monitored. If you set the **START\_MOUNTD** variable to 1, the `rpc.mountd` process is monitored.

- 5 Set the **SUBNET** variable to the subnet that will be monitored for the package, as in the following example:

```
SUBNET 15.13.112.0
```

You can use the default values for the rest of the variables in the package configuration file, or you can change them as needed. For instructions on modifying the default values, see the *Managing MC/ServiceGuard* manual, or read the comments in the `/opt/cmcluster/nfs/nfs.conf` template file.

## Editing the Control Script (nfs.cntl)

- 1 Create a separate **VG[n]** variable for each volume group, as in the following example:

```
VG[0]=/dev/vg01  
VG[1]=/dev/vg02
```

- 2 Create a separate **LV[n]** and **FS[n]** variable for each volume group and file system that will be mounted on the server, as in the following example:

```
LV[0]=/dev/vg01/lvol1;FS[0]=/ha_root  
LV[1]=/dev/vg01/lvol2;FS[1]=/users/scaf  
LV[2]=/dev/vg02/lvol1;FS[2]=/ha_data
```

- 3 Create a separate **XFS[n]** variable for each NFS directory to be exported. Specify the directory name and any export options. Following is an example:

```
XFS[0]="/ha_root"  
XFS[1]="/users/scaf"  
XFS[2]="-o ro /ha_data"
```

Do not configure these exported directories in the `/etc/exports` file. When an NFS server boots up, it attempts to export all file systems in its `/etc/exports` file. If those file systems are not currently present on the NFS server node, the node cannot boot properly. This happens if the server is an adoptive node for a file system, and the file system is available on the server only after failover of the primary node.

- 4 Specify the IP address for the package and the address of the subnet to which the IP address belongs, as in the following example:

```
IP[0]=15.13.114.243  
SUBNET[0]=15.13.112.0
```

The IP address you specify is the relocatable IP address for the package. NFS clients that mount the file systems in the package will use this IP address to identify the server. You should configure a name for this address in the DNS or NIS database, or in the `/etc/hosts` file.

- 5 If you want to run the NFS monitor script, set the **NFS\_SERVICE\_NAME** variable to the value of the **SERVICE\_NAME** variable in the package configuration file, as in the following example. Each package must have a unique

## Installing and Configuring Highly Available NFS

### Configuring a Highly Available NFS Package

service name.

```
NFS_SERVICE_NAME[0]=nfs1.monitor
```

If you do not want to run the NFS monitor script, comment out the `NFS_SERVICE_NAME` and `NFS_SERVICE_CMD` variables:

```
# NFS_SERVICE_NAME[0]=nfs.monitor  
# NFS_SERVICE_CMD[0]=/etc/cmcluster/nfs/nfs.mon
```

By default, the `NFS_SERVICE_NAME` and `NFS_SERVICE_CMD` variables are commented out, and the NFS monitor script is not run.

You do not have to run the NFS monitor script. If your NFS package configuration file specifies `PKG_SWITCHING_ENABLED YES` and `NET_SWITCHING_ENABLED YES` (the defaults), the package will switch to the next adoptive node or to a standby network interface in the event of a node or network failure. The NFS monitor script causes the package to fail over if any of the monitored NFS services fails. NFS services are monitored if they are configured in the `/etc/rc.config.d/nfsconf` file to be started at system boot. If you set the `NFS_SERVER` variable to 1, the `nfsd`, `rpc.lockd`, and `rpc.statd` processes are monitored. If you set the `PCNFS_SERVER` variable to 1, the `pcnfsd` process is monitored. If you set the `START_MOUNTD` variable to 1, the `rpc.mountd` process is monitored.

- 6 If you will run the NFS monitor script, set the `NFS_SERVICE_CMD` variable to the full path name of the NFS monitor script, as in the following example:

```
NFS_SERVICE_CMD[0]=/etc/cmcluster/nfs/nfs.mon
```

The path name for the executable script does not have to be unique to each package. Every package can use the same script. Multiple instances of the monitor script can run on the same node without any problems, and if a package fails over, only the instance associated with that package is killed.

If you do not want to run the NFS monitor script, comment out the `NFS_SERVICE_NAME` and `NFS_SERVICE_CMD` variables:

```
# NFS_SERVICE_NAME[0]=nfs.monitor  
# NFS_SERVICE_CMD[0]=/etc/cmcluster/nfs/nfs.mon
```

By default, the `NFS_SERVICE_NAME` and `NFS_SERVICE_CMD` variables are commented out, and the NFS monitor script is not run.

- 7 If two packages have the same adoptive node, and you want to prevent the adoptive node from adopting both packages at once, specify the `cmmodpkg` command with the package control option (`-d`) in the `customer_defined_run_cmds`, as in the following example:

```
function customer_defined_run_cmds
{
    cmmodpkg -d -n `hostname` pkg02 &
}
```

The package control option can prevent an adoptive node from becoming overloaded when multiple packages fail over. If an adoptive node becomes overloaded, it can fail.

In this example, if a host is an adoptive node for both `pkg01` and `pkg02`, the above `cmmodpkg -d` command, in the control script for `pkg01`, would prevent the host that is running `pkg01` from adopting `pkg02`. A similar line in the control script for `pkg02` could prevent the host that is running `pkg02` from adopting `pkg01`. The ampersand (&) causes the `cmmodpkg` command to run in the background. It must run in the background to allow the control script to finish bringing up the package. The `cmmodpkg` command will hang until the package is up, so it must run in the background to allow the control script to complete.

There is a short time, after one package has failed over but before the `cmmodpkg` command has executed, when the other package can fail over and the host will adopt it. In other words, if two packages fail over at approximately the same time, a host may adopt both packages, even though the package control option is specified.

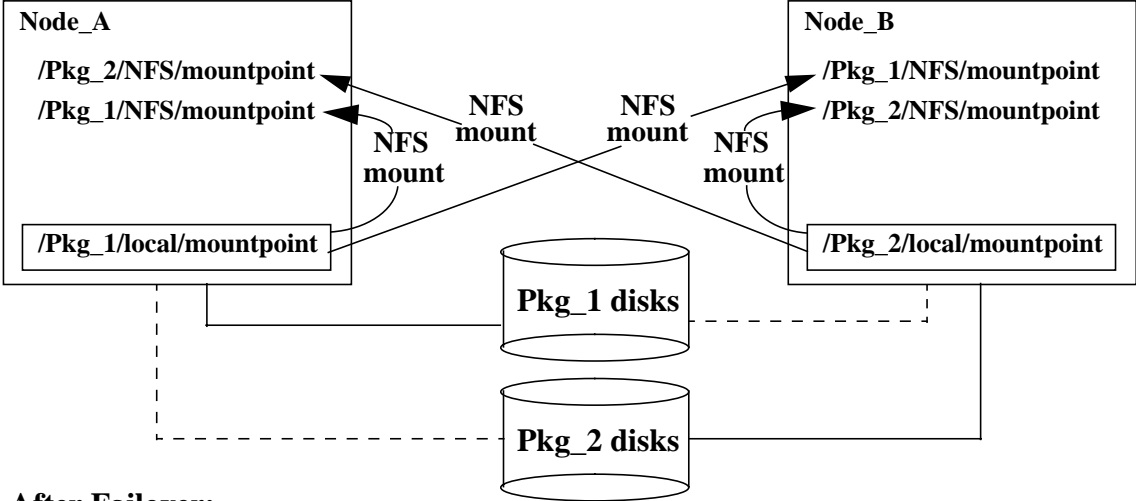
See “One Adoptive Node for Two Packages” on page 50 for a sample configuration using the package control option.

You can use the default values for the rest of the variables in the NFS control script, or you can change them as needed. For instructions on modifying the default values, see the *Managing MC/ServiceGuard* manual, or read the comments in the `/opt/cmcluster/nfs/nfs.cnt1` template file.

### Configuring Server-to-Server Cross-Mounts (Optional)

Two NFS server nodes may NFS-mount each other's file systems and still act as adoptive nodes for each other's NFS server packages. Figure 6 illustrates this configuration.

**Before Failover:**



**After Failover:**

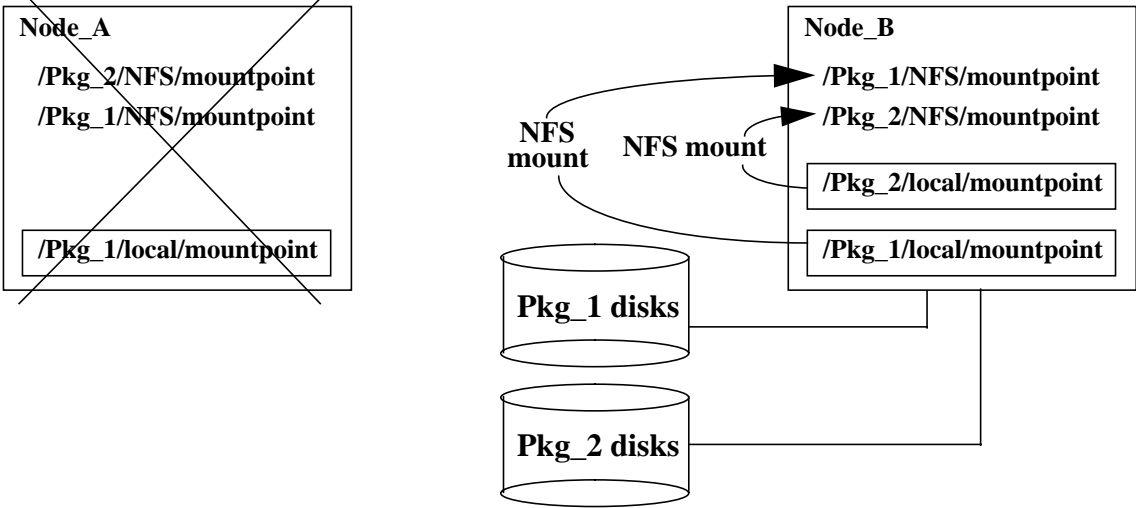


Figure 6

Server-to-Server Cross-Mounting

The advantage of server-to-server cross-mounting is that every server has an identical view of the file systems. The disadvantage is that, on the node where a file system is locally mounted, the file system is accessed through an NFS mount, which has poorer performance than a local mount.

In order to make a highly available file system available to all servers, all servers must NFS-mount the file system. That way, access to the file system is not interrupted when the package fails over to an adoptive node. An adoptive node cannot access the file system through the local mount, because it would have to unmount the NFS-mounted file system before it could mount it locally. And in order to unmount the NFS-mounted file system, it would have to kill all processes using the file system.

Follow these steps to set up an NFS package with file systems that are NFS-mounted by highly available NFS servers:

- 1 Make a copy of the `/etc/cmcluster/nfs/nfs_xmnt` script:

```
cd /etc/cmcluster/nfs
cp nfs_xmnt nfs1_xmnt
```

- 2 In the copy of the `nfs_xmnt` script, create an `SNFS[n]` and `CNFS[n]` variable for each file system in the package that will be NFS-mounted by servers. The `SNFS[n]` variable is the server location of the file system, and the `CNFS[n]` variable is the client mount point of the file system.

```
SNFS[0]="nfs1:/hanfs/nfsu011";CNFS[0]="/nfs/nfsu011"
```

In this example, “`nfs1`” is the name that maps to the package’s relocatable IP address. It must be configured in the name service used by the server (DNS, NIS, or the `/etc/hosts` file).

If a server for the package will NFS-mount the package’s file systems, the client mount point (`CNFS`) *must* be different from the server location (`SNFS`).

- 3 Copy the script you have just modified to all the servers that will NFS-mount the file systems in the package.
- 4 After the package is active on the primary node, execute the `nfs_xmnt` script on each server that will NFS-mount the file systems:

```
/etc/cmcluster/nfs/nfs1_xmnt start
```

## Installing and Configuring Highly Available NFS

### Configuring a Highly Available NFS Package

Hewlett-Packard recommends that you execute the `nfs_xmnt` script from the command line after the package is active on the primary node. However, you can configure the `nfs_xmnt` script to be executed by the NFS control script in the `customer_defined_run_cmds` function, as in the following example:

```
function customer_defined_run_cmds
{
    /etc/cmcluster/nfs/nfs1_xmnt start
    remsh sage /etc/cmcluster/nfs/nfs1_xmnt start
}
```

The second line in the function invokes `remsh` to run the `nfs_xmnt` script on remote host `sage`.

Running the `nfs_xmnt` script from the NFS control script guarantees that the package is active before the `mount` command executes. It prevents cross-mounted servers from becoming deadlocked while each server hangs on the `mount` command, waiting for the other server's package to become active. However, if the package fails to activate, or if the `remsh` command fails, the file systems will not be mounted, and no error will be returned. The only way to be sure the file systems are mounted successfully is to run the `nfs_xmnt` script manually on each host where the file systems should be mounted.

For an example of a configuration with cross-mounted servers, see “Two Servers with NFS Cross-Mounts” on page 64.

## Creating the MC/ServiceGuard Binary Configuration File

- 1 Use the **cmapplyconf** command to verify the content of your cluster and package configuration and to copy the binary configuration file to all the nodes in the cluster. In the following example, the cluster configuration file is **/etc/cmcluster/cluster.conf**, and the NFS package configuration file is **/etc/cmcluster/nfs/nfs.conf**. On your system, use the names of your own cluster configuration and package configuration files.

```
# cmapplyconf -v -C /etc/cmcluster/cluster.conf \  
-P /etc/cmcluster/nfs/nfs.conf
```

- 2 Use your favorite copy utility (for example, **ftp** or **rcp**) to copy the package control and monitor scripts to the same path names on all the nodes in the cluster. For example, to copy the files from host **thyme** to host **basil**, issue the following command from host **thyme**:

```
# rcp /etc/cmcluster/nfs/* basil:/etc/cmcluster/nfs
```



---

**Sample Configurations**

## Sample Configurations

This chapter gives sample cluster configuration files, package configuration files, and control scripts for the following configurations:

- Three-server mutual takeover. This configuration has three servers and three highly available NFS packages. Each server is the primary node for one package and an adoptive node for the other two packages.
- One adoptive node for two packages. This configuration has two packages, each owned by a different server. A third server is the adoptive node for both packages. This sample configuration uses the package control option, which prevents the adoptive node from adopting more than one package at a time.
- Three-server cascading failover. This configuration has three servers and two packages. One server is the primary node for both packages, and the other two servers are adoptive nodes for both packages.
- Two servers with NFS cross-mounts. This configuration has two servers and two packages. The primary node for each package NFS-mounts the file systems from its own package and the other package.

The sample configuration files in this chapter show only the configured values. Most of the comments have been omitted.

## Three-Server Mutual Takeover

This configuration has three servers and three highly available NFS packages. Each server is the primary node for one package and an adoptive node for the other two packages. Figure 7 illustrates this configuration. Dotted lines indicate which servers are adoptive nodes for the packages. Figure 8 illustrates the configuration after host `basil` fails.

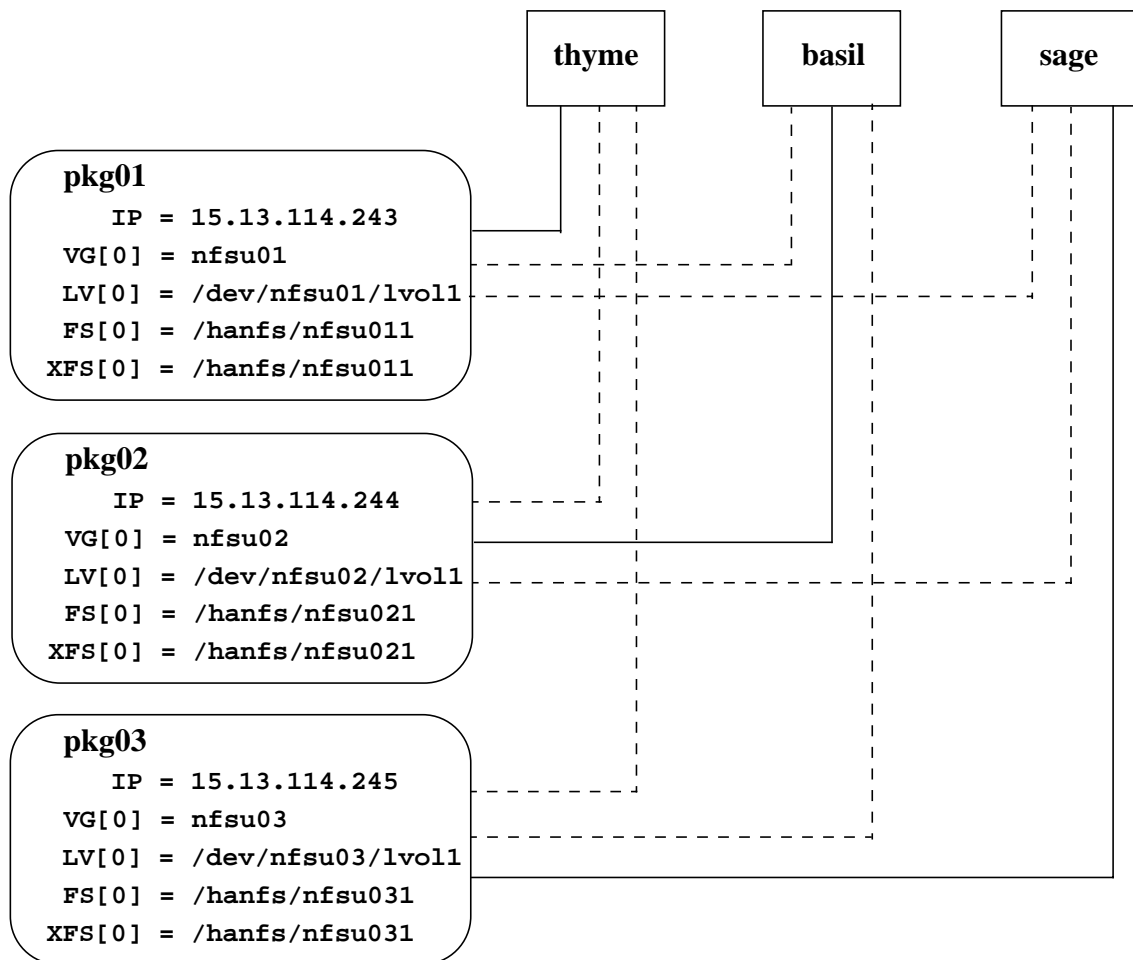


Figure 7

Three-Server Mutual Takeover

Sample Configurations  
Three-Server Mutual Takeover

Figure 8 shows the three-server mutual takeover configuration after host **basil** has failed and host **sage** has adopted **pkg02**. Dotted lines indicate which servers are adoptive nodes for the packages.

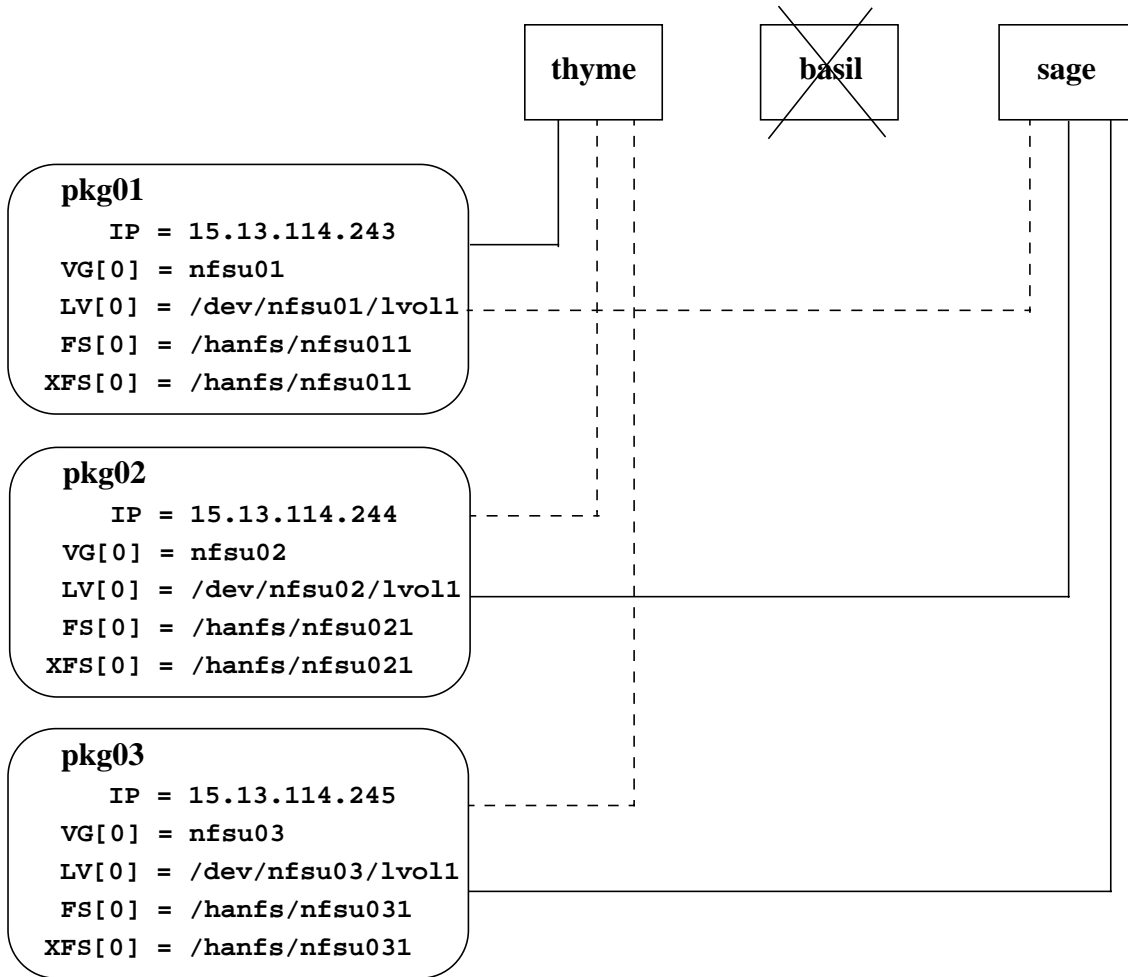


Figure 8

Three-Server Mutual Takeover After One Server Fails

## Cluster Configuration File for Three-Server Mutual Takeover

This section shows the cluster configuration file (`cluster.conf`) for this configuration example. The comments are not shown.

<code>CLUSTER_NAME</code>	<code>MutTakOvr</code>
<code>FIRST_CLUSTER_LOCK_VG</code>	<code>/dev/nfsu01</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.119.146</code>
<code>NETWORK_INTERFACE</code>	<code>lan1</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c0t1d0</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.113.168</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c1t1d0</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.115.184</code>
<code>NETWORK_INTERFACE</code>	<code>lan1</code>
<code>NETWORK_INTERFACE</code>	<code>lan2</code>
<code>NETWORK_INTERFACE</code>	<code>lan3</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c0t1d0</code>
<code>HEARTBEAT_INTERVAL</code>	<code>1000000</code>
<code>NODE_TIMEOUT</code>	<code>2000000</code>
<code>AUTO_START_TIMEOUT</code>	<code>600000000</code>
<code>NETWORK_POLLING_INTERVAL</code>	<code>2000000</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu01</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu02</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu03</code>

### Package Configuration File for pkg01

This section shows the package configuration file (`nfs1.conf`) for the package `pkg01` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg01</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs1.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs1.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs1.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

## NFS Control Script for pkg01

This section shows the NFS control script (`nfs1.cnt1`) for the package `pkg01` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin

#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e" # Default

VG[0]=nfsu01

LV[0]=/dev/nfsu01/lvol1; FS[0]=/hanfs/nfsu011

XFS[0]=/hanfs/nfsu011

IP[0]=15.13.114.243
SUBNET[0]=15.13.112.0

NFS_SERVICE_NAME[0]="nfs1.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=
```

### Package Configuration File for pkg02

This section shows the package configuration file (`nfs2.conf`) for the package `pkg02` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg02</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs2.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs2.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs2.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

## NFS Control Script for pkg02

This section shows the NFS control script (`nfs2.cnt1`) for the package `pkg02` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin

#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e" # Default

VG[0]=nfsu02

LV[0]=/dev/nfsu02/lvol1; FS[0]=/hanfs/nfsu021

XFS[0]=/hanfs/nfsu021

IP[0]=15.13.112.244
SUBNET[0]=15.13.112.0

NFS_SERVICE_NAME[0]="nfs2.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=
```

### Package Configuration File for pkg03

This section shows the package configuration file (`nfs3.conf`) for the package `pkg03` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg03</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs3.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs3.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs3.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

### NFS Control Script for pkg03

This section shows the NFS control script (`nfs3.cnt1`) for the package `pkg03` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin
#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e" # Default
VG[0]=nfsu03
LV[0]=/dev/nfsu03/lvol1; FS[0]=/hanfs/nfsu031
XFS[0]=/hanfs/nfsu031
IP[0]=15.13.114.245
SUBNET[0]=15.13.112.0
NFS_SERVICE_NAME[0]="nfs3.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=
```

---

## One Adoptive Node for Two Packages

This configuration has two packages, each owned by a different server. The adoptive node for both packages is the same host. This sample configuration uses the package control option, which prevents the adoptive node from adopting another package if it has already adopted one. Figure 9 illustrates this configuration. Figure 10 shows the same configuration after one primary server has failed.

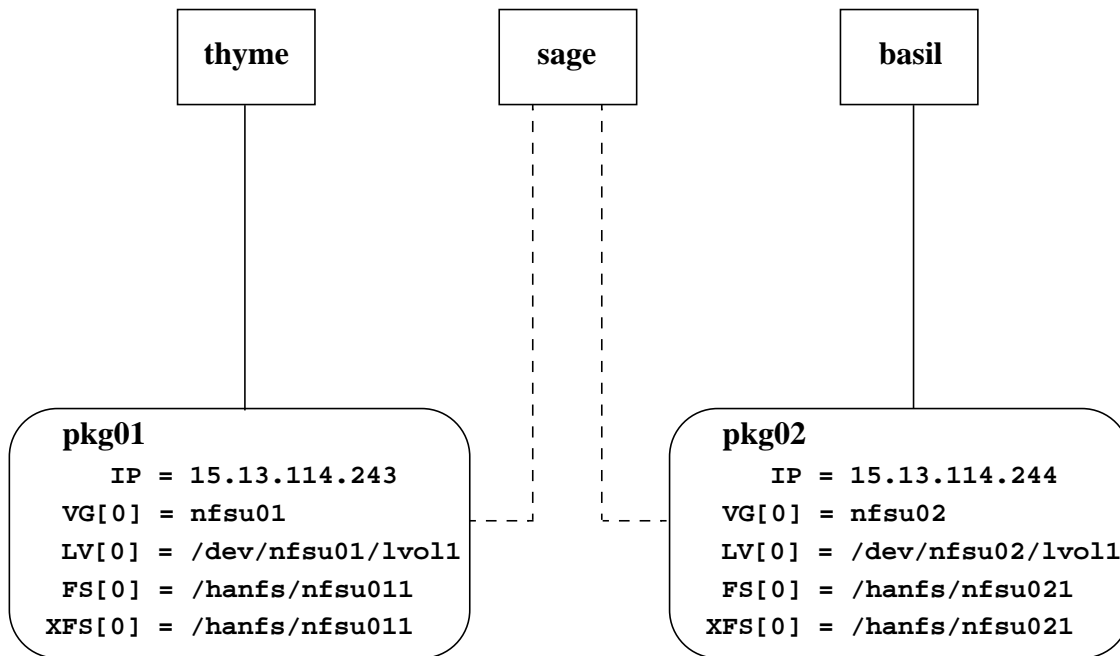


Figure 9 One Adoptive Node for Two Packages

Figure 10 shows this sample configuration after host **basil** has failed. Host **sage** has adopted **pkg02**. The package control option prevents host **sage** from adopting another package, so host **sage** is no longer an adoptive node for **pkg01**.

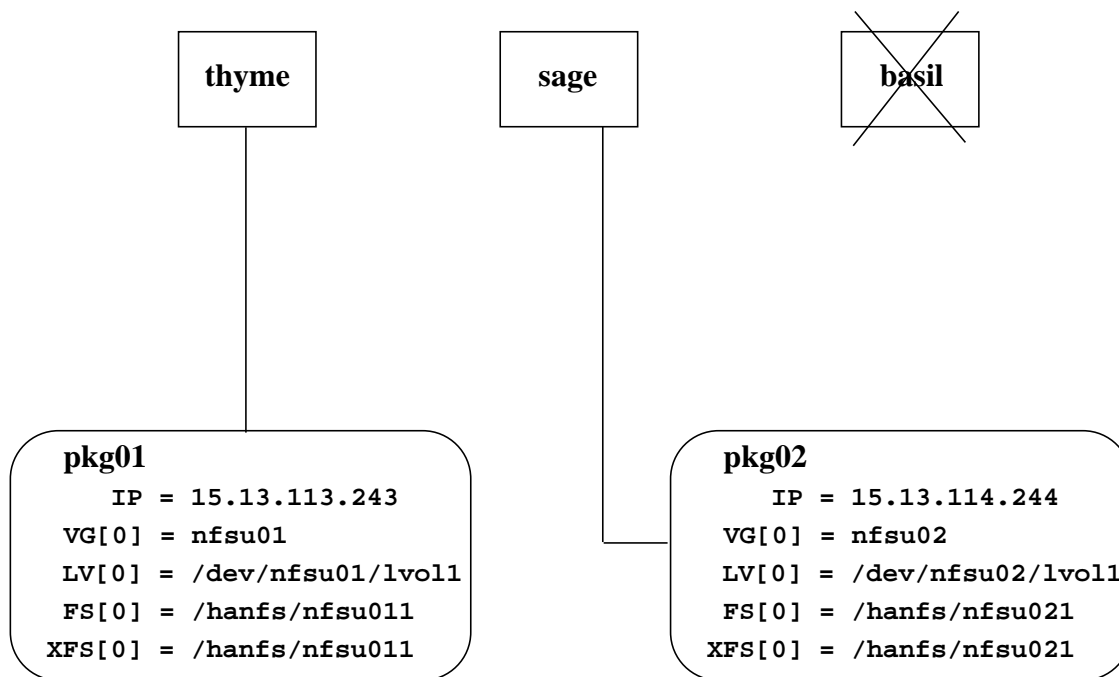


Figure 10 One Adoptive Node for Two Packages After One Server Fails

### Cluster Configuration File for Adoptive Node for Two Packages

This section shows the cluster configuration file (`cluster.conf`) for this configuration example. The comments are not shown.

<code>CLUSTER_NAME</code>	<code>PkgCtrl</code>
<code>FIRST_CLUSTER_LOCK_VG</code>	<code>/dev/nfsu01</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.119.146</code>
<code>NETWORK_INTERFACE</code>	<code>lan1</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c0t1d0</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.113.168</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c1t1d0</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.115.184</code>
<code>NETWORK_INTERFACE</code>	<code>lan1</code>
<code>NETWORK_INTERFACE</code>	<code>lan2</code>
<code>NETWORK_INTERFACE</code>	<code>lan3</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c0t1d0</code>
<code>HEARTBEAT_INTERVAL</code>	<code>1000000</code>
<code>NODE_TIMEOUT</code>	<code>2000000</code>
<code>AUTO_START_TIMEOUT</code>	<code>600000000</code>
<code>NETWORK_POLLING_INTERVAL</code>	<code>2000000</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu01</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu02</code>

### Package Configuration File for pkg01

This section shows the package configuration file (`nfs1.conf`) for the package `pkg01` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg01</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs1.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs1.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs1.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

## NFS Control Script for pkg01

This section shows the NFS control script (`nfs1.cnt1`) for the package `pkg01` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin

#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e"           # Default

VG[0]=nfsu01

LV[0]=/dev/nfsu01/lvol1; FS[0]=/hanfs/nfsu011

XFS[0]=/hanfs/nfsu011

IP[0]=15.13.114.243
SUBNET[0]=15.13.112.0

NFS_SERVICE_NAME[0]="nfs1.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=

function customer_defined_run_cmds
{
    cmmodpkg -d -n `hostname` pkg02 &
}
```

The function `customer_defined_run_cmds` calls the `cmmodpkg` command with the package control option (`-d`). This command prevents the host that is running `pkg01` from adopting `pkg02`. The ampersand (`&`) causes the `cmmodpkg` command to run in the background. It must run in the background to allow the control script to complete.

There is a short time, after one primary node has failed but before the `cmmodpkg` command has executed, when the other primary node can fail and the adoptive node will adopt its package. In other words, if both `thyme` and `basil` fail at approximately the same time, host `sage` may adopt two packages, even though the package control option is specified.

If you omit the `cmmodpkg -d` command from the NFS control script, host `sage` can adopt both `pkg01` and `pkg02` if their primary nodes fail.

### Package Configuration File for pkg02

This section shows the package configuration file (`nfs2.conf`) for the package `pkg02` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg02</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs2.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs2.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs2.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

## NFS Control Script for pkg02

This section shows the NFS control script (`nfs2.cnt1`) for the package `pkg02` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin

#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e"           # Default

VG[0]=nfsu02

LV[0]=/dev/nfsu02/lvol1; FS[0]=/hanfs/nfsu021

XFS[0]=/hanfs/nfsu021

IP[0]=15.13.114.244
SUBNET[0]=15.13.112.0

NFS_SERVICE_NAME[0]="nfs2.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=

function customer_defined_run_cmds
{
    cmmodpkg -d -n `hostname` pkg01 &
}
```

The function `customer_defined_run_cmds` calls the `cmmodpkg` command with the package control option (`-d`). This command prevents the host that is running `pkg02` from adopting `pkg01`. The ampersand (&) causes the `cmmodpkg` command to run in the background. It must run in the background to allow the control script to complete.

There is a short time, after one primary node has failed but before the `cmmodpkg` command has executed, when the other primary node can fail and the adoptive node will adopt its package. In other words, if both `thyme` and `basil` fail at approximately the same time, host `sage` may adopt two packages, even though the package control option is specified.

If you omit the `cmmodpkg -d` command from the NFS control script, host `sage` can adopt both `pkg01` and `pkg02` if their primary nodes fail.

## Three-Server Cascading Failover

This configuration has two packages and three servers. One server is the primary node for both packages. The other servers are adoptive nodes for the two packages. Figure 11 illustrates this configuration. Dotted lines indicate which servers are adoptive nodes for the packages. Figure 12 illustrates the configuration after host `thyme` fails.

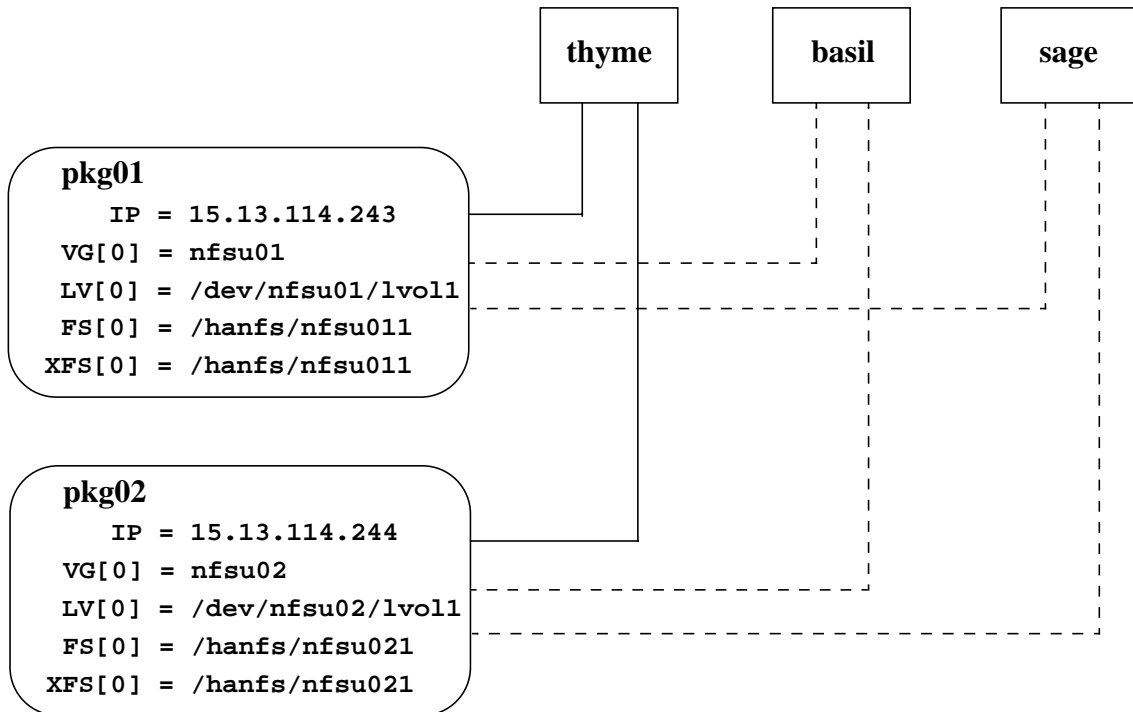


Figure 11

Cascading Failover with Three Servers

Sample Configurations  
Three-Server Cascading Failover

Figure 12 shows the cascading failover configuration after host **thyme** has failed. Host **basil** is the first adoptive node configured for **pkg01**, and host **sage** is the first adoptive node configured for **pkg02**.

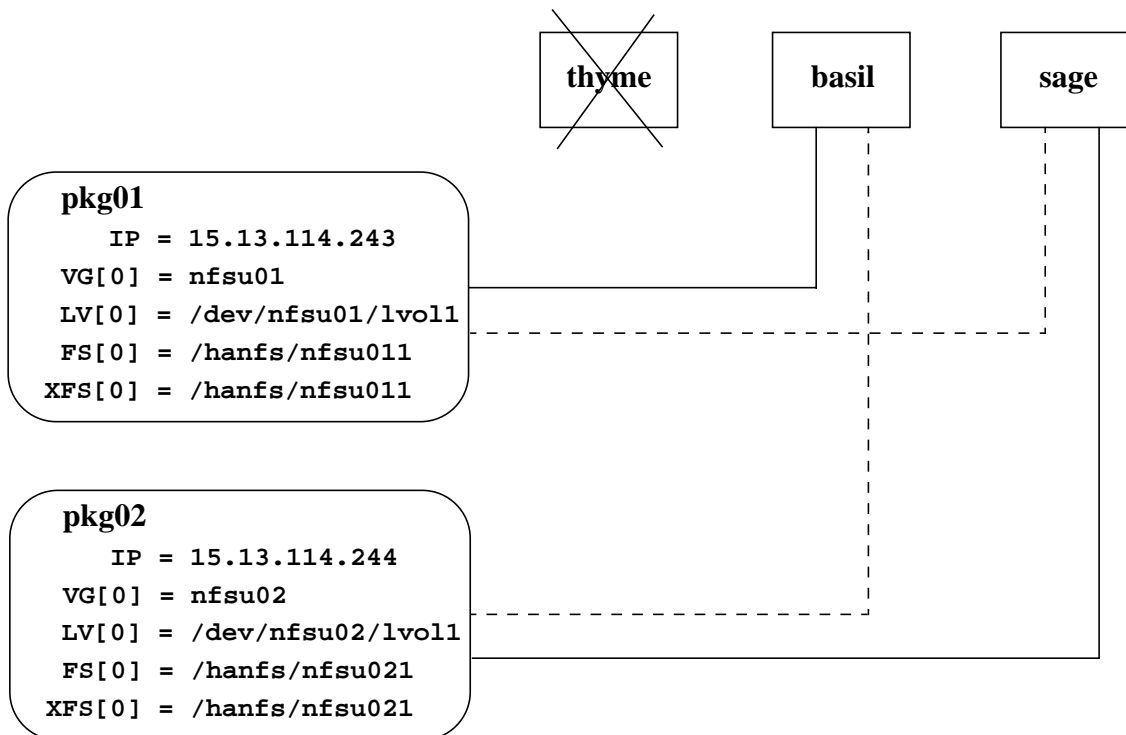


Figure 12 Cascading Failover with Three Servers After One Server Fails

## Cluster Configuration File for Three-Server Cascading Failover

This section shows the cluster configuration file (`cluster.conf`) for this configuration example. The comments are not shown.

<code>CLUSTER_NAME</code>	<code>Cascading</code>
<code>FIRST_CLUSTER_LOCK_VG</code>	<code>/dev/nfsu01</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.119.146</code>
<code>NETWORK_INTERFACE</code>	<code>lan1</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c0t1d0</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.113.168</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c1t1d0</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.115.184</code>
<code>NETWORK_INTERFACE</code>	<code>lan1</code>
<code>NETWORK_INTERFACE</code>	<code>lan2</code>
<code>NETWORK_INTERFACE</code>	<code>lan3</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c0t1d0</code>
<code>HEARTBEAT_INTERVAL</code>	<code>1000000</code>
<code>NODE_TIMEOUT</code>	<code>2000000</code>
<code>AUTO_START_TIMEOUT</code>	<code>600000000</code>
<code>NETWORK_POLLING_INTERVAL</code>	<code>2000000</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu01</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu02</code>

### Package Configuration File for pkg01

This section shows the package configuration file (`nfs1.conf`) for the package `pkg01` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg01</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs1.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs1.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs1.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

## NFS Control Script for pkg01

This section shows the NFS control script (`nfs1.cnt1`) for the package `pkg01` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin
#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e" # Default
VG[0]=nfsu01
LV[0]=/dev/nfsu01/lvol1; FS[0]=/hanfs/nfsu011
XFS[0]=/hanfs/nfsu011
IP[0]=15.13.114.243
SUBNET[0]=15.13.112.0
NFS_SERVICE_NAME[0]="nfs1.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=
```

### Package Configuration File for pkg02

This section shows the package configuration file (`nfs2.conf`) for the package `pkg02` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg02</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NODE_NAME</code>	<code>sage</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs2.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs2.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs2.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

## NFS Control Script for pkg02

This section shows the NFS control script (`nfs2.cnt1`) for the package `pkg02` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin
#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e" # Default
VG[0]=nfsu02
LV[0]=/dev/nfsu02/lvol1; FS[0]=/hanfs/nfsu021
XFS[0]=/hanfs/nfsu021
IP[0]=15.13.114.244
SUBNET[0]=15.13.112.0
NFS_SERVICE_NAME[0]="nfs2.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=
```

## Two Servers with NFS Cross-Mounts

This configuration has two servers and two packages. The primary node for each package NFS-mounts the file systems from its own package and the other package. Figure 13 illustrates this configuration. If one server fails, the other server adopts its package. The NFS mounts are not interrupted when a package fails over. Figure 14 shows the same configuration after one server has failed.

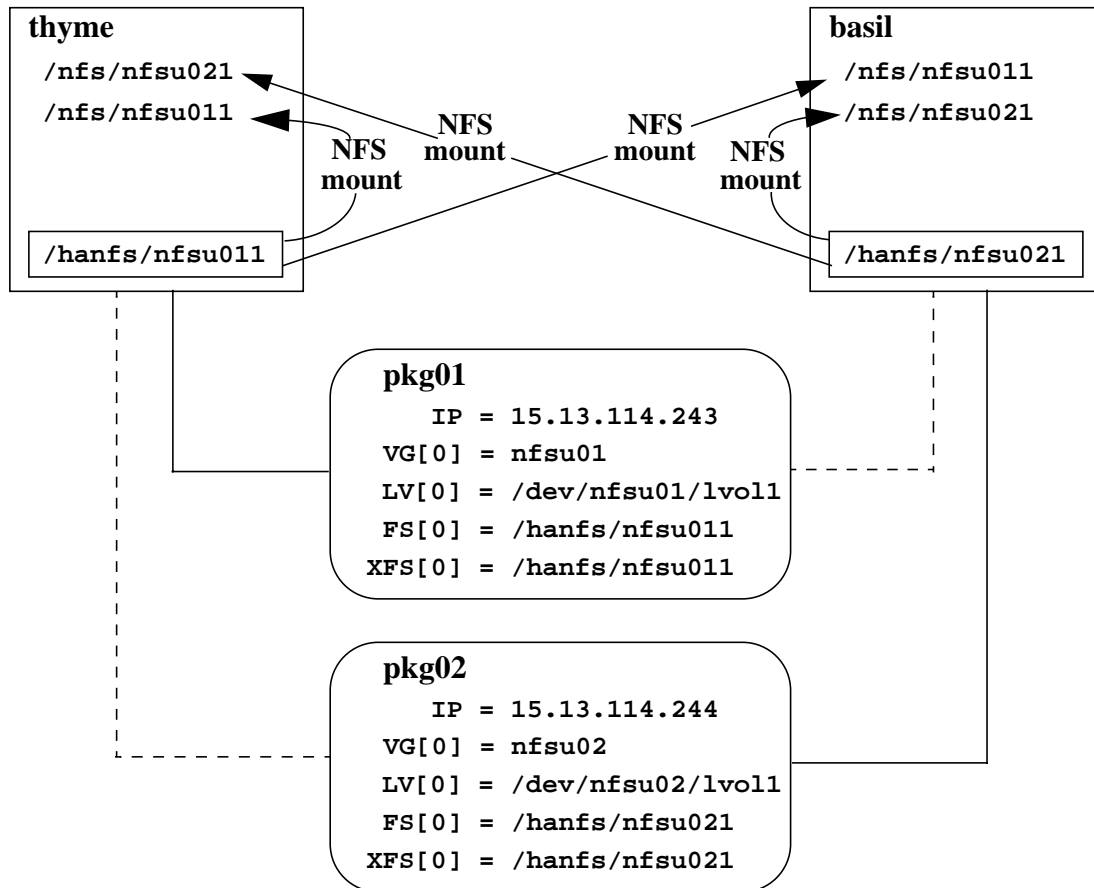


Figure 13

Two Servers with NFS Cross-Mounts

Figure 14 shows two servers with NFS cross-mounted file systems after server **thyme** has failed. The NFS mounts on server **basil** are not interrupted.

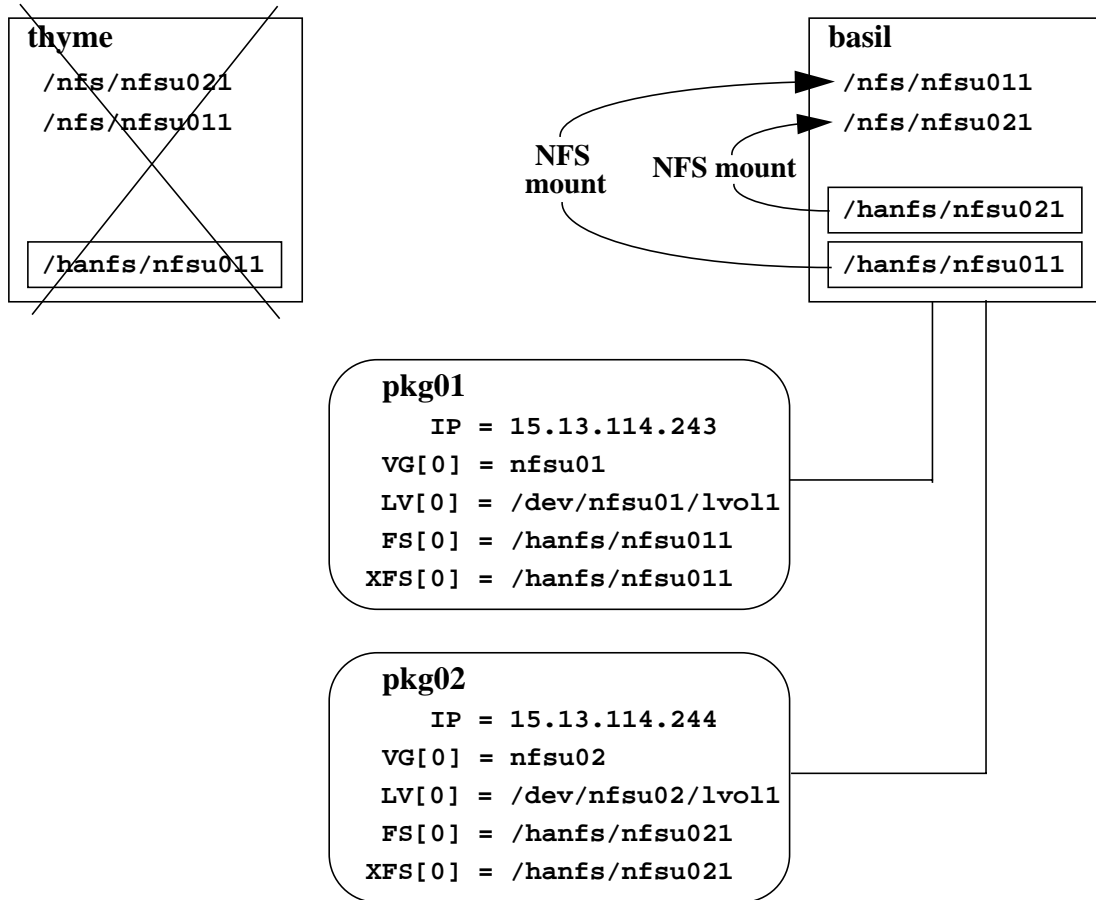


Figure 14

Two Servers with NFS Cross-Mounts

### Cluster Configuration File for Two-Server NFS Cross-Mount

This section shows the cluster configuration file (`cluster.conf`) for this configuration example. The comments are not shown.

<code>CLUSTER_NAME</code>	<code>XMnt</code>
<code>FIRST_CLUSTER_LOCK_VG</code>	<code>/dev/nfsu01</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.119.146</code>
<code>NETWORK_INTERFACE</code>	<code>lan1</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c0t1d0</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NETWORK_INTERFACE</code>	<code>lan0</code>
<code>HEARTBEAT_IP</code>	<code>15.13.113.168</code>
<code>FIRST_CLUSTER_LOCK_PV</code>	<code>/dev/dsk/c1t1d0</code>
<code>HEARTBEAT_INTERVAL</code>	<code>1000000</code>
<code>NODE_TIMEOUT</code>	<code>2000000</code>
<code>AUTO_START_TIMEOUT</code>	<code>600000000</code>
<code>NETWORK_POLLING_INTERVAL</code>	<code>2000000</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu01</code>
<code>VOLUME_GROUP</code>	<code>/dev/nfsu02</code>

### Package Configuration File for pkg01

This section shows the package configuration file (`nfs1.conf`) for the package `pkg01` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg01</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs1.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs1.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs1.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

## NFS Control Script for pkg01

This section shows the NFS control script (`nfs1.cnt1`) for the package `pkg01` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin

#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e" # Default

VG[0]=nfsu01

LV[0]=/dev/nfsu01/lvol1; FS[0]=/hanfs/nfsu011

XFS[0]=/hanfs/nfsu011

IP[0]=15.13.114.243
SUBNET[0]=15.13.112.0

NFS_SERVICE_NAME[0]="nfs1.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=

function customer_defined_run_cmds
{
    /etc/cmcluster/nfs/nfs1_xmnt start
    remsh basil /etc/cmcluster/nfs/nfs1_xmnt start
}
```

The function `customer_defined_run_cmds` calls a script called `nfs1_xmnt`. This script NFS-mounts the file system exported by the package `pkg01`. If you configured the file system in the `/etc/fstab` file, the package might not be active yet when the servers tried to mount the file system at system boot. By configuring the NFS control script to NFS-mount the file system, you ensure that the package is active before the `mount` command is invoked.

The first line in the `customer_defined_run_cmds` function executes the `nfs1_xmnt` script locally on host `thyme` (the primary node for `pkg01`). The second line, beginning with `remsh`, executes the `nfs1_xmnt` script remotely on host `basil`.

If **pkg01** fails to come up, or if the **remsh** to host **basil** fails, the file system will not be mounted, and no error will be returned. The only way to be sure the file system was mounted successfully is to run the **nfs1\_xmnt** script manually on both host **thyme** and host **basil**.

The only user-configurable values in the **nfs1\_xmnt** script are the **SNFS[n]** and **CNFS[n]** variables. These specify the server location of the file system and the client mount point for the file system. The following line is the from the **nfs1\_xmnt** script in this example configuration:

```
SNFS[0]="nfs1:/hanfs/nfsu011"; CNFS[0]="/nfs/nfsu011"
```

In the **SNFS[0]** variable, "**nfs1**" is the name that maps to the relocatable IP address of **pkg01**. It must be configured in the name service the host is using (DNS, NIS, or the **/etc/hosts** file). If you do not want to configure a name for the package, you can just specify the IP address in the **SNFS[0]** variable, as follows:

```
SNFS[0]="15.13.114.243:/hanfs/nfsu011"; CNFS[0]="/nfs/nfsu011"
```

The client mount point, specified in the **CNFS[0]** variable, *must* be different from the location of the file system on the server (**SNFS[0]**).

### Package Configuration File for pkg02

This section shows the package configuration file (`nfs2.conf`) for the package `pkg02` in this sample configuration. The comments are not shown.

<code>PACKAGE_NAME</code>	<code>pkg02</code>
<code>NODE_NAME</code>	<code>basil</code>
<code>NODE_NAME</code>	<code>thyme</code>
<code>RUN_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs2.cnt1</code>
<code>RUN_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>HALT_SCRIPT</code>	<code>/etc/cmcluster/nfs/nfs2.cnt1</code>
<code>HALT_SCRIPT_TIMEOUT</code>	<code>NO_TIMEOUT</code>
<code>SERVICE_NAME</code>	<code>nfs2.monitor</code>
<code>SERVICE_FAIL_FAST_ENABLED</code>	<code>NO</code>
<code>SERVICE_HALT_TIMEOUT</code>	<code>300</code>
<code>SUBNET</code>	<code>15.13.112.0</code>
<code>PKG_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NET_SWITCHING_ENABLED</code>	<code>YES</code>
<code>NODE_FAIL_FAST_ENABLED</code>	<code>NO</code>

## NFS Control Script for pkg02

This section shows the NFS control script (`nfs2.cnt1`) for the package `pkg02` in this sample configuration. Only the user-configured part of the script is shown; the executable part of the script and most of the comments have been omitted.

```
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin

#VGCHANGE="vgchange -a e -q n"
#VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e" # Default

VG[0]=nfsu02

LV[0]=/dev/nfsu02/lvol1; FS[0]=/hanfs/nfsu021

XFS[0]=/hanfs/nfsu021

IP[0]=15.13.114.244
SUBNET[0]=15.13.112.0

NFS_SERVICE_NAME[0]="nfs2.monitor"
NFS_SERVICE_CMD[0]="/etc/cmcluster/nfs/nfs.mon"
#NFS_SERVICE_RESTART[0]=

function customer_defined_run_cmds
{
    /etc/cmcluster/nfs/nfs2_xmnt start
    remsh thyme /etc/cmcluster/nfs/nfs2_xmnt start
}

```

The function `customer_defined_run_cmds` calls a script called `nfs2_xmnt`. This script NFS-mounts the file system exported by the package `pkg02`. If you configured the file system in the `/etc/fstab` file, the package might not be active yet when the servers tried to mount the file system at system boot. By configuring the NFS control script to NFS-mount the file system, you ensure that the package is active before the `mount` command is invoked.

The first line in the `customer_defined_run_cmds` function executes the `nfs2_xmnt` script locally on host `basil` (the primary node for `pkg02`). The second line, beginning with `remsh`, executes the `nfs2_xmnt` script remotely on host `thyme`.

## Sample Configurations

### Two Servers with NFS Cross-Mounts

If **pkg02** fails to come up, or if the **remsh** to host **thyme** fails, the file system will not be mounted, and no error will be returned. The only way to be sure the file system was mounted successfully is to run the **nfs2\_xmnt** script manually on both host **basil** and host **thyme**.

The only user-configurable values in the **nfs2\_xmnt** script are the **SNFS[n]** and **CNFS[n]** variables. These specify the server location of the file system and the client mount point for the file system. The following line is the from the **nfs2\_xmnt** script in this example configuration:

```
SNFS[0]="nfs2:/hanfs/nfsu021"; CNFS[0]="/nfs/nfsu021"
```

In the **SNFS[0]** variable, “**nfs2**” is the name that maps to the relocatable IP address of **pkg02**. It must be configured in the name service the host is using (DNS, NIS, or the **/etc/hosts** file). If you do not want to configure a name for the package, you can just specify the IP address in the **SNFS[0]** variable, as follows:

```
SNFS[0]="15.13.114.244:/hanfs/nfsu021"; CNFS[0]="/nfs/nfsu021"
```

The client mount point, specified in the **CNFS[0]** variable, *must* be different from the location of the file system on the server (**SNFS[0]**).

---

## Index

---

- A**  
adoptive nodes, 10  
  configuring, 29  
  example of cascading failover, 57  
  example of package control option, 50  
  for multiple packages, 10, 13, 33, 50  
  illustration of cascading failover, 14  
automounter timeout, 19
- B**  
binary configuration files, creating, 37
- C**  
cascading failover, 10  
  example configuration, 57  
  illustration of, 14  
client behavior, 8, 19  
cluster configuration file  
  (**cluster.conf**)  
  example, 43, 52, 59, 66  
**cmapplyconf** command, 37  
**cmmodpkg -d** (package control option),  
  33, 54, 56  
**CNFS** variable, in **nfs\_xmmt** script, 35,  
  69, 72  
configuration, 28  
  control script (**nfs.cnt1**), 31  
  cross-mounted servers, 34  
  disks, 25  
  examples, 40  
  illustrations of supported, 11  
  package file (**nfs.conf**), 28  
  prerequisites, 24  
  volume groups and logical volumes, 25  
configuration files, 8, 28  
  copying to all cluster nodes, 37  
  creating binary, 37  
  default values, 30  
  installing new, 23  
  location of, 23  
configurations supported, 10  
control script (**nfs.cnt1**), 17, 28, 31  
  default values, 33  
  example, 45, 47, 49, 54, 56, 61, 63, 68,  
  71  
  specified in **nfs.conf**, 29  
cross-mounted NFS servers  
  configuration example, 64  
  
**customer\_defined\_run\_cmds**, in  
  **nfs.cnt1**, 35, 36, 54, 56, 68, 71
- D**  
**-d** option, **cmmodpkg**, 33, 54, 56  
diskless NFS, 9  
disks, configuring, 25  
DNS, 26, 31, 35, 69, 72  
documentation  
  MC/ServiceGuard, 24  
  Software Distributor (SD-UX), 23
- E**  
**/etc/cmcluster/nfs** directory, 23  
**/etc/exports** file, 24, 31  
**/etc/fstab** file, 68, 71  
**/etc/group** file, 26  
**/etc/hosts** file, 26, 31, 35, 69, 72  
**/etc/passwd** file, 26  
**/etc/rc.config.d/nfsconf** file,  
  18, 24  
executables, where to locate, 26  
exported file systems, 17, 24  
  definition of, 8  
  naming, 25  
  specifying in **nfs.cnt1**, 31
- F**  
file locking, during package failover, 9  
file systems  
  journalled (xvfs), 26  
  mounting, 17  
  specifying in **nfs.cnt1**, 31  
  unmounting, 18  
**FS** variable, in **nfs.cnt1** file, 31
- G**  
group IDs, 26
- H**  
**HALT\_SCRIPT**, in **nfs.conf**, 29  
**HALT\_SCRIPT\_TIMEOUT**, in  
  **nfs.conf**, 29  
Highly Available NFS  
  configuration, 28  
  control script, 17, 28  
  definition of, 8  
  installation, 23
- Highly Available NFS (cont.)  
  limitations, 9  
  location of installed files, 8  
  monitor script, 18  
  package configuration file, 28  
  prerequisites for configuration, 24  
  sample configurations, 40  
  supported configurations, 10  
  hung client, 8, 27
- I**  
**inetd**, starting **rpc.mountd**, 24  
installation, 23  
internet address, for package, 17, 31  
  mapping to logical name, 26  
interruptible NFS mounts, 26  
IP address, for package, 17, 31  
  mapping to logical name, 26  
**IP** variable, in **nfs.cnt1** script, 31
- J**  
journalled file systems (xvfs), 26
- L**  
**lockd**  
  monitoring, 18  
  restarting, 18  
  stopping, 18  
locked files, during package failover, 9  
logging, NFS monitor script, 18  
logical volumes  
  configuration, 25  
  specifying in **nfs.cnt1**, 31  
**LV** variable, in **nfs.cnt1** script, 31
- M**  
MC/ServiceGuard documentation, 24  
monitor script (**nfs.mon**), 18  
  logging, 18  
  specified in **nfs.cnt1**, 32  
  specified in **nfs.conf**, 29  
  starting, 17  
  stopping, 17  
  unconfiguring, 29, 32  
mount points, 25  
mount retry, 19  
**mountd**, starting, 24  
mounting file systems, 17

---

## Index

---

- mutual takeover
  - sample configuration, 41
- N**
- NET\_SWITCHING\_ENABLED**, 30, 32
- NFS client behavior, 8, 19
- NFS control script (**nfs.cnt1**), 17, 28, 31
  - default values, 33
  - example, 45, 47, 49, 54, 56, 61, 63, 68, 71
  - specified in **nfs.conf**, 29
- NFS diskless, 9
- NFS monitor script (**nfs.mon**), 18
  - logging, 18
  - specified in **nfs.cnt1**, 32
  - specified in **nfs.conf**, 29
  - starting, 17
  - stopping, 17
  - unconfiguring, 29, 32
- NFS mount points, 25
- NFS servers
  - cross-mounting NFS file systems, 10, 15, 34, 35, 36, 64
  - definition of, 8
  - diskless, 9
  - multiple active, 10, 41
  - starting, 25
- nfs.cnt1** (control script), 17, 28, 31
  - default values, 33
  - example, 45, 47, 49, 54, 56, 61, 63, 68, 71
  - specified in **nfs.conf**, 29
- nfs.cnt1.log** file, 18
- nfs.conf** package configuration file, 28
  - default values, 30
  - example, 44, 46, 48, 53, 55, 60, 62, 67, 70
- nfs.mon** (monitor script), 18
  - logging, 18
  - specified in **nfs.cnt1**, 32
  - specified in **nfs.conf**, 29
  - unconfiguring, 29, 32
- nfs.server** script, 25
- NFS\_SERVER** variable, 24
- NFS\_SERVICE\_CMD**, in **nfs.cnt1**, 32
- NFS\_SERVICE\_NAME**, in **nfs.cnt1**, 32
- nfs\_xmmt** script, 35, 36, 68, 71
- nfsconf** file, 18, 24
- nfsd** daemons, number of, 25
- NFS-mounted file systems
  - cross-mounted servers, 64
  - interruptible, 26
  - on highly available servers, 10, 15, 34, 35, 36
- NIS, 26, 31, 35, 69, 72
- NODE\_NAME**, in **nfs.conf**, 29
- nointr** option, **mount**, 26
- NUM\_NFSD** variable, 25
- O**
  - /opt/cmcluster/nfs** directory, 23
- P**
  - package configuration file (**nfs.conf**), 28
    - default values, 30
    - example, 44, 46, 48, 53, 55, 60, 62, 67, 70
  - package control option (**cmmodpkg -d**), 33, 54, 56
  - package name, 26
  - PACKAGE\_NAME**, in **nfs.conf**, 29
  - PCNFS\_SERVER** variable, 24
  - pcnfsd**, 18, 24
  - PKG\_SWITCHING\_ENABLED**, 30, 32
- R**
  - retry** option, **mount** command, 19
  - rpc.lockd**
    - monitoring, 18
    - restarting, 18
    - stopping, 18
  - rpc.mountd**, starting, 24
  - rpc.statd**
    - monitoring, 18
    - restarting, 18
    - stopping, 18
  - rpcinfo** command, 18
  - RUN\_SCRIPT**, in **nfs.conf**, 29
  - RUN\_SCRIPT\_TIMEOUT**, in **nfs.conf**, 29
- S**
  - sample configurations, 40
  - SD-UX (Software Distributor), 23
  - servers, multiple active, 10, 41
  - SERVICE\_NAME**, in **nfs.conf**, 29
  - SNFS** variable, in **nfs\_xmmt** script, 35, 69, 72
  - Software Distributor (SD-UX), 23
  - start** parameter, control script, 17
  - START\_MOUNTD** variable, 24
  - statd**
    - monitoring, 18
    - restarting, 18
    - stopping, 18
  - stop** parameter, control script, 17
  - SUBNET** variable
    - in **nfs.cnt1**, 31
    - in **nfs.conf**, 30
  - swinstall** command, 23
  - swremove** command, 23
- T**
  - timeout, automounter, 19
- U**
  - unexporting file systems, 17
  - unmounting file systems, 18
  - user IDs, 26
- V**
  - VG** variable, in **nfs.cnt1** script, 31
  - volume groups
    - activating, 17
    - configuring, 25
    - deactivating, 18
    - major and minor numbers, 25
    - specifying in **nfs.cnt1**, 31
- X**
  - XFS** variable, in **nfs.cnt1** script, 31
  - xvfs (journalled file systems), 26