

Managing Serviceguard NFS for Linux



Manufacturing Part Number : T1442-90014

September 2006

Legal Notices

© Copyright 2006 Hewlett-Packard Development Company, L.P.

Publication Date: 2006

Confidential computer software. Valid license from HP required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Serviceguard® is a registered trademark of Hewlett-Packard Company, and is protected by copyright.

NFS® is a registered trademark of Sun Microsystems, Inc.

NIS™ is a trademark of Sun Microsystems, Inc.

UNIX® is a registered trademark of The Open Group.

Linux® is a registered trademark of Linus Torvalds.

Red Hat® is a registered trademark of Red Hat Software, Inc.

1. Serviceguard NFS for LINUX Introduction

Overview of Serviceguard NFS	6
Limitations of Serviceguard NFS	7
Supported Configurations	8
Failover to an Idle Node	9
Failover between Active Nodes	10
Failover with an Adoptive Node for Multiple Packages	11
Cascading Failover with Multiple Adoptive Nodes	12
Understanding the Serviceguard NFS Files	13
How the Control and Monitor Scripts Work	14
Starting the NFS Services	14
Halting the NFS Services	15
Monitoring the NFS Services	15
Remote mount table synchronization	16
On the Client Side	16

2. Installing and Configuring Serviceguard NFS for Linux

Installing Serviceguard NFS for Linux	18
Copying the Template Files	20
Before Creating an Serviceguard NFS Package	21
Configuring a Serviceguard NFS Package	25
Editing the Package Configuration File (pkg.conf)	25
Editing the Package Control Scripts (pkg.cntl)	27
Editing the NFS Configuration File (hanfs.conf)	30
Editing the NFS Control Script (hanfs.sh)	30
Creating the Serviceguard Binary Configuration File	31
Housekeeping Suggestions	32

3. Sample Configurations

Configuring Multiple Nodes to Support Failover of Multiple Packages	34
Cluster Configuration File for Three-Server Mutual Takeover	36
Package Configuration File for pkg01	37
Package Control Script for pkg01	38
NFS Control Script for pkg01	38
Package Configuration File for pkg02	39
Package Control Script for pkg02	40
NFS Control Script for pkg02	40
Package Configuration File for pkg03	41

Contents

Package Control Script for pkg03	42
NFS Control Script for pkg03	42
Configuring One Adoptive Node to Support Failover of Multiple Packages	43
Cluster Configuration File for Adoptive Node for Two Packages	45
Package Configuration File for pkg01	46
Package Control Script for pkg01	47
NFS Control Script for pkg01	48
Package Configuration File for pkg02	49
Package Control Script for pkg02	50
NFS Control Script for pkg02	51
Configuring Multiple Adoptive Nodes for Cascading Failover	52
Cluster Configuration File for Three-Server Cascading Failover	54
Package Configuration File for pkg01	55
Package Control Script for pkg01	56
NFS Control Script for pkg01	56
Package Configuration File for pkg02	57
Package Control Script for pkg02	58
NFS Control Script for pkg02	58
Index	59

1 Serviceguard NFS for LINUX Introduction

This manual describes how to install and configure an Serviceguard NFS toolkit on a Linux system. Serviceguard NFS® is a toolkit that allows you to use Serviceguard to set up highly available NFS servers.

The information presented in this manual assumes you are familiar with Serviceguard and NFS operations. Refer to your Serviceguard and/or NFS documentation for additional information.

NOTE

You must set up a Serviceguard cluster before you can set up Highly Available NFS. For instructions on setting up an Serviceguard Linux cluster, see the *Managing Serviceguard for Linux* manual.

The NFS server programs must also be installed on your Linux system before you install, configure, and test your NFS package.

Overview of Serviceguard NFS

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.

Serviceguard allows you to create high availability clusters of HP Linux computers (nodes). A high availability computer system allows applications to continue in spite of a hardware or software failure. Serviceguard 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 Serviceguard coordinates the transfer between components.

Serviceguard NFS is a separate set of shell scripts, and a binary file. One shell script (NFS control script) is provided as a template for an NFS server package. Customize this script to meet your specific needs.

In the event of failure, the NFS server package containing the exported file systems moves to a different node in the Serviceguard cluster. After 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 Serviceguard NFS

The following limitations apply to Serviceguard NFS:

- File locks are not maintained when an NFS server package moves to an adoptive 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.
- A system administrator may need to manually maintain or remove persistent file-lock states for the failed node. The previous file-lock states may remain available on the failed node.

Supported Configurations

Serviceguard NFS supports the following configurations and are illustrated in the following sections:

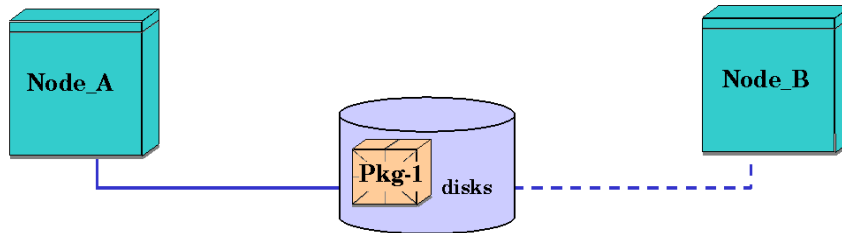
- Simple failover from an active NFS server node to an idle NFS server node.
- Failover from one active NFS server node to another active NFS server node, where the adoptive node supports more than one NFS package after the failover.
- A host configured as an adoptive node for more than one NFS package. 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 several adoptive nodes.

Failover to an Idle Node

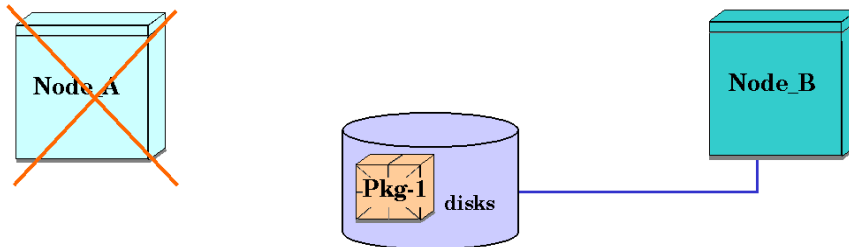
Figure 1-1 shows a simple failover from an active NFS server node to an idle NFS server node.

Figure 1-1 Simple Failover to an Idle NFS Server

Before Failover



After Failover



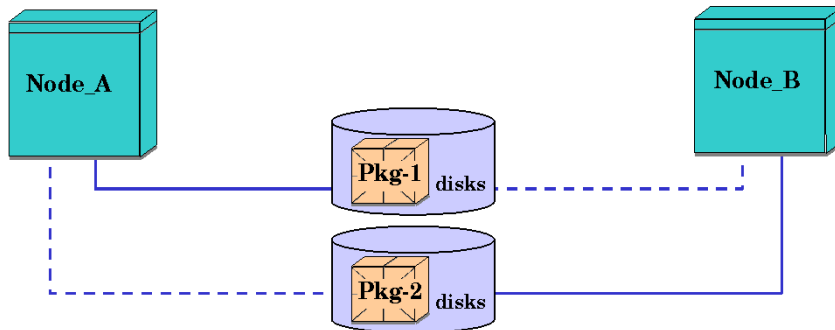
Node_A is the primary node for NFS server package Pkg_1. When Node_A fails, Node_B adopts Pkg_1. This means that Node_B locally mounts the file systems associated with Pkg_1 and exports them. Both Node_A and Node_B must have access to the disks that hold the file systems for Pkg_1.

Failover between Active Nodes

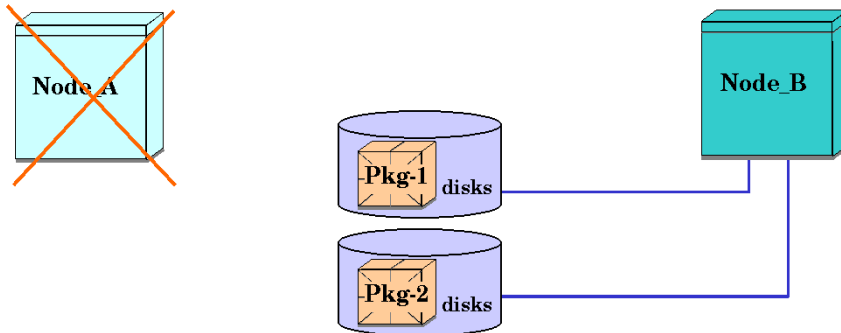
Figure 1-2 shows a failover from one active NFS server node to another active NFS server node. In Figure 1-2, Node_A is the primary node for Pkg_1, and Node_B is the primary node for Pkg_2. When Node_A fails, Node_B adopts Pkg_1 and becomes the server for both Pkg_1 and Pkg_2.

Figure 1-2 Failover from One Active NFS Server to Another

Before Failover



After Failover



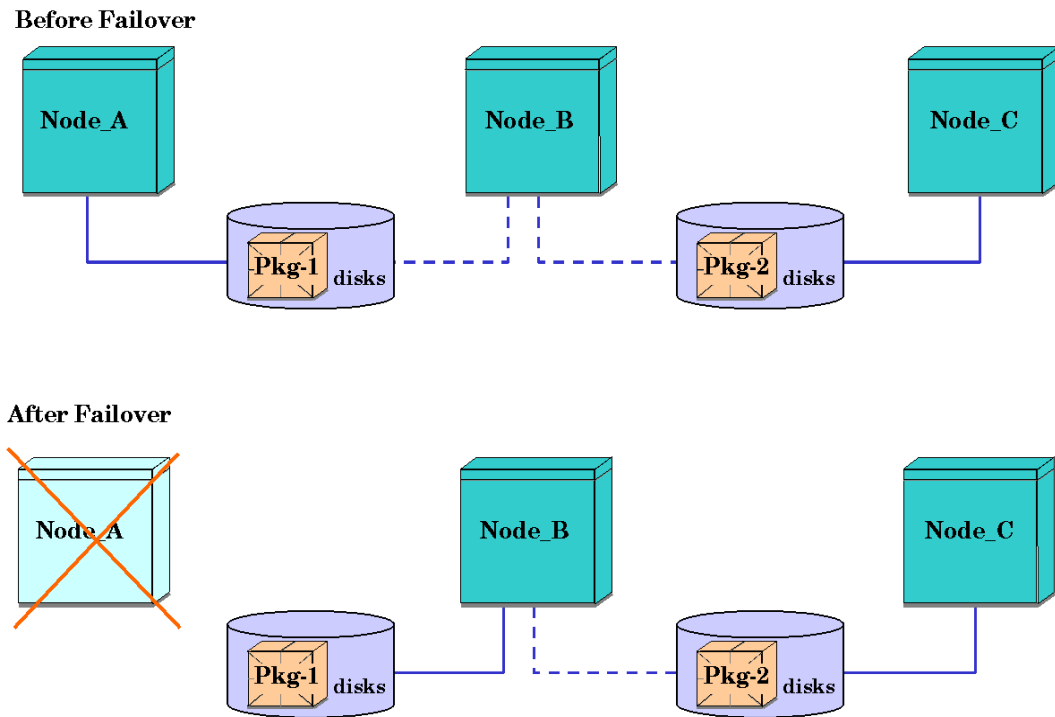
NOTE

During a package failover, the NFS server may receive a status information messages such as: Input/output error, Stale NFS file handle, or Write error: Stale NFS file handle. The result is a retry. If the package fails over during a user file access, the client may experience a momentary hang. Access continues as soon as the package has completed failover to the other node.

Failover with an Adoptive Node for Multiple Packages

Figure 1-3 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. 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.

Figure 1-3 A Host Configured as Adoptive Node for Multiple Packages

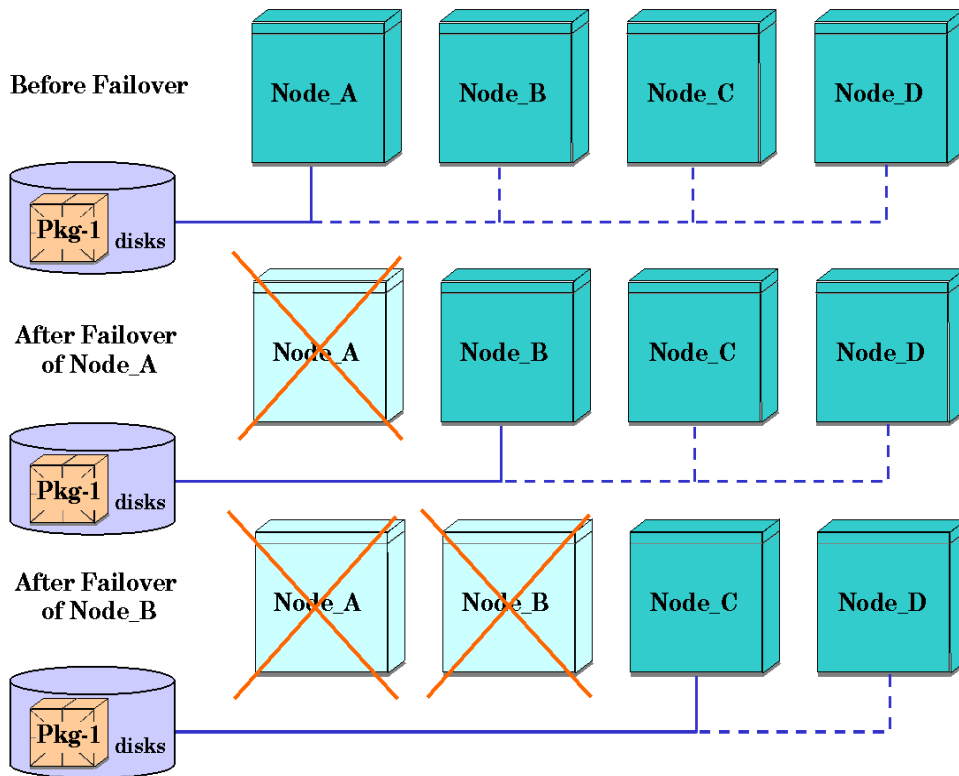


Alternatively, you can prevent Node_B from adopting more than one package at a time by setting a package control function in the package control script. With the package control function set, Node_B may adopt the package of the first node that fails, but if the second node fails, Node_B will not adopt its package. The package control function prevents a node from becoming overloaded by adopting too many packages. If an adoptive node becomes overloaded, it can fail. Refer to “Package Control Script for pkg02” on page 40 for function usage.

Cascading Failover with Multiple Adoptive Nodes

Consider a package that is configured up to three adoptive nodes. Figure 1-4 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, which was generated by using `cmmakepkg -p file` command (see *Managing Serviceguard for Linux*, Chapter 6) 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.

Figure 1-4 Cascading Failover, with Three Adoptive Nodes



Understanding the Serviceguard NFS Files

Serviceguard NFS uses files similar and typical to Serviceguard. These include configuration files, control scripts, monitoring scripts, and templates. As is true for all Serviceguard packages, you configure and view a small number of files. The following is a brief description of the files:

- Files that apply to the whole cluster:
 - cluster configuration file, `cluster.conf`
Defines the cluster nodes, shared networks, and max # of packages
 - `raidtab.sg`
Defines the HA storage devices (MDs) for all the packages.
- Files that apply to each NFS package (<pkg>):
 - package configuration file, `<pkg>.conf`
The master package configuration file. It defines the package's nodes and failover behavior, and points to the package's control script, `pkg.cntl`.
 - package control script, `<pkg>.cntl`
This defines the start and stop behavior, especially the activation and de-activation of the HA storage, and calls the `toolkit.sh` file.
 - Toolkit configuration script, `hanfs.conf`
 - NFS control script, `hanfs.sh`
This exports the HA file system managed by this package. If specified, it also starts monitoring the NFS services.
 - `<pkg.cntl>.log` and `hanfs.sh.log` files
These are generated automatically.
 - NFS monitoring script, `nfs.mon`
 - Toolkit interface script, `toolkit.sh`
This script is invoked by `<pkg>.cntl`, which invokes the `hanfs.sh`.

How the Control and Monitor Scripts Work

As with all Serviceguard packages, the package control scripts start and stop the NFS package and determine how the package will operate once it becomes available on a particular node. Each 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 package control script `pkg.cntl` can be generated by using `cmmakepkg -s pkg.cntl`. The template script `hanfs.sh` is provided in `/usr/local/cmcluster/nfstoolkit` directory for RedHat environments, and `/opt/cmcluster/nfstoolkit` for SLES environments.

Refer to the *Managing Serviceguard for Linux*-Chapter 6 for additional information on creating the files. Refer to “Editing the Package Control Scripts (`pkg.cntl`)” on page 27 and “Editing the NFS Control Script (`hanfs.sh`)” on page 30 for information on how to modify this package control script template file for your own packages.

Starting the NFS Services

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

- Activates MD devices that are used by column groups.
- Activates the volume group or volume groups associated with the package.
- Mounts each file system associated with the package.
- Invoke `toolkit.sh` to run the NFS start script.
- The NFS script `hanfs.sh`, exports each file system associated with the package so that it can later be NFS-mounted by clients.
- The NFS script 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.
- 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.
- The package control script invokes the `toolkit.sh` to run the NFS script and to halt the NFS related process.
- The NFS script un-exports all file systems associated with the package so that they can no longer be NFS-mounted by clients.
- The NFS script halts the monitor process.
- Unmounts each file system associated with the package.
- Deactivates each volume group associated with the package.
- Deactivates each MD device.

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 `nfs.mon`, located in the file `/usr/local/cmcluster/nfstoolkit` for RedHat environments, and `/opt/cmcluster/nfstoolkit` for SLES environments), 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 NFS services including:

- `portmap`
- `rpc.statd`
- `nfsd`
- `rpc.mountd`
- `rpc.rquotad`, if `QUOTO_MON` is set to "YES" in `hanfs.conf`
- `lockd`

If any of the services are dead or hangs, the `nfs.mon` will cause the package to fail.

NOTE

To configure NFS for maximal availability, you must do the following:

- To have the NFS package start automatically when the cluster starts up, and to start on an adoptive node after a failure, you need to specify `AUTO_RUN=YES` in the package configuration file.
- Also, the default NFS control script does not invoke the NFS monitoring script, `nfs.mon`. To invoke this script (see Chapter 3, “Sample Configurations,”) to trigger a failover if one of the package’s NFS services goes down while the node and network remain up.

Whenever the monitor script detects an event, it logs information to a file using the same name as your NFS control script adding a `.log` extension. Each NFS package has its own log file. For example, if your control script is called `pkg1.cntl`, the package log file is called `pkg1.cntl.log`. The NFS monitor log file, which is on the same directory as the NFS control script, is always called `hanfs.sh.log`

Remote mount table synchronization

With NFS toolkit, a remote mount table synchronization binary code is installed in `/usr/bin/sync_rmtab`. This program is provided for synchronizing the client current mount table, `/var/lib/nfs/rmtab`, in the case of a NFS package failover. This synchronization process ensures NFS clients access NFS seamlessly in the case of the NFS package failover. The NFS control script, `hanfs.sh`, calls the synchronization program when the remote mount table needs to be synchronized.

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 auto-mounter. The timeout should be greater than the total end-to-end recovery time for the Serviceguard NFS package—that is, running `fsck`, mounting file systems, and exporting file systems on the new node. 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.

2 Installing and Configuring Serviceguard NFS for Linux

This chapter explains how to configure Serviceguard NFS.

NOTE

You must set up your Serviceguard cluster and make sure NFS server is installed before configuring Serviceguard NFS.

For instructions on setting up an Serviceguard cluster, see Chapters 5 and 6 of the *Managing Serviceguard for Linux* user's guide.

This chapter contains the following sections:

- Installing Serviceguard NFS for Linux
- Before Creating an Serviceguard NFS Package
- Configuring a Serviceguard NFS Package

Installing Serviceguard NFS for Linux

The following describes the Serviceguard NFS for Linux installation process.

NOTE

The following procedures assume you are working in a RedHat environment. If your environment is SLES, replace all occurrences of `/usr/local` with `/opt`.

1. Check for and remove any previous version of Serviceguard NFS for Linux:

Query the rpm database for the NFS Toolkit:

```
# rpm -qa |grep nfstoolkit
```

If any part of the NFS Toolkit is installed, the grep returns the version number.

Erase older versions of the NFS Toolkit, if needed:

```
# rpm -e nfstoolkit<release_version>
```

2. Use the Redhat and SuSE Package Management `rpm` command to install the Serviceguard NFS file set. Change to your RPM file directory, then issue the following commands:

RedHat:

Proliant Servers:

```
# rpm -i nfs-toolkit-A.01.04-0.product.redhat.i386.rpm
```

Integrity Servers:

```
# rpm -i nfs-toolkit-A.01.04-0.product.redhat.ia64.rpm
```

SUSE LINUX Enterprise Server:

Proliant Servers:

```
# rpm -i nfs-toolkit-A.01.04-0.product.suse.i386.rpm
```

Integrity Servers:

```
# rpm -i nfs-toolkit-A.01.04-0.product.suse.ia64.rpm
```

The files will be installed in the `/usr/local/cmcluster/nfstoolkit` and `/usr/bin` directories. The following files are part of the toolkit:

NOTE

The following procedures assume your environment is RedHat. If your environment is SLES, replace all occurrences of `"/usr/local"` with `"/opt"`.

- `/usr/local/cmcluster/nfstoolkit/README`. Description of the toolkit contents.
 - `/usr/local/cmcluster/nfstoolkit/hanfs.conf`. The NFS configuration file which allows the option to monitor the quotad daemon. If `QUOTA_MON` is set to `YES`, the toolkit will start `rpc.quotad` (if not already running) and will monitor this daemon. If it is set to `NO`, the toolkit will not start or monitor `rpc.quotad`. The default value for the `QUOTA_MON` variable is `YES`.
 - `/usr/local/cmcluster/nfstoolkit/hanfs.sh`. The NFS control script template that starts and stops NFS daemons and exports and unexports file systems.
 - `/usr/local/cmcluster/nfstoolkit/nfs.mon`. The NFS monitor script.
 - `/usr/bin/sync_rmtab`. Remote mount table synchronization binary code.
 - `/usr/local/cmcluster/nfstoolkit/toolkit.sh`. The interface script between the package control script and `hanfs.sh`.
3. Run `cmmakepkg` command to generate a package configuration file and package control script template to the `/usr/local/cmcluster/nfstoolkit` directory with the following command:

```
# cd /usr/local/cmcluster/nfstoolkit
# cmmakepkg -p pkg.conf
# cmmakepkg -s pkg.cnt1
```

4. Create a directory for your package files, for example:

```
# mkdir /usr/local/cmcluster/<pkg_name>
```

5. Issue the following command to copy the Serviceguard NFS template files to the newly created package directory:

```
# cp /usr/local/cmcluster/nfstoolkit/* \  
/usr/local/cmcluster/<pkg_name>
```

Copying the Template Files

If you will run only one Serviceguard NFS package in your Serviceguard cluster, technically you do not have to copy the template files. Though, it is recommended that you keep your template file in its original form for future use. If you will run multiple Serviceguard NFS packages, each package *must* have its own package directory, package configuration file and control scripts.

For each Serviceguard NFS package you plan to run, make a copy of all the package files including the package configuration file (`pkg.conf`), package control script (`pkg.cntl`), toolkit interface script (`toolkit.sh`), NFS Control Script (`hanfs.sh`), NFS configuration file (`hanfs.conf`), and NFS monitor script (`nfs.mon`). You can rename the package control script with a package specific identification, such as `pkg1.conf` and `pkg1.cntl`.

NOTE

`pkg.cntl`, `toolkit.sh`, `hanfs.conf`, `nfs.mon`, and `hanfs.sh` should be in the same directory. Do not rename `hanfs.conf`, `hanfs.sh`, `toolkit.sh`, and `nfs.mon`. These files are hard coded in the control scripts.

Before Creating an Serviceguard NFS Package

Before creating a Serviceguard NFS package, perform the following tasks:

NOTE

The following procedures assume your environment is RedHat. If your environment is SLES, replace all occurrences of “/usr/local” with “/opt”.

1. Select the NFS Server package during Red Hat Linux installation and verify that the NFS is properly installed.

After RedHat installation is complete, check for both the NFS kernel and NFS utility to verify NFS installation:

- Verify the NFS utility, run the command:

```
# rpm -qa | grep nfs
```

If the output contains `nfs-utils-<release_version>`, the utility is installed.

- Verify NFS kernel, do the following: (the following steps are for Linux run-level 5)

- Go into your kernel source directory.

For example, if your kernel source directory is `/usr/src/linux-<version>-<release>`, use the following command to change directory

```
# cd /usr/src/linux-<version>-<release>
```

- Run the `xconfig` command.

This displays a kernel configuration window.

- From the configuration window, click the File Systems button.

A File System configuration window is displayed

Before Creating an Serviceguard NFS Package

- From the File Systems window, click the Network File Systems button.
- Verify that both NFS file system support and NFS server support items are checked.

If both are checked, NFS is included in Linux Kernel.

2. Set up your Serviceguard cluster according to the instructions in the *Managing Serviceguard for Linux* user's guide.

3. Configure the disk hardware for high availability. Disks may be protected using disk mirroring or an HP High Availability Disk Array. Data disks associated with Serviceguard NFS must be external disks. All the nodes that support the Serviceguard 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.

Create appropriate multiple device (MD) definitions for your storage. For information on configuring multiple devices, see the *Managing Serviceguard for Linux*, Chapter 5.

4. Use LVM commands to set up volume groups, logical volumes, and file systems as needed for the data that will be exported to clients. Refer to the *Managing Serviceguard* manual.
 - a. Create a directory for each NFS package. For example:

```
/usr/local/cmcluster/nfs1
```

- b. In the directory create a `raidtab.nfs1` file describing this package's MD definitions.

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 location after movement of the package to a different node.

5. Make sure the user IDs and group IDs of those who access the Serviceguard NFS file system are the same on all nodes that can run the package.

Make sure the user IDs and group IDs in 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.

Before Creating an Serviceguard NFS Package

6. 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.
7. 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 Serviceguard 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:

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

Configuring a Serviceguard NFS Package

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

- Editing the Package Configuration File (pkg.conf)
- Editing the Package Control Scripts (pkg.cntl)
- Editing the NFS Configuration File (hanfs.conf)
- Editing the NFS Control Script (hanfs.sh)
- Creating the Serviceguard Binary Configuration File

NOTE

Repeat the configuration process for each NFS package.

Editing the Package Configuration File (pkg.conf)

The following steps describe the required modifications to the Package Configuration File. Make one Package Configuration file for each package.

1. Except for the variables listed below, use the default values for the variables in the package configuration file, or change them as needed.

For instructions on modifying the default values, see the *Managing Serviceguard* manual, or read the comments in the package configuration file.

2. Set the `PACKAGE_NAME` variable. For example:

```
PACKAGE_NAME pkg01
```

You can use the default package name if you are planning to run only one Serviceguard NFS package on your Serviceguard cluster. Each package must have a unique name.

3. 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. For example:

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

4. Set the `RUN_SCRIPT` and `HALT_SCRIPT` variables to the full path name of the control script. You do not have to specify a timeout for either script. For example:

```
RUN_SCRIPT /usr/local/cmcluster/pkg1/pkg1.cnt1  
RUN_SCRIPT_TIMEOUT NO_TIMEOUT  
HALT_SCRIPT /usr/local/cmcluster/pkg1/pkg1.cnt1  
HALT_SCRIPT_TIMEOUT NO_TIMEOUT
```

5. If you want to run the NFS monitor script, set the `SERVICE_NAME` variable. For example:

```
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 NFS control script.

If you do not wish to run the NFS monitor script, comment out the `SERVICE_NAME` variable. For example:

```
SERVICE_NAME nfs.monitor
```

If your NFS package configuration file specifies `AUTO_RUN YES`, the package switches to the next adoptive node in the event of a node or package failure. The NFS monitor script causes the package to fail over if any of the monitored NFS services fails.

6. Set the `SUBNET` variable to the subnet that is monitored for the package. For example:

```
SUBNET 192.100.112.0
```

Editing the Package Control Scripts (pkg.cnt1)

The following steps describe the required modifications to the Package Control Scripts. Make one Package Control Script for each package.

NOTE

The following procedures assume your environment is RedHat. If your environment is SLES, replace all occurrences of “/usr/local” with “/opt”.

1. Specify the configuration file that will be used to define the MD raid devices for each package. For example:

```
RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
```

2. Specify which MD devices are used by this package. For example:

```
MD[0]=/dev/md0  
MD[1]=/dev/md1
```

The MD devices are defined in the RAIDTAB file specified in Step 1.

3. Create a separate VG[n] variable for each volume group. For example:

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

4. Create a separate LV[n], FS[n], FS_TYPE[n], and FS_MOUNT_OPT[n] variable for each volume group and file system that will be mounted on the server. For example:

```
LV[0]=/dev/vg01/lvol1;FS[0]=/ha_root;  
FS_TYPE[0]=ext2;FS_MOUNT_OPT[0]="-o rw"  
LV[1]=/dev/vg01/lvol2;FS[1]=/users/scaf;  
FS_TYPE[0]=ext2;FS_MOUNT_OPT[1]="-o rw"  
LV[2]=/dev/vg02/lvol1;FS[2]=/ha_data;  
FS_TYPE[0]=ext2;FS_MOUNT_OPT[2]="-o rw"
```

This example defines the variable for three NFS mounted file systems, ha_root, users/scaf, and ha_data.

5. Specify the IP address for the package and the address of the subnet to which the IP address belongs. For example:

```
IP[0]=15.13.114.243  
SUBNET[0]=192.100.112.0
```

Configuring a Serviceguard NFS Package

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.

6. Specify that this package uses the high availability NFS server by uncommenting the `HA_APP_SERVER` variable. Uncomment the following line:

```
HA_APP_SERVER="pre-IP"
```

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

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

This package control function 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`.

Add a similar line in the control script for `pkg02` to prevent the host that is running `pkg02` from adopting `pkg01`.

The ampersand (`&`) causes the `cmmodpkg` command to run in the background. The `cmmodpkg` command in the background allows the control script to complete and finish bringing up the package.

There is a small window of time, during which if one package has begun to fail over but the `cmmodpkg` command has not executed, 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 “Configuring One Adoptive Node to Support Failover of Multiple Packages” on page 43 for a sample configuration using the package control option.

8. Use the default values for the rest of the variables in the control script, or change them as needed. For instructions on modifying the default values, see the *Managing Serviceguard for Linux* manual, or read the comments in the `/usr/local/cmcluster/nfstoolkit/pkg.cntl` template file.

Editing the NFS Configuration File (`hanfs.conf`)

The following steps describe the required modifications to the NFS Configuration file:

1. If you want to start and monitor `rpc.quotad` daemon, set `QUOTA_MON` to YES. For example:

```
QUOTA_MON=YES
```

2. If you do not want to start and monitor `rpc.quotad` daemon, set `QUOTA_MON` to NO. For example:

```
QUOTA_MON=NO
```

Editing the NFS Control Script (`hanfs.sh`)

The following steps describe the required modifications to the NFS Control Script.

NOTE

The following procedures assume your environment is RedHat. If your environment is SLES, replace all occurrences of `/usr/local` with `/opt`.

1. Create a separate `XFS[n]` variable for each NFS directory to be exported. Specify the directory name and any export options. For 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.

2. 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. Each package must have a unique service name. For example:

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

- Set the `NFS_SERVICE_CMD` variable to the full path name of the NFS monitor script. For example:

```
NFS_SERVICE_CMD[0]=/usr/local/cmcluster/pkg1/nfs.\  
mon
```

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.

3. If you do not want to run the NFS monitor script:

Comment out the `NFS_SERVICE_NAME` and `NFS_SERVICE_CMD` variables. For example:

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

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

Creating the 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 `/usr/local/cmcluster/cluster.conf`. On your system, use the names of your own cluster configuration and package configuration files.

```
# cmapplyconf -v -C /usr/local/cmcluster/cluster.conf \  
-P /usr/local/cmcluster/pkg1/pkg1.conf
```

Configuring a Serviceguard NFS Package

2. Use your favorite copy utility (for example, `ftp` or `rcp`) to copy the package control, NFS 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 /usr/local/cmcluster/cluster.conf/pkg1/* \  
      basil:/usr/local/cmcluster/cluster.conf/pkg1
```

Housekeeping Suggestions

After the shell scripts are installed they are located in `/usr/local/cmcluster/nfstoolkit` and the binary file is located in `/usr/bin` on your Linux platforms. It is recommended that you set up directories to keep your various package and script files grouped for organization. Set up one directory for each package and keep the associated control and monitoring scripts in that directory.

3 Sample Configurations

This chapter gives sample cluster configuration files, package configuration files, package control script, and NFS control script for configurations supporting the following failover options:

- Failover between multiple active nodes. The sample configuration has three servers and three Serviceguard NFS packages and supports a three-server mutual takeover. Each server is the primary node for one package and an adoptive node for the other two packages.
- One adoptive node for two packages. The sample 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.
- Multiple node cascading failover. The sample 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.

NOTE

Examples in this chapter are RedHat specific

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

Configuring Multiple Nodes to Support Failover of Multiple Packages

This configuration has three servers and three Serviceguard NFS packages. Each server is the primary node for one package and an adoptive node for the other two packages. Figure 3-1 illustrates this configuration. Dotted lines indicate which servers are adoptive nodes for the packages. Figure 3-2 illustrates the configuration after host basil fails.

Figure 3-1 Three-Server Mutual Takeover

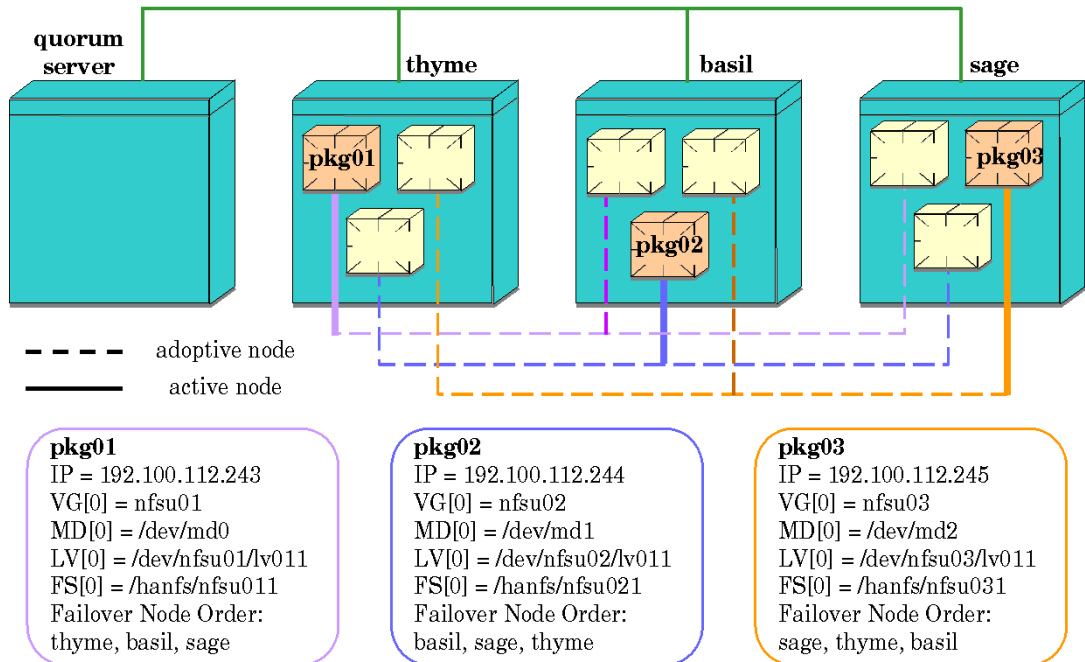
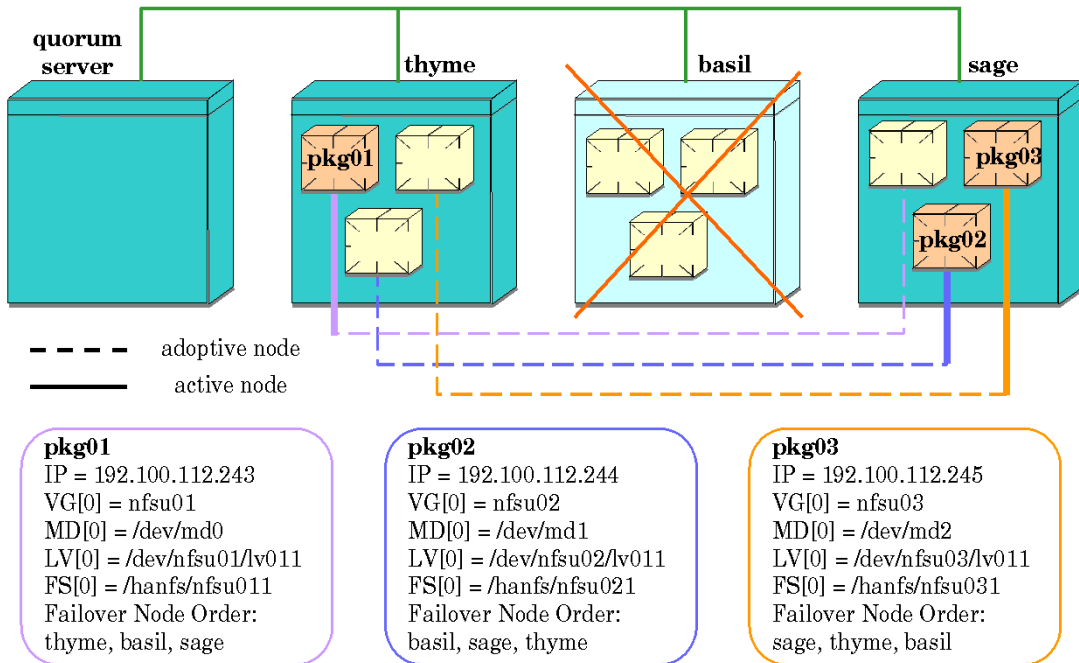


Figure 3-2 shows the three-server mutual takeover configuration after host `basil` has failed and host `sage` has adopted `pkg02`. Dotted lines indicate which servers are adoptive nodes for the packages.

Figure 3-2 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.

```
CLUSTER_NAME                MutTakOvr

QS_HOST                      qs
QS_POLLING_INTERVAL         300000000

NODE_NAME                    thyme
  NETWORK_INTERFACE          eth0
  HEARTBEAT_IP               192.100.112.146
  NETWORK_INTERFACE          eth1

NODE_NAME                    basil
  NETWORK_INTERFACE          eth0
  HEARTBEAT_IP               192.100.112.168

NODE_NAME                    sage
  NETWORK_INTERFACE          eth0
  HEARTBEAT_IP               192.100.112.184
  NETWORK_INTERFACE          eth1
  NETWORK_INTERFACE          eth2

HEARTBEAT_INTERVAL          1000000
NODE_TIMEOUT                 5000000

AUTO_START_TIMEOUT          600000000
NETWORK_POLLING_INTERVAL    2000000

MAX_CONFIGURED_PACKAGES     4
```

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.

```

PACKAGE_NAME                pkg01
PACKAGE_TYPE                 FAILOVER
FAILOVER_POLICY              CONFIGURED_NODE
FAILBACK_POLICY              MANUAL
NODE_NAME                    thyme
NODE_NAME                    basil
NODE_NAME                    sage
AUTO_RUN                     YES
NODE_FAIL_FAST_ENABLED      NO
RUN_SCRIPT                   /usr/local/cmcluster/pkg1/pkg1.conf
RUN_SCRIPT_TIMEOUT           NO_TIMEOUT
HALT_SCRIPT                   /usr/local/cmcluster/pgk1/pkg1.conf
HALT_SCRIPT_TIMEOUT          NO_TIMEOUT
SERVICE_NAME                nfs1.monitor
SERVICE_FAIL_FAST_ENABLED   NO
SERVICE_HALT_TIMEOUT        300
SUBNET                        192.100.112.0

```

Package Control Script for pkg01

This section shows the package control script (`pkg1.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:usr/local/
cmcluster/bin
RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
RAIDSTART="raidstart -c ${RAIDTAB}"
RAIDSTOP="raidstop -c ${RAIDTAB}"
VGCHANGE="vgchange -a y"                # Default
VG[0]="nfsu01"
MD[0]="/dev/md0"
LV[0]="/dev/nfsu01/lvol1;
FS[0]="/hanfs/nfsu011;
FS_TYPE[0]="ext2";
FS_MOUNT_OPT[0]="-o rw"
# FS_UMOUNT_COUNT=" "
# FS_MOUNT_RETRY_COUNT=" "
IP[0]="192.100.112.243"
SUBNET[0]="192.100.112.0"
HA_APP_SERVER="pre-IP"
#SERVICE_NAME[0]=" "
#SERVICE_CMD[0]=" "
#SERVICE_RESTART[0]=" "

```

NFS Control Script for pkg01

This section shows the NFS control script (`hanfs.sh`) for the package `pkg01` on this sample configuration on the user-configured part of the script is shown:

```

XFS[0]="-o rw *:/hanfs/nfsu011"
NFS_SERVICE_NAME[0]="nfs1.monitor"
NFS_SERVICE_CMD[0]="/usr/local/cmcluster/pkg1/nfs.mon"
NFS_SERVICE_RESTART[0]="-r 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.

```
PACKAGE_NAME           pkg02
PACKAGE_TYPE           FAILOVER
FAILOVER_POLICY        CONFIGURED_NODE
FAILBACK_POLICY        MANUAL
NODE_NAME              basil
NODE_NAME              sage
NODE_NAME              thyme
AUTO_RUN               YES
NODE_FAIL_FAST_ENABLED NO
RUN_SCRIPT             /usr/local/cmcluster/pkg2/pkg2.conf
RUN_SCRIPT_TIMEOUT     NO_TIMEOUT
HALT_SCRIPT            /usr/local/cmcluster/pgk2/pkg2.conf
HALT_SCRIPT_TIMEOUT    NO_TIMEOUT
SERVICE_NAME          nfs2.monitor
SERVICE_FAIL_FAST_ENABLED NO
SERVICE_HALT_TIMEOUT  300
SUBNET                 192.100.112.0
```

Package Control Script for pkg02

This section shows the package control script (`pkg2.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:usr/local/
cmcluster/bin
RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
RAIDSTART="raidstart -c ${RAIDTAB}"
RAIDSTOP="raidstop -c ${RAIDTAB}"
VGCHANGE="vgchange -a y"           # Default
VG[0]="nfsu02"
MD[0]="/dev/md1"
LV[0]=/dev/ngsu02/lvol1;
FS[0]=/hanfs/nfsu021;
FS_TYPE[0]="ext2";
FS_MOUNT_OPT[0]="-o rw"
# FS_UMOUNT_COUNT=" "
# FS_MOUNT_RETRY_COUNT=" "
IP[0]="192.100.112.244"
SUBNET[0]="192.100.112.0"
HA_APP_SERVER="pre-IP"
#SERVICE_NAME[0]=" "
#SERVICE_CMD[0]=" "
#SERVICE_RESTART[0]=" "

```

NFS Control Script for pkg02

This section shows the NFS control script (`hanfs.sh`) for the package `pkg02` on this sample configuration on the user-configured part of the script is shown:

```

XFS[0]="-o rw *: /hanfs/nfsu021"
NFS_SERVICE_NAME[0]="nfs2.monitor"
NFS_SERVICE_CMD[0]="/usr/local/cmcluster/pkg2/nfs.mon"
NFS_SERVICE_RESTART[0]="-r 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.

```
PACKAGE_NAME           pkg03
PACKAGE_TYPE           FAILOVER
FAILOVER_POLICY        CONFIGURED_NODE
FAILBACK_POLICY        MANUAL
NODE_NAME              sage
NODE_NAME              thyme
NODE_NAME              basil
AUTO_RUN               YES
NODE_FAIL_FAST_ENABLED NO
RUN_SCRIPT             /usr/local/cmcluster/pkg3/pkg3.conf
RUN_SCRIPT_TIMEOUT     NO_TIMEOUT
HALT_SCRIPT            /usr/local/cmcluster/pgk3/pkg3.conf
HALT_SCRIPT_TIMEOUT    NO_TIMEOUT
SERVICE_NAME          nfs3.monitor
SERVICE_FAIL_FAST_ENABLED NO
SERVICE_HALT_TIMEOUT  300
SUBNET                 192.100.112.0
```

Package Control Script for pkg03

This section shows the NFS control script (`pkg3.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:usr/local/
cmcluster/bin
RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
RAIDSTART="raidstart -c ${RAIDTAB}"
RAIDSTOP="raidstop -c ${RAIDTAB}"
VGCHANGE="vgchange -a y" # Default
VG[0]="nfsu03"
MD[0]="/dev/md2"
LV[0]="/dev/ngsu03/lvol1;"
FS[0]="/hanfs/nfsu031;"
FS_TYPE[0]="ext2";
FS_MOUNT_OPT[0]="-o rw"
# FS_UMOUNT_COUNT=" "
# FS_MOUNT_RETRY_COUNT=" "
IP[0]="192.100.112.245"
SUBNET[0]="192.100.112.0"
HA_APP_SERVER="pre-IP"
#SERVICE_NAME[0]=" "
#SERVICE_CMD[0]=" "
#SERVICE_RESTART[0]=" "

```

NFS Control Script for pkg03

This section shows the NFS control script (`hanfs.sh`) for the package `pkg03` on this sample configuration on the user-configured part of the script is shown:

```

XFS[0]="-o rw */hanfs/nfsu031"
NFS_SERVICE_NAME[0]="nfs3.monitor"
NFS_SERVICE_CMD[0]="/usr/local/cmcluster/pkg3/nfs.mon"
NFS_SERVICE_RESTART[0]="-r 0"

```

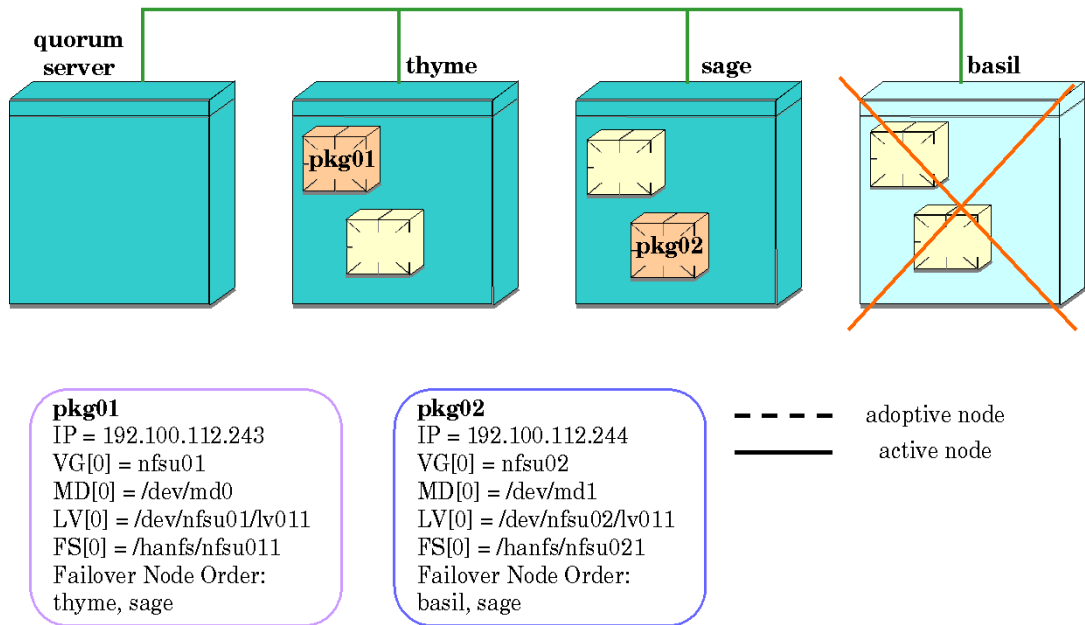

Configuring One Adoptive Node to Support Failover of Multiple Packages

Figure 3-4 shows this sample configuration after host `basil` has failed. Host `sage` has adopted `pkg02`.

NOTE

Setting the package control option (`-d`) of the `cmmodpkg` command, prevents host `sage` from adopting another package, so host `sage` is no longer an adoptive node for `pkg01`. This prevent the adoptive node (`sage`) from becoming overloaded when multiple packages failover.

Figure 3-4 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.

```
CLUSTER_NAME                                PkgCtrl

QS_HOST                                     qs
QS_POLLING_INTERVAL                         300000000

NODE_NAME                                    thyme
  NETWORK_INTERFACE                         eth0
  HEARTBEAT_IP                              192.100.112.146
  NETWORK_INTERFACE                         eth1

NODE_NAME                                    basil
  NETWORK_INTERFACE                         eth0
  HEARTBEAT_IP                              192.100.112.168

NODE_NAME                                    sage
  NETWORK_INTERFACE                         eth0
  HEARTBEAT_IP                              192.100.112.184
  NETWORK_INTERFACE                         eth1
  NETWORK_INTERFACE                         eth2

HEARTBEAT_INTERVAL                         1000000
NODE_TIMEOUT                               5000000

AUTO_START_TIMEOUT                         600000000
NETWORK_POLLING_INTERVAL                   2000000

MAX_CONFIGURED_PACKAGES                    4
```

Package Configuration File for pkg01

This section shows the package configuration file (`pkg1.conf`) for the package `pkg01` in this sample configuration. The comments are not shown.

```
PACKAGE_NAME           pkg01
PACKAGE_TYPE           FAILOVER
FAILOVER_POLICY        CONFIGURED_NODE
FAILBACK_POLICY        MANUAL
NODE_NAME              thyme
NODE_NAME              sage
AUTO_RUN               YES
NODE_FAIL_FAST_ENABLED NO
RUN_SCRIPT             /usr/local/cmcluster/pkg1/pkg1.conf
RUN_SCRIPT_TIMEOUT    NO_TIMEOUT
HALT_SCRIPT            /usr/local/cmcluster/pgk1/pkg1.conf
HALT_SCRIPT_TIMEOUT   NO_TIMEOUT
SERVICE_NAME          nfs1.monitor
SERVICE_FAIL_FAST_ENABLED NO
SERVICE_HALT_TIMEOUT  300
SUBNET                 192.100.112.0
```

Package Control Script for pkg01

This section shows the package control script (`pkg1.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:usr/local/
cmcluster/bin
RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
RAIDSTART="raidstart -c ${RAIDTAB}"
RAIDSTOP="raidstop -c ${RAIDTAB}"
VGCHANGE="vgchange -a y" # Default
VG[0]="nfsu01"
MD[0]="/dev/md0"
LV[0]="/dev/nfsu01/lvol1;"
FS[0]="/hanfs/nfsu011;"
FS_TYPE[0]="ext2";
FS_MOUNT_OPT[0]="-o rw"
# FS_UMOUNT_COUNT=" "
# FS_MOUNT_RETRY_COUNT=" "
IP[0]="192.100.112.243"
SUBNET[0]="192.100.112.0"
HA_APP_SERVER="pre-IP"
#SERVICE_NAME[0]=" "
#SERVICE_CMD[0]=" "
#SERVICE_RESTART[0]=" "
```

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.

NFS Control Script for pkg01

This section shows the NFS control script (`hanfs.sh`) for the package `pkg03` on this sample configuration on the user-configured part of the script is shown:

```
XFS[0]="-o rw *:/hanfs/nfsu011"  
NFS_SERVICE_NAME[0]="nfs1.monitor"  
NFS_SERVICE_CMD[0]="/usr/local/cmcluster/pkg1/nfs.mon"  
NFS_SERVICE_RESTART[0]="-r 0"
```

Package Configuration File for pkg02

This section shows the package configuration file (`pkg2.conf`) for the package `pkg02` in this sample configuration. The comments are not shown.

```
PACKAGE_NAME           pkg02
PACKAGE_TYPE           FAILOVER
FAILOVER_POLICY        CONFIGURED_NODE
FAILBACK_POLICY        MANUAL
NODE_NAME              basil
NODE_NAME              sage
AUTO_RUN               YES
NODE_FAIL_FAST_ENABLED NO
RUN_SCRIPT             /usr/local/cmcluster/pkg2/pkg2.conf
RUN_SCRIPT_TIMEOUT     NO_TIMEOUT
HALT_SCRIPT            /usr/local/cmcluster/pgk2/pkg2.conf
HALT_SCRIPT_TIMEOUT    NO_TIMEOUT
SERVICE_NAME          nfs2.monitor
SERVICE_FAIL_FAST_ENABLED NO
SERVICE_HALT_TIMEOUT  300
SUBNET                 192.100.112.0
```

Package Control Script for pkg02

This section shows the package control script (`pkg2.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:usr/local/
cmcluster/bin
RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
RAIDSTART="raidstart -c ${RAIDTAB}"
RAIDSTOP="raidstop -c ${RAIDTAB}"
VGCHANGE="vgchange -a y" # Default
VG[0]="nfsu02"
MD[0]="/dev/md1"
LV[0]="/dev/nfsu02/lvol1;"
FS[0]="/hanfs/nfsu021;"
FS_TYPE[0]="ext2";
FS_MOUNT_OPT[0]="-o rw"
# FS_UMOUNT_COUNT=" "
# FS_MOUNT_RETRY_COUNT=" "
IP[0]="192.100.112.244"
SUBNET[0]="192.100.112.0"
HA_APP_SERVER="pre-IP"
#SERVICE_NAME[0]=" "
#SERVICE_CMD[0]=" "
#SERVICE_RESTART[0]=" "
```

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.

NFS Control Script for pkg02

This section shows the NFS control script (`hanfs.sh`) for the package `pkg02` on this sample configuration on the user-configured part of the script is shown:

```
XFS[0]="-o rw *:/hanfs/nfsu021"  
NFS_SERVICE_NAME[0]="nfs2.monitor"  
NFS_SERVICE_CMD[0]="/usr/local/cmcluster/pkg2/nfs.mon"  
NFS_SERVICE_RESTART[0]="-r 0"
```

Configuring Multiple Adoptive Nodes for 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 3-5 illustrates this configuration. Dotted lines indicate which servers are adoptive nodes for the packages. Figure 3-6 illustrates the configuration after host `thyme` fails.

Figure 3-5 Cascading Failover with Three Servers

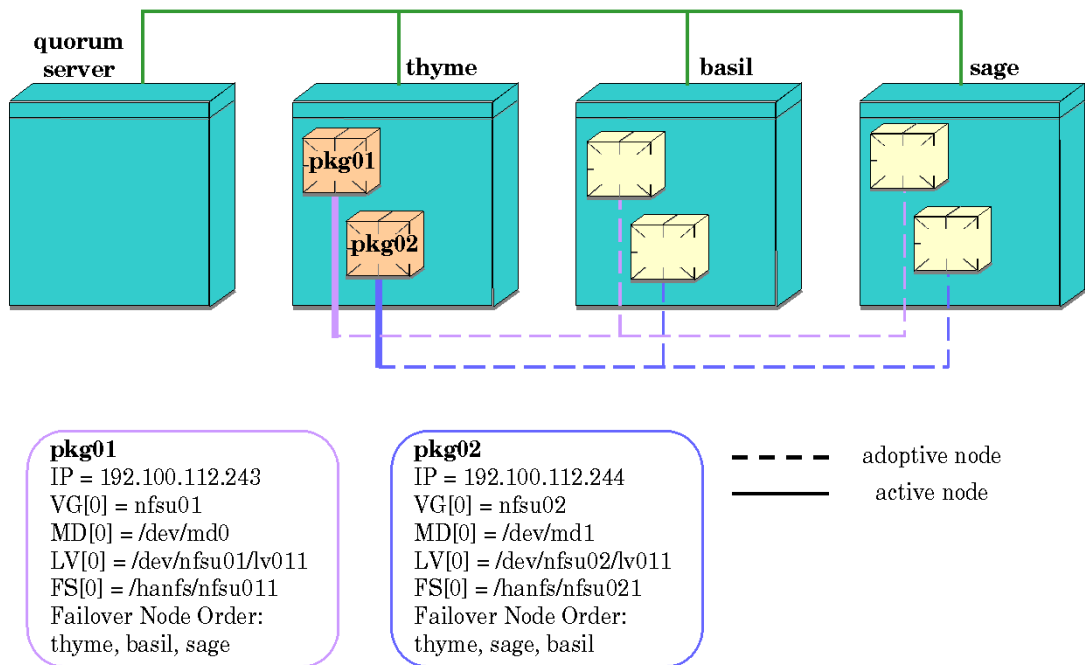
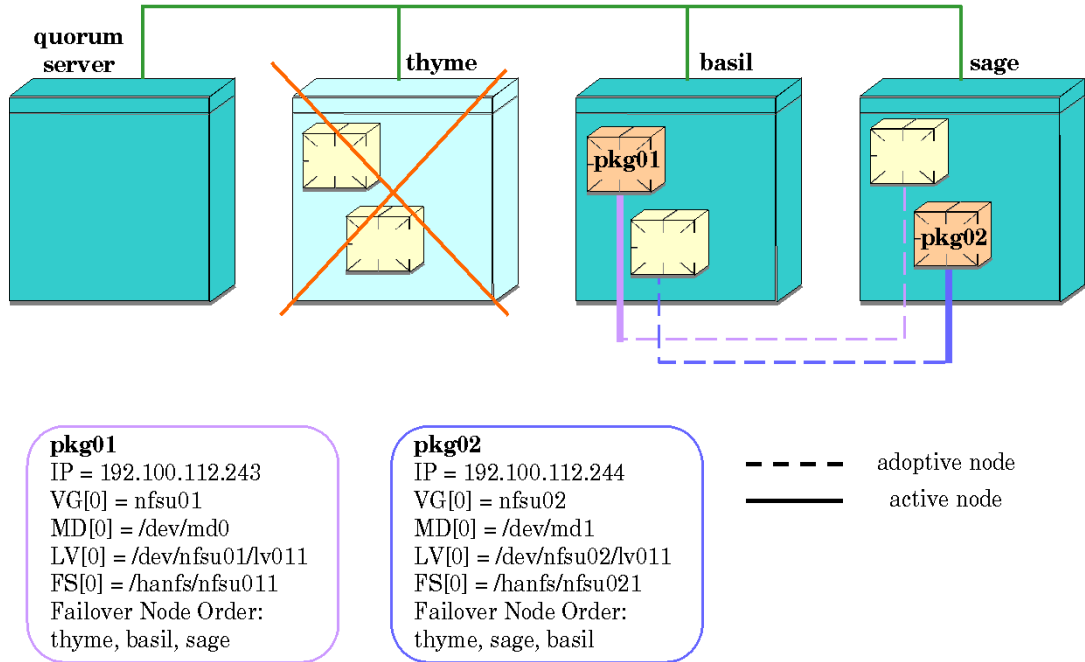


Figure 3-6 shows the cascading failover configuration after host *thyme* has failed. Host *basil* is the first adoptive node configured for *pkg01*, and host *sage* is the first adoptive node configured for *pkg02*.

Figure 3-6 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.

```
CLUSTER_NAME                Cascading

QS_HOST                      qs
QS_POLLING_INTERVAL         300000000

NODE_NAME                    thyme
  NETWORK_INTERFACE          eth0
  HEARTBEAT_IP               192.100.112.146
  NETWORK_INTERFACE          eth1

NODE_NAME                    basil
  NETWORK_INTERFACE          eth0
  HEARTBEAT_IP               192.100.112.168

NODE_NAME                    sage
  NETWORK_INTERFACE          eth0
  HEARTBEAT_IP               192.100.112.184
  NETWORK_INTERFACE          eth1
  NETWORK_INTERFACE          eth2

HEARTBEAT_INTERVAL          1000000
NODE_TIMEOUT                 5000000

AUTO_START_TIMEOUT          600000000
NETWORK_POLLING_INTERVAL    2000000

MAX_CONFIGURED_PACKAGES     4
```

Package Configuration File for pkg01

This section shows the package configuration file (`pkg1.conf`) for the package `pkg01` in this sample configuration. The comments are not shown.

```
PACKAGE_NAME          pkg01
PACKAGE_TYPE          FAILOVER
FAILOVER_POLICY       CONFIGURED_NODE
FAILBACK_POLICY       MANUAL
NODE_NAME             thyme
NODE_NAME             basil
NODE_NAME             sage
AUTO_RUN              YES
NODE_FAIL_FAST_ENABLED NO
RUN_SCRIPT            /usr/local/cmcluster/pkg1/pkg1.conf
RUN_SCRIPT_TIMEOUT    NO_TIMEOUT
HALT_SCRIPT           /usr/local/cmcluster/pgk1/pkg1.conf
HALT_SCRIPT_TIMEOUT   NO_TIMEOUT
SERVICE_NAME         nfs1.monitor
SERVICE_FAIL_FAST_ENABLED NO
SERVICE_HALT_TIMEOUT 300
SUBNET                192.100.112.0
```

Package Control Script for pkg01

This section shows the package control script (`pkg1.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:usr/local/
cmcluster/bin
RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
RAIDSTART="raidstart -c ${RAIDTAB}"
RAIDSTOP="raidstop -c ${RAIDTAB}"
VGCHANGE="vgchange -a y" # Default
VG[0]="nfsu01"
MD[0]="/dev/md0"
LV[0]="/dev/nfsu01/lvol1;"
FS[0]="/hanfs/nfsu011;"
FS_TYPE[0]="ext2";
FS_MOUNT_OPT[0]="-o rw"
# FS_UMOUNT_COUNT=" "
# FS_MOUNT_RETRY_COUNT=" "
IP[0]="192.100.112.243"
SUBNET[0]="192.100.112.0"
HA_APP_SERVER="pre-IP"
#SERVICE_NAME[0]=" "
#SERVICE_CMD[0]=" "
#SERVICE_RESTART[0]=" "
```

NFS Control Script for pkg01

This section shows the NFS control script (`hanfs.sh`) for the package `pkg01` on this sample configuration on the user-configured part of the script is shown:

```
XFS[0]="-o rw */hanfs/nfsu011"
NFS_SERVICE_NAME[0]="nfs1.monitor"
NFS_SERVICE_CMD[0]="/usr/local/cmcluster/pkg1/nfs.mon"
NFS_SERVICE_RESTART[0]="-r 0"
```

Package Configuration File for pkg02

This section shows the package configuration file (`pkg2.conf`) for the package `pkg02` in this sample configuration. The comments are not shown.

```
PACKAGE_NAME          pkg02
PACKAGE_TYPE          FAILOVER
FAILOVER_POLICY       CONFIGURED_NODE
FAILBACK_POLICY       MANUAL
NODE_NAME             thyme
NODE_NAME             sage
NODE_NAME             basil
AUTO_RUN              YES
NODE_FAIL_FAST_ENABLED NO
RUN_SCRIPT            /usr/local/cmcluster/pkg2/pkg2.conf
RUN_SCRIPT_TIMEOUT    NO_TIMEOUT
HALT_SCRIPT           /usr/local/cmcluster/pgk2/pkg2.conf
HALT_SCRIPT_TIMEOUT   NO_TIMEOUT
SERVICE_NAME         nfs2.monitor
SERVICE_FAIL_FAST_ENABLED NO
SERVICE_HALT_TIMEOUT 300
SUBNET                192.100.112.0
```

Package Control Script for pkg02

This section shows the package control script (`pkg2.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:usr/local/
cmcluster/bin
RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
RAIDSTART="raidstart -c ${RAIDTAB}"
RAIDSTOP="raidstop -c ${RAIDTAB}"
VGCHANGE="vgchange -a y" # Default
VG[0]="nfsu02"
MD[0]="/dev/md1"
LV[0]="/dev/nfsu02/lvol1;"
FS[0]="/hanfs/nfsu021;"
FS_TYPE[0]="ext2";
FS_MOUNT_OPT[0]="-o rw"
# FS_UMOUNT_COUNT=" "
# FS_MOUNT_RETRY_COUNT=" "
IP[0]="192.100.112.244"
SUBNET[0]="192.100.112.0"
HA_APP_SERVER="pre-IP"
#SERVICE_NAME[0]=" "
#SERVICE_CMD[0]=" "
#SERVICE_RESTART[0]=" "

```

NFS Control Script for pkg02

This section shows the NFS control script (`hanfs.sh`) for the package `pkg02` on this sample configuration on the user-configured part of the script is shown:

```

XFS[0]="-o rw *: /hanfs/nfsu021"
NFS_SERVICE_NAME[0]="nfs2.monitor"
NFS_SERVICE_CMD[0]="/usr/local/cmcluster/pkg2/nfs.mon"
NFS_SERVICE_RESTART[0]="-r 0"

```

A

adoptive nodes, 8
 configuring, 25
 example of cascading failover, 52
 example of package control option, 43
 for multiple packages, 8, 11, 29, 43
 illustration of cascading failover, 12
AUTO_RUN, 26
automounter timeout, 16

B

binary configuration files, creating, 31

C

cascading failover, 8
 example configuration, 52
 illustration of, 12
client behavior, 6, 16
cluster configuration file (cluster.conf)
 example, 36, 45, 54
cmapplyconf command, 31
cmmmodpkg -d (package control option), 29,
 47, 50
configuration, 20
 control script (nfs.cntnl), 27
 disks, 23
 examples, 33
 illustrations of supported, 9
 package file (nfs.conf), 20
 prerequisites, 21
 volume groups and logical volumes, 23
configuration files, 20, 32
 copying to all cluster nodes, 31
 creating binary, 31
 default values, 25
 location of, 18
configurations supported, 8
control script (nfs.cntnl), 14, 20, 27
 default values, 30
 example, 38, 40, 42, 47, 50, 56, 58
 specified in nfs.conf, 26
customer_defined_run_cmds, in nfs.cntnl, 47,
 50

D

-d option, cmmmodpkg, 29, 47, 50
disks, configuring, 23
DNS, 24, 28
documentation

MC/ServiceGuard, 22

E

/etc/exports file, 31
/etc/group file, 23
/etc/hosts file, 24, 28
/etc/passwd file, 23
/etc/rc.config.d/nfsconf file, 15
executables, where to locate, 24
exported file systems, 14
 definition of, 6
 naming, 23
 specifying in nfs.cntnl, 30

F

file locking, during package failover, 7
file systems
 mounting, 14
 specifying in nfs.cntnl, 27
 unmounting, 15
FS variable, in nfs.cntnl file, 27

G

group IDs, 23

H

HALT_SCRIPT, in nfs.conf, 26
HALT_SCRIPT_TIMEOUT, in nfs.conf, 26
Highly Available NFS
 configuration, 20
 control script, 14, 20
 definition of, 5
 installation, 18
 limitations, 7
 location of installed files, 32
 monitor script, 15
 package configuration file, 20
 prerequisites for configuration, 21
 sample configurations, 33
 supported configurations, 8
hung client, 6

I

installation, 18
internet address, for package, 14, 15, 27
 mapping to logical name, 24
IP address, for package, 14, 15, 27
 mapping to logical name, 24
IP variable, in nfs.cntnl script, 27

Index

L

- lockd
 - monitoring, 15
- locked files, during package failover, 7
- logging, NFS monitor script, 16
- logical volumes
 - configuration, 23
 - specifying in `nfs.cntl`, 27
- LV variable, in `nfs.cntl` script, 27

M

- MC/ServiceGuard documentation, 22
- MD devices
 - activation, 14
 - deactivation, 15
- monitor script (`nfs.mon`), 15
 - logging, 16
 - specified in `nfs.cntl`, 31
 - specified in `nfs.conf`, 26
 - starting, 14
 - stopping, 15
 - unconfiguring, 26, 31
- mount points, 23
- mount retry, 16
- mounting file systems, 14
- mutual takeover
 - sample configuration, 34

N

- NFS client behavior, 6, 16
- NFS control script (`nfs.cntl`), 14, 20, 27
 - default values, 30
 - example, 38, 40, 42, 47, 50, 56, 58
 - specified in `nfs.conf`, 26
- NFS monitor script (`nfs.mon`), 15
 - logging, 16
 - specified in `nfs.cntl`, 31
 - specified in `nfs.conf`, 26
 - starting, 14
 - stopping, 15
 - unconfiguring, 26, 31
- NFS mount points, 23
- NFS servers
 - definition of, 6
 - multiple active, 8, 34
- `nfs.cntl` (control script), 14, 20, 27
 - default values, 30
 - example, 38, 40, 42, 47, 50, 56, 58
 - specified in `nfs.conf`, 26

- `nfs.cntl.log` file, 16
- `nfs.conf` package configuration file, 20
 - default values, 25
 - example, 37, 39, 41, 46, 49, 55, 57
- `nfs.mon` (monitor script), 15
 - logging, 16
 - specified in `nfs.cntl`, 31
 - specified in `nfs.conf`, 26
 - unconfiguring, 26, 31
- `NFS_SERVICE_CMD`, in `nfs.cntl`, 31
- `NFS_SERVICE_NAME`, in `nfs.cntl`, 31
- `nfsconf` file, 15
- NIS, 23, 28
- `NODE_NAME`, in `nfs.conf`, 25

P

- package configuration file (`nfs.conf`), 20
 - default values, 25
 - example, 37, 39, 41, 46, 49, 55, 57
- package control option (`cmmodpkg -d`), 29, 47, 50
- package name, 24
- `PACKAGE_NAME`, in `nfs.conf`, 25
- `pcnfsd`, 15

R

- Red Hat Package Management (RPM), 18
- retry option, mount command, 16
- `rpc.lockd`
 - monitoring, 15
- `rpc.statd`
 - monitoring, 15
- `rpcinfo` command, 15
- `rpm` command, 18
- `RUN_SCRIPT`, in `nfs.conf`, 26
- `RUN_SCRIPT_TIMEOUT`, in `nfs.conf`, 26

S

- sample configurations, 33
- servers, multiple active, 8, 34
- `SERVICE_NAME`, in `nfs.conf`, 26
- start parameter, control script, 14
- `statd`
 - monitoring, 15
- stop parameter, control script, 15
- `SUBNET` variable
 - in `nfs.cntl`, 27
 - in `nfs.conf`, 26

T

timeout, automounter, 16

U

unexporting file systems, 15

unmounting file systems, 15

user IDs, 23

V

VG variable, in `nfs.cntd` script, 27

volume groups

 activating, 14

 configuring, 23

 deactivating, 15

 major and minor numbers, 23

 specifying in `nfs.cntd`, 27

X

XFS variable, in `nfs.cntd` script, 30