IRIX® FailSafe™ NFS Administrator's Guide

007–3949–003

CONTRIBUTORS

Written by Lori Johnson and Anita Manders

Illustrated by Dany Galgani

Edited by Susan Wilkening

Production by Glen Traefald

Engineering contributions by Gilberto Arnaiz, Chander Kant, Bill Sparks, and Paddy Sreenivasan

Cover Design By Sarah Bolles, Sarah Bolles Design, and Dany Galgani, SGI Technical Publications.

New Features in This Guide

This guide supports the IRIX FailSafe NFS 2.2 plug-in release and contains the following changes:

- The new `change-export-options` script allows users to change export options for an NFS filesystem when the resource group is online; see "Changing Export Options", page 17.
- A caution against cross-mounting filesystems using NFS in a FailSafe cluster and using NFS over TCP; see "Planning NFS Filesystems", page 2.
- Information about the `fsid` option in the `/etc/exports` file that allows you to handle some Linux or other operating system NFS clients that are written to interpret or place restrictions on the `fsid` value returned in the attributes. See "Changing Export Options", page 17.
- The start executable timeout for the `statd_unlimited` resource type has been increased to 50 seconds. If you are upgrading from an earlier release, you may want to increase the start executable timeout for the `statd_unlimited` resource type already installed on your system. You can do this by using the **Modify a Resource Type** task the FailSafe GUI or of the `modify resource_type` operation in the `cmgr` command. In FailSafe 2.2, the recovery for NFS locks will take at most 30 seconds, even when NFS clients are not responding. For more information, see "Installing the `NFS` and `statd_unlimited` Resource Types", page 9.
- The guide has been reorganized. Old information has been deleted and new examples have been added.

Record of Revision

Version	Description
001	March 1999 Incorporates information for the IRIS FailSafe 2.0 release.
002	October 2000 Incorporates information for the IRIS FailSafe 2.1 release.
004	January 2003 Incorporates information for the IRIX FailSafe NFS 2.2 release, which supports the IRIX FailSafe 2.1.*x* releases.

Contents

Figures

Tables

About This Guide

This guide provides information about configuring IRIX FailSafe 2.1.*x* systems with the IRIX FailSafe NFS 2.2 plug-in. This plug-in enables NFS resources to be failed over from one node to another if a component fails. This guide is intended as a supplement to the information included in the *IRIS FailSafe Version 2 Administrator's Guide*.

Audience

This guide is written for system administrators who are responsible for configuring and administering an IRIX FailSafe system with the optional IRIX FailSafe NFS software. These system administrators must be able to customize several shell scripts and must be familiar with NFS configuration and NFS startup and shutdown procedures.

This book also assumes that you are familiar with the basic components of FailSafe described in the *IRIS FailSafe Version 2 Administrator's Guide*.

Related Documentation

For NFS installation information, see the *ONC3/NFS Administrator's Guide*.

Besides this guide, other documentation for the IRIS FailSafe system includes:

- *IRIS FailSafe Version 2 Administrator's Guide*
- *IRIS FailSafe Version 2 Programmer's Guide*
- *IRIS FailSafe 2.0 INFORMIX Administrator's Guide*
- *IRIS FailSafe 2.0 Netscape Server Administrator's Guide*
- *IRIS FailSafe 2.0 Oracle Administrator's Guide*
- *IRIS FailSafe Version 2 Samba Administrator's Guide*

- `build_cmgr_script`(1M)
- `cdbBackup`(1M)
- `cdbRestore`(1M)
- `cluster_mgr`(1M)
- `crsd`(1M)
- `failsafe`(7M)
- `fs2d`(1M)
- `ha_cilog`(1M)
- `ha_cmsd`(1M)
- `ha_exec2`(1M)
- `ha_fsd`(1M)
- `ha_gcd`(1M)
- `ha_http_ping2`(1M) (IRIS FailSafe Netscape Web option)
- `ha_ifd`(1M)
- `ha_ifdadmin`(1M)
- `ha_ifmx2`(1M) (IRIS FailSafe INFORMIX option)
- `ha_macconfig2`(1M)
- `ha_srmd`(1M)
- `ha_statd2`(1M)
- `haStatus`(1M)

Release notes are included with each IRIS FailSafe product. The names of the release notes are as follows:

Release Note	Product
`cluster_admin`	Cluster administration services
`cluster_control`	Cluster node control services
`cluster_services`	Cluster services
`failsafe2`	IRIX FailSafe 2.1.*x* release
`failsafe2_informix`	FailSafe INFORMIX
`failsafe2_nfs`	FailSafe NFS
`failsafe2_oracle`	FailSafe Oracle
`failsafe2_samba`	FailSafe Samba
`failsafe2_web`	FailSafe Netscape web
patch_number	FailSafe patch release

Obtaining Publications

You can obtain SGI documentation in the following ways:

- See the SGI Technical Publications Library at: `http://docs.sgi.com`. Various formats are available. This library contains the most recent and most comprehensive set of online books, release notes, man pages, and other information.
- If it is installed on your SGI system, you can use InfoSearch, an online tool that provides a more limited set of online books, release notes, and man pages. With an IRIX system, select **Help** from the Toolchest, and then select **InfoSearch**. Or you can type `infosearch` on a command line.
- You can also view release notes by typing either `grelnotes` or `relnotes` on a command line.
- You can also view man pages by typing `man` *title* on a command line.

Conventions

The following conventions are used throughout this guide:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. (Output is shown in nonbold, fixed-space font.)
[ ]	Brackets enclose optional portions of a command or directive line.
GUI	This font denotes the names of graphical user interface (GUI) elements such as windows, screens, dialog boxes, menus, toolbars, icons, buttons, boxes, fields, and lists.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, contact SGI. Be sure to include the title and document number of the manual with your comments. (Online, the document number is located in the front matter of the manual. In printed manuals, the document number is located at the bottom of each page.)

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

- Use the Feedback option on the Technical Publications Library Web page:

  `http://docs.sgi.com`

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.

- Send mail to the following address:

  Technical Publications
  SGI
  1600 Amphitheatre Parkway, M/S 535
  Mountain View, California 94043–1351

- Send a fax to the attention of "Technical Publications" at +1 650 932 0801.

SGI values your comments and will respond to them promptly.

Introduction

The FailSafe NFS plug-in lets you add NFS to the highly available services that are failed over in a cluster. A *plug-in* is the set of software required to make an application highly available, including a resource type and action scripts. There are plug-ins provided with the base FailSafe release, optional plug-ins available for purchase from SGI, and customized plug-ins you can write using the instructions in the *IRIS FailSafe Version 2 Programmer's Guide*. The optional FailSafe NFS plug-in enables FailSafe to provide failover protection for filesystems and directories that are exported using NFS. The plug-in supports NFS version 2 and version 3.

The `rpc.lockd` and `rpc.statd` daemons manage NFS file locks.

FailSafe operates on the concept of resources. A *resource* can be a filesystem, an IP address, or any entity that can be moved from one node to another when a problem (or scheduled downtime) occurs.

In a FailSafe 2.1.*x* cluster, one or more nodes can export NFS resources. A resource group can contain multiple NFS resources and a single node in the cluster may have multiple resource groups that contain NFS resources. If a node that exports NFS resources fails, another node provides backup service.

The FailSafe NFS plug-in provides the `NFS` and `statd_unlimited` resource types, which includes the following set of action scripts for each resource type:

- `start`, which starts the resource
- `stop`, which stops the resource
- `monitor`, which tests to see if the resource is running
- `exclusive`, which tests to see if the resource is already running
- `restart`, which restarts the resource (however, by default the `NFS` and `statd_unlimited` resource types are not configured to be restarted)

These scripts are found in the following directories:

```
/var/cluster/ha/resource_types/NFS
/var/cluster/ha/resource_types/statd_unlimited
```

You can use the NFS resource type to export the following:

- XFS filesystems by using the filesystem resource type.
- CXFS filesystems (clustered XFS filesystems) by using the CXFS resource type. A CXFS filesystem is accessible in multiple nodes in the cluster but can be exported using NFS from only one node in cluster; therefore, a CXFS resource can appear in only one resource group along with an NFS resource.

Planning NFS Filesystems

Caution: Do not cross-mount filesystems using NFS in a FailSafe cluster (that is, do not mount a locally mounted filesystem on a different node using NFS). This configuration is not reliable and will not work with FailSafe. Instead, you should use the CXFS (clustered XFS) plug-in, which provides this functionality. For more information, see *IRIX FailSafe NFS Administrator's Guide*.

Use of NFS over TCP is not recommended. If the client loses the TCP connection and does not reconnect, it can cause the client to hang on a failover. You should use UDP rather than TCP. If TCP is the default for your NFS clients, you must reconfigure them to use UDP. One method to accomplish this is to create the /etc/config/nfsd.options file with the content -p UDP, which will prevent the server from accepting TCP mount requests.

In a FailSafe cluster, one or more nodes can export NFS filesystems. If a node that exports NFS filesystems fails, the NFS filesystem is failed over to the next node that is listed in the application failover domain for the resource group.

When the NFS server on one node fails, a surviving node must take over the NFS file-locking information from the failed node. FailSafe does this by storing the NFS locking state for each exported filesystem. The NFS lock information for each exported NFS directory is stored in a directory that is in the shared disk. The filesystem that contains the directory should also be in the same resource group. This directory must be named statmon.

The locks held by clients on a filesystem are stored in the statmon directory for each exported filesystem. The statmon directory is maintained in the shared disk, along with the exported filesystem or directory.

NFS lock failover is required if the application running on the client using the NFS filesystem obtains file locks. In this case, you must also provide a resource of type `statd_unlimited`. (The resource type `statd_unlimited` replaces the old `statd` resource type. It provides NFS lock failover functionality but is not restricted to two resource groups per cluster, as was `statd`.) The resources that are type `statd_unlimited` can be present in any number of resource groups in the FailSafe cluster.

Do not place NFS filesystems on shared disks in the `/etc/exports` file. FailSafe exports these filesystems only after ensuring that another node does not have these filesystems exported.

Note: The use of the `wsync` option is applicable only to NFS version 2, because NFS version 3 implements a safe, asynchronous write.

You can reconfigure NFS exports and XFS local mounts on the node to perform writes synchronously to disk (when received) by adding the parameter `wsync` to the `mode` parameter in all filesystem blocks and to the `export-info` parameter in all NFS blocks. In this case, a reply to an NFS write will not be returned until the NFS write data is written to the server's disk. However, performance can be greatly affected by adding the `wsync` option in the filesystem blocks and in the NFS blocks. You must balance the risk of NFS data corruption during a node failure against the performance gain from using asynchronous NFS writes. Some applications perform their own error checking of data or perform writes in which data corruption does not occur.

Required Software

The required software for NFS failover is as follows:

- NFS software. See the *ONC3/NFS Administrator's Guide* for more information about NFS.
- Base IRIS FailSafe software. See the *IRIS FailSafe Version 2 Administrator's Guide* for a complete list of required base software.
- IRIX FailSafe NFS software:
  - `failsafe2_nfs.books.book_AG` contains this book.
  - `failsafe2_nfs.man.man` contains the man pages.

- `failsafe2_nfs.man.relnotes` contains the release notes.
- `failsafe2_nfs.sw.base` contains the base software.

Overview of the Configuration Process

To configure a FailSafe cluster for failover of NFS, follow these steps:

1. Install, configure, and test the base IRIX FailSafe software as described in the *IRIS FailSafe Version 2 Administrator's Guide*.
2. Install the NFS software. See "Required Software", page 3.
3. Configure the NFS file-locking information. See Chapter 2, "Configuring NFS", page 5.
4. If needed, install the `NFS` and `statd_unlimited` resource types. See "Installing the `NFS` and `statd_unlimited` Resource Types", page 9.
5. Add the individual instances of `NFS` and (if needed) `statd_unlimited` resources to the cluster database. See "Defining Resources", page 11.
6. Create the resource group that will be failed over. See "Creating a Resource Group", page 16.
7. Test the NFS failover. See the section "Testing the `start` Script", page 22.

Configuring NFS

You can use the following procedure to configure the NFS filesystems that will be failed over. There are no entries in `/etc/exports` that are required for these filesystems; FailSafe software exports the NFS filesystems.

1. Create or identify the filesystems to be failed over. They must follow the guidelines in "Planning NFS Filesystems", page 2.
2. For each exported filesystem, create a directory named `statmon` beneath the export point. This directory will hold the NFS file-locking information using the NFS client interface IP addresses.

   The directory must have the following characteristics:

   - Owner: `root`
   - Group: `sys`
   - Minimum permission: `0700` (at least read/write/execute for owner `root`)

   Note: The NFS clients mount the highly available (HA) NFS filesystem using the HA IP address (not the hostname of the server).

3. On each node, edit the `/etc/config/statd.options` file and put `-h` on the first line of the file.

   If you must create the `/etc/config/statd.options` file, set the permissions as follows :

   ```
   # chown root.sys /etc/config/statd.options
   # chmod 644 /etc/config/statd.options
   ```

Suppose you have the following configuration:

- A node named `Stocks` (IP address `192.56.50.1`) that exports the `/shared1` NFS filesystem
- A node named `Bonds` (IP address `192.56.51.2`) that exports the `/shared2` NFS filesystem

To make these filesystem highly available with FailSafe, you must create the following directories, giving them the appropriate permissions (644):

- `/shared1/statmon` on the node `Stocks`. This directory stores NFS lock information that lets the node `Bonds` recover the NFS locks for the filesystem `/shared1` if `Stocks` fails.
- `/shared2/statmon` on the node `Bonds`. This directory stores NFS lock information that lets the node `stocks` recover NFS locks for the filesystem `/shared2` if `Bonds` fails.

Figure 2-1 shows an example failover for a resource group named `nfs1` from the node named `Stocks` to the node named `Bonds`.

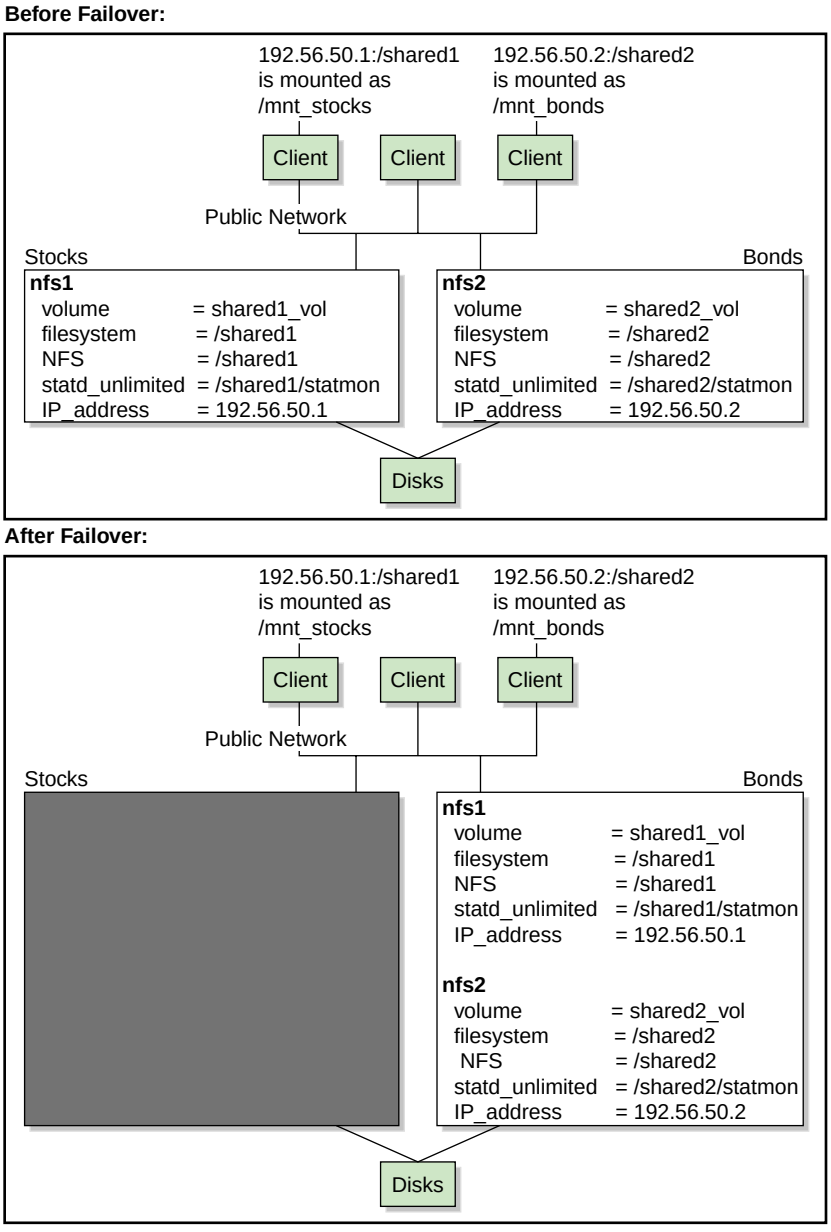

Figure 2-1 Failover Example

Adding NFS to the Cluster Database

This chapter discusses the tasks required to add NFS to the FailSafe cluster database. The major sections in this chapter are as follows:

- "Installing the `NFS` and `statd_unlimited` Resource Types"
- "Defining Resources", page 11
- "Creating a Resource Group", page 16
- "Changing Export Options", page 17

These procedures assume that the FailSafe cluster database has already been created, installed, and tested as described in the *IRIS FailSafe Version 2 Administrator's Guide*.

To verify hat the `NFS` and `statd_unlimited` resources have been correctly configured, see Chapter 4, "Testing the Resources", page 19.

Installing the `NFS` and `statd_unlimited` Resource Types

If you have the FailSafe NFS software installed before you create a FailSafe cluster, the `NFS` and `statd_unlimited` resource types will be automatically installed at cluster creation time. However, if you already have a cluster created before you install the FailSafe NFS software, you must manually install the resource types.

To determine if the resource types have been installed, enter the following `cmgr`(1M) command:

`show resource_types in cluster` *clustername*

For example, the following output shows that the resource types are not present:

```
cmgr> show resource_types in cluster nfscluster
template
Netscape_web
Oracle_DB
MAC_address
IP_address
INFORMIX_DB
```

```
filesystem
volume
```

You can also display this information by selecting **View: Types** in the FailSafe GUI.

To install the resource types, use the following `cmgr` command:

`install resource_type` *ResourceTypename* `in cluster` *Clustername*

For example, to install the `NFS` resource type in a cluster named `nfscluster`, enter the following `cmgr` command:

```
cmgr> install resource_type NFS in cluster nfscluster
```

To do this in the GUI, use the following menu selection and choose the name of the resource type:

Tasks
> Resource Types
> Load a Resource Type

Note: As of FailSafe NFS 2.2, the `start` executable timeout has been increased to 50 seconds. If you are **upgrading from a FailSafe NFS 2.1 or earlier**, you may want to increase the `start` executable timeout for the `statd_unlimited` resource type by using the **Modify a Resource Type** task the FailSafe GUI or the `modify resource_type` operation in the `cmgr` command. In FailSafe 2.2, the recovery for NFS locks will take at most 30 seconds, even when NFS clients are not responding.

For example, for a cluster named `mycluster`:

```
cmgr> modify resource_type statd_unlimited in cluster mycluster
Enter commands, when finished enter either "done" or "cancel"

resource_type statd_unlimited ? modify action start
Enter action parameters, when finished enter "done" or "cancel"

Current action start parameters:
        exec_time : 40000ms
        monitor_interval : 0ms
        monitor_time : 0ms

Action - start ? set exec_time to 50000
Action - start ? done
resource_type statd_unlimited ? done
Successfully modified resource_type statd_unlimited
```

Defining Resources

You can use the FailSafe GUI or the `cmgr`(1M) command to define specific instances of `NFS` and `statd_unlimited` resources. You must provide a name for each resource and information about any type-specific attributes; the resource name can be a maximum length of 255 characters and cannot begin with an underscore. For more information about using the GUI or `cmgr`, see the *IRIS FailSafe Version 2 Administrator's Guide*.

If there are more than 50 clients mounting the NFS filesystem, the NFS lock recovery for the filesystem will take longer. In such cases, you may have to increase the maximum execution time of the `start` action script for the `statd_unlimited` resource type.

NFS Attributes

The NFS resource name is the export-point; therefore, all standard IRIX directory name requirements must be followed.

You must define the following:

- **Export Information:** Specifies the options for the filesystem used by the exportfs(1M) command. In cmgr, this is the export-info attribute.

  An exportfs(1M) command option that is a relevant to FailSafe NFS is specifying the filesystem ID (fsid) for an exported filesystem. The fsid option specifies the value that will be returned in the fsid field of the attributes returned to the client. The value is an arbitrary number that must uniquely identify the exported filesystem. This option allows you to handle some Linux or other operating system NFS clients that are written to interpret or place restrictions on the fsid value returned in the attributes. If you use this option, you must use it for all exported filesystems and you must not use it in conjunction with the nohide option. This option only affects the fsid contained in the file attributes. File handles are constructed differently and do not use the fsid value in the file attributes.

  For more information about options, see the exports(4) man page.

- **Filesystem:** Specifies the export disk name used as input to the exportfs(1M) command. In cmgr, this is the filesystem attribute.

statd_unlimited Attributes

The statd_unlimited resource specifies the NFS lock directory, which stores information about the NFS client locks. The statd_unlimited resource type requires the **Export Point** attribute, which specifies the NFS export point that corresponds to the NFS lock directory. In cmgr, this is the ExportPoint attribute.

Interactive cmgr Example

This example assumes that the highly available cluster named nfscluster and two resources (/shared1 and /shared2) of the filesystem resource type have been created in the cluster database.

```
node1# /usr/cluster/bin/cmgr
Welcome to IRIS FailSafe Cluster Manager Command-Line Interface

cmgr> set cluster nfscluster
cmgr> define resource /shared1 of resource_type NFS
Enter commands, when finished enter either "done" or "cancel"

Type specific attributes to create with set command:

Type Specific Attributes - 1: export-info
Type Specific Attributes - 2: filesystem

Resource type dependencies to add:

Resource Dependency Type - 1: filesystem

resource /shared1 ? set export-info to "rw,wsync,anon=root"
resource /shared1 ? set filesystem to /shared1
resource /shared1 ? add dependency /shared1 of type filesystem
resource /shared1 ? done
Successfully created resource /shared1

cmgr> define resource /shared1/statmon of resource_type statd_unlimited
Enter commands, when finished enter either "done" or "cancel"

Type specific attributes to create with set command:

Type Specific Attributes - 1: ExportPoint

Resource type dependencies to add:

Resource Dependency Type - 1: NFS

resource /shared1/statmon ? set ExportPoint to /shared1
resource /shared1/statmon ? add dependency /shared1 of type NFS
resource /shared1/statmon ? done
```

```
Successfully created resource /shared1/statmon

cmgr> show resource /shared1 of resource_type NFS
export-info: rw,wsync,anon=root
filesystem: /shared1

Resource dependencies
filesystem /shared1

cmgr> define resource /shared2 of resource_type NFS
Enter commands, when finished enter either "done" or "cancel"

Type specific attributes to create with set command:

Type Specific Attributes - 1: export-info
Type Specific Attributes - 2: filesystem

Resource type dependencies to add:

Resource Dependency Type - 1: filesystem

resource /shared2 ? set export-info to "rw,wsync,anon=root"
resource /shared2 ? set filesystem to /shared2
resource /shared2 ? add dependency /shared2 of type filesystem
resource /shared2 ? done
Successfully created resource /shared2

cmgr> define resource /shared2/statmon of resource_type statd_unlimited
Enter commands, when finished enter either "done" or "cancel"

Type specific attributes to create with set command:

Type Specific Attributes - 1: ExportPoint

Resource type dependencies to add:

Resource Dependency Type - 1: NFS

resource /shared2/statmon ? set ExportPoint to /shared2
resource /shared2/statmon ? add dependency /shared2 of type NFS
resource /shared2/statmon ? done
```

```
Successfully created resource /shared2/statmon

cmgr> show resource /shared2 of resource_type NFS
export-info: rw,wsync,anon=root
filesystem: /shared2

Resource dependencies
filesystem /shared2

cmgr> quit

node1#
```

Table 3-1 shows examples using the values as shown with the `cmgr` command; you can also set these values by using the FailSafe GUI.

Table 3-1 Example Resources and Attributes for the Resource Groups

Resource Type	Attribute	Dependency	Resource Group `Stocks_rg`	Resource Group `Bonds_rg`
`filesystem`			`/shared1`	`/shared2`
	`volume-name`		`/shared1_vol`	`/shared2_vol`
	`mount-options`		`rw`	`rw`
	`kill-nfsds-before-umount`		`true`	`true`
	`monitoring-level`		`2`	`2`
`IP_address`			`192.56.50.1`	`192.56.51.2`
	`NetworkMask`		`0xffffff00`	`0xffffff00`
	`interfaces`		`ef0`	`ef0`
`NFS`			`/shared1`	`/shared2`
	`export-info`		`rw,wsync,anon+root`	`rw,wsync,anon+root`
	`filesystem`		`/shared1`	`/shared2`
		`filesystem`	`/shared1`	`/shared2`

Resource Type	Attribute	Dependency	Resource Group Stocks_rg	Resource Group Bonds_rg
statd_unlimited			/shared1/statmon	/shared2/statmon
	ExportPoint		/shared1	/shared2
		NFS	/shared1	/shared2
volume			shared1_vol	shared2_vol
	devname-group		sys	sys
	devname_owner		root	root
	devname_mode		0600	0600

Creating a Resource Group

You can use the FailSafe GUI or the cmgr(1M) command to define a resource group. You must include all of the resources that the NFS resource is dependent on, such as filesystems, volumes, and IP addresses. An NFS resource should have a filesystem or a CXFS resource as a dependency.

The statd_unlimited resource is dependent upon the NFS resource. For more information about using the GUI or cmgr, see the *IRIS FailSafe Version 2 Administrator's Guide*.

The following example shows the creation of a typical resource group:

```
cmgr> create resource_group nfs1 in cluster nfscluster
Enter commands, when finished enter either "done" or "cancel"

resource_group nfs ? set failover_policy to ordered
resource_group nfs ? add resource /shared1 of resource_type NFS
resource_group nfs ? add resource 192.56.50.1 of resource_type IP_address
resource_group nfs ? add resource /shared1 of resource_type filesystem
resource_group nfs ? add resource shared1_vol of resource_type volume
resource_group nfs ? add resource /shared1/statmon of resource_type statd_unlimited


resource_group nfs ? done
Successfully created resource group nfs
```

```
cmgr> show resource_group nfs1 in cluster nfscluster

Resource Group: nfs1
        Cluster: nfscluster
        Failover Policy: ordered

Resources:
        /shared1 (type: NFS)
        192.56.50.1 (type: IP_Addresses)
        /shared1 (type: filesystem)
        shared1_vol (type: volume)
        /shared1/statmon (type: statd_unlimited)
```

Changing Export Options

The `/var/cluster/cmgr-scripts/change-export-options` script lets you change export options for an NFS filesystem when the resource group containing the NFS resource is online. Use the following command line format:

`change-export-options "`*ExportOptions*`"` *NFS-ExportPoint ResourceGroup* `[`*statd_unlimited_ResourceName*`]`

If you use multiple export options, you must enclose them in double-quotation marks. If there are multiple `statd_unlimited` resources present in the resource group, you must also specify the `statd_unlimited` resource name.

For example, the following command line changes the export options for the `/hafs` NFS resource in the `nfs-group` resource group to be read-only, and specifies that the `statd_unlimited` resource to be used is `/hafs/statmon`:

```
# change-export-options "ro" /hafs nfs-group /hafs/statmon
Modifying NFS export option for /hafs in resource group nfs-group in cluster nfs-cluster
Putting resource group nfs-group in maintenance mode
Resume resource group nfs-group monitoring
change-export-options completed successfully
```

Testing the Resources

To ensure that the NFS and statd_unlimited resources have been correctly configured, you should test the set of action scripts located in the following directories:

```
/var/cluster/ha/resource_types/NFS
/var/cluster/ha/resource_types/statd_unlimited
```

After executing an action script, see the contents of the output file. The output file must contain the resource name and status of action for the resource. If resource status is zero, the action was successful for the resource; if the status is nonzero, there was an error that could mean the resource was not configured properly for FailSafe. For more information on error codes, see the *IRIS FailSafe Version 2 Programmer's Guide*.

For example, to test an NFS resource named /shared1, you would test each action script by starting with the following commands, using an input file named /tmp/ipfile and an outputfile named /tmp/opfile:

```
$ cd /var/cluster/ha/resource_types/NFS
$ echo /shared1 > /tmp/ipfile
```

You can then execute each action script with the following command:

```
$ ./actionscript /tmp/ipfile /tmp/opfile
```

where *actionscript* is one of the following action script names:

```
start
stop
monitor
exclusive
restart
```

Note: By default, the NFS and statd_unlimited resource types are not configured to be restarted. Although an action script file by the name of restart must exist, it is not necessary to test it unless you change the configuration to use the script.

For more information about action scripts, see the *IRIS FailSafe Version 2 Administrator's Guide*.

For example:

```
# cd /var/cluster/ha/resource_types/NFS
# echo /shared1 > /tmp/ipfile
# ./start /tmp/ipfile /tmp/opfile
# cat /tmp/opfile
/shared1 0
# rm /tmp/opfile
# ./stop /tmp/ipfile /tmp/opfile
# cat /tmp/opfile
/shared1 0
etc.
```

The output for each action script is logged to the following file:

`/var/cluster/ha/log/script_`*hostname*

You can increase the amount of data logged from scripts by changing the `HA_CURRENT_LOGLEVEL` variable in the following file on each node:

`/var/cluster/ha/common_scripts/scriptlib`

The default value for `HA_CURRENT_LOGLEVEL` is 2; to make logging more verbose, you can change it to 11. For more information about log levels, see the *IRIS FailSafe Version 2 Administrator's Guide*.

The rest of this chapter discusses the following:

Obtaining More Details

To view the individual action script executions, you must edit the action scripts and add `set -x` to the function.

In the following example, to see what the `start_nfs()` function does, edit the `start` script for the `NFS` resource and add `set -x` in the `start` function.

```
# Start the resource on the local machine.
# Return HA_SUCCESS if the resource has been successfully started on
# the local machine and HA_CMD_FAILED otherwise. The resource name is
# the nfs instance id.
#
start_nfs()
{
set -x
 # for all nfs resource configured
...
```

Following is an example showing output obtained when `set -x` is used:

```
# cd /var/cluster/ha/resource_types/NFS
# echo /shared1 > /tmp/ipfile
# ./start /tmp/ipfile /tmp/opfile
+ HA_SCRIPTNAME=/var/cluster/ha/resource_types/NFS/start
+ set_global_variables
+ set_local_variables
+ ha_check_args /tmp/ipfile /dev/null
+ [ 0 -ne 0 ]
+ ha_read_infile
+ start_nfs
+ NFSFILEDIR=/var/cluster/ha/tmp/shared1
+ HA_CMD=/sbin/mkdir -p /var/cluster/ha/tmp/shared1
+ ha_execute_cmd creating nfs status file directory
+ get_nfs_info /shared1
+ [ 0 -ne 0 ]
+ ha_get_field export-info rw,wsync,anon=root
filesystem /shared1 export-point
+ [ 0 -ne 0 ]
+ ha_get_field export-info rw,wsync,anon=root
filesystem /shared1 export-info
+ [ 0 -ne 0 ]
+ export_opts=rw,wsync,anon=root
```

```
+ retstat=0
+ /usr/etc/exportfs
+ grep /shared1$
+ 1> /dev/null 2>& 1
+ retstat=1
+ [ 1 -eq 1 ]
+ /usr/etc/exportfs
+ grep /shared1
+ grep rw,wsync,anon=root$
+ 1> /dev/null 2>& 1
+ retstat=1
+ [ 1 -eq 1 ]
+ HA_CMD=/sbin/grep /shared1 /etc/mtab > /dev/null 2>&1
+ ha_execute_cmd check if the export-point exists
+ [ 0 -eq 0 ]
+ HA_CMD=/usr/etc/exportfs -i -o rw,wsync,anon=root /shared1
+ ha_execute_cmd export /shared1 directories to NFS clients
+ [ 0 -ne 0 ]
+ ha_write_status_for_resource /shared1 0
+ exit_script 0
```

Testing the `start` Script

Before testing the `start` script, you must ensure that the file system has not already been exported. You can use the `exportfs`(1M) command to check if the file system has been exported.

If it has already been exported, you must unexport the file system by using the `exportfs -u` command, as in the following example:

```
# exportfs
/dmf_home
/move_fs
/usr/cluster/bin
/shared2
shared1 -rw,wsync,anon=root
# exportfs -u /shared1
```

After running the test, you can use the `exportfs`(1M) command again to verify that the resource is exported.

Testing the `stop` Script

Before testing the `stop` script, you must use the `exportfs`(1M) command to ensure that the NFS file system is already exported. If it has not already been exported, you must export it by using the `exportfs` command before running the `stop` script.

After running the test, you can use the `exportfs`(1M) command again to verify that the resource has been stopped by the `stop` script and is therefore not exported.

Testing the `monitor` Script

Use the `tail`(1) command on the script log to verify that the `monitor` script works as expected. For example:

```
# tail /var/cluster/ha/log/script_node1
Mon Dec  9 17:34:12 <D0 /var/cluster/ha/resource_types/NFS/monitor script 182977:0>
/var/cluster/ha/resource_types/NFS/monitor called with  /var/cluster/ha/tmp/srmMCXa004xn,
/var/cluster/ha/tmp/srmNCXa004xn and /var/cluster/ha/tmp/srmOCXa004xn
Mon Dec  9 17:34:13 <D0 /var/cluster/ha/resource_types/NFS/monitor script 183688:0> Entry:
monitor_nfs()
Mon Dec  9 17:34:14 <D0 /var/cluster/ha/resource_types/NFS/monitor script 183688:0> check the
filesystem /shared1 is exported
Mon Dec  9 17:34:14 <D0 /var/cluster/ha/resource_types/NFS/monitor script 183688:0> awk '{print $1}'
/var/cluster/ha/tmp/exportfs.183688 | grep -x /shared1 exited with status 0
Mon Dec  9 17:34:14 <D0 /var/cluster/ha/resource_types/NFS/monitor script 183688:0> Entry:
exec_rpcinfo()
Mon Dec  9 17:34:14 <D0 /var/cluster/ha/resource_types/NFS/monitor script 183688:0> trying rpcinfo
on localhost for nfs ver 3
Mon Dec  9 17:34:15 <D0 /var/cluster/ha/resource_types/NFS/monitor script 183688:0>
/usr/cluster/bin/ha_exec2 5 2 /usr/etc/rpcinfo -n 2049 -u 127.0.0.1 nfs 3 exited with status 0
Mon Dec  9 17:34:15 <D0 /var/cluster/ha/resource_types/NFS/monitor script 183688:0> Return:
exec_rpcinfo() value 0
Mon Dec  9 17:34:15 <D0 /var/cluster/ha/resource_types/NFS/monitor script 183688:0> Exit:
exit_script()
```

Testing the `exclusive` Script

Use the `tail`(1) command on the script log to verify that the `exclusive` script works as expected. For example:

```
# tail /var/cluster/ha/log/script_node1
Mon Dec  9 17:37:19 <D0 /var/cluster/ha/resource_types/NFS/exclusive script 183918:0>
/var/cluster/ha/resource_types/NFS/exclusive called with /var/cluster/ha/tmp/srmYLXa004xn,
/var/cluster/ha/tmp/srmZLXa004xn and /var/cluster/ha/tmp/srmAMXa004xn
Mon Dec  9 17:37:19 <D0 /var/cluster/ha/resource_types/NFS/exclusive script 183918:0> Entry:
exclusive_nfs()
Mon Dec  9 17:37:20 <D0 /var/cluster/ha/resource_types/NFS/exclusive script 183918:0> checking for
/shared1 exported directory
Mon Dec  9 17:37:20 <N /var/cluster/ha/resource_types/NFS/exclusive script 183918:0> checking for
/shared1 exported directory failed
Mon Dec  9 17:37:20 <N /var/cluster/ha/resource_types/NFS/exclusive script 183918:0>
Mon Dec  9 17:37:20 <D0 /var/cluster/ha/resource_types/NFS/exclusive script 183918:0> /sbin/grep
/shared1 /var/cluster/ha/tmp/showmount.183918 >> /dev/null 2>&1 exited with status 1
Mon Dec  9 17:37:20 <N /var/cluster/ha/resource_types/NFS/exclusive script 183918:0> resource
/shared1 exclusive status: NOT RUNNING
Mon Dec  9 17:37:20 <D0 /var/cluster/ha/resource_types/NFS/exclusive script 183918:0> Exit:
exit_script()
```

Testing the `restart` Script

You should test the `restart` script if the resource is configured to restart; by default, the `NFS` and `statd_unlimited` resource types are not configured to restart.

After running the test, you can use the `exportfs`(1M) command to verify that the resource is exported.

Testing Resource Group Failovers

You can test the failover policy by using either the `cmgr`(1M) or the FailSafe GUI to move the resource group to another node in the cluster and then display the resource group states. The following example uses `cmgr` to test the failover policy:

```
cmgr> admin offline resource_group NFS in cluster nfscluster

cmgr> admin move resource_group NFS in cluster eagan to node node2

cmgr> admin online resource_group NFS in cluster nfscluster
```

For more information about states, see the *IRIS FailSafe Version 2 Administrator's Guide*.

Glossary

action scripts

The set of scripts that determine how a resource is started, monitored, and stopped. There must be a set of action scripts specified for each resource type. The possible set of action scripts is: `exclusive`, `start`, `stop`, `monitor`, and `restart`.

active/backup configuration

A configuration in which all resource groups have the same primary node. The backup node does not run any highly available resource groups until a failover occurs.

cluster

A *cluster* is the set of systems (nodes) configured to work together as a single computing resource. A cluster is identified by a simple name and a cluster ID.

There is only one cluster that may be formed from a given pool of nodes.

Disks or logical units (LUNs) are assigned to clusters by recording the name of the cluster on the disk (or LUN). Thus, if any disk is accessible (via a Fibre Channel connection) from machines in multiple clusters, then those clusters must have unique names. When members of a cluster send messages to each other, they identify their cluster via the cluster ID. Thus, if two clusters will be sharing the same network for communications, then they must have unique cluster IDs.

Because of the above restrictions on cluster names and cluster IDs, and because cluster names and cluster IDs cannot be changed once the cluster is created (without deleting the cluster and recreating it), SGI advises that you choose unique names and cluster IDs for each of the clusters within your organization. Clusters that share a network and use XVM must have unique names.

cluster database

Contains configuration information about all resources, resource types, resource groups, failover policies, nodes, and the cluster.

cluster node

A node that is defined as part of the cluster. See also *node*.

control network

The network that connects nodes through their network interfaces (typically Ethernet) such that FailSafe can maintain a cluster's high availability by sending heartbeat messages and control messages through the network to the attached nodes. FailSafe uses the highest priority network interface on the control network; it uses a network interface with lower priority when all higher-priority network interfaces on the control network fail.

A node must have at least one control network interface for heartbeat messages and one for control messages (both heartbeat and control messages can be configured to use the same interface). A node can have no more than eight control network interfaces.

database

See *cluster database*.

dependency list

See *resource dependency* or *resource type dependency*.

failover

The process of allocating a *resource group* to another *node* according to a *failover policy* A failover may be triggered by the failure of a resource, a change in the FailSafe membership (such as when a node fails or starts), or a manual request by the administrator.

failover attribute

A string that affects the allocation of a resource group in a cluster. The administrator must specify system-defined attributes (such as `Auto_Failback` or `Controlled_Failback`), and can optionally supply site-specific attributes.

failover domain

The ordered list of nodes on which a particular *resource group* can be allocated. The nodes listed in the failover domain must be within the same cluster; however, the failover domain does not have to include every node in the cluster. The administrator defines the *initial failover domain* when creating a failover policy. This list is transformed into the *run-time failover domain* by the *failover script* the run-time failover domain is what is actually used to select the failover node. FailSafe stores the run-time failover domain and uses it as input to the next failover script invocation. The initial

and run-time failover domains may be identical, depending upon the contents of the failover script. In general, FailSafe allocates a given resource group to the first node listed in the run-time failover domain that is also in the FailSafe membership; the point at which this allocation takes place is affected by the *failover attributes*.

failover policy

The method used by FailSafe to determine the destination node of a failover. A failover policy consists of a *failover domain*, *failover attributes*, and a *failover script*. A failover policy name must be unique within the *pool*.

failover script

A failover policy component that generates a *run-time failover domain* and returns it to the FailSafe process. The process applies the failover attributes and then selects the first node in the returned failover domain that is also in the current FailSafe membership.

FailSafe database

See *cluster database*.

heartbeat messages

Messages that cluster software sends between the nodes that indicate a node is up and running. Heartbeat messages and *control messages* are sent through a node's network interfaces that have been attached to a control network. A node can be attached to multiple control networks.

initial failover domain

The ordered list of nodes, defined by the administrator when a failover policy is first created, that is used the first time a cluster is booted. The ordered list specified by the initial failover domain is transformed into a *run-time failover domain* by the *failover script*; the run-time failover domain is used along with failover attributes to determine the node on which a resource group should reside. With each failure, the failover script takes the current run-time failover domain and potentially modifies it; the initial failover domain is never used again. Depending on the run-time conditions and contents of the failover script, the initial and run-time failover domains may be identical. See also *run-time failover domain*.

log configuration

A log configuration has two parts: a *log level* and a *log file*, both associated with a *log group*. The cluster administrator can customize the location and amount of log output, and can specify a log configuration for all nodes or for only one node. For example, the `crsd` log group can be configured to log detailed level-10 messages to the `/var/cluster/ha/log/crsd-foo` log only on the node `foo` and to write only minimal level-1 messages to the `crsd` log on all other nodes.

log file

A file containing notifications for a particular *log group*. A log file is part of the *log configuration* for a log group. By default, log files reside in the `/var/cluster/ha/log` directory, but the cluster administrator can customize this. Note: FailSafe logs both normal operations and critical errors to `/var/adm/SYSLOG`, as well as to individual logs for specific log groups.

log group

A set of one or more FailSafe processes that use the same log configuration. A log group usually corresponds to one daemon, such as `gcd`.

log level

A number controlling the number of log messages that FailSafe will write into an associated log group's log file. A log level is part of the log configuration for a log group.

node

A *node* is an operating system (OS) image, usually an individual computer. (This use of the term *node* does not have the same meaning as a node in an SGI Origin 3000 or SGI 2000 system.)

A given node can be a member of only one pool (and therefore) only one cluster.

plug-in

The set of software required to make an application highly available, including a resource type and action scripts. There are plug-ins provided with the base FailSafe release, optional plug-ins available for purchase from SGI, and customized plug-ins you can write using the instructions in the *IRIS FailSafe Version 2 Programmer's Guide*.

pool

The *pool* is the set of nodes from which a particular cluster may be formed. Only one cluster may be configured from a given pool, and it need not contain all of the available nodes. (Other pools may exist, but each is disjoint from the other. They share no node or cluster definitions.)

A pool is formed when you connect to a given node and define that node in the cluster database using the CXFS GUI or `cmgr`(1M) command. You can then add other nodes to the pool by defining them while still connected to the first node, or to any other node that is already in the pool. (If you were to connect to another node and then define it, you would be creating a second pool).

resource

A single physical or logical entity that provides a service to clients or other resources. For example, a resource can be a single disk volume, a particular network address, or an application such as a web server. A resource is generally available for use over time on two or more nodes in a cluster, although it can be allocated to only one node at any given time. Resources are identified by a resource name and a resource type. Dependent resources must be part of the same resource group and are identified in a resource dependency list.

resource dependency

The condition in which a resource requires the existence of other resources.

resource dependency list

A list of resources upon which a resource depends. Each resource instance must have resource dependencies that satisfy its resource type dependencies before it can be added to a resource group.

resource group

A collection of resources. A resource group is identified by a simple name; this name must be unique within a cluster. Resource groups cannot overlap; that is, two resource groups cannot contain the same resource. All interdependent resources must be part of the same resource group. If any individual resource in a resource group becomes unavailable for its intended use, then the entire resource group is considered unavailable. Therefore, a resource group is the unit of failover.

resource name

The simple name that identifies a specific instance of a resource type. A resource name must be unique within a given resource type.

resource type

A particular class of resource. All of the resources in a particular resource type can be handled in the same way for the purposes of failover. Every resource is an instance of exactly one resource type. A resource type is identified by a simple name; this name must be unique within a cluster. A resource type can be defined for a specific node or for an entire cluster. A resource type that is defined for a node overrides a cluster-wide resource type definition with the same name; this allows an individual node to override global settings from a cluster-wide resource type definition.

resource type dependency

A set of resource types upon which a resource type depends. For example, the `filesystem` resource type depends upon the `volume` resource type, and the `Netscape_web` resource type depends upon the `filesystem` and `IP_address` resource types.

resource type dependency list

A list of resource types upon which a resource type depends.

run-time failover domain

The ordered set of nodes on which the resource group can execute upon failures, as modified by the failover script. The run-time failover domain is used along with failover attributes to determine the node on which a resource group should reside. See also *initial failover domain*.

start/stop order

Each resource type has a start/stop order, which is a nonnegative integer. In a resource group, the start/stop orders of the resource types determine the order in which the resources will be started when FailSafe brings the group online and will be stopped when FailSafe takes the group offline. The group's resources are started in increasing order, and stopped in decreasing order; resources of the same type are started and stopped in indeterminate order. For example, if resource type `volume` has order 10 and resource type `filesystem` has order 20, then when FailSafe brings a

resource group online, all volume resources in the group will be started before all file system resources in the group.

type-specific attribute

Required information used to define a resource of a particular resource type. For example, for a resource of type `filesystem` you must enter attributes for the resource's volume name (where the file system is located) and specify options for how to mount the file system (for example, as readable and writable).

Index