Migrating from IRIS FailSafe 1.2 to IRIS FailSafe 2.1.*x*

007–4630–001

CONTRIBUTORS

Written by Lori Johnson

Production by Glen Traefald

Engineering contributions by Paddy Sreenivasan

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

Record of Revision

Version	Description
001	July 2003 Original publication

Contents

Chapter 1

Introduction

IRIS FailSafe 2.1.*x* is not a new release of the IRIS FailSafe 1.2 product. Instead, it is a new set of files and scripts that provides many additional possibilities for the size and complexity of a highly available system.

If you wish to migrate a 1.2 system to a 2.1.*x* system to take advantage of these features, you must upgrade your system configuration. There is no upgrade installation option to automatically upgrade 1.2 to 2.1.*x*.

This chapter discusses the following:

- "Summary of Changes"
- "Hardware Changes", page 3
- "Software Changes", page 3
- "Configuration Changes", page 4
- "Script Changes", page 5
- "System Status Changes", page 5
- "Additional 2.1.*x* Tasks", page 5

See also Chapter 2, "Configuration Examples", page 7 and Chapter 3, "Script Changes for Programmers", page 17.

Summary of Changes

In 1.2, the unit of failover is the node. In 2.1.*x*, the unit of failover is the resource group. Because of this, the concepts of node failover, node failback, and node state do not apply to 2.1.*x*. In addition, all FailSafe scripts differ between the two releases.

Table 1-1 summarizes the differences between the 1.2 and 2.1.*x* releases.

Table 1-1 Differences between 1.2 and 2.1.*x*

FailSafe 1.2	FailSafe 2.1.*x*
`ha.conf` configuration file.	Cluster database at `/var/cluster/cdb/cdb/db`. The database is automatically copied to all nodes in the pool. Much of the data contained in the 1.2 `ha.conf` file will be used in the 2.1.*x* database, but the format is completely different. You will configure the database using the FailSafe Manager graphical user interface (GUI) or the `cmgr` command.
Node states (standby, normal, degraded, booting or up).	Resource group states (online, offline, pending, maintenance, error).

Scripts:
`giveaway, giveback`
`takeover, takeback`
`check`
(no equivalent)
	Scripts:
	`stop`
	`start`
	`monitor`
	`exclusive`, `probe`, `restart`
	Failover script
	Failover attributes

All common functions and variables are kept in the `/var/ha/actions/common.vars` file.	All common functions and variables are kept in the `/var/cluster/ha/common_scripts/scriptlib` file.
Configuration information is read using the `ha_cfginfo` command.	Configuration information is read using the `ha_get_info()` and `ha_get_field()` shell functions.
Software links specify application ordering.	Software links are not used for ordering.
Scripts use `/sbin/sh`.	Scripts use `/sbin/ksh`.
Scripts require configuration checksum verification.	There is no configuration checksum verification in the scripts.
Scripts require resource ownership.	Action scripts have no notion of resource ownership.
Scripts do not run in parallel.	Multiple instances of action scripts can be run at the same time.
Each service has its own log in `/var/ha/logs`.	Action scripts use cluster logging and all scripts log to the same file using the `ha_cilog` command.
There are two units of failover, one for each node in the cluster.	There is a unit of failover (a *resource group*) for each highly available service.

Hardware Changes

There are no hardware changes that are required when you upgrade a system to 2.1.*x*. A 1.2 system will be a dual-hosted storage with reset ring two-node configuration in 2.1.*x*.

With 2.1.*x*, you can test the hardware configuration with FailSafe diagnostic commands. (See *IRIS FailSafe Version 2 Administrator's Guide*, for instructions on using FailSafe to test the connections.) These diagnostics are not run automatically when you start FailSafe 2.1.*x*; you must run them manually.

You can also use the `admin ping` command to test the serial reset line in 2.1.*x*. This command replaces the `ha_spng` command you used with FailSafe 1.2.

The following 1.2 command tests serial reset lines:

```
# /usr/etc/ha_spng -i 1 -d msc -f /dev/ttyd2
# echo $status
```

The following 2.1.*x* `cmgr` command tests serial reset lines:

```
cmgr> admin ping dev_name /dev/ttyd2 of dev_type tty with sysctrl_type msc
```

If the `crsd` daemon is running, this command will not run.

See the *IRIS FailSafe Version 2 Administrator's Guide* for information on using `cmgr` commands.

Software Changes

FailSafe 2.1.*x* consists of a different set of files than 1.2. The 1.2 and 2.1.*x* software can exist on the same node, but you cannot run both versions of FailSafe at the same time.

FailSafe 1.2 contains a configuration file, `ha.conf`. In 2.1.*x*, configuration information is contained in a cluster database at `/var/cluster/cdb/cdb.db` that is kept in all FailSafe nodes in the pool. You configure the cluster database using the `cmgr` command or the GUI.

Note: If you are running 2.1.*x* in coexecution with CXFS, there may be some CXFS client-only nodes in the cluster that do not contain the cluster database.

The 2.1.*x* cluster database is automatically copied to all administration nodes in the pool. The 2.1.*x* configuration is kept in all administration nodes in the pool.

Configuration Changes

You must reconfigure your 1.2 system by using the 2.1.*x* FailSafe Manager GUI or the 2.1.*x* `cmgr` command. For information on using these administration tools, see the *IRIS FailSafe Version 2 Administrator's Guide*.

To update a 1.2 configuration, consider how the 1.2 configuration maps onto the concept of resource groups:

- A dual-active 1.2 configuration contains two resource groups, one for each node.
- An active/standby 1.2 configuration contains one resource group, consisting of an entire node (the active node).

Each resource group contains all the applications that were primary on each node and backed up by the other node.

When you configure a 2.1.*x* system, you perform the following steps:

1. Add nodes to the pool.
2. Define the cluster.
3. Add nodes to the cluster.
4. Set HA parameters (FailSafe 2.1.*x* can be started at this point, if desired).
5. Define resources.
6. Define failover policies.
7. Define resource groups.
8. Add resources to resource groups.
9. Put resource groups online.

The guided configuration tasks in the GUI lead you through these steps.

Script Changes

All customized 1.2 scripts must be rewritten for 2.1.*x*. For more information, see Chapter 3, "Script Changes for Programmers", page 17.

System Status Changes

In 1.2, you produced a display of the system status with the `ha_admin -a` command. In 2.1.*x*, you can display the system status in the following ways:

- You can keep continuous watch on the state of a cluster using the GUI.
- You can query the status of an individual resource group, node, or cluster using either the GUI or `cmgr`.
- You can use the `/var/cluster/cmgr-scripts/haStatus` script provided with the `cmgr` command to see the status of the cluster, nodes, resources, and resource groups.

For information on performing these tasks, see the *IRIS FailSafe Version 2 Administrator's Guide*.

Additional 2.1.*x* Tasks

After you have defined your nodes, cluster, and resources, you define your resource groups, a task which has no equivalent in FailSafe 1.2. When you define a resource group, you specify the resources that will be included in the resource group and the failover policy that determines which node will take over the services of the resource group on failure.

After you have configured your system, you can start FailSafe services, as described in the *IRIS FailSafe Version 2 Administrator's Guide*.

Chapter 2

Configuration Examples

To migrate from a 1.2 system to a 2.1.*x* system, you must examine your `ha.conf` file in order to determine how to define the equivalent parameters in the 2.1.*x* cluster database.

The following sections show upgrade examples for the following tasks:

- "Defining a Node"
- "Defining the Cluster", page 9
- "Setting HA Parameters", page 10
- "Defining an XLV Volume Resource", page 12
- "Defining an XFS Filesystem Resource", page 13
- "Defining an IP Address Resource", page 14

Note: The examples in this paper use the `cmgr` command in prompting mode (`-p`) and reflect the FailSafe 2.1.5 release. For details about `cmgr`, see the *IRIS FailSafe Version 2 Administrator's Guide*.

Defining a Node

The following example shows node definition in the 1.2 `ha.conf` file. Parameters that you must use when configuring a 2.1.*x* system are indicated in bold:

```
Node node1
{
interface node1-fxd
{
name = rns0
ip-address = 54.3.252.6
netmask = 255.255.255.0
broadcast-addr = 54.3.252.6
}
heartbeat
{
```

```
hb-private-ipname = 192.0.2.3
hb-public-ipname = 54.3.252.6
hb-probe-time = 6
hb-timeout = 6
hb-lost-count = 4
}
reset-tty = /dev/ttyd2

sys-ctlr-type = MSC
}
```

In this configuration example, you will use the following values when you define the same node in 2.1.*x*:

- Node name: `node1`
- Primary network interface: `node1`
- Type of system controller: `msc`
- System control device name: `/dev/ttyd2`
- Control networks: `192.0.2.3`, `54.3.252.6`

Use the following `cmgr` command to use these values to define a node in 2.1.*x*.

Note: There are additional parameters you must specify when you define this node.

```
cmgr> define node node1
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional]? node1
Is this a FailSafe node <true|false> ? true
Is this a CXFS node <true|false> ? false
Operating System <IRIX|Linux32|Linux64|AIX|HPUX|Solaris|Windows> ? Irix
Node Function <server_admin|client_admin> ? server_admin
Node ID[optional] ?
Partition ID[optional] ? (0)
Reset type <powerCycle> ? (powerCycle)
Do you wish to define system controller info[y/n]:y
Sysctrl Type <msc|mmsc|l2|l1>? (msc)
Sysctrl Password [optional]? ( )
Sysctrl Status <enabled|disabled>? enabled
Sysctrl Owner? node2
Sysctrl Device? /dev/ttyd2
Sysctrl Owner Type <tty> [tty]?
Number of Network interfaces [1]? 2
NIC 1 - IP Address? 192.0.2.3
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false>? true
NIC 1 - (use network for control messages) <true|false>? true
NIC 1 - Priority <1,2,...>? 1
NIC 2 - IP Address? 192.0.2.4
NIC 2 - Heartbeat HB (use network for heartbeats) <true|false>? true
NIC 2 - (use network for control messages) <true|false>? true
NIC 2 - Priority <1,2,...>? 2
```

For information on setting monitoring values in 2.1.*x*, see "Setting HA Parameters", page 10.

Defining the Cluster

Although 1.2 does not require the definition of clusters, you specify a parameter in the `ha.conf` file that 2.1.*x* uses in its cluster definition: the e-mail address to use to notify the system administrator when problems occur in the cluster.

The `ha.conf` file includes the following:

```
system configuration
{
mail-dest-addr = root@localhost
...
}
```

When you define a cluster in 2.1.*x*, you can use this as the e-mail address for problem notification.

There are other things you must provide in addition to this parameter when you define a 2.1.*x* cluster, such as the e-mail program to use for this notification and, of course, the nodes to include in the cluster.

Use the following `cmgr` command to define a cluster:

```
cmgr> **define cluster apache-cluster**
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? **true**
Is this a CXFS cluster <true|false> ? **false**
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster HA mode <normal|experimental> [optional] ?

No nodes in cluster apache-cluster
Add nodes to or remove nodes from cluster clustername
Enter "done" when completed or "cancel" to abort
clustername ? **add node node1**
clustername ? **add node node2**
...
clustername ? **done**
Successfully defined cluster apache-cluster
```

Setting HA Parameters

The following example shows the sections of a 1.2 `ha.conf` file that are used to set monitoring and timeout values. Parameters that you must use when configuring a 2.1.*x* system are indicated in bold.

```
system-configuration
{
      pwrfail = **true**
      ...
}


Node node1
{
...
heartbeat
{
       hb-private-ipname = 192.0.2.3
       hb-public-ipname = 54.3.252.6
       hb-probe-time = **6**
```

```
        hb-timeout = 6
        hb-lost-count = 4
}
...
}
```

As this `ha.conf` node-definition shows, in 1.2 you defined `hb-probe-time`, `hb-timeout`, and `hb-lost-count` parameters to set the values that determined how often to send monitoring messages and the length of time without a response that would indicate a failure. The 2.1.*x* release uses a different method for monitoring the nodes in a cluster than 1.2 uses, sending out continuous messages to the other nodes in a cluster and, in turn, maintaining continuous monitoring of the messages the other nodes are sending.

Because of the different monitoring methods between the two systems, there is no one-to-one correspondence between the values you set in the 1.2 `ha.conf` file and the timeout and heartbeat intervals you set in 2.1.*x* when you set FailSafe HA parameters. However, if you wish to maintain approximately the same time interval before which your system determines that failure has occurred, you can use the following formula to determine the value to which you should set your node timeout interval:

node_timeout = (*probetime* + *timeout*) * *lostcount*

This formula should account for the same total node-to-node communication time.

All 2.1.*x* timeouts are in milliseconds, and can be changed when 2.1.*x* is running. Timeouts can be specified for the cluster for a specific node in the cluster.

There is no long-timeout value in 2.1.*x*. The long-timeout value equivalent is set with the resource type `start` and `stop` timeouts. The resource type `start`, `monitor`, and `stop` timeouts can be changed using the GUI or `cmgr`.

Use the following `cmgr` command to modify the HA parameters for `node1` in 2.1.*x*:

Note: Node-specific cluster database information is not replicated to other nodes in the cluster. Therefore, you must change node-specific information when running `cmgr` on that node. You must execute `cmgr -p` on `node1` in the following example.

```
cmgr> modify ha_parameters on node node1 in cluster apache-cluster
Enter commands, when finished enter either "done" or "cancel"

Node Timeout (60000) ? 24000
Heartbeat Period (1000)? 6000
Node wait time (12000)?
Tie Breaker Node?
Run Powerfail (true)?
Successfully modified ha_parameters
```

Defining an XLV Volume Resource

The following example shows a volume definition in the 1.2 `ha.conf` file. Parameters that you must use when configuring the same volume as a volume resource in a 2.1.*x* system are indicated in bold:

```
volume apache-vol
{
   server-node = node1
   backup-node = node2
   devname = apache-vol
   devname-owner = root
   devname-group = sys
   devname-mode = 600
}
```

In this configuration example, you will use the following values when you define the same volume in 2.1.*x*:

- Volume name: `apache-vol`
- User name of device file owner: `root`
- Group name of device file: `sys`
- Device file permissions: `600`

To create an XLV volume resource, use the following `cmgr` command and the default values for each argument:

```
cmgr> define resource apache-vol of resource_type volume in cluster apache-cluster
Enter commands, when finished enter either "done" or "cancel"

devname-group [optional] ? (sys)
devname-owner [optional] ? (root)
devname-mode [optional] ? (600)

No resource type dependencies

Add dependencies to or remove from resource apache-vol:
Enter "done" when completed or "cancel" to abort

apache-vol ? done
```

Defining an XFS Filesystem Resource

The following example shows an XFS filesystem definition in the 1.2 `ha.conf` file. Parameters that you must use when configuring the same filesystem as a filesystem resource in a 2.1.*x* system are indicated in bold:

```
filesystem apache-fs
{
   mount-point = /apache-fs
   mount-info
{
   fs-type = xfs
   volume-name = apache-vol
   mode = rw, noauto
}
}
```

In this configuration example, you will use the following values when you define the same filesystem in 2.1.*x*:

- Resource name (mount point): `/apache-vol`
- XLV volume: `apache-vol`
- Mount options: `rw`, `noauto`

To create a filesystem resource, use the following `cmgr` command:

```
cmgr> define resource /apache-fs of resource_type filesystem in cluster apache-cluster
Enter commands, you may enter "done" or "cancel" at any time to exit

volume-name [optional] ? apache-vol
mount-options [optional] ? (rw) rw,noauto
kill-nfsds-before-umount [optional] ? (true)
monitoring-level [optional] ? (2)

Resource type dependencies to add or remove:
Resource dependency type - 1: volume

Add dependencies to or remove from resource /lori-fs:
Enter "done" when completed or "cancel" to abort

/apache-fs ? done
Successfully defined resource /apache-fs
```

Defining an IP Address Resource

The following example shows an IP address definition in the 1.2 `ha.conf` file. Parameters that you must use when configuring the same IP address as a highly available resource in a 2.1.*x* system are indicated in bold:

```
interface-pair FDDI_1
{
   primary-interface = node-fxd
   secondary-interface = node2-fxd
   re-mac = false
   netmask = 0xffffff00
   broadcast-addr = 54.3.252.255

   ip-aliases = ( 54.3.252.7 )
}
```

In this configuration example, you will use the following values when you define the same IP Address in 2.1.*x*:

- Resource name: `54.3.252.7`
- Broadcast address: `54.3.252.255`
- Network mask: `0xffffff00`

To create an IP address resource, use the following `cmgr` commands:

```
cmgr> define resource 54.3.252.7 of resource_type IP_address in cluster apache-cluster
Enter commands, you may enter "done" or "cancel" at any time to exit

NetworkMask [optional] ? 0xffffff00
interfaces [optional] ? rns0
BroadcastAddress [optional] ? 54.3.252.255
No resource type dependencies

Add dependencies to or remove from resource 54.3.252.7:
Enter "done" when completed or "cancel" to abort

54.3.252.7 ? done
Successfully defined resource 54.3.252.7
```

Chapter 3

Script Changes for Programmers

This chapter provides guidelines for migrating your 1.2 resources and monitor script information to 2.1.*x* action scripts. It covers the following:

- "Resource Types"
- "Reading Information", page 22
- "Parameter Parsing", page 23
- "Action Scripts", page 23
- "Ordering Script Actions", page 27

Caution: Multiple instances of 2.1.*x* action scripts may be executed at the same time. To avoid this, you can use the `ha_execute_lock` command.

The software for 2.1.*x* and 1.2 can coexist in the same node. However, 2.1.*x* and 1.2 cannot run at the same time.

There is no configuration checksum verification in scripts.

Resource Types

In 2.1.*x*, the `ha.conf` configuration file has been replaced by the cluster database. The cluster database is automatically copied to all FailSafe nodes in the pool. See the *IRIS FailSafe Version 2 Administrator's Guide* for information about configuring a 2.1*x* system.

If you require new resource types, you will create them using either the FailSafe Manager GUI or the `cmgr` command. See the *IRIS FailSafe Version 2 Administrator's Guide*.

You may be able to reuse the following monitoring information from the 1.2 `ha.conf` file with regard to 2.1.*x* resource types:

- `start-monitor-time`
- `lmon-probe-time` (equivalent in 2.1 to the `monitor` script's interval parameter)

- `lmon-timeout`

Note: All 2.1.*x* time-outs are in milliseconds.

The following examples show information (in bold) that is used in the 1.2 `ha.conf` file and reused when creating a new resource type in 2.1.*x*.

Suppose a portion of the 1.2 `ha.conf` file had the following:

```
action apache
{
        local-monitor = /var/ha/actions/ha_apache_lmon
}

action-timer apache
{
        start-monitor-time = 120
        lmon-probe-time = 120
        lmon-timeout = 60
}
```

You would reuse the information when creating a resource type in 2.1*x*, as follows:

```
cmgr> create resource_type apache in cluster apache-cluster


(Enter "cancel" at any time to abort)

Node[optional] ?
Order ? 500
Restart Mode ? (0)


DEFINE RESOURCE TYPE OPTIONS

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option: 1

No current resource type actions

Action name ? start
Executable timeout (in milliseconds) ? 20000
        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

Current resource type actions:
        start

Action name ? stop
Executable timeout (in milliseconds) ? 20000

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)
```

```
Enter option: 1
Current resource type actions:
        start
        stop

Action name ? monitor
Executable timeout (in milliseconds) ? 60000
Monitoring Interval (in milliseconds) ? 120000
Start Monitoring Time (in milliseconds) ? 120000

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

Current resource type actions:
        start
        stop
        monitor

Action name ? exclusive
Executable timeout (in milliseconds) ? 60000

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)
```

```
Enter option:3

No current type specific attributes

Type Specific Attribute ? search-string
Datatype ? string
Default value[optional] ? httpd

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:5

No current resource type dependencies

Dependency name ? IP_address


        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:7

Current resource type actions:
        Action - 1: start
        Action - 2: stop
        Action - 3: monitor
        Action - 4: exclusive

Current type specific attributes:
        Type Specific Attribute - 1: search-string

No current resource type dependencies

Resource dependencies to be added:
        Resource dependency - 1: IP_address

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully defined resource_type apache
```

Reading Information

In 2.1*x*, configuration information is read using the `ha_get_info()` and `ha_get_field()` shell functions. These functions are equivalent to the 1.2 `ha_cfginfo` command.

In 2.1*x*, all common functions and variables are kept in the following file:

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

This file is equivalent to the following 1.2 file:

`/var/ha/actions/common.vars`

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

Parameter Parsing

In 2.1*x*, action script parameters are passed in a file and information is also returned in a file. The script takes a list of resource names as parameters.

Action Scripts

Table 3-1, summarizes the differences in scripts between the releases.

Table 3-1 Differences between 1.2 and 2.1.*x* Scripts

FailSafe 1.2	FailSafe 2.1.*x*
`giveaway, giveback`	`stop`
`takeover, takeback`	`start`
`check`	`monitor`
(no equivalent)	`exclusive`, `restart`

In 2.1.*x*, the action scripts are installed in the following directory, where *Resource_Type_Name* is the name of the resource type (such as `NFS` and *Action_Name* is the name of the action script (such as `start`):

`/var/cluster/ha/`*Resource_Type_Name*`/`*Action_Name*

For example, the `start` action script for the `NFS` resource type would be located in the following directory:

```
/var/cluster/ha/NFS/start
```

Templates of the action scripts (`start`, `stop`, `monitor`, `exclusive`, `restart`) are provided in the following directory:

```
/var/cluster/ha/resource_types/template
```

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

The following sections provide example portions of 1.2 scripts and their 2.1.*x* equivalents:

- `giveback` and `stop`
- `takeover` and `start`
- `monitor` and `monitor`

Note: There are no 1.2 equivalents for the 2.1.*x* `exclusive` and `restart` scripts.

In the following examples, only the relevant portions of the scripts are shown. Areas in common between 1.2 and 2.1.*x* are shown in bold.

1.2 giveback / 2.1.*x* stop

For example, suppose you had the following in the `giveback` script in 1.2:

```
giveback()
{
    for i in`$CFG_INFO ${T_APACHE}`
    do
        SEARCH="$CFG_INFO ${T_APACHE}${CFG_SEP}${i}${CFG_SEP}${T_BACKUP}"
        BACKUP=`$SEARCH`
        if [ $? -eq 1 ]; then
            ${LOGGER} "$0: Trouble finding backup-node for apache ($SEARCH)"
            exit $INCORRECT_CONF_FILE;
        fi
        # If I am the backup
        if [ ${BACKUP} = ${HOST} ]; then
            ${LOGGER} "$0: Stopping apache for backup server."
            killall -9 /apache-fs/usr/local/apache_1.2.0/src/httpd
            if [ $? -ne "0" ]; then
                ${LOGGER} "$0: halt of apache on backup server failed."
            fi
        fi

        exit $SUCCESS
    done
}
```

In 2.1.*x*, you would have the following in the `stop` script:

```
stop_apache()
{
    for server in $HA_RES_NAMES
    do
        ${HA_DBGLOG} "Stopping apache server $server"
        killall -9 /apache-fs/usr/local/apache_1.2.0/src/httpd
        if [ $? -ne "0" ]; then
            ${HA_LOG} "halt of apache server $server failed."
            ha_write_status_for_resource $server $HA_CMD_FAILED;
        else
            ${HA_DBGLOG} "halt of apache server $server successful"
            ha_write_status_for_resource $server $HA_SUCCESS;
        fi
    done
}
```

1.2 takeover / 2.1.*x* start

For example, suppose you had the following in the `takeover` script in 1.2:

```
takeover()
{
    for i in`$CFG_INFO ${T_APACHE}`
    do
        SEARCH="$CFG_INFO ${T_APACHE}${CFG_SEP}${i}${CFG_SEP}${T_BACKUP}"
        BACKUP=`$SEARCH`
        if [ $? -eq 1 ]; then
            ${LOGGER} "$0: Trouble finding backup-node for apache ($SEARCH)"
            exit $INCORRECT_CONF_FILE;
        fi
        # If I am the backup
        if [ ${BACKUP} = ${HOST} ]; then
            ${LOGGER} "$0: Starting apache for backup server."
            /apache-fs/usr/local/apache_1.2.0/src/httpd -d \
/apache-fs/usr/local/apache_1.2.0
           if [ $? -ne "0" ]; then
                ${LOGGER} "$0: start of apache on backup server failed."
                exit $FAILED
            fi
        fi
        exit $SUCCESS
    done
}
```

In 2.1.*x*, you would have the following in the `start` script:

```
start_apache()
{
    for server in $HA_RES_NAMES
    do
        ${HA_DBGLOG} "Starting apache server $server"
        /apache-fs/usr/local/apache_1.2.0/src/httpd -d \
/apache-fs/usr/local/apache_1.2.0
        if [ $? -ne "0" ]; then
            ${HA_LOG} "start of apache server $server failed."
            ha_write_status_for_resource $server $HA_CMD_FAILED;
        else
            ${HA_DBGLOG} "start of apache server $server successful"
            ha_write_status_for_resource $server $HA_SUCCESS;
        fi
    done
}
```

1.2 monitor/ 2.1.*x* monitor

For example, suppose you had the following in the `monitor` script in 1.2:

```
monitor()
{

    # Read the search string entry
    for i in`$CFG_INFO ${T_APACHE}`
    do
        SEARCH="$CFG_INFO ${T_APACHE}${CFG_SEP}${i}${CFG_SEP}${T_SEARCH_STR}"
        SEARCH_STR=`$SEARCH`
        ${SEARCH_STR:=httpd};
    done

    EXEC="${KILLALL} -0 ${SEARCH_STR}";
    execute_cmd "check if apache server processes are running"

}
```

In 2.1.*x*, you would have the following in the `monitor` script:

```
monitor_apache()
{
    for server in $HA_RES_NAMES
    do
        get_apache_info $server
        if [ $? -eq 0 ]; then
            APACHE_FIELDS=${HA_STRING
            ha_get_field "${APACHE_FIELDS}" search-string;
            if [ $? -eq 0 ]; then
                SEARCH_STR=${HA_FIELD_VALUE};
            fi
        fi
        ${SEARCH_STR:=httpd};
        HA_CMD=${KILLALL} -0 ${SEARCH_STR}";
        ha_execute_cmd "check if server $server processes are running"
        if [ $? -ne 0 ]; then
            ${HA_LOG} "monitor of apache server $server failed."
            ha_write_status_for_resource $server $HA_CMD_FAILED;
        else
            ${HA_DBGLOG} "monitor of apache server $server successful"
            ha_write_status_for_resource $server $HA_SUCCESS;
        fi
    done
}
```

Ordering Script Actions

In 2.1.*x*, each resource type has a start/stop order, which is a positive integer. In a resource group, the start/stop orders of the component 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.

Note: 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.

There is no need to create software links similar to those used in 1.2.

Index