HIGHAVAILABILITY

RSF-1 Reference Guide

For RSF-1 Version 3.0+

High-Availability.Com Limited

Suite B,
Pentland House,
Village Way,
Wilmslow,
Cheshire,
SK9 2GH,
United Kingdom.

http://www.high-availability.com

Normal hours	+44 (0)844 736 1434
Outside hours	+44 (0)844 736 1974

Issue 2.5c, last updated 12/11/2013 11:16:00

Table of Contents

Table of Contents 5
List of Tables and Figures 10
1 Preface 11
1.1 Scope 11
1.2 Who should use this manual 11
1.3 Typographic conventions used in this guide 11
2 Overview of RSF-1 12
2.1 Concepts and definitions 12
2.2 Introduction 12
2.3 RSF-1 Terminology 12
2.3.1 Server/Node 12
2.3.2 Cluster 12
2.3.3 Service 12
2.3.4 Service states 13
2.3.5 Important considerations about broken_unsafe mode 14
2.3.6 Primary and Secondary Servers 16
2.3.7 Automatic and manual switchover mode 16
2.3.8 Blocked and unblocked mode 17
2.3.9 Heartbeats 17
2.3.9.1 Structure of a Heartbeat Packet 18
2.3.10 Network ports used by RSF-1 18
2.3.11 Floating addresses (VIP's) 18
2.3.11.1 Gratuitous ARP 18
2.3.11.2 Multiple network interface monitoring 19
2.3.12 Understanding how RSF-1 uses service timeouts 20
2.4 Scope of RSF-1 20
2.5 Conclusion 21
3 Components of RSF-1 22
3.1 Chapter overview 22
3.2 What the components do 22
3.3 The configuration file 22
3.4 Service start-up and shutdown 22
3.5 Heartbeats and monitoring 22
3.6 Controlling RSF-1 23
3.7 RSF-1 log file 23
4 Service failover definition 24
4.1 Chapter overview 24
4.2 RSF-I initiation 24
4.3 Service control 24
4.4 Host monitoring 24
4.5 Failover 25
4.6 Server recovery 25
5 Software Requirements for RSF-1 26
5.1 Chapter overview 26
5.2 Operating system 26
5.3 Network 26
5.4 Disk management 26
5.5 Applications 26
5.5.1 Installation 27
6 Building and Configuring a Cluster 28
6.1 Chapter overview 28
6.1.1 Network setup 29
6.1.2 Serial setup 29
6.1.3 Disk setup 29
6.2 Building the configuration file 29

7 Configuration file reference 34
7.1 Chapter overview 34
7.2 Location 34
7.2.1 Warning about differing configurations in a cluster 34
7.3 Distribution to cluster nodes 34
7.4 Syntax reference 34
7.4.1 Template configuration file 35
7.4.2 Syntactic rules 35
7.4.3 Variable affecting the whole cluster 35
7.4.3.1 DISC_HB_BACKOFF 36
7.4.3.2 CLUSTER_NAME 36
7.4.3.3 POLL_TIME 36
7.4.3.4 SCRIPT_TRIES 36
7.4.3.5 REALTIME 37
7.4.3.6 PRIORITY 37
7.4.3.7 NETBEAT_TRANSMITS 37
7.4.3.8 SCRIPT_TIMEOUT 37
7.4.3.9 SYSLOG_FACILITY 37
7.4.3.10 UI_HINT 37
7.4.3.11 IPDEVICE_MONITOR 38
7.4.3.12 NETBEAT_SOURCE_PORT 38
7.4.4 Machine definitions 39
7.4.4.1 MACHINE 39
7.4.4.2 The SERIAL heartbeat definition 40
7.4.4.3 The DISC heartbeat definition 40
7.4.4.4 The NET heartbeat definition 41
7.4.4.5 REALTIME, PRIORITY, NETBEAT_TRANSMITS 41
7.4.4.6 Summarising machine names 41
7.4.5 Service definition 42
7.4.5.1 SERVICE 42
7.4.5.2 OPTION 43
7.4.5.3 SERVER 43
7.4.5.4 IPDEVICE 43
7.4.5.5 INITIMEOUT 43
7.4.5.6 RUNTIMEOUT 44
7.4.5.7 MOUNT_POINT - *recommended use where applicable* 44
7.4.5.8 RSF_RESERVATION_DRIVE_n 45
7.4.5.9 A complete service example 45
7.4.6 Updating RSF-1 after changing configurations 45
7.4.7 Configuration database 45
7.5 Service start and stop scripts 47
7.5.1 Understanding the script directories 47
7.5.2 Creating the scripts 48
7.5.2.1 Writing your own 48
7.5.2.2 Using the templates provided 48
7.5.2.3 Example service directories supplied with RSF-1 49
7.5.2.4 Using application rc scripts 50
7.5.3 Exit codes 50
7.5.4 Adding the Scripts 51
7.5.5 Testing your scripts 51
7.5.6 rsfklink 52
7.6 RSF-1 Access control 52
7.6.1 Network access control 52
7.6.2 User access control 53
7.7 Summary 54
8 Starting and Stopping RSF-1 55
8.1 Chapter overview 55
8.2 Starting RSF-1 for the first time 55

8.2.1 Starting the heartbeats 55
8.3 Stopping RSF-1 55
8.3.1 Stopping RSF-1 and services 55
8.3.1.1 Using the console 56
8.3.1.2 Using the shutdown script 56
8.3.1.3 Sending a TERM Signal 56
8.3.2 Shutting down RSF-1 only (not recommended) 56
8.3.2.1 Using the console 56
8.3.2.2 Using the command line interface 56
8.3.2.3 Killing RSF-1 manually 57
8.3.3 Important notes about Stopping RSF-1 57
8.3.4 Signals and RSF-1 57
8.4 Restarting RSF-1 57
8.4.1 Restarting RSF-1 with Services Left Running 57
8.4.2 Restarting RSF-1 and Services 57
8.4.3 Note about Restarting RSF-1 58
8.5 Summary 58
9 Other RSF-1 Utilities 59
9.1 Chapter overview 59
9.2 Introduction 59
9.3 RSF-1 Log 59
9.4 Miscellaneous 59
9.4.1 rsfcli (Command Line Interface) 59
9.4.1.1 List 60
9.4.1.2 Heartbeats 61
9.4.1.3 Vname 61
9.4.1.4 Start 61
9.4.1.5 Stop 61
9.4.1.6 Auto 61
9.4.1.7 Manual 61
9.4.1.8 Bounce 62
9.4.1.9 Restart 62
9.4.1.10 Pass 62
9.4.1.11 Shutdown 62
9.4.1.12 Move 62
9.4.1.13 Repair 62
9.4.1.14 Dump 62
9.4.1.15 Debug 63
9.4.1.16 Is running 63
9.4.1.17 Status 63
9.4.1.18 Modes 63
9.4.1.19 RSF Log 63
9.4.1.20 Using rsfcli vname in scripts 64
9.4.2 hacdiag 65
10 Modifying RSF-1 Operation 66
10.1 Chapter overview 66
10.2 Changing Service Parameters 66
10.2.1 Changing a service name or floating name 66
10.2.2 Changing the service description 66
10.2.3 Changing the hostname of a server 66
10.2.4 Changing the timeouts 66
10.2.5 Changing the heartbeats 66
10.3 Changing the service scripts 67
10.4 Removing or replacing a service 67
10.5 Changing the ports used by RSF-1 67
10.6 Changing a Server 67
10.6.1 Hardware 67

10.6.2 Software 68
10.6.3 Network 68
10.7 Upgrading RSF-1 68
11 Towards Total Availability 69
11.1 Introduction 69
11.2 Physical security 69
11.3 System security 69
11.4 Related components 69
11.5 Maintenance 69
11.6 Training 69
11.7 Disaster planning 70
Appendix A. Glossary of Terms 71
Appendix B. RSF-1 for System Engineers 73
B.1 To the Administrator 73
B.2 To the Engineer 73
B.3 About RSF-1 High-Availability software 73
B.4 Identifying RSF-1 in operation 73
B.5 Disabling failover 74
B.6 Failing over running services 74
B.7 Restoring normal operation 74
B.8 Other Issues 74
B.9 In Case Of Difficulties 75
Appendix C. Service Script Environment 76
C.1 Introduction 76
C.2 Environment 76
C.3 Environment variables for scripts 76
C.4 Shell Header File 77
C.5 Start-up Parameters 77
Appendix D. Troubleshooting RSF-1 78
D.1 General Advice 78
D.2 Common Queries About RSF-1 78
D.2.1 The rsfmon daemon exits immediately 78
D.2.2 Some of the heartbeats are not working 78
D.2.3 A service isn't starting up or a fail over isn’t occurring 78
D.2.4 Shared disks are not accessible 79
D.2.5 The clients can't access the service 79
D.2.6 Admin console can't connect to a Server 79
D.2.7 A service failover occurred for no apparent reason 79
D.2.8 One or more servers rebooted but the services are not available 79
D.2.9 How can I be alerted by RSF-1 when a failure occurs? 79
D.2.10 Can I run multiple consoles? 80
D.2.11 Network attack resistance in RSF-1 version 2 80
D.3 Further Help 80
Appendix E. RSF-1 Log File Entries 81
E.1 Introduction 81
E.2 Heartbeat delays 81
Appendix F. Memory and CPU allocation by RSF-1 83
Appendix G. Further Support 84
G.1 Contacting High-Availability.Com 84
G.1.1 Support Calls 84
G.2 RSF-1 Support Mailing List 84
G.3 References and Additional Support 84
G.3.1 Solaris 84
G.3.2 AIX 85
G.3.3 Linux 85
G.3.4 OpenServer 85
G.3.5 HPUX 85
Appendix H. Summary of changes in RSF-1 versions 86

H.1 Changes to Version 2.1 86
H.1.1 FQDN hostnames 86
H.1.2 Process locking 86
H.2 Changes in Version 2.4 86
H.2.1 Real time scheduling 86
H.2.2 Syslog facility codes 86
H.2.3 Service start-up and shutdown timeouts 86
H.2.4 A machines name need not match its hostname 86
H.2.5 Service script start-up and shutdown exit conditions 86
H.3 Changes in Version 2.6 86
H.3.1 Disabling service script timeouts 86
H.3.2 Single node clusters 86
H.3.3 Directory structure 86
H.3.4 RSF release file 87
H.3.5 New exit codes 87
H.3.6 Rsf_install 87
H.3.7 Licensing on Linux can use any Ethernet card 87
H.3.8 Controlling user name logging 87
H.4 Changes in 2.7 87
H.4.1 Opteron support 87
H.4.2 RSFCLI list extensions 87
H.4.3 Broadcast listener 87
H.4.4 Downloadable configuration file 87
H.4.5 Mount points 87
H.4.6 Configuration file fix 87
H.4.7 Rsfcli changes 87
H.4.8 Linux mount checks 87
H.4.9 New port number 87
H.5 Changes in 2.8 87
H.5.1 Differing CRC's in heartbeats 87
H.5.2 Check for RSF-1 running 88
H.5.3 Rsfcli updates 88
H.5.4 RSF-1 minimum service start-up timeout 88

List of Tables and Figures

Table 1 Typographic Conventions 11
Table 2 State transition table 13
Table 3 RSF-1 port numbers 18
Table 4 Global variables 35
Table 5 UI Hints 37
Table 6 Scripts in /opt/HAC/RSF-1/etc/service/scripts 48
Table 7 Scripts in /opt/HAC/RSF-1/etc/service/disks 49
Table 8 Scripts in /opt/HAC/RSF-1/etc/service/network 49
Table 9 Scripts in /opt/HAC/RSF-1/etc/service/tables 49
Table 10 Script exit codes 50
Table 11 Summary of rsfctl arguments 58
Table 12 RSF-1 syslog priorities 59
Table 13 rsfcli switches 60
Table 14 RSF-1 Service Script Environment Variables 76
Table 15 RSF-1 shell variables 77
Table 16 RSF-1 shell functions 77

Figure 1 Cluster: three node example 28
Figure 2 Read/Write offsets for shared discs. 31
Figure 3 Syntax for machine definition 39
Figure 4 Machine definition example 41
Figure 5 Syntax for service definition 42
Figure 6 Service definition example 45
Figure 7 Service directory for the example cluster 52
Figure 8 Testing the service scripts 52
Figure 9 Example access control entries 53

1 Preface

1.1 Scope

This manual describes release 2.6 and above of RSF-1 on all supported platforms. It does not cover the version 1 (two node cluster) releases of RSF-1.

The latest release of this manual is available via the web at http://www.high-availability.com/. Should you encounter problems using RSF-1, please refer to the appendices section at the end of the document before contacting your vendor or High-Availability.Com.

1.2 Who should use this manual

This document is both an easy installation guide and an extensive reference guide. This guide is for both the new RSF-1 user and the experienced RSF-1 user, offering full step-by-step installation and configuration instructions, as well as featuring extensive reference material by documenting all configuration features and support areas.

It assumes you have familiarity with your chosen operating system and hardware platforms, specifically in areas of:

- Shell scripting.
- System administration.
- Storage management.
- TCP/IP networking.
- Application support.

If necessary, refer to your system documentation for details of particular commands and procedures. Additionally, you may find the RSF-1 Applications Guide useful. Do not be perturbed by the size of this manual. RSF-1 is extremely simple to configure and use, although some of the basic High Availability concepts may be unfamiliar to you initially. This manual is intended to form a complete guide to all aspects of RSF-1. You will not necessarily need to know all the information contained within it.

1.3 Typographic conventions used in this guide

The majority of this manual is set in Bitstream Vera Sans 10pt

Important information is highlighted in boxes like this.

Certain words and phrases are picked out in different text styles, as shown in Table 1.

Table 1 Typographic Conventions

Typeface	Meaning	Example
`RSF-1`	Command, file or directory names; on-screen computer output.	The `/etc/hosts` file.
`RSF-1`	What you type; keyboard input	rsfhost1% **`rsfdump`**
`RSF-1`	Command line placeholder; replace with an actual name or value of the indicated type.	`vi` *`script_file`*
RSF-1	Object or element in graphical user interface.	Use the View menu.
RSF-1	Titles, new terms or important concepts.	***Floating IP addresses***.
RSF-1	Keywords in configuration file(s).	Edit the **MACHINE** definitions.

2 Overview of RSF-1

2.1 Concepts and definitions

This chapter describes some of the concepts used within RSF-1. It also defines common High Availability terms and some specific RSF-1 terms.

2.2 Introduction

RSF-1 (Resilient Server Facility) provides *High-Availability* (H.A.) failover for software applications running on server systems. High Availability means that a particular service is accessible almost continuously. The service should not be affected by local hardware failure, maintenance or local software failure.

Note that HA software does not protect the hardware components from failure. Indeed, we assume that the underlying host hardware may not be continually reliable. In the event of a hardware failure, the application will be migrated or reconfigured so that it can continue to run on operating hardware.

Under RSF-1, applications are part of *services* that can be run on any available RSF-1 *server* in a *cluster* without affecting the users accessing those applications. When a server fails, these services are migrated to a standby RSF-1 server, where they are restarted and continue running normally. We call this *migration & failover*.

Clearly, this implies that the applications you wish to use under RSF-1 must be independent of the underlying host systems. When a service migrates to another server, RSF-1 manages the following elements:

- **Network identity**, by configuring a new network interface with a floating address seen by the clients.
- **Application data**, usually by transferring the ownership of a shared disk or volume.
- **Application start-up**, by running the required initialisation code and procedures.

Effectively, RSF-1 implements a set of logical rules that decide when a service should be started and stopped and where it should run. The physical locations of the services are *transparent* to the users.

2.3 RSF-1 Terminology

Each RSF-1 installation is defined in the following terms.

2.3.1 Server/Node

A server or node is a host running the RSF-1 software, consisting of the monitoring programs and control utilities. It is also capable of running one or more services in a predefined set, simultaneously if required.

2.3.2 Cluster

A set of RSF-1 servers each communicating and able to participate in running a defined set of services is called a cluster. There can be from two to sixteen RSF-1 servers in a cluster. These servers are interconnected by means of various communication *channels*, through which they exchange information about their states and the services running on them *(heartbeats)*. Servers in a cluster that share services (and by extension, heartbeats) are called *siblings*.

2.3.3 Service

A service can consist of any combination of the following elements:

- A network identity for the service, by which external clients contact the service and interact with it. This is typically a unique IP address and hostname independent of any particular node on the network.
- Application data which can be accessed by every RSF-1 server that may run the service. This data is usually stored on a disk subsystem with connections to each of those servers. This disk storage is often referred to as shared disk.
- Application initialisation and shutdown code. These are often the scripts and programs supplied with an application to start it at system boot and stop it when the system is halted.

RSF-1 can control up to 200 services, each capable of running on some or all of the available servers dependent on the configuration. This allows every server to perform useful work and, in the event of a failure, other specified servers to take over the lost services.

2.3.4 Service states

A service will always be in one of several states on each server:

- **Unknown** – The service has never been in contact with the relevant server, typically this state occurs when RSF-1 is first starting up. This is typically a transient state and is quickly updated to one of the other states given below.
- **Stopped** - The service is not currently running on the server (it may be running elsewhere).
- **Starting** - The start-up sequence for the service is currently underway on the server.
- **Running** - The service is currently running on the server.
- **Stopping** - The shutdown sequence for the service is currently underway on the server.
- **Broken_safe** - A problem occurred while starting the service on the server, but it has been stopped safely and may be run elsewhere.
- **Broken_unsafe** – A fatal problem occurred while starting or stopping the service on the server. The service cannot be run on any other server in the cluster until it has been repaired.
- **Panicking** – A problem has been detected, and an attempt to take some corrective action is being made.
- **Panicked** – A fatal error occurred whilst running a service and it may not run on any other server in the cluster until it is repaired.
- **Aborting** – The start-up scripts for a service have failed, and the stop scripts are being run to try to make the service safe to run elsewhere.
- **Bouncing** – At user request, the stop scripts for a service are running. If they succeed, the service will be immediately restarted on this server.

During normal operation, the state of a service may go through several transitions (for instance on start-up it will go from **unknown**, to **starting** and finally to **started**). The following table lists the known states and the transition (with reason) that may occur to a state.

Table 2 State transition table

Initial State	Trigger Condition	New State
unknown		
	Finish start-up, read state files, or initialise start	any state

stopped[1]		
	User starts the service	starting
starting; start scripts running		
	Scripts OK	running
	Scripts request retry	starting
	Scripts fail	aborting
	Scripts exit broken_safe	broken_safe
	Scripts exit broken_unsafe	broken_unsafe
	Split brain detected	panicking
running		
	User bounce request	bouncing
	User stop request	stopping
	user restart request	starting
	Sysdown state entered	stopping
	Split brain detected	panicking
stopping; stop scripts running		
	Scripts OK	stopped
	Scripts request retry	stopping
	Scripts fail	broken_unsafe
	Scripts exit broken_safe	broken_safe
	Scripts exit broken_unsafe	broken_unsafe
	Split brain detected	panicking
panicking; panic scripts running		
	Scripts OK	panicked
	Scripts request retry	panicking
	Scripts fail	broken_unsafe
	Scripts exit broken_safe	broken_safe
	Scripts exit broken_unsafe	broken_unsafe
panicked		
	User repair request	stopped
broken_unsafe		
	User repair request	stopped
broken_safe		
	User repair request	stopped
aborting; abort scripts running		
	Scripts OK	broken_safe
	Scripts request retry	aborting
	Scripts fail	broken_unsafe
	Scripts exit broken_safe	broken_safe
	Scripts exit broken_unsafe	broken_unsafe
	Split brain detected	panicking

2.3.5 Important considerations about broken_unsafe mode

`Broken_unsafe` mode is triggered when a service, during shutdown (which can be during panicking,) fails to complete successfully and the scripts exit with an exit code indicating **`broken_unsafe`** (note that a failed shutdown can follow a failed or incomplete start-up). In effect this should be read as the service is broken **plus** it is in an unsafe, undetermined mode. When a service is in this state, RSF-1 sets all other cluster members to `manual` for this service.

[1] Service found which is:
In automatic mode.
Not active on any other servers (active states are running, started, stopping, bouncing).
Not in automatic mode on any other higher priority running server.

This behaviour is the default and it is highly recommended it is left this way for reasons explained below.

Should a service become **broken_unsafe** then the server administrator should investigate and fix the error before attempting to either restart the service or even reboot the host the service failed on. To explain the reasoning behind these recommended measures let us take an example scenario and work through the stages of possible failure. Initially we have the following set-up:

- A cluster with two machines, `personnel_master` and `personnel_slave`
- The hosts are connected to a dual ported SCSI disk array
- The hosts run a single Oracle RSF-1 service referred to as `AdminDB`, which utilises a database located on the shared disk
- The administrator has decided to shut down the service on `personnel_master` for routine maintenance. Because `personnel_slave` is in automatic mode it is assumed the service will fail over. However, due to an error in the shutdown scripts, the service fails to shutdown and is left in a **broken_unsafe** mode. Let us further assume the error occurred early on in the SQL shutdown scripts so there is still a database running on the shared disk. At this point a number of scenarios can occur:

  Corrective action is taken

  The server administrator investigates and fixes the error that caused the shutdown to fail. Running the scripts manually now shuts down the `AdminDB` service correctly. RSF-1 is now put back into automatic mode for the service and again it is shut down, this time successfully. The service is now in a known safe state.

  The failing server is rebooted – service in manual mode

  By rebooting the failed server it is probable that the service will be shut down to a manageable state, although if done by the operating system default shutdown methods (such as **SIGHUP**) then the state of open transactions etc may be lost. Because the other servers in the cluster are set to manual then the onus is on the server administrator to restart the service. Note that also there is still the need to fix the shutdown scripts.

  The failing server is rebooted – service in automatic mode

  Although by default RSF-1 will set all other servers to manual mode for the *broken_unsafe service only*, there are certain situations where automatic mode may be enabled:

  1. A system was down when RSF-1 attempted to set the service manual mode and therefore automatic mode remains in persistence.
  2. A user of the system has switched the service from manual to automatic.
  3. RSF-1 is configured not to switch the service to manual mode when a **broken_unsafe** state occurs.

  This scenario could lead to possible corruptions and should be avoided in favour of taking corrective action. To understand why, consider the following when applied to the Oracle database:

  - Host `personnel_master` is shutdown as part of a reboot sequence.
  - RSF-1 exits immediately (as it no longer has any services active)
  - Host `personnel_slave` immediately starts up the AdminDB service as it is now no longer running anywhere
  - Host `personnel_master` is still shutting down the Oracle database and syncing disks
  - The database and filesystem become corrupt as they were being accessed on two hosts simultaneously

- In conclusion, whenever **`broken_unsafe`** mode is encountered the recommended corrective action is that documented in the first scenario **Corrective action is taken**.

2.3.6 Primary and Secondary Servers

Each RSF-1 service has a designated primary and one or more secondary servers. In fact, this differentiation is largely semantic; it has little effect on the operation of a service.

The *primary* server is the host on which the service normally runs. After a simultaneous reboot of all servers, the service will typically start up on the primary server.

A *secondary* server is an alternate host, or hosts (as there may be several servers for each service), on which the service is capable of running. A secondary usually acts as a backup to a primary server and possibly other sibling secondary servers. It will start the service if it is enabled and detects that other servers are not running it.

The order in which server lines appear in the RSF-1 configuration file dictate a priority in which servers will run the service. The primary server is the one at the top of the list of servers; if a service fails to run on the primary server, then RSF-1 attempts to start it on the next one in the list, and so on until the list is exhausted. In this way you have the ability to control the ordering of service failover.

Beyond that, assuming that the hardware is similar, there is no difference between a service running on a primary or secondary server. Its location is completely transparent to the users of that service.

It is inaccurate to identify particular hosts as primary or secondary servers without specifying a service, as the terms are bound to a service, not a server. A server can be *both* a primary and a secondary, for different services.

If a server is the primary for a service, that service is said to be a *primary service* on that server. Similarly, if a server is the secondary for a service, that service is said to be a *secondary service* on that server.

There are configured *timeouts* for each service on each server that define how long they must wait when a service is unavailable and no other RSF-1 server is running it, before starting the service. This delay has two purposes:

1. It prevents a secondary server from starting a service ahead of a primary server that may still be in the process of rebooting.
2. It allows recovery from brief host outages and transient conditions, such as overloading, accidentally disconnected cables, halts and even reboots (if the timeout is long enough) without triggering failover.

2.3.7 Automatic and manual switchover mode

The *switchover mode* defines whether or not a server will attempt to start a service when it is not running. There are separate switchover mode settings for each server that can run a service. The combination of server and service is known as an instance.

The switchover modes can be set to *automatic* or *manual*. In automatic mode, the server will attempt to start the service in question when it detects that no sibling server in the cluster is available or running it. In manual mode, it will not attempt to start the service but will generate warnings when it is unavailable. If the server cannot obtain a definitive answer regarding the state of the service (because it cannot contact its siblings in the cluster) or the service is not running anywhere else, the appropriate timeout must expire before any action can be taken.

The primary service switchover modes are typically set to automatic to ensure that a server starts its primary service(s) on boot up. The secondary service switchover modes are typically set to automatic to ensure that services will be taken over in the event of a failure. Alternatively, the secondary switchover modes can be left in manual, allowing the administrator to initiate failover themselves. Note that putting a service into manual mode when the service is already running does not stop that service, it only prevents the service from being started on that server.

2.3.8 Blocked and unblocked mode

Rsfmon implements the notion of each instance of a service being 'blocked' – i.e. unable to start. The normal state of an instance is 'unblocked', so that it can start as required. This state is similar to the automatic / manual state, but is intended to make it possible for other programmes in the RSF-1 suite to control a service, where the automatic / manual state are for (human) administrators to exert control. In operation, an instance of a service may only be started when it is both automatic and unblocked.

In the past, services were always marked as being unblocked when rsfmon was (re)started. A change has been made to allow the blocked state to be explicitly set, or the old state preserved, when rsfmon starts. To control this rsfmon now recognises a "-b" option, which takes a single letter argument:

-b <arg> set service blocked states according to <arg>, as follows:

- b force all services to be blocked
- u force all services to be unblocked
- p preserve blocked state from the last known setting

The reporting of the mode of a service has been extended to shown the new modes as one of the following:

```
auto / blocked
auto / unblocked
manual / blocked
manual / unblocked
```

auto / unblocked will behave as the old auto mode; the other three modes will behave like the old manual mode.

2.3.9 Heartbeats

Heartbeats are small packets of information exchanged between two or more RSF-1 servers. The heartbeats indicate whether a server is still up and what services it is running. They are key to detecting failures in a cluster. A server capable of running a particular service (i.e. a primary or secondary for that service) must have *direct* links to each and every other server capable of running that service (i.e. all its siblings).

Because RSF-1 servers must be certain that a sibling is down before taking over its services, RSF-1 is normally configured to use several communication channels through which to exchange heartbeats. Only the loss of *all* heartbeat channels represents a failure. If a server wrongly detects a failure, it may attempt to start a service that is already running on another server, leading to so-called *split brain syndrome*. This can result in confusion and data corruption. Multiple, redundant heartbeats prevent this occurring. If no services are shared between two particular servers, then no direct heartbeats are required between them. However, at least one heartbeat must be transmitted to each member of a cluster for control and monitoring requests to be propagated.

Note that siblings must not be a subset within a cluster. If a group of siblings share services but none of them share a service or exchange heartbeats with any other server in the cluster, this is effectively a subset within the cluster. This subset cannot communicate with the rest of the cluster and hence cannot be properly managed.

There is no benefit to be gained by creating subsets like this; if required, they should be entirely separate clusters sharing no configuration details.

Heartbeat channels should be independent to ensure that a partial failure of one element does not result in the loss of all heartbeats. RSF-1 lets you use three different types of channels: serial links, disk partitions and network connections. You may use more than one of each heartbeat type, and in fact should use multiple heartbeats for resilience.

Note that the purpose of the heartbeats is to monitor servers and services, not to monitor the status of the links themselves; heartbeats cannot be used to verify network or disk connectivity. Separate agents are required for this function.

All these heartbeats use dedicated protocols. They carry information about the state of the cluster nodes, rather than simple request/response packets.

2.3.9.1 Structure of a Heartbeat Packet

Every heartbeat packet contains the following information:

- Status of all services on the originating server.
- Request for remote RSF-1 monitoring to terminate, if applicable.
- Incrementing sequence number.

In themselves, the heartbeat packets also act as an indication that the originating end is still functioning correctly.

Despite the information encoded in them, the heartbeat packets are very small – from a few tens of bytes on a small cluster, up to around 1500 bytes on a large cluster. They do not consume much processing power or bandwidth.

2.3.10 Network ports used by RSF-1

RSF-1 listens on two network ports; their numbers and use are given in Table 3. Ports are looked up in the `/etc/services` file first, and if not found default to those given in the table. If necessary, the ports RSF-1 listens on can be changed by modifying the entries in the `/etc/services` file on **all** nodes in the cluster.[2]

Table 3 RSF-1 port numbers

Port Number	service name	Description
1195	rsfnet	Used to transmit and receive heartbeat packets.
1195	rsfreq	Used to listen and then respond to control and monitoring requests.

2.3.11 Floating addresses (VIP's)

Each service has an associated network identity in the form of an IP address and name. These addresses are not tied to particular hosts, but float between the RSF-1 servers depending where the service is running.

Floating addresses are implemented using *virtual network interfaces*. These allow a single physical network interface to respond to several different IP addresses. These addresses can be added or removed at any time without interrupting connectivity.

Most recent operating systems support the use of virtual interfaces, sometimes called *IP aliases*.

When a service is started on a server, RSF-I configures its floating address on the public interface in addition to the 'fixed' server address. From then on, requests for that service from clients are directed to that server (see the section on Gratuitous ARP below). Note that on routed networks some reconfiguration of the network may be required in order for remote clients to access floating addresses on all servers in a cluster – check with the site network administrator.

When installing RSF-1, clients should be configured to use a *new* floating address, rather than the fixed machine address. Later, during failover, the clients will not need to be reconfigured.

2.3.11.1 Gratuitous ARP

Hosts that communicate with each other on a local network (i.e. not through a gateway or router) do so by mapping IP addresses to MAC[3] addresses. Hosts translate IP addresses to MAC addresses by means of an internal table known as the ARP cache (ARP caches are maintained on a per host basis). Generally, when an IP address is configured on a host (as is what happens when RSF-1 configures a floating address on service start-up or failover,) the host broadcasts out an

[2] You will need to restart all RSF-1 instances at the same in order for the change to take effect.
[3] Media Access Controller

update[4] so that other hosts can update their ARP caches. In this way, when a failover occurs, clients quickly direct packets to the new host.

However, some operating systems (variants of Linux for example) do not issue this gratuitous ARP and clients have to wait till the old ARP cache entry times out (typically 20-30 seconds) before they can access the new host. RSF-1 can avoid this problem by issuing its own gratuitous ARP if necessary; see the `hac_grarp` command for details.

2.3.11.2 Multiple network interface monitoring

For each running service, RSF-1 can monitor the network interface which the associated VIP is bound to. If the interface becomes unavailable, then RSF-1 will check the remote host to see if it has connectivity (the remote host will report through heartbeats whether or not it can run the service – see below). If the remote node has connectivity, then a failover will be initiated in order to allow the service to remain highly available.

It is possible to specify multiple VIPs for a single service, and these VIPs may be bound to multiple physical network interfaces. It is therefore desirable to have the ability to monitor each of these interfaces for connectivity.

Network interfaces are defined in the config file with the keywords `IPDEVICE` and `EXTRA_IPDEVICE`, using the format `'IPDEVICE "<device_name>"'` or `'EXTRA_IPDEVICE "<device_name>"'`. If these interfaces are to be monitored, then the lines in the config file should also include the keyword `MONITOR_TAG` following the device name:

```
IPDEVICE "<device_name>" MONITOR_TAG <tag name> , or
EXTRA_IPDEVICE "<device_name>" MONITOR_TAG <tag name>
```

The names used in these tags are in a unique, dedicated namespace, and therefore duplicate tag names are prohibited. Note, `MONITOR_TAG` names may be the same as service or machine names, although this could cause confusion, and should be avoided.

The "`MONITOR_TAG name`" part of each line defining the interface is optional, and if absent the interface will not be monitored. `IPDEVICE` and `EXTRA_IPDEVICE` are also optional, as a service not associated with any network interfaces is legitimate in a configuration file.

Note that the `MONITOR_TAG` keyword should NOT be used for the default IPDEVICE in the service (see section 7.4.5.4). It should only be used in lines after a `SERVER` line, i.e. `MONITOR_TAG` keywords must only be used on server specific IP device lines.

Monitoring information is used to compute a single 'blocked' or 'unblocked' state for each instance of a service. A service which is 'unblocked' may be started and run as usual. A 'blocked' instance of a service cannot run successfully, and so a service will never be started on a machine where it is blocked. It can, however carry on if it is already running before the server changes its state to 'blocked'. If a service is running when it becomes blocked, then a check is made to see if it can run elsewhere, and if so the blocked service will be stopped (and therefore failover to an available server). If there is no other server which can run the service (all other nodes are down or have that service marked as blocked, in manual mode etc.) then the service is left running. Action will only be taken when a server becomes available with the ability to run the service.

Note that when a server checks to see if a service can be run elsewhere, if it finds more than one possible remote server, it does not select the 'best' choice based on the number of available interfaces, as this could result in a service being moved to a server which does not have the capability (CPU power, etc.) to run it. Rather, it selects based on the order in which the servers are listed in the config file (see section 2.3.6).

Finally, in the config file, after the `IPDEVICE` and `EXTRA_IPDEVICE` lines, there can be an optional `MONITOR` line. If present, this line contains a boolean expression describing what monitored conditions are required for a service. If the expression evaluates to TRUE the service can be run, and is FALSE the service is blocked. The expression is composed of names appearing as `MONITOR_TAGs`, parenthesis, the operators `AND`, `OR`, `EOR`, `XOR`, `NOT`, and the constants TRUE and FALSE. If no MONITOR line is present, the service will never be blocked.

[4] Known as a Gratuitous ARP

Formally, the syntax is:

```
exp:                                    operator:
      name                                    AND
      '(' exp ')'                             EOR | XOR
      exp operator exp                        OR
      NOT exp
```

The following extract from a config file (extract contains the service section for a single service only - not a full config file) illustrates this functionality:

```
SERVICE service1 192.168.55.2 / 255.255.255.0 "Corp service 1"
 INITIMEOUT 20
 RUNTIMEOUT 12
 SERVER nodea
 IPDEVICE "igb0" MONITOR_TAG a0
 EXTRA_IPDEVICE "igb1" MONITOR_TAG a1
 EXTRA_IPDEVICE "igb2" MONITOR_TAG a2
 MONITOR a0 AND (a1 OR a2)
 SERVER nodeb
 IPDEVICE "igb0" MONITOR_TAG b0
 EXTRA_IPDEVICE "igb1" MONITOR_TAG b1
 EXTRA_IPDEVICE "igb2" MONITOR_TAG b2
 MONITOR (b0 AND b1) OR (b1 AND b2) OR (b0 AND b2)
```

2.3.12 Understanding how RSF-1 uses service timeouts

In RSF-1 terms, a service time out is a value that dictates how long (in seconds) RSF-1 will wait before starting a service. There are differing conditions (and differing timeout values that are considered) that effect how RSF-1 performs. These are:

Heartbeat timeout

When taking over a service on a failed machine, first the machine failure must be detected. This is done by counting consecutive missed heartbeats, and will be several poll times. No log events are generated until this period elapses, though missing 2 or more heartbeats then resuming without deciding the remote machine is down will be logged as a missed heartbeats message.

Init timeout & run timeout

When one or more of the servers is out of contact, the configured timeouts are always used. If the service has never been seen to be active on any server, `initimeout` is used, and if it has been seen to be active, then `runtimeout` is taken.

2.4 Scope of RSF-1

Confusion sometimes arises over the benefits of implementing H.A., and its capabilities. This section clarifies some typical misunderstandings.

No assumptions are made about hardware reliability

RSF-1 provides high availability only when there is redundant hardware available to continue running a service. You should ensure that your disk storage is protected by means of techniques such as mirroring and backups, and that your network is reliable.

Applications only run on a single node at any time

Although a shared disk is often used in the same RSF-1 clusters, this does not imply simultaneous shared access to the same data by several servers in the cluster. You can enable a form of joint access by sharing your data via NFS to each server and implementing a locking mechanism within your application if necessary. Alternatively, your application may natively support *distribution*. But only one host is allowed direct access to the disk at any time.

You may still run multiple, *different* instances of the same application, which use data on different file-systems, on separate nodes. These must be implemented as separate services.

Load balancing

RSF-1 can be used to easily migrate applications to idle servers and take advantage of unused capacity for intensive processing. This process is controlled by the administrator.

It may also be possible to segment an application into several services running on different servers; e.g. a database backend accessed by a middleware product on another server. This is only practical if a failure of any one component does not cause failure of the others.

Distributed processing

The term 'cluster' is sometimes used to refer to distributed computing, or 'high performance' clusters, in which processes are managed collectively across several hosts. This is an entirely different form of clustering, which is not implemented by RSF-1.

Not all applications must be under the control of RSF-1

A host running RSF-1 can also perform other duties without special arrangements. You can run additional applications outside the control of RSF-1 (and hence not highly available,) without any interaction between them. However, this is inadvisable if the other applications may in any way jeopardise the stability of the server.

Transfer of state depends on the application

The application must be capable of recovery or restart following interruption, for processing state to be maintained following failover. (Stateless protocols such as HTTP do not require this capability).

In practice, an application that can survive a crash on a single server (a desirable trait!) will usually migrate to a new server without serious problems.

RSF-1 monitors servers

Servers are monitored only via the heartbeat links. The heartbeats are not intended for monitoring the actual channels they use, only to indicate that the originating server is up.

RSF-1 monitors and responds to failures of server hardware. Agents available from High-Availability.Com can be incorporated with RSF-1 to provide application monitoring. Similarly, network links can also be monitored using the product NetMon; as with agent monitoring NetMon integrates seamlessly with RSF-1.

2.5 Conclusion

Now that you understand the basic terms and concepts underlying the operation of RSF-1, read the next chapter to learn about the various components of RSF-1 and how they support this functionality. If you need a reminder, refer to the glossary in Appendix A.

3 Components of RSF-1

3.1 Chapter overview

This chapter provides a general overview of the various components in RSF-1 and explains how they are related.

3.2 What the components do

RSF-1 contains programs and files to configure services, start-up and shutdown services safely, monitor services and hosts, and control services and failovers.

3.3 The configuration file

RSF-1 uses a single configuration file to specify the characteristics of servers, heartbeats and services. These include:

- Identifying names.
- Service timeouts.
- Floating addresses.
- Heartbeat channel parameters and endpoints.

The configuration files must be identical on every server in a cluster. If there is no configuration file present, RSF-1 will not run.

3.4 Service start-up and shutdown

Each service has a dedicated subdirectory on every participating server, containing scripts for:

- Taking, checking and mounting the disks containing application data for the service.
- Initialisation steps such as renaming files or clearing locks.
- Starting application programs or daemons (located elsewhere).

During service shutdown, the opposite tasks to these are performed in reverse order.

The service scripts are arranged in a similar way to the SVR4 Unix system boot & shutdown scripts. You must provide and install these scripts yourself. Templates for common service tasks are supplied on the RSF-1 distribution media, while application scripts can often be moved unchanged from their normal installation directories.

The service scripts are run by a special handler within RSF-1 called `rsfexec`. This program is run when RSF-1 determines that a service should be started or stopped. It uses the exit codes from the scripts to decide how to proceed. It also configures the floating network identity of the service automatically.

The service scripts must do the same things on all RSF-1 servers that are capable of running the service. RSF-1 does not provide any mechanism for synchronising these files across several servers. Later, we suggest methods of performing this.

3.5 Heartbeats and monitoring

The `rsfmon` daemon sends out heartbeat packets and listens for incoming heartbeats.

It also decides when a service should be started, failed over, or stopped. It bases this information on the present switchover states, the current status of other RSF-1 servers and requests received from the control console. It is very much the central component of RSF-1.

rsfmon spawns multiple copies of itself to serve each configured type of heartbeat. The parent process will restart any children that die. If the parent process dies, the children will also exit.

rsfmon listens for network heartbeats on a single UDP/IP port. It listens for control requests on a separate TCP/IP port. The default numbers of these ports can be altered by the administrator, but must be the same for all servers in a cluster.

3.6 Controlling RSF-1

RSF-1 is controlled and monitored by the administrator through a Java-based graphical console (rsfgui) that will run on any compliant Java-enabled platform. This console can be run remotely on non-RSF-1 hosts.

The console allows the administrator to view the current status of each service, to control the switchover modes, to shutdown RSF-1 for maintenance and to start, stop or move particular services.

There is also a command line interface available, rsfcli, which provides similar functionality within shell scripts.

3.7 RSF-1 log file

rsfmon logs all its actions to a file called rsfmon.log located in the directory /opt/HAC/RSF-1/log. It also logs via the UNIX syslog mechanism, which allows the messages to be filtered and redirected to remote hosts or particular users.

The log should be your first resort when you are unsure how RSF-1 is behaving.

4 Service failover definition

4.1 Chapter overview

This chapter describes how RSF-1 operates and what happens when a server fails.

4.2 RSF-I initiation

RSF-1 runs automatically on boot. The main `rsfmon` daemon is started and reads the RSF-1 configuration file. It then forks additional copies of itself for each configured heartbeat type.

These child processes send out heartbeats and poll for incoming heartbeats from sibling servers at regular intervals. Once these are received, RSF-1 can establish which services are running on the siblings.

If a service is not active on any sibling, and its local switchover mode is set to automatic, and there is no higher priority server ready to run that service in the cluster, it is started on the local server by executing the service start-up scripts[5]. The rsfmon daemons send out heartbeats informing the other servers which services are running locally.

If RSF-1 cannot establish contact with the siblings for a service, it begins to count down the initial timeout for that service. If the service timeout expires, and the switchover mode is automatic, the service is started.

4.3 Service control

While the parent RSF-1 monitor daemon is active, it listens for requests on the request network port. There are various types of requests that can be made, such as requests for details of last received heartbeats, currently running services, log messages, service shutdowns and switchover mode changes. These requests are generated by either the administration console utilities or the command line program `rsfcli`.

4.4 Host monitoring

Recall that RSF-1 servers monitor each other by exchanging heartbeat packets. The heartbeats are sent simultaneously on all configured channels at regular intervals, and are expected to be received from sibling servers within those intervals.

RSF-1 only recognises a server failure when all the heartbeats from it are lost. There is no other circumstance in which a failure is flagged by RSF-1. Loss of a single heartbeat generates a warning message but is not critical as long as at least one or more heartbeat between these hosts is still active.

RSF-1 supports monitoring and failover of applications using separate agents supplied by High-Availability.Com. You may also write your own monitoring programs that can instruct RSF-1 to restart a service when an application fails (see the RSF-1 Applications Guide).

The loss of all heartbeats does not immediately trigger any action. Instead, RSF-1 on the highest priority available sibling for each service begins to count down the local timeout(s) associated with the service(s) running on the remote host, while continuing to poll for heartbeats. Each of these siblings also logs the failure and the remaining time at intervals. If any of the heartbeats return within the timeout period, normal operation is resumed and the timeout is cancelled.

If the timeout on a sibling expires, RSF-1 assumes that the service is dead. When this happens, RSF-1 initiates a start-up of the failed service.

[5] Note that if the server starting a service is the only one in the cluster currently running, the service start-up timeouts will be honored. This is the normal action RSF-1 takes. If RSF-1 can determine that the service is not being run on any other servers (i.e. all servers in the cluster are communicating) then this timeout is bypassed.

Should the service fail from its primary server to a secondary, and then the primary recover, it will be taken back should the secondary then fail; i.e. the service does not 'cascade' to lower priority servers by default if a higher priority server is available.

4.5 Failover

Service failover is accomplished simply by starting up a service as normal on the new server:

- The service floating IP address and hostname is configured as a virtual interface on the public network interface. The Address Resolution Protocol (ARP) ensures that the new MAC address for the service is broadcast to clients - see section 2.3.11.1 Gratuitous ARP for more information. Traffic redirection is transparent and clients do not need to be rebooted or reconfigured.
- Any disks required for the shared data area are taken, checked and mounted.
- Application recovery and initiation scripts are run.

Any further recovery steps must be handled by your applications. Although you may run recovery procedures as part of the service start-up, either you or the application must provide them. You can leave the secondary switchover modes in manual if you wish to control application start-up manually.

Once the application has started, clients can resume or restart processing. Until that time, client applications may hang or exit depending on their implementation. For example, NFS clients will display the familiar "`NFS server not available, still trying`" message until the file-systems become available again, at which point normal operation is resumed.

4.6 Server recovery

When the services have recovered and are safely running on standby servers, you will be able to repair the failed host. This may involve calling out an engineer or replacing internal components. When the host is repaired, you can reboot it without any special measures. The service MUST NOT be run on this server without stopping it elsewhere.

When RSF-1 is restarted on the recovered server, it immediately contacts its siblings and establishes whether the services are running. Assuming they are, *it will not take any further action*.

After a fail over switchover modes remain at their previous settings. At this point, there is no requirement to fallback or otherwise alter the services. There is no difference within RSF-1 whether a service is running on a primary or secondary server. However, if many services are now running on a standby server, or a secondary server is less powerful than the primary, you may experience degraded performance from your applications. For these reasons, many users prefer to *fail-back* a service to its primary server.

Fail-back is accomplished by shutting down the service on the current server, thus allowing another server to start it up again. If a service is starting on a server which is in contact with all the siblings for that service, the normal timeout, intended to allow other servers to regain contact, is bypassed and thus the fail-back occurs immediately and safely. The service is shut down using the normal procedures, the disks released and the floating hostname un-configured.

To ensure that no confusion can occur, manually stopping a service on a server also sets the service switchover mode to manual, preventing the service from starting on this server again. Start-up then occurs on the highest priority available server (assuming the switchover mode is automatic). The normal timeout is bypassed if all siblings are in contact. A failover when in contact with all the servers can only occur as a result of user action. The 'next available' is the first one listed in the configuration that is up and has the switchover mode for the service in automatic; normally, this should be the primary server.

Because this process may cause interruption to your clients (as mentioned previously,) you should consider defering it to a convenient time.

5 Software Requirements for RSF-1

5.1 Chapter overview

This chapter describes additional software that may be required by RSF-1. This includes software and configuration involving the operating system, network, disk management and applications.

5.2 Operating system

RSF-1 is supplied as packages of pre-compiled binaries for each supported operating system and hardware platform. Before installing it, you should ensure that you are using a supported release of your operating system. Please visit http://www.high-availability.com/links/26-0-try_buy.php for the latest supported system packages.

5.3 Network

You will require the following:

- A static IP address and interface on each connected network for each.
- An unused IP address and associated name on your local network for each service. This should be the public network and every server for the service must have an interface on it.

All the assigned hostnames and IP addresses should be entered into your network naming service or local hosts' files. The IP network numbers in use should have appropriate entries in the local `/etc/netmasks` or `/etc/inet/netmasks` file, especially if you are using subnets.

5.4 Disk management

Disk management software typically allows you to combine physical disks into logical volumes via concatenation or striping, and to mirror these volumes for resilience. It may also contain features to allow groups of disks to be shared between several hosts[6].

You do not necessarily have to use disk management software on your chosen platform, but it is often advantageous to do so. Details of supported disk management solutions for each platform are given in the RSF-1 *Applications Guide*. You should select, install and configure any such product before installing RSF-1. Configuration usually requires an understanding of the RSF-1 services and what file-systems are required to support them.

The file-systems your applications require should all be held on disks directly connected to the servers. It is unwise to rely on file-systems remotely mounted off other servers, unless these servers are also highly available.

5.5 Applications

Because of its unique flexibility, RSF-1 can provide high availability for many off-the-shelf applications without adaptation. However, this often depends on the suitability of the application.

Applications that are not tied to particular nodes or physical network addresses are most easily failed over by RSF-1. Often, applications are licensed for use on individual machines. In this case, you will require a licence for each RSF-1 server.

Applications that do not preserve any concept of *state* are most easily failed over without interruption. Examples include NFS servers and HTTP daemons (web servers,) both of which use *stateless* protocols. Where a persistent state or session is established between an application and its users, for example database connections or remote logins, these sessions must be re-

[6] This may require additional hardware and does not imply simultaneous shared access.

established after the application has failed over. Transactions that were in progress at the time of the failure must usually be resubmitted.

Applications that use the node name of the host they are running on or do not interact well with virtual hostnames may require additional configuration and special arrangements for failover. For example, in the case of a NIS server, this will mean changing the node name of the host and restarting the RPC port mapper, or rebuilding the NIS maps from source. Although these procedures can be scripted, when recovering a failed host you must be careful to prevent conflicts with the replacement NIS server.

The application must be capable of recovering from sudden failure, preferably without intervention. For example, most database engines are able to rollback failed transactions and return the database to a known, consistent state. If your application may require assistance, you should set the secondary switchover mode for the service to manual and use the logs to alert you when a failure has occurred. You can then oversee the failover at the console. As already noted, when an application is stateless, no recovery is necessary.

In certain cases, users may prefer to have multiple instances of an application, running on several RSF-1 servers. One instance acts as the *live* server in active use, and the others form *standby* servers, ready for immediate use. RSF-1 will ensure that network traffic from clients only reaches the current live server[7]

5.5.1 Installation

Applications typically consist of two parts:

1. Binary files, configuration files, documentation and other static elements, usually installed as part of the installation process.
2. Data areas, table spaces, scratch areas or other dynamic data required by the application and its users in normal operation.

You can choose to install both these parts on the shared disk storage, or you can install the static part on each RSF-1 server and the dynamic part on the shared disk. Which you choose depends how easily the static files can be shared between two separate systems and which method you prefer.

Installing the entire application on the shared disk means that you only have one copy to maintain and configure. However, check that the application does not install additional system-specific components such as kernel drivers or */etc* files, or perform other system-specific installation tasks. You will have to duplicate these parts on all the siblings in the cluster. Also, you may encounter problems if the application locks some of its files while running.

Storing only the data areas on shared disk leaves you with two entirely independent copies of the application. This may be desirable if you wish to carry out host-specific configuration, for example, when using heterogeneous systems. It is also unavoidable when the application is contained within the operating system, such as NFS.

In either case, you may need to perform additional post-installation procedures to reconfigure your application accordingly. Consult your application vendor or manuals.

Remember that all the data your application requires must be capable of migrating to the standby system. This usually means that it must be located on shared disk storage and must not contain host-specific references.

Some applications provide other means to synchronise data between multiple instances. In this case, you may not need a shared data area. RSF-1 does not mandate the use of shared disk.

Additional Help

High-Availability.Com has a wealth of experience in configuring RSF-1 to work with a wide variety of different applications. Much of this experience is documented within the RSF-1 Applications Guide, but please contact us if you need further advice.

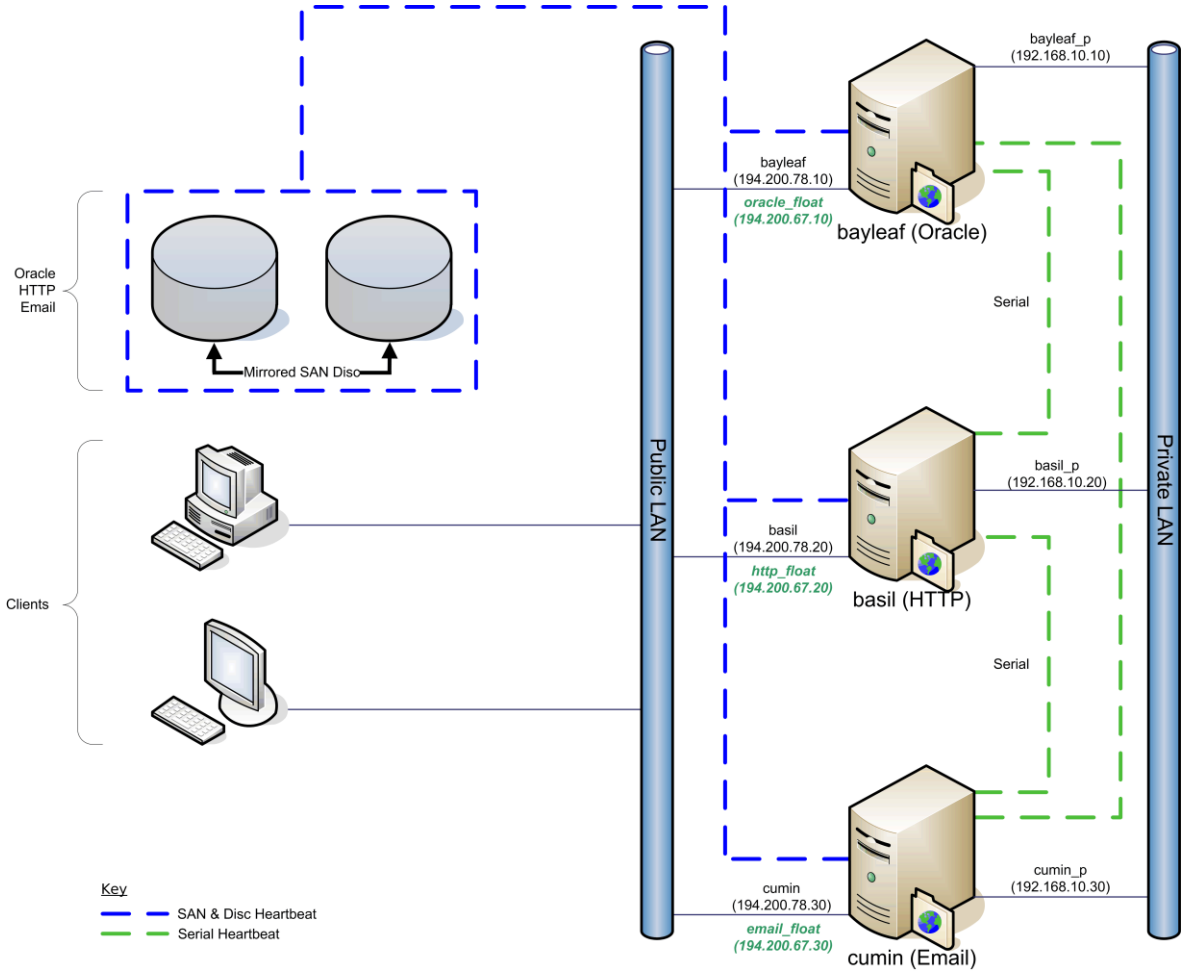
[7] RSF-1 does not provide facilities for automatic load-balancing across multiple servers, although it can be used within such scenarios to redirect requests for unavailable servers.

6 Building and Configuring a Cluster

6.1 Chapter overview

This chapter walks through the process of building a cluster configuration from scratch. It is based on the conceptual model depicted in Figure 1, consisting of three nodes with three distinct data services (applications) – understanding how to configure a cluster from this conceptual model will provide the knowledge of how to apply it to your own cluster requirements.

Figure 1 Cluster: three node example



All nodes in the cluster are attached to both a public and private LAN. Clients connect to the servers via the public LAN. All the cluster nodes have access to a mirrored SAN containing a filesystem for each of the data services (each filesystem mounted by a single node when running that service).

Nodes monitor each other via four different communication channels: a serial link, the shared SAN, the public network and a private network. Should any server fail all communication channels to the other nodes will be lost and one of the remaining servers will resume the services the failed machine was providing.

The RSF-1 package contains the full configuration file being created in this section. It is located in the directory `/opt/HAC/RSF-1/etc/` as `config-example.txt`. Disc and serial port paths in the example are specified generically, i.e. `/dev/hb_serial_port_1` - host specific naming conventions should be substituted when interpreting paths. Please refer to vendor specific documentation for these conventions.

6.1.1 Network setup

There are two network segments involved: the public LAN (which connects the servers to their clients) and a separate, optional, private network connecting the servers using a second switch and their private interfaces. The public LAN is the 194.200.78.0/24 network. The separate private network uses the reserved Internet 192.168.10.0/24, address range. Finally, access to RSF-1 services on these hosts is via the 194.200.67.0/24 network. To ease configuration, the same host part of the network number is used for addresses on each server; it is recommended you follow this convention.

It is important to understand that the 194.200.78.0/24 and 192.168.10.0/24 addresses are permanently assigned to the interfaces on their relevant hosts. RSF-1 does not perform this task and as such the names and addresses should be entered into the local naming service and the local hosts file on each server and the operating system configured to assign them on boot.

In contrast the 194.200.67.0/24 addresses are paired with RSF-1 services and **not** the hosts themselves. RSF-1 configures (and removes) these addresses on a host where the associated service is running or being stopped.

6.1.2 Serial setup

Serial communication between the hosts is configured over point to point null modem cables directly attached to unused serial ports on the hosts. RSF-1 uses its own transport protocol for communications rather than relying on underlying protocols like PPP; it is therefore not necessary to configure any services on these serial ports.

6.1.3 Disk setup

In the example all the hosts in the cluster have access to a mirrored SAN partition over a fibre storage network.

RSF-1 uses its own pseudo filesystem tables located in the `/opt/HAC/RSF-1/etc/fs` directory. These tables are specific to each RSF-1 service and have a naming convention of *`tablename.servicename`*. The *`tablename`* is dependent upon the operating system RSF-1 is running on as different systems have different filesystem table formats and RSF-1 needs to support these different formats.

Note that it is not necessary to have any RSF-1 controlled filesystems in the standard filesystems tables (i.e. `/etc/fstab`) – indeed it is recommended that references should be removed to avoid confusion and possible incorrect mounting or unmounting.

Filesystems requiring exporting before clients can access them should be declared in files located in the directory `/opt/HAC/RSF-1/etc/fs`. The file `/opt/HAC/RSF-1/README.examples` contains descriptions and locations of Operating System specific export tables used by RSF-1.

6.2 Building the configuration file

The first logical section of the configuration file declares the cluster name along with any global options to be declared. Following this the second logical section declares nodes in the cluster and the heartbeat communications between them. Initially defining the nodes and cluster a name gives:

```
#
# config-example.txt
CLUSTER_NAME three_node_example

MACHINE bayleaf
MACHINE basil
MACHINE cumin
```

As all machines in the cluster are attached via a public network, the configuration file is expanded to include heartbeats between nodes on that network. This is done using the NET keyword, followed by the host name the heartbeat is directed to (which must appear in a MACHINE declaration). Adding the heartbeats for just bayleaf to communicate with basil and cumin would be made to the configuration file as:

```
#
# config-example.txt
CLUSTER_NAME three_node_example

MACHINE bayleaf
 NET    basil
 NET    cumin

MACHINE basil
MACHINE cumin
```

The model in Figure 1 is also configured with a private LAN; secondary network heartbeats can therefore be added to take advantage of this. To ensure routing of the heartbeats over this network, the remote ***private*** address of the heartbeat endpoint is appended to the network keyword:

```
#
# config-example.txt
CLUSTER_NAME three_node_example

MACHINE bayleaf
 NET    basil
 NET    basil basil-p
 NET    cumin
 NET    cumin cumin-p

MACHINE basil
MACHINE cumin
```

Note that when the NET keyword only specifies a machine name with no remote endpoint (as in the case of the first NET entries for basil and cumin), then the end point address is taken to be the machine name itself.

Next the serial lines are added to the configuration using the SERIAL keyword, followed again by the host name the heartbeat is being sent to, and finally the serial port to send the heartbeat down. In this example a generic name of `/dev/hb_serial_port_` is used to identify the serial port:

```
#
# config-example.txt
CLUSTER_NAME three_node_example

MACHINE bayleaf
 NET    basil
 NET    basil basil-p
 NET    cumin
 NET    cumin cumin-p
 SERIAL basil /dev/hb_serial_port_1
 SERIAL cumin /dev/hb_serial_port_2

MACHINE basil
MACHINE cumin
```

Finally disc heartbeats are added using the DISC keyword. Again the host name follows this keyword and finally the raw disc device itself. The disc heartbeat media is in fact a small partition on shared disc that is reserved exclusively for RSF-1 use. It is declared as follows:

```
#
# config-example.txt
CLUSTER_NAME three_node_example

MACHINE bayleaf
 NET    basil
 NET    basil basil-p
 NET    cumin
 NET    cumin cumin-p
 SERIAL basil /dev/hb_serial_port_1
```

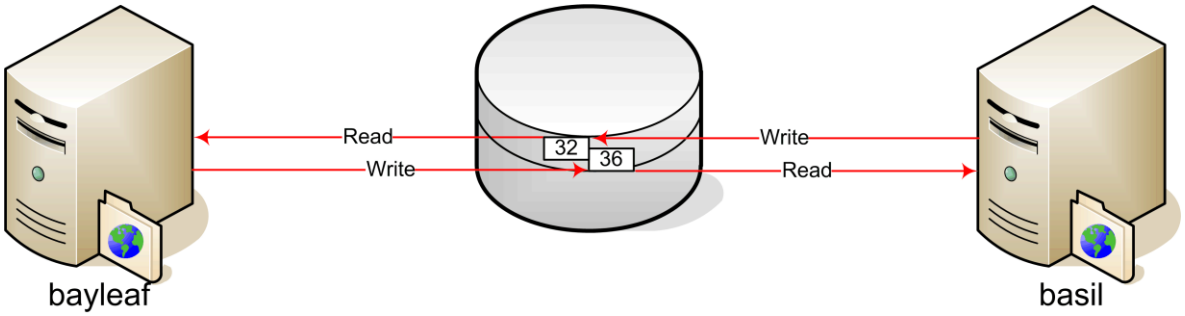
```
    SERIAL cumin /dev/hb_serial_port_2
    DISC   basil /dev/rdisc/hb_bayleaf_basil:32:36:40
    DISC   cumin /dev/rdisc/hb_bayleaf_cumin:32:36:40

  MACHINE basil
  MACHINE cumin
```

The syntax for naming a disc device is slightly different than that of the network and serial ones. This is because a single disc heartbeat partition is shared between hosts, with each reading and writing to the same partition. Therefore the hosts need to know where to read and write from on the disc.

To specify this, colon separated disc block offsets from the start of the partition are appended to the device name. So in the example above for bayleaf we have the device `/dev/rdisc/hb_bayleaf_basil:32:36:40`. The first number (32) is the read offset, the second (36) is the write offset and the third (40) is a signature check block; therefore when bayleaf heartbeats to basil it reads heartbeat data from block offset 32 and writes it to block offset 36. Similarly on basil a heartbeat device is specifies to the same area of shared disc, but with the read and write block offsets swapped, i.e. `/dev/rdisc/hb_bayleaf_basil:36:32:40`. This is depicted in Figure 2.

Figure 2 Read/Write offsets for shared discs.



Finally, the check block offset (40) is initialised manually with a predefined bit pattern by running `rsfmon -w` before starting RSF-1. Whenever RSF-1 starts subsequently, it checks to ensure this bit pattern exists on the shared disc partition before any writes are attempted. This is done as an extra safety check to ensure data corruption does not occur should a disc heartbeat be incorrectly specified or linked on any host.

Expanding the rest of the machine entries to add the remaining heartbeats results in:

```
  #
  # config-example.txt
  CLUSTER_NAME three_node_example

  MACHINE bayleaf
   NET    basil
   NET    basil   basil-p
   NET    cumin
   NET    cumin   cumin-p
   SERIAL basil   /dev/hb_serial_port_1
   SERIAL cumin   /dev/hb_serial_port_2
   DISC   basil   /dev/rdisc/hb_bayleaf_basil:32:36:40
   DISC   cumin   /dev/rdisc/hb_bayleaf_cumin:32:36:40

  MACHINE basil
   NET    bayleaf
   NET    bayleaf bayleaf-p
   NET    cumin
   NET    cumin   cumin-p
   SERIAL bayleaf /dev/hb_serial_port_1
   SERIAL cumin   /dev/hb_serial_port_2
   DISC   bayleaf /dev/rdisc/hb_bayleaf_basil:36:32:42
   DISC   cumin   /dev/rdisc/hb_basil_cumin:32:36:40

  MACHINE cumin
   NET    bayleaf
```

```
    NET    bayleaf bayleaf-p
    NET    basil
    NET    basil   basil-p
    SERIAL bayleaf /dev/hb_serial_port_1
    SERIAL basil   /dev/hb_serial_port_2
    DISC   bayleaf /dev/rdisc/hb_bayleaf_cumin:36:32:42
    DISC   basil   /dev/rdisc/hb_basil_cumin:36:32:42
```

At this point the configuration file contains only information about the nodes and their heartbeats, the services the cluster is providing need to be added next (this is the third logical section of the configuration file). A service is introduced by the SERVICE keyword, followed by its name, virtual IP address and description:

```
   SERVICE oracle   oracle_float      "Oracle database"
```

Next the nodes that can run this service are introduced using the SERVER keyword. Note that server names must correspond to previously defined MACHINE names (in this case bayleaf, basil and cumin):

```
   SERVICE oracle   oracle_float      "Oracle database"
    SERVER bayleaf
    SERVER basil
    SERVER cumin
```

Finally, some service parameters are added along with a network device logical alias. For a description of INITIMEOUT and RUNTIMEOUT please see sections 7.4.5.6 and 7.4.5.5.

The keyword IPDEVICE specifies which network logical device the virtual IP address should be bound to when the service is running. The convention of `eth0:1` specifies a logical address on a physical device (in this case the physical device is eth0 and the logical address is 1).

Different services specify different logical addresses. This is so a server can run multiple services at the same time without logical device conflicts. Note also that the IPDEVICE keyword applies to all servers of that service, except any that specify a different IPDEVICE after the SERVER keyword[8] . In the example below bayleaf uses the device `hme0:1`.

```
   SERVICE oracle   oracle_float      "Oracle database"
    INITIMEOUT      60
    RUNTIMEOUT      30
    IPDEVICE        "eth0:1"
    SERVER bayleaf
     IPDEVICE       "hme0:1"
    SERVER basil
    SERVER cumin
```

When the other two services are added in the final configuration becomes:

```
   SERVICE oracle   oracle_float      "Oracle database"
    INITIMEOUT      60
    RUNTIMEOUT      30
    IPDEVICE        "eth0:1"
    SERVER bayleaf
     IPDEVICE       "hme0:1"
    SERVER basil
    SERVER cumin

   SERVICE HTTP     http_float        "Web server"
    INITIMEOUT      60
    RUNTIMEOUT      30
    IPDEVICE        "eth0:2"
    SERVER basil
    SERVER cumin
    SERVER bayleaf
     IPDEVICE       "hme0:2"
```

[8] This is true of most keywords that appear after the SERVICE line – see Table 4 for details.

```
SERVICE Email    email_float       "Electronic mail"
 INITIMEOUT      60
 RUNTIMEOUT      30
 IPDEVICE        "eth0:3"
 SERVER cumin
 SERVER bayleaf
  IPDEVICE       "hme0:3"
 SERVER basil
```

The combined machine and service definitions provide a basic cluster configuration for the example in Figure 1 Cluster: three node example. The configuration must be duplicated to each node in the cluster and RSF-1 (re)started.

At this stage in the build, a cluster has been defined to support the applications as described in the configuration file. The next stage is to integrate the applications themselves with the cluster service definitions. This is achieved simply by adding appropriate start and stop scripts to the individual service definition runtime configuration directories within RSF-1. This task is covered in detail in section 7.5 Service start and stop scripts.

7 Configuration file reference

7.1 Chapter overview

This chapter describes how to configure an RSF-1 cluster, its nodes, its heartbeats and its services and how to setup service start-up and shutdown scripts.

7.2 Location

RSF-1 cluster nodes take all their configuration information from the single file `/opt/HAC/RSF-1/etc/config`. The configuration file describes all the servers, heartbeats and services that go to make up the cluster. There is an exact copy of this configuration file on each server that is a member of the cluster (a server cannot be a member of more than one cluster).

A cluster may have up to 200 services, each shared between one or more servers. Servers for a service are specified in priority order: the first listed server becomes the *primary server* for the service and the others are then *secondaries*. When a server fails or does not start the service, the next listed server that has the service switchover mode set to *automatic* will attempt to start the service. This continues until either the service is successfully running or there are no available servers remaining.

Note that a server may have primary *and/or* secondary services (it need not have any services, but then it would have no purpose in the cluster either). Each service will have a *primary* and one or more *secondary* servers.

Each service also requires a valid and unique IP address/name not used by any other host or network node. Each service may optionally (but normally) run a set of *service scripts* to perform a sequence of actions, including application control, on start-up and shutdown.

RSF-1 will use the appropriate information from the configuration file for its local parameters, i.e. the information that is specific to the server it is running on.

7.2.1 Warning about differing configurations in a cluster

The configuration file must be identical on each server in the cluster; RSF-1 will detect differing configurations in a cluster and *will reject heartbeats from another server whose configuration differs from its own*. This is a necessary precaution as differing configuration files between nodes in a cluster will cause indeterminate behaviour and possible false actions. If a server is receiving heartbeats from another cluster it will **not** start any services; such heartbeats often indicate a mis-configuration of a cluster.

7.3 Distribution to cluster nodes

RSF-1 does not currently provide a method for synchronising configurations across participating servers. We recommend that you nominate one RSF-1 server in each cluster as the *master configuration server*. Make all your changes on this host and ensure that the other servers are updated from it.

Because the configuration is held in a static file, you can replicate it using simple methods such as remote copying (`rcp`) or FTP. Alternatively, you may prefer something more sophisticated such as `rdist`.

Do not share the configuration file via NFS or store it on shared disks, as these can be single points of failure which could potentially disable an entire cluster.

7.4 Syntax reference

The configuration file contains three sections that are specified in order:

1. Global cluster definitions specify or alter default parameters affecting all the servers and services.

2. Machine definitions describe the servers in the cluster and the heartbeat channels between them.
3. Service definitions describe each service and the servers able to run it.

Do not mix definitions from these different sections together; the first machine definition is assumed to end the global definitions, likewise the first service definition ends the machine definitions.

7.4.1 Template configuration file

Before reading further review the template configuration file supplied with RSF-1 in `/opt/HAC/RSF-1/etc/config.template`. The template contains extensive comments describing the various keywords and their syntax and is structured in the correct manner.

For a new cluster, make a copy of this file as `/opt/HAC/RSF-1/etc/config` and edit that copy using any standard text editor (avoid editing the original as you may need it for later reference).

7.4.2 Syntactic rules

There are some general syntactic rules that apply to the entire configuration file:

- The file must consist entirely of printable characters from the ASCII set.
- Lines must be terminated with the standard UNIX line terminator (ASCII 10).
- Quotes must be double quotes ".
- Indentation and white space within lines does not matter, except within parameters. You should choose a style of indentation that makes the configuration easier to read. White space within parameters must be quoted.
- All keywords are alphanumeric and begin with a letter. They may contain any number of letters, digits, underscores or hyphens. All numbers are decimal integers, except for hexadecimal representations where A-F and a-f are allowed.
- A parameter may be any sequence of characters except white space, new lines, and the characters **( ) = : , #**. If a string contains these special characters then they must be enclosed in double. Note that the colon **:** character often appears in network interface device names.
- A comment is introduced by a **#** character at the start of a token and extends to the end of the line.
- The **#** character must be quoted if it appears at the start of a string.
- Each non-empty line consists only of recognised RSF-1 configuration keywords, associated parameters and/or comments and white space.
- Machine and service names must not be duplicated.

7.4.3 Variable affecting the whole cluster

These variables set some basic parameters that apply to the entire cluster. In some cases these global values can be overridden by context specific values, for example the DISC_HB_BACKOFF may appear in the global section to set a default for all machines, and in a machine section to set values for all following disc heartbeats **in that machine section only**.

Table 4 lists the available variable, their default values and if they can be overridden further on in the configuration file (i.e. entry specific declarations override the global definition).

Table 4 Global variables

Global Keyword	Default value	Overridable?
DISC_HB_BACKOFF	450, 8, 32720	yes
CLUSTER_NAME	Unnamed_cluster_XXXX	no
POLL_TIME	2	no
SCRIPT_TRIES	10	no

REALTIME	1	yes
PRIORITY	4	yes
NETBEAT_TRANSMITS	2	yes
SCRIPT_TIMEOUT	900	no
SYSLOG_FACILITY	LOCAL0	no
UI_HINT		no
IPDEVICE_MONITOR		no

7.4.3.1 DISC_HB_BACKOFF

Format: `DISC_HB_BACKOFF <start>,<multiplier>,<max>`

Declares a disc heartbeat suspension interval should an I/O error occur on the disc being used for the heartbeat. When specifically setting values in a `MACHINE` section the line may be repeated, in which case the new values will be applied to disc heartbeats following the declaration (thus allowing different values to be applied to differing disc heartbeats within a machine section).

Disc heartbeat suspension intervals are needed as disc I/O errors can cause a machine to hang (for instance waiting on SCSI timeouts). The back off interval is therefore useful in avoiding continuous disc writes, as in the case of heart beating, causing the host machine to repeatedly hang – the back off algorithm provides a widening window of time between writes and subsequent possible hangs. Of course, this is specific only to writes performed by RSF-1 – other applications writing to a disc producing I/O errors will cause their own hangs.

Should a disc error occur during disc heartbeat I/O, the heartbeat is suspended for a period of time. Each time the I/O error occurs the suspension time is increased until an upper limit is reached. The first suspension time is specified in the `<start>` parameter. On subsequent errors the suspension time is increased by multiplying it by the `<multiplier>`, up until `<max>` seconds is reached. The allowable values for these parameters are:

```
<start>          12 – 300
<multiplier>     1 – 20
<max>            12 – 86400
```

If values are specified that are outside these ranges, RSF-1 will print an error message and terminate.

7.4.3.2 CLUSTER_NAME

Format: `CLUSTER_NAME <name of cluster>`

Provides a descriptive name for the cluster. This is used in the cluster management console and the command line interface for identification purposes. When no cluster name is declared in the configuration file, a default is assigned (i.e. all clusters will have a name of some form). The default is `Unnamed_cluster_XXXX`, where the *XXXX* part is replaced by the configuration file CRC (so as to always generate unique names).

7.4.3.3 POLL_TIME

Format: `POLL_TIME <seconds>`

Declares an interval in seconds at which new heartbeats are transmitted, and expected to be received from siblings. The default is 2 seconds. The minimum time is 1 second (allowing server failure to be detected earlier, but increasing heartbeat traffic and processing times). A longer interval reduces traffic and load but increases the time to detect a failure. Unless you have specific or unusual requirements, the heartbeat interval should be left unchanged.

7.4.3.4 SCRIPT_TRIES

Format: `SCRIPT_TRIES <number of attempts>`

Declares the total number of attempts made to run service scripts when continual restarts are requested. This setting acts as a brake on services that fail to start but instead loop with repeated errors. If this limit is reached, the service will be aborted, shut down and put into manual mode on the server in question. The default is 10 attempts.

7.4.3.5 REALTIME

Format: `REALTIME <setting>`

Declares a request for the system to run RSF-1 in real time mode (on those systems where real time is supported). The setting is an integer between -1 and a system dependent upper limit. A positive *number* requests round robin real time scheduling of that level, `REALTIME -1` turns off real time scheduling.

Generally real time 0 is the highest available priority on a system, real time 1 being one less than this and so on. If `REALTIME` is not specified in the configuration file, the default is `REALTIME 1`.

The keyword can appear in the global section of the configuration file, where it sets a default value for all machines, and in the machine sections, where is sets the value for that machine only.

Real time should definitely be used on heavily loaded machines or with other applications using real time processing, i.e. fire walls.

7.4.3.6 PRIORITY

Format: `PRIORITY <setting>`

Declares a request for a scheduling priority boost. The priority is normally raised to ensure that RSF-1 receives regular CPU time even when the server is busy. By default, RSF-1 boosts itself by 4. If real time priority is in use then this value is ignored.

7.4.3.7 NETBEAT_TRANSMITS

Format: `NETBEAT_TRANSMITS <number of attempts>`

Declares the number of times to retransmit each UDP network heartbeat. The default is twice. On busy or unreliable networks, you may need to increase this value.

7.4.3.8 SCRIPT_TIMEOUT

Format: `SCRIPT_TIMEOUT <seconds>`

Declares a time in seconds by which service scripts have to run before being terminated and considered to have failed. When terminating scripts, RSF-1 first sends a **SIGTERM** signal, and waits for a further 30 seconds before sending a **SIGKILL** signal. The default timeout is 900 seconds (15 minutes). A value of 0 disables script timeout processing.

7.4.3.9 SYSLOG_FACILITY

Format: `SYSLOG_FACILITY <facility>`

Declares the facility code used by RSF-1 to log messages using the UNIX `syslog` system. If unset, RSF-1 defaults to a facility code of `LOCAL0`.

7.4.3.10 UI_HINT

Format: `UI_HINT <name=value>`

Declares hints to be passed to the cluster management console. Hints are used by the console so it can graphically better represent the cluster. Where more than one hint is specified each hint should start on its own line. Table 5 list the supported hints.

Table 5 UI Hints

Hint	Description
SVM=true	Indicates the cluster uses Solstice Volume Manager for disc management.
VVM=true	Indicates the cluster uses Veritas Volume Manager for disc

	management.
ZONES=true	Indicates this cluster uses Solaris zones.

7.4.3.11 IPDEVICE_MONITOR

Format: `IPDEVICE_MONITOR [IFUP] <polls-to-mark-down>,<polls-to-mark-up>`

Indicates RSF-1 should monitor all network interfaces that the primary VIP is bound to for each service. It is not necessary to specify the actual device, this is worked out on a machine by machine basis from the VIP device declaration further on in the configuration file.

The `polls-to-mark-down` argument represents the number of tests (one each poll interval) that should fail before the interface is marked as unavailable. The `polls-to-mark-up` argument represents the number of successful tests before an unavailable interface is marked as available again. Both poll values must be an integer between 1 to 99.

Without the `IFUP` option, the monitor tests just the link status of the interface; if `IFUP` is included in the declaration then the interface up/down status (as reported by the Operating system) is also checked on each poll.

If during monitoring an interface becomes unavailable that currently has a VIP assigned to it, **and** the remote interface which the VIP would be assigned to it is marked as available, then RSF-1 will perform a failover of the service to the working machine/interface.

7.4.3.12 NETBEAT_SOURCE_PORT

Format: `NETBEAT_SOURCT_PORT [REOPEN] <port>`

This tells RSF-1 what port number to use for transmitting network heartbeats. If the line is not specified, or if a port number of 0 is given, then the same port is used for transmitting and receiving heartbeats (1195).

A fault has been found in some Solaris based operating systems that means if one network heartbeat interface fails, all network heartbeats go down, even if they are connected to different interfaces.

To work around this, the 'REOPEN' argument can be used. This causes RSF-1 to open and close its network socket for each network heartbeat, instead of leaving it open for the entire time it is running. To use this functionality, the same port cannot be used for both transmitting and receiving network heartbeats, so specifying the same port in this line (1195) is an error. Specifying a port number of 0 allows the operating system to use any unused port for each packet transmitted.

This line can be specified in the global section as a default for the whole cluster, and in the individual MACHINE sections.

7.4.4 Machine definitions

A MACHINE defines a single RSF-1 server and the heartbeats it exchanges with other machines. Note that, to be affective, the definitions of those machines must include related heartbeats to the first machine (i.e. a logical link is provided). There can be up to sixteen MACHINE definitions (although RSF-1 does not enforce this restriction, using more than sixteen servers will result in degraded cluster performance).

The maximum number of heartbeats a server may send or receive is limited by the number of file descriptors available to processes under the OS (typically 1024) and the available memory. In practice, you are unlikely to reach these limits. There is an overall limit of 32767 heartbeats within a cluster, but the available network bandwidth is likely to impose a more practical limit. There are three channel types over which heartbeats may be exchanged: network, serial and disk.

The heartbeat definitions are all optional, but obviously you must include at least one, but preferably two or more for sensible operation. Ideally, you should use as many heartbeats as practicable, of different types, to each other server with which the machine shares a service. The syntax for a machine definition is shown in Figure 3.

Figure 3 Syntax for machine definition

```
MACHINE <nodename> [<hostid> <ipaddress>]
  [SERIAL <remote_nodename> <local_serial_port>]
  [DISC <remote_nodename> <read_dev:block:check_block,write_dev:block:check_block>]
  [NET <remote_nodename> [remote_address]]
  […]
```

RSF-1 can be configured with a machine name which is not the same as the hostname of the machine on which it is running. This allows multiple hosts in a cluster to have the same hostname, which is required by a few applications.

To do this, the MACHINE line of the configuration file allows an optional third and fourth parameter at the end of the line. If present, this must be the HAC host ID of the machine, in hexadecimal, preceded by "0x" (to indicate hex,) followed by the IP address of the machine. In this case RSF-1 will identify that machine by the name on the MACHINE line, even if it does not match the real hostname. e.g.:

```
MACHINE slug 0x02ae5747 192.168.200.34
```

The IPADDRESS part is required as the machine name need only exist in RSF-1 space (i.e. is not required to be a part of any naming service[9]). Support programs such as the Cluster Management Console use this information when making initial contact.

RSF-1 also sets the environment variable RSF_MACHINE_NAME in all start-up/shutdown scripts executed to be the machine name in use. This allows scripts to create log messages using the same machine name as RSF-1 itself.

Finally RSF-1 can be configured to have clusters consisting of just a single node so that it can simply be used as a management interface.

7.4.4.1 MACHINE

Format: MACHINE <*nodename*> [[<*hostid*>] <*ipaddress*>]

Begins a machine definition and specifies the name of the node in question. Either the name must match the hostname of the server or the host id must match the HAC hostid[10]. That server will read this part of the configuration file for its heartbeat definitions. If the second form of machine entry is used then the host matches on hostid.

[9] Naming services such as the /etc/hosts file, the Domain Name System or NIS.

[10] The HAC hostid is obtained by running the program /opt/HAC/bin/hac_hostid, which is part of the HACbase package.

7.4.4.2 The SERIAL heartbeat definition

Format: `SERIAL <remote node> <local serial port>`

Defines a serial heartbeat to the specified remote node, sending out, and listening for serial heartbeats via the given local serial port device (e.g. `/dev/cua/b; /dev/ttyS1`).

7.4.4.3 The DISC heartbeat definition

Format: `DISC <remote node> <read device:block,write device:block>`

Defines a disc heartbeat to the specified remote node, using the given devices and block offsets for reading from the remote end and writing to it.

For example:

```
DISC machine2 /dev/disc1/cltld0s0:34,/dev/disc2/c1t1d0s0:38
```

specifies reading from block offset 34 and writing to 38 on device c1t1d0s0; the remote machine will have a definition of:

```
DISC machine2 /dev/disc1/c1t1d0s0:38,/dev/disc2/c1t1d0s0:34
```

so it reads from the device the remote end is writing to and visa-versa. The block offsets must differ by at least 4 to allow enough space for each heartbeat area.

Disk heartbeats must use *raw* (i.e. un-buffered) devices. (See the discussion of the disk heartbeat in the RSF-1 installation guide). The initial block offset should be at least 32 in case the partition is the first one on the disk. This is because space is needed at the start of the partition (on some systems) for system boot blocks, partition tables and other information.

As some operating systems can change the names of the disc device files on every reboot (most notably Apple OS-X) RSF-1 provides a mechanism to check a partition really is a heartbeat partition before writing into it using a check block. The check block, if specified, is a ":block_number" immediately following a disc heartbeat write block number.

The check block is scanned for an RSF-1 signature before heartbeats are enabled on this device.

Check block signatures are initialised by running the `rsfmon` command with the `-W` switch (check block specification is taken from the configuration file and therefore need not be specified). Check block initialisation need only be done once, after manually confirming that all the configured heartbeat partitions have the expected names. Note that the initialisation needs to be performed on *all* nodes in the cluster that have been configured to check and write disc heartbeats.

To verify check blocks run the command `rsfmon -R` (or `-RR` for more verbose output).

A shortened form of the DISC definition is supported:

DISC <*remote node*> <*device:read:write*>

or

DISC <*remote node*> <*device:read:write:check*>

The shortened form can be used if both read and write are on the same disc device, i.e.:

```
DISC machine1 /dev/disc1:32:36
```

or

```
DISC machine1 /dev/disc1:32:36:40
```

Finally, the DISC line can include a service tag at the end of the line. This can be either

`SERVICE <service>` or `TAG <service>`.

This tells RSF-1 to associate this heartbeat with one service, and causes it to suspend it while that service is starting. This is to ensure the heartbeat does not conflict with any transient SCSI reservations that may be placed on the disk by the service start scripts. The heartbeat will then be resumed when the service has finished running start scripts.

7.4.4.4 The NET heartbeat definition

Format: `NET <remote node> [<remote hostname|address>]`

Defines a network heartbeat to the specified remote node. If a remote hostname or address is given, then the heartbeat is sent to that destination rather than the hostname associated with the machine[11]. Examples of valid conventions are:

```
MACHINE rsfserver2 192.168.44.56
…
NET rsfserver2 192.168.44.56
```

This could also be represented as:

```
MACHINE rsfserver2 192.168.44.56
…
NET rsfserver2
```

Additionally, if `rsfserver2` resolves to a valid IP address then this can further be reduced to:

```
MACHINE rsfserver2
…
NET rsfserver2
```

Specifying the remote address in a NET definition provides a mechanism by which a heartbeat can be directed down different network links identified by the remote endpoint address (for example to utilise a private network link between hosts).

7.4.4.5 REALTIME, PRIORITY, NETBEAT_TRANSMITS

The global parameters, described in section 7.4.3, can also be defined on a per-server basis. Server-specific parameters take precedence over global settings.

The following figure shows an example of a typical machine definition. Here, `korma` has serial, disk and two network heartbeats (via public and private nets) to `masala`.

Figure 4 Machine definition example

```
# defines for korma
MACHINE korma
  # serial hb to masala via port B
  SERIAL masala /dev/cua/b
  # disk hb on device clt1d0s0 with check block
  DISC masala /dev/rdsk/cltld0s0:34,/dev/rdsk/cltld0s0:38:42
  # public net hb
  NET masala
  # private link hb
  NET masala masala-priv
```

7.4.4.6 Summarising machine names

As there are a number of places in the RSF-1 configuration file where machine names, host names & IP addresses can appear, this section is included to summarise and clarify the differences.

Firstly, there are two names which are associated with each node in an RSF-1 cluster, which may look the same, but in fact have subtle differences. They are:

Machine Names

These are the names defined on the `MACHINE` lines of the configuration file, and are used by RSF-1 to identify the nodes in a cluster. The machine name of a node will normally be the same as the network name of that node, but this is not required. The `MACHINE` line may contain a host ID to identify the node to which it refers, and in this case the machine name may be different from the network name of the node.

[11] See the Network Name paragraphs in section 7.4.4.6.

Machine names are defined on MACHINE lines. They are used on heartbeat (NET, DISK and SERIAL) lines to indicate what machine the heartbeat is being sent to, and on SERVER lines to indicate which machines may act as a server for a service.

Network Name

These are used to resolve network addresses for nodes, and may be names which can be looked up to find IP addresses, or may be numeric IP addresses.

Network names are used in three types of configuration line:

a) On MACHINE lines to indicate the network address of a node. This address is passed to RSF-1 control programmes as the address to which monitoring & control requests for that node should be sent.

b) On NET heartbeat lines to give the address to which the heartbeat is sent.

c) On SERVICE lines to indicate the floating network name/address for the service.

There are two types of line in the configuration file in which both a machine name and a network name may be given, and in both cases the network name may be omitted. These lines are:

MACHINE *<machine_name> [<optional_network_name>]*

For the machine line, if no network name is provided then it is taken to be the same as the machine name.

NET *<machine_name> [<optional_network_name>]*

For the net line, if no network name is given, the network name associated with that machine, from a previously declared MACHINE line, is used.

Note: If a MACHINE line contains a hexadecimal host ID to identify the node it refers to, then the network_name is required (i.e. no longer an optional parameter). In this case it is likely that the network name will also be needed in all NET heartbeat lines which refer to this machine.

7.4.5 Service definition

A service defines the characteristics of a service and lists the servers available to run it. The first server listed is the *primary;* the remaining servers are *secondary* in priority order. Also specified are the floating address and description. The network device for the address and the timeouts can be given as defaults applying to all servers or for individual servers. The default is used unless a parameter for the server in question is supplied; if no default is given, the parameter must be specified separately for each server.

There may be from 1 to 200 service definitions[12]. Using more services than this will cause problems with RSF-1. The syntax for a service definition is shown in Figure 5.

Figure 5 Syntax for service definition

```
SERVICE <service_name> <floating_addr> <description>
   [OPTION <options string>]
   [IPDEVICE <network_device>]
   [INITIMEOUT <seconds>]
   [RUNTIMEOUT <seconds>]
 SERVER <servername>
   [IPDEVICE <network_device>]
   [INITIMEOUT <seconds>]
   [RUNTIMEOUT <seconds>]
 [ SERVER ...]
```

7.4.5.1 SERVICE

Format: SERVICE *<service name> <floating address> <description>*

Begins the definition and specifies the service name, floating address and description. The name must be a short identifier containing no white space (maximum 64 characters). The floating address is a unique, virtual network name or IP address. In particular, it must *not* be the name of

[12] 200 service definitions is the maximum allowed.

a server and should preferably not be based on the name of a particular host either (since it is migratory). The description is a more verbose form of the name and may contain white space if quoted. The service line may be followed by parameter definitions that will apply to all the servers listed below which do not supply their own.

7.4.5.2 OPTION

Format: `OPTION <options line>`

A service line may be immediately followed by an option line. This line specifies a quoted string containing space separated options to be processed on service start-up and shutdown. The recognised options are:

`ip_up_after`

Causes the floating address for the service to be enabled after the service start-up scripts have been run. The address is normally attached before the scripts are run.

`ip_down_before`

Causes the floating address for the service to be disabled before the service shutdown scripts are run. Normally the address is disabled after the scripts are run.

7.4.5.3 SERVER

Format: `SERVER <server name>`

Defines the name of a server that can run the service. It must be a server previously defined as a MACHINE in the cluster. A server may have its own definitions for the following parameters that take precedence over any defaults.

The order in which servers are specified for a service is the order of preference for which server the service should run on. The first server listed is the **primary** server for that service.

7.4.5.4 IPDEVICE

Format: `IPDEVICE <server name>`

Defines the *device name* of the network device on which the *floating address* specified previously is configured. There can be only one `IPDEVICE` per server or service. If a default is given, then a device of that name must exist on every server that does not specify a different one. Remember that the device name must be quoted if it contains non-alphanumeric characters (e.g. "`hme1:1`"; `en0`).

The `IPDEVICE` must be unique to the service on each server; several services cannot share a specific interface device (physical or logical) although it is quite permissible to have many logical interfaces, each with its own service, on a single physical interface. Note that your OS may limit the number of logical interfaces or aliases possible on a single physical interface.

If the `IPDEVICE` is set to `FILE` (in upper case,) the file `/opt/HAC/RSF-1/etc/rc.`*`servicename`*`.d/interface.`*`hostname`* is used on each server to determine the correct network device name. For example, on Solaris it could contain the network device `le0:1`.

If this file is executable, it is run by RSF-1 and expected to give the device name as output. Otherwise, it is expected to contain the device name on a single line. This file can be used to change the current network interface while RSF-1 is running. (Note: changes only take effect when the service is started or stopped. If the service is currently running, you must switch the interface manually).

7.4.5.5 INITIMEOUT

Format: `INITIMEOUT <initial timeout>`

Defines the *initial timeout* for the service in seconds. This must be long enough for all the servers in the cluster to complete booting following a simultaneous power-up. You may wish to make this timeout shorter for the primary server, as this will not result in the service starting on a lower priority server which happens to boot faster. The maximum length of the initial timeout is 10800 seconds (3 hours).

Once all the servers in the cluster have successfully contacted each other any remaining timeout is discarded to allow services to start up quicker (i.e. their state will now be known).

7.4.5.6 RUNTIMEOUT

Format: `RUNTIMEOUT <timeout>`

Defines the *timeout* for the service in seconds. This timeout is used after all the siblings for the service have been contacted, and subsequently contact is lost with one or more of them. It will normally be shorter than the initial timeout, but should be long enough to detect failure of another server with certainty, taking into account high transient server and network loads where applicable.

7.4.5.7 MOUNT_POINT - *recommended use where applicable*

Format: `MOUNT_POINT <mount point>`

The mount point directive specifies a directory upon which shared file-systems should be mounted when this service is running, and likewise un-mounted when the service is stopped. There may be multiple lines to indicate multiple mount points to check. Mount points specified for the service apply to all server entries for that service, *unless* a server entry specifies its own mount points, in which case all applicable mount points for the service must be specified (see Figure 6 for an example configuration file).

Should RSF-1 detect that a file-system has failed to mount or un-mount correctly, it takes preventative measures to ensure no corruptions can occur; in reality this means that a service claiming to be stopped, but which still has file-systems mounted, is prevented by RSF-1 from running on any other nodes (which in effect could result in file-systems being dual mounted with the possibility of data corruption occurring). This is an important safety feature of RSF-1 and its use is highly recommended with those services which mount any shared file-systems.

The sequence of events that occur should a mount/un-mount failure be detected by RSF-1 are given below. Note that these descriptions assume that a `MOUNT_POINT` directive exists in the configuration for the scenario described.

Service start up

When the service scripts indicate that the service has successfully started (exit code 0) RSF-1 checks any mount directories mentioned in `MOUNT_POINT` entries for that service. If all the directories listed contain valid mounted file-systems then the service is considered successfully started and its RSF-1 state is set to running. If however, any of the mount directories do **not** contain mounted file-systems then RSF-1 considers the service has failed to start correctly and runs that services stop scripts. The state the service is left in finally depends on the result of running the stop scripts (see below).

Note that if the service start up scripts exit indicating that an error has occurred then the mount check is skipped and the stop scripts are run (as is normal for this situation, again see below).

Service shutdown and abort

When the service shutdown or stop scripts are run and they exit indicating successful completion, RSF-1 then checks the mount directories to ensure none of the listed file-systems remain mounted. Should any mounts be found then RSF-1 sets the state of the service to broken_unsafe. In the case where no mounted file-systems are found then for normal service stop the service state is set to stopped, and for service aborts, i.e. stop scripts are run because the start scripts failed, then the service state is set to broken_safe.

Failure avoidance using mount point checks

To demonstrate how RSF-1 uses the MOUNT_POINT directive to avoid potential serious issues, let us consider a two machine cluster with a single database service. In a controlled failover from one machine to the other the typical sequence of shutdown events would be:

1. Database shutdown
2. Fil-esystem un-mount
3. VIP address removal

At this point RSF-1 would normally migrate the service to the secondary machine (assuming the secondary machine is in automatic mode). However, if stage 1 or 2 fails (but erroneously reports success) then actual failover could result in a cross mount of file-systems, with the inevitable possibility of data corruption. Using the `MOUNT_POINT` directive avoids this scenario as during the final service shutdown checks RSF-1 will detect the still mounted file-system(s) and set the service state to broken_unsafe (and thus prevent service migration).

7.4.5.8 RSF_RESERVATION_DRIVE_n

This is a variable used by the start/stop scripts for ZFS appliances (rc.appliance.c). It should be used to tell RSF-1 which disks to place a SCSI reservation on while the service is running. RSF-1 has the ability to quickly issue PGR3 reservations to disks in order to fence out the other node and avoid a split brain. With ZFS, we recommend reserving enough disks to make it impossible for the other node to import the pool (e.g. Both disks in one mirror, or two disks from one raidz vdev). The form of the config lines should be:

```
SETENV RSF_RESERVATION_DRIVE_1 c0t0Ad0
SETENV RSF_RESERVATION_DRIVE_2 c0t0Bd0
```

If the disk names are not identical on both nodes, these variables can be set individually for each server in the config file.

7.4.5.9 A complete service example

An example service definition is shown in Figure 6. Here the service Sybase is defined with Korma the primary server and `Masala` the secondary. `Korma` has shorter timeouts than the default, has the mount points `/oracle` and `/oradata` and uses the service setting of `hme0:1` for the network device. `Masala` specifies its own network device and uses the default timeouts and mount points for the service.

Figure 6 Service definition example

```
SERVICE Sybase sybserv "Sybase production database"
 IPDEVICE "hme0:1"
 INITIMEOUT 300
 RUNTIMEOUT 30
 MOUNT_POINT /oracle
 MOUNT_POINT /data

 SERVER korma         # primary
   INITIMEOUT 180
   RUNTIMEOUT 12
   MOUNT_POINT /oracle
   MOUNT_POINT /oradata

 SERVER masala        # secondary
   IPDEVICE "le0:1"
```

7.4.6 Updating RSF-1 after changing configurations

After creating or updating the configuration file, copy it to the same location on all other RSF-1 servers in the cluster. Changes made to a service configuration will have no effect until RSF-1 is restarted. It is recommended that all services be put in manual mode, or stopped, when changing the configuration file, and only be set back to automatic when **all** the machines in the cluster have been restarted (see chapter 8 for more details).

If a service name is altered, or removed from the configuration, the service should be shutdown before stopping RSF-1, as it will be harder to modify when RSF-1 no longer knows about it.

7.4.7 Configuration database

The RSF-1 installation includes a configuration database. Currently there are several database properties that are referenced in the start/stop scripts. These properties can be listed using the following command:

`/opt/HAC/RSF-1/bin/rsfcdb list_props`

The current list of properties includes:

Property	Default	Function

prop_comstar_support	true	Support failover of COMSTAR luns
prop_zpool_sync_cache	false	Regularly sync ZFS cache files between nodes
prop_no_zfs_cache	false	Don't use ZFS cache files
prop_mhdc_use_pgr3	true	Use PGR3 SCSI reservations instead of SCSI-2
prop_mhdc_use_failfast	true	Use failfast as a split brain prevention
prop_event_notify	true	
prop_scsi2_drive_count	2	Number of disks to use for reservations (SCSI-2)
prop_zpool_fail_mode	panic	Zpool failmode property (panic, continue or wait)
prop_zpool_export_option_c	false	Use -c option with zpool export (NexentaStor only)

To update a property in the database, run the following command:
`/opt/HAC/RSF-1/bin/rsfcdb update <property> <new value>`

Note that all services should be stopped before updating the 'prop_mhdc_use_pgr3' property.

To add a new property to the database for use with your own custom scripts, run the following command:
`/opt/HAC/RSF-1/bin/rsfcdb create <property> <value>`

7.5 Service start and stop scripts

Each service typically requires some start up and corresponding shutdown procedures. These procedures are executed by means of scripts or programs similar to those run when UNIX boots up or shuts down. You may be able to acquire these scripts ready-made from various locations on your system, but it will help if you are familiar with shell script programming. The choice of shell interpreter is yours; typically, the Bourne shell (`/bin/sh`) is used.

7.5.1 Understanding the script directories

The service scripts for a particular service are all held in a subdirectory below `/opt/HAC/RSF-1/etc`, called `rc.`*`servicename`*`.d`[13] for example `rc.`*`nfsserv`*`.d`. A script directory for a service that is not defined in the configuration will be ignored. Within this directory, there are a number of scripts. Each script is named with the convention:

```
[S|K|P]XXname
```

where S stands for 'Start', K stands for 'Kill', P stands for 'Panic' and XX is a two digit number from 00 to 99. The name is simply an identifier and is not significant. E.g. `S01announce`, `K20disks`

In fact, the 'K' scripts are usually symbolic links to the matching 'S' scripts with the numbers in reverse order, i.e. the same script handles both start-up and shutdown tasks. You can quickly generate these 'K' script links using the command `rsfklink`, see section 7.5.6 for details.

When a service is started, RSF-1 runs all the 'S' scripts in numerical order (based on the 'XX' digits in the name,) with an argument of *start*. When a service is shutdown, RSF-1 runs all the 'K' scripts in numerical order, with an argument of *stop*. It is simple to write a script that tests its first argument and takes the appropriate action within a *case* or *switch* statement.

The shutdown actions are typically taken in reverse order to the start-up actions. This is accomplished by using the correct numerical sequences for the 'S' and 'K' scripts.

Note that RSF-1 uses these scripts in a slightly different way to the normal SVR4 Unix semantics. Only the 'S' scripts *or* the 'K' scripts are run when the state of the service changes; they are never executed together in alphabetical order.

The 'P' scripts are panic scripts and in normal operation will not be run. They are intended to try to limit the damage which may be caused if a service is started on more than one server at the same time (this can only happen if ALL the heartbeats between siblings have failed, the service has been started and then the heartbeats restored).

The following is a typical start-up sequence (each step is handled by a separate script):

1. Announce that the service is starting.
2. Take, check and mount the disk devices required.
3. Start the application(s).
4. Announce that the service has now started.

The following is a typical shutdown sequence:

1. Announce that the service is shutting down.
2. Stop or kill the application(s) safely.
3. Unmount and release the disks, after killing any processes still using them.
4. Announce that the service has stopped.

13 This notion stands for "runtime configuration directory."

7.5.2 Creating the scripts

There are three ways to obtain suitable service scripts, described in the following sections. Before starting to gather your scripts, think about what is required to start and stop your applications, how they will be split between servers and what tasks your scripts need to perform.

7.5.2.1 Writing your own

You must write your own scripts if there are no suitable ones available or if you wish to take advantage of additional RSF-1 functionality. Scripts can be written in any available script language of your choice, or you can develop and compile programs in a high level language such as C.

Your scripts should accept two arguments that will be supplied by RSF-1:

- The first argument will either be start or stop, depending on whether the service is starting or stopping. Your script should take the appropriate action, or no action at all if none is applicable, depending on this argument.
- The second argument is the number of attempts that have been made to start this service (a service is restarted every time one of the scripts returns an exit code of 2; see section 7.5.3). Above a certain threshold, you may wish to give up and signal an abort back to RSF-1. If not, you can ignore this argument.

If the script executes processes and commands which are to be left running, they must be back-grounded. Execution of further scripts does not proceed until the current one has finished. Scripts are executed as the superuser.

Output to stdout and stderr from your script is sent to the RSF-1 log file. The exit code from the script can be used to signal problems to RSF-1.

Be careful not to create dependencies outside RSF-1 control in your scripts. Remember that these scripts must be capable of running on any RSF-1 server in the cluster.

See Appendix D and the RSF-1 *Applications Guide* for further details on service script capabilities.

7.5.2.2 Using the templates provided

The installed distribution contains the directory `/opt/HAC/RSF-1/etc/service` with many examples of typical service scripts, some taken from actual installations (see chapter 5). You should not use these examples without modification. Most of them will need the service details, disk groups or application paths altering (comments advising on specific changes required will be found at the head of each script).

At the time of writing, examples are provided for NFS, Sybase, Oracle, Oracle9i, Ingress and HTTP (apache) daemons and generic applications. There are also templates to migrate shared file-systems and some typical disk management software volumes. Finally, a set of scripts is provided to assist in VIP failover and ARP cache updating.

We strongly recommend that you study the provided examples to fully understand how the service scripts work.

The following tables list the service scripts provided with RSF-1 along with a brief description of the intended use of each script.

Table 6 Scripts in /opt/HAC/RSF-1/etc/service/scripts

Script	Description
P10template	This is a generic template script for use as a panic script.
S00template	This is a generic template for use as a start and stop script.
S01announce	Timestamps the RSF-1 log file when starting the service.
S10apc_snmp	APC UPS power supply interface start-up using SNMP.
S30_9ias	Oracle 9i database start-up and shutdown.
S30oracle	Oracle database start-up and shutdown.
S30sybase	Sybase database start-up and shutdown.
S30ingres	Ingres database start-up and shutdown.
S80httpd	Generic WEB server service script.

S80netscape_https	Specific to the Netscape WEB server.
S80generic_app	This is a generic template for application start-up and shutdown.
S90notify_email	Email notification that a service change has occurred.
S99announce	Timestamps the RSF-1 log file when the service has started.

Table 7 Scripts in /opt/HAC/RSF-1/etc/service/disks

	Script	Description
SOLARIS	S20sds	Imports and starts Solstice disksets located in `fs/disksets.<servicename>` associated with the service (disabled by default).
SOLARIS	S20ufs	
SOLARIS	S20vxm	Imports and starts Veritas diskgroups located in `fs/disksets.<servicename>` associated with the service (disabled by default).
SOLARIS	S25vfs	Checks, mounts and/or exports filesystems located in `fs/vfstab.<servicename>` and `fs/dfstab.<servicename>` associated with the service.
LINUX	S20fstab	Checks, mounts and starts older style Linux filesystems located in `fs/fstab.<servicename>` associated with the service.
LINUX	S20ext2	Checks, mounts and starts Linux ext2 filesystems located in `fs/fstab.<servicename>` associated with the service.
HP-UX	S20vg	Imports, starts, checks, mounts and/or exports volume groups located in `fs/diskgroups.<servicename>` associated with the service.
HP-UX	S25vfs	Checks, mounts and/or exports filesystems located in `fs/vfstab.<servicename>` and `fs/dfstab.<servicename>` associated with the service.

Table 8 Scripts in /opt/HAC/RSF-1/etc/service/network

Script	Description
S10arp	Sends a gratuitous ARP update for the RSF VIP on start-up.
S30interface	Brings up the VIP configured into RSF-1.
S31default_route	Enable/disable the default route through the VIP.
S31route_order	Re-orders the network routing table to make the specified interface the preferred route.

Table 9 Scripts in /opt/HAC/RSF-1/etc/service/tables

	Script	Description
SOLARIS	dfstab.service	An example dfstab table for Solaris export commands.
SOLARIS	diskgroups.service	An example diskgroups table for use with Veritas VM.
SOLARIS	disksets.service	An example disksets table for use with SDS.
SOLARIS	vfstab.service	An example vfstab table for use with Solaris default filesystems.
HP-UX	diskgroups.service	An example diskgroups table for use with HP-UX Volume Manager.
HP-UX	dfstab.service	An example dfstab table for HP-UX export command.
HP-UX	vfstab.service	An example vfstab table for use with HP-UX default filesystems.
LINUX	fstab.service	An example fstab for use with Linux filesystems.

7.5.2.3 Example service directories supplied with RSF-1

rc.9ias.d/

Contains `S30_9ias` and `K70_9ias` scripts to start and stop Oracle9i Application Server (9iAS). The `ORA_HOME` and `ORA_OWNER` environment variables will need to be set to reflect your own 9iAS configuration and environment.

rc.apache.d/

Contains `S80httpd` and `K20https` scripts to start and stop Apache/NCSA compliant web servers. The `HTHOME` and `HTTPD` environment variables will need to be set to reflect the location of your web server home directory and httpd binary.

rc.ingress.d/

Contains `S30ingres` and `K70ingres` scripts to start and stop Ingres II. The `II_SYSTEM` and `PATH` environment variables will need to be set to reflect your own Ingres II installation location and associated directories.

rc.nfs.d/

Contains no additional start and stop scripts by default. Assumes NFS related daemons are started at system boot time, either via dummy entries appearing in `/etc/dfs/dfstab` or via NFS exported filesystems outside of RSF-1's control, i.e. shares not associated with any RSF-1 service.

If all NFS exported filesystems are to be under RSF-1's control, it is acceptable to copy the `/etc/rc3.d/S15nfs.server` script to `rc.nfs.d/` and start the NFS related daemons before the common `S25vfs` script exports the filesystems from `fs/dfstab.<servicename>` associated with the service.

rc.oracle.d/

Contains `S30oracle` and `K70oracle` scripts to start and stop Oracle8i databases. The `ORA_HOME` and `ORACLE_OWNER` environment variables will need to be set to reflect your own Oracle8i installation and configuration.

7.5.2.4 Using application rc scripts

Many applications supply and install their own start-up and shutdown scripts in the standard system directories, such as `/etc/rc2.d` and `/etc/rc0.d`. These scripts can be moved from their normal locations and used directly, usually without modification. The only changes likely are to ensure they return and set the exit code appropriately to indicate any problems.

If you choose to copy the start-up scripts, *ensure that you delete the original ones*. Failure to do so will cause the application to be started every time the host is booted, regardless of the configuration; this may lead to two copies of the application being run.

7.5.3 Exit codes

The exit code from your service script or program is used by RSF-1 to determine whether or not it executed successfully, and what to do next. The recognised exit codes are listed in the following table.

When RSF-1 receives an exit code of 2, it attempts to start the service again by running all the start-up scripts. If any of the earlier scripts already ran successfully, they must exit silently without taking any action, rather than reporting errors. (Note: the shutdown sequence is not run before restarting). RSF-1 will only restart a service up to a maximum of SCRIPT_TRIES times, as defined in the configuration file, or ten by default.

If the exit code is 3, RSF-1 runs the stop sequence and changes to manual switchover mode, if the stop sequence executes OK then the service will be marked broken/safe and may fail over to the other RSF-1 server if that server has the switchover mode set to automatic. Should the standby server also abort the start-up, the service will be shut down and will not be available on any server.

Because commands terminate with individual exit codes, you should explicitly include recognised exit statements within your script.

The symbolic names (RSF_OK etc) are defined in the `rsf.sh` script, and should normally be used in preference to the numeric values.

The following table documents the script exit codes and what they mean to RSF-1.

Table 10 Script exit codes

Exit	Symbolic name	Meaning	Description

code			
0	RSF_OK	OK	Success; the script worked, action complete
1	RSF_WARN	WARN	Warning; the script worked, but warnings were issued, action complete
2	RSF_RESTART	RESTART	Failure; script failed but a re-run may work, action incomplete. Will re-run ALL the scripts for this service until the retry count is reached.
3	RSF_ABORT	ABORT	Failure; script failed, re-run won't work, service unusable, action incomplete. If the action was "start", the shutdown scripts are run. If the shutdown scripts succeed the service is marked "broken safe". If the action was "stop", even if caused by a failing "start", the service is marked "broken unsafe".
4	RSF_SAFE	SAFE	Failure; no further actions or retries are taken, the service is marked "broken safe".
5	RSF_UNSAFE	UNSAFE	Failure; no further actions or retries are taken, the service is marked "broken unsafe".

If a service start/stop scrip exits because it was killed by a signal (including a signal sent by rsfmon after a timeout,) then the signal used is examined. If the process was killed by a HUP signal, then the scripts are treated as if they had exited with code 2, failure with restart possible. Any other signal is treated as an exit with code 3, scripts failed, can not be restarted.

Finally refer to Appendix C for information regarding the environment provided by RSF-1 to service scripts.

7.5.4 Adding the Scripts

When you have collected all the required scripts, decide their running order and use this to assign sequence numbers to them. The sequence numbering should leave gaps to allow you to insert additional scripts in the future. Remember that your 'Kill' scripts should be symbolic links (made using `ln -s` or `rsfklink`, see section 7.5.6 for details) to your 'Start' scripts, simplifying maintenance, and that their sequence numbers should run in reverse.

Create the appropriate `rc.`*`servicename`*`.d` directories under `/opt/HAC/RSF-1/etc` and populate them with your scripts, e.g.

```
# cd /opt/HAC/RSF-1/etc
# mkdir rc.oracle.d
# mkdir rc.nfsserv.d
```

Example

A directory listing for the *oracle* service in the example cluster is shown in figure 12 (`/opt/HAC/RSF-1/etc/rc.oracle.d/`). The `S*announce` scripts log a message to the `rsfmon` log file when the service is starting and has started successfully. `S02audio` plays a sound sample when the service starts to alert operators. `S20disks` parses the `vfstab` file to find the entry for the `/oracle` file-system, then checks and mounts it (from `/dev/dsk/c1t2d0s3`). `S30oracle` uses the Oracle server manager to start the database instance in the normal way. The `K*` scripts perform the opposite actions in reverse order.

7.5.5 Testing your scripts

Your scripts can be tested by running each one individually from the command line. Execute them in the correct sequence with the normal arguments. After each script, check that the tasks

have been performed correctly. You should run this test on each RSF-1 server. If all is well, you may proceed to start RSF-1 and the services.

Figure 7 Service directory for the example cluster

```
Total 22
lrwxrwxrwx 1 root   other   11 Jan 12 12:25 K01announce -> S01announce
lrwxrwxrwx 1 root   other    9 Jan 12 12:25 K20oracle   -> S30oracle
lrwxrwxrwx 1 root   other    8 Jan 12 12:25 K60disks    -> S20disks
lrwxrwxrwx 1 root   other    8 Jan 12 12:25 K98audio    -> S02audio
lrwxrwxrwx 1 root   other   11 Jan 12 12:25 K99announce -> S99announce
-rwxr-xr-x 1 root   other  263 Sep 24 16:26 S01announce
-rwxr-xr-x 1 root   other  289 Dec 23 12:04 S02audio
-rwxr-xr-x 1 root   other 1916 Oct  2 17:21 S20disks
-rwxr-xr-x 1 root   other  389 Sep 24 16:31 S30oracle
-rwxr-xr-x 1 root   other  275 Sep 24 16:27 S99announce
```

Figure 8 Testing the service scripts

```
# sh S02audio start
# sh S20disks start
# sh S30oracle start
# sh K70oracle stop
# sh KSOdisks stop
# sh K98audio stop
```

7.5.6 rsfklink

Format: `rsfklink <servicename>`

`rsfklink` is a utility that automatically creates the correct service shutdown sequence on the local server (K* symbolic links in the service script directory). It will ensure that the numbering is the reverse of the existing start-up sequence in the directory it is linking.

When run without arguments, it processes all existing service directories or configured services (if any). Alternatively, you can specify a particular service or services.

7.6 RSF-1 Access control

RSF-1 implements certain access controls to decide which users and/or hosts are able to monitor and control it. For security reasons, you are strongly advised to configure these mechanisms correctly before using your cluster.

7.6.1 Network access control

By default, there is no network access control in RSF-1 and all client requests are allowed. If the file `/opt/HAC/RSF-1/etc/access` exists, it is used to determine whether client requests from remote nodes to `rsfmon` should be accepted.

The file contains a sequence of lines that either ALLOW or DENY access from specified IP hostnames, addresses or networks. The source address of the incoming request is checked against each line and the *first* match used to accept or reject it. If no matches are found, the request is rejected; hence an empty file will deny all client requests.

Each line is of the form:

```
ALLOW | DENY hostname | address [/bits | netmask]
```

The case of the leading keyword is not significant. Comments are allowed if preceded by a # character.

The first argument is a hostname, network name or IP address to be matched against. This must be followed either by a forward slash and the number of significant bits in the preceding address to use, or white space and a netmask in dotted quad format. Figure 9 lists some example entries.

Figure 9 Example access control entries

```
ALLOW adminclient/32                    # single host
ALLOW 192.168.78.0/24
# allow all hosts in 192.168.78.0 network
DENY 10.12.0.0 255.255.0.0
# deny all hosts in 10.12.0.0 net
# these are all equivalent
DENY 193.62.192.131 255.255.255.192
DENY 193.62.192.128 255.255.255.192     # low bits in 131 are masked out
DENY 193.62.192.128/26                  # 255.255.255.192 is 26 ls+6 0s
DENY net.bio.net/26                     # net.bio.net = 193.62.192.131
#
# We want to deny access to all 1.2.*.* except 1.2.3.*
ALLOW 1.2.3.0/24      # Matches & allows 1.2.3.4, doesn't match 1.2.8.9
DENY  1.2.0.0/16      # Matches & denies 1.2.8.9
ALLOW 0.0.0.0/0       # Matches and allows everything
```

Note that a `/32` or `/255.255.255.255` netmask must be used to match single hosts, where all parts of the address are significant.

Because only the first match is used, exact matches should precede more general matches in the file.

When using network access control, remember to allow access to the loop-back network on the local machine, as this is used by the RSF-1 user interface utilities rsfcli and rsfgui, e.g. `ALLOW 127.0.0.0/8`.

Note that RSF-1 network access controls do not apply to applications or services. These must implement their own access control mechanisms if required.

7.6.2 User access control

RSF-1 maintains its own list of user accounts for authentication of individual administrators using monitoring clients. User access control is applied to all client requests that involve a change of cluster state (e.g. starting or stopping services, changing switchover modes), *after* being cleared by any network access controls.

RSF-1 user names and passwords are *separate* to those used by the underlying operating system and must be maintained separately.

RSF-1 user names and passwords are stored in a file called `/opt/HAC/RSF-1/etc/passwd` on *each* server. The passwords are stored in an encrypted form. To maintain consistency, it is easier if this file is only ever altered on one server and then copied to all the others using a similar mechanism to the configuration file.

The command used to maintain the RSF-1 user names and passwords is `rsfpasswd`.

This only changes the password file on the local server. It has four possible actions:

- `rsfpasswd -a` *`username`* adds the specified user, prompting you to enter a password for that user and repeat it to verify.
- `rsfpasswd -c` *`username`* changes the password for the specified user, prompting you for the new password twice. An error message appears if the user does not already exist in the password file.
- `rsfpasswd -d` *`username`* deletes the specified user.
- `rsfpasswd -h` will list the possible options.

To make full use of `rsfcli` (see section 9.4.1,) you may wish to create a `default _rsfadmin` account initially. This may then be used when running as root by using the `-i 0` option of `rsfcli`, and allows scripts to control rsfmon without needing to put plain text passwords in

them (the password is still chosen by you and will be encrypted in the same way as all the other accounts).

Please note that, although RSF-1 always transmits requests and passwords across the network in an encrypted form, you should still choose safe, secure passwords that cannot easily be guessed. Avoid dictionary words, names or personal details. Use a combination of letters and numbers, perhaps mnemonically.

7.7 Summary

The following files and directories should normally be kept identical on every RSF-1 server in a cluster:

- `/opt/HAC/RSF-1/etc/config`
- `/opt/HAC/RSF-1/etc/rc.*.d/`
- `/opt/HAC/RSF-1/etc/passwd`
- `/opt/HAC/RSF-1/etc/access`

In the standard RSF-1 distribution an example cluster file (`/opt/HAC/RSF-1/etc/config-example.txt`) is included. It provides a complete example of the areas covered in this section of the manual.

8 Starting and Stopping RSF-1

8.1 Chapter overview

This chapter describes how to start RSF-1 initially, the various methods for stopping it, and how to restart it.

Note: In our experience, incorrect starting and stopping of RSF-1 and RSF-1 servers can sometimes lead to service confusion and even *split brain syndrome*. Whilst every effort has been made to prevent such problems occurring High-Availability.Com will not be held responsible for data loss or downtime incurred as a result of failure to follow the correct procedures. You should take particular care that all administrators and engineers are aware of the following information.

8.2 Starting RSF-1 for the first time

This section assumes you have completed the installation procedure. Before beginning, check that RSF-1 is not already running. If you have rebooted the servers after installation and configuration, it will have started automatically (but the services will be dormant). Commands to verify this are illustrated below. Otherwise, use the following procedures to initialise RSF-1.

8.2.1 Starting the heartbeats

To initiate heart-beating and monitoring, start the `rsfmon` processes on each RSF-1 server by executing:

```
# rsfctl start
```

This will not start the services (the initial switchover modes will be manual). If `rsfmon` is already running, this will be detected, an error will be logged and the new daemon will exit.

Check that the `rsfmon` processes are running on each server using the `ps` command, usually:

```
# ps -ef | grep rsfmon
```

You should see at least two such processes. There is one parent, and a child for each configured heartbeat type.

Also check the RSF-1 log-file in `/opt/HAC/RSF-1/log/rsfmon.log` for error messages, such as typographical mistakes in the configuration files. Ensure that all the configured heartbeats are running correctly.

If the `rsfmon` processes do not start correctly, you must investigate and fix the problem before proceeding.

To start RSF-1 services, use the administration console, see the Cluster Management Console guide for more details

8.3 Stopping RSF-1

This section describes how to stop RSF-1 and the services completely before power outages or other critical situations.

8.3.1 Stopping RSF-1 and services

There are three methods for safely shutting down RSF-1 and the services together:

1. Using the console.
2. Using the shutdown script.
3. By sending a TERM signal to the monitoring process.

When RSF-1 is shut down using the following procedures, it first executes the shutdown scripts for all services currently running on the server before terminating.

As discussed in the previous section, we recommend that you ensure switchover modes are set to manual before shutting down services.

8.3.1.1 Using the console

Use the console to shut down each running service, as described in the Cluster Management Console guide, then follow the procedures described in 8.3.2 to stop RSF-1.

8.3.1.2 Using the shutdown script

Execute the shutdown script as root with an argument of stop:

```
# rsfctl stop
```

There is a short delay before shutdown commences, to give you the opportunity to interrupt the process. When the script terminates, RSF-1 and the services on this server have been shut down.

8.3.1.3 Sending a TERM Signal

Use the Unix kill command to kill the parent `rsfmon` process on each server by sending a TERM signal (default):

```
# kill `cat /etc/rsfmon.pid`
```

On receiving a TERM signal, rsfmon first stops all running services before terminating.

As before, check that the processes have died. If any remain, execute the kill command again with the PID(s) from the output of the `ps` command.

Remember that in this case RSF-1 will wait until it has shutdown all services. This may take several minutes for some applications databases, so allow enough time for such service shutdowns to complete before sending any other signals.

8.3.2 Shutting down RSF-1 only (not recommended)

Stopping RSF-1 monitoring on only one server will result in service failover if the switchover mode for it is set to automatic elsewhere! Do not perform this action on individual servers unless you have ensured that the relevant switchover modes are manual. In fact, we recommend that you confirm this is the case before proceeding with *any* RSF-1 maintenance, to guarantee that undesirable failovers cannot occur (changing the switchover modes can be done from the console).

Remember, by shutting down a single instance of RSF-1, other RSF-1 servers in the cluster assume the node has gone down and, if in automatic mode, will initiate fail over for services they are capable of running. Please note that shutting down `rsfmon` is not the recommended way to stop it and is done so at your own risk.

There are three ways to stop the RSF-1 monitoring processes:

1. Using the console.
2. Using the shutdown script.
3. By sending a KILL signal to the parent rsfmon process.

8.3.2.1 Using the console

See the Cluster Management Console guide for instructions on using the graphical administration console to shut down the cluster.

8.3.2.2 Using the command line interface

Execute the RSF-1 shutdown command line interface with an argument of *kill* on each server:

```
# rsfctl kill
```

(Note: do not use an argument of *stop*). There will be a short delay before the `rsfmon` processes are killed.

Using the commands in the previous section, check that the rsfmon processes have died. If the parent process remains on any server, proceed to the next section.

8.3.2.3 Killing RSF-1 manually

Use the UNIX kill command to kill the parent `rsfmon` process on each server by sending a KILL signal:

```
# kill -KILL 'cat /etc/rsfmon.pid'
```

As before, check that the processes have died. If any remain, execute the kill command again with the PID(s) from the output of the ps command.

8.3.3 Important notes about Stopping RSF-1

Should a server be halted or rebooted after terminating rsfmon, the services it is running will not be safely stopped by RSF-1 if it remains disabled.

RSF-1 and the enabled services will be started automatically if a server is rebooted after stopping RSF-1. If only one server is rebooted and a sibling switchover mode for a service it was running is set to automatic, a failover will occur.

Therefore, avoid stopping RSF-1 prior to shutting down the system(s,) unless the services are also safely stopped.

8.3.4 Signals and RSF-1

`rsfmon` catches the TERM signal and first stops all running services before terminating. The KILL signal cannot be trapped, therefore causing only the monitoring to cease (no other clean-up is necessary and `rsfmon` can be safely restarted).

Most operating systems send a TERM signal to all running processes when the halt or reboot commands are used; if there is time, this will cause `rsfmon` to shut down running services first. Because of this behaviour, halting a server is not an effective method of testing failover, as it does not properly simulate a sudden crash or unclean termination.

Be very careful when killing the `rsfmon` process - as it has the potential to lead to split brain situations. Only use the kill signal if you understand all the implications of this.

8.4 Restarting RSF-1

RSF-1 can be restarted either without affecting currently running services, or in a default state where it will start the services normally.

Unless you have left switchover modes in manual as recommended, you should restart RSF-1 on each server simultaneously (or at least within the shortest defined timeout period).

8.4.1 Restarting RSF-1 with Services Left Running

Use this procedure when you have stopped RSF-1 without shutting down the services, perhaps for maintenance.

RSF-1 creates a lock file for each service by which it can remember service states between restarts. To restart RSF-1 so that it will assume the previous state is still valid (i.e. the services have not altered in the interim,) execute the start-up script with an argument of restart on each stopped server:

```
# rsfctl restart
```

Check that all the rsfmon processes have been restarted. Connect to the services using the console and ensure that they are operating normally and that the heartbeats are polling.

8.4.2 Restarting RSF-1 and Services

Use this procedure when both RSF-1 and the services have been shut down and you wish to restart both.

Execute the start-up script with an argument of *start:*

```
# rsfctl start
```

Executing this command is equivalent to starting RSF-1 during a normal system boot.

The previously set switchover modes are retained and RSF-1 will start services that are not running if it determines that it should.

Use the `rsfmon` log file and the console to verify that the services are available once more.

8.4.3 Note about Restarting RSF-1

Do not manually alter the service states outside the control of RSF-1, e.g. by executing service scripts from the command line. Changes such as these will not be noticed by RSF-1, leading to possible problems later.

8.5 Summary

Table 11 Summary of rsfctl arguments

start	Start RSF-1 and services
stop	Stop RSF-1 and services
restart	Restart RSF-1 after kill without affecting services
kill	Stop RSF-1 without affecting services
dump	Same as `rsfdump`
debug	Same as `rsfdebug`

9 Other RSF-1 Utilities

9.1 Chapter overview

This chapter describes the RSF-1 log file and miscellaneous administration utilities.

9.2 Introduction

RSF-1 provides three ways to monitor its operation:

- Using the console utility to give a real time display of RSF-1 status.
- Using the RSF-1 and system logs to observe cluster actions and service activities.
- Using miscellaneous utilities such as `rsfdump` to gain low level information.

Each has a role to play in enabling you to understand and influence how RSF-1 behaves.

9.3 RSF-1 Log

The main RSF-1 monitoring process, `rsfmon`, logs messages to its log file, `/opt/HAC/RSF-1/log/rsfmon.log`.

This log file is never cleaned out or recycled by RSF-1, but will be rotated through ten iterations every time RSF-1 is started afresh (e.g. at system boot).

In addition, `rsfmon` sends log messages at appropriate levels to `syslogd` using the facility code specified in the configuration file, LOCAL0 by default. You can configure `syslogd` to redirect these messages to users, files, or remote hosts (usually by editing `/etc/syslog.conf`). The priority levels used for various types of log message are shown in Table 12.

Table 12 RSF-1 syslog priorities

debug	Certain debugging output, if enabled.
notice	Service start-ups and shutdowns.
alert	Service state change requests.
crit	Script execution notices, process monitoring checks.

You should always view the log file when testing or debugging your services, or when diagnosing RSF-1 problems. RSF-1 can log a wide variety of messages, including events such as:

- Starting or stopping RSF-1.
- Starting or stopping a service.
- Losing or regaining a heartbeat channel.
- Losing or regaining contact with a remote host.
- Losing a service.
- Requesting switchover mode changes or service switchovers.

9.4 Miscellaneous

9.4.1 rsfcli (Command Line Interface)

The `rsfcli` command provides access to some simple monitoring and control functions. It is intended for use in scripts.

`rsfcli` has some generic switches that can be used with any of the subcommands to specify a particular server or the authentication details. These are shown in the following table.

If the username and password are not specified, `rsfcli` attempts to find a working combination:

- If running as root in non-interactive mode, if `-i 0` is specified as an argument to `rsfcli`, or the environment variable `RSFCLI_INTERACTIVE` is set to 0, then `rsfcli` looks for a user called `_rsfadmin` in the local RSF-1 password file (assuming there is one). It partially decrypts the password for this user and uses it as the key for requests. Effectively, no username or password need be specified, making it suitable to use in scripts and monitoring agents. You must create the `_rsfadmin` account before using this feature of `rsfcli`.
- If `rsfcli` is not running as root and/or it is in interactive mode, it will prompt the user to supply a username and password if they are required.
- If `rsfcli` is not running as root and it is in non-interactive mode (no attached tty) it uses an empty username and password. Under normal circumstances this will fail as `rsfcli` will attempt to read the `_rsfadmin` entry from the local RSF-1 password file which should only be readable as root.

(The environment variable `RSFCLI_INTERACTIVE` can also be set to the digit zero or one, to force a particular mode).

Table 13 rsfcli switches

Switch	Purpose
-d level	Set amount of debugging information (0-9).
-v	Verbose mode, gives more information in status displays.
-l	Logging output.
-a subcommand	Subcommand to use (the switch is optional).
-c server name	Initial server to use when connecting to the cluster (default: localhost).
-s service name	Service to which subcommand applies.
-h server name	Server to which subcommand applies (default: initial cluster server).
-u username	User name for authentication.
-p password	Password for authentication.
-H	Print help message for given subcommand and exit.
-V	Print version number of `rsfcli` and exit.
-i 0/1	Disable/enable interactive input (1=enable, default: interactive when attached to a tty, non-interactive otherwise).

The `-h` switch can be used to query or manipulate remote RSF-1 servers, and can be given with any subcommand.

Some of the `rsfcli` subcommands have alternative forms (listed second) that are backwards-compatible with RSF-1 version l. However, you should avoid using these legacy forms where possible and update any existing scripts that already use them.

The available commands are:

9.4.1.1 List

Format: `rsfcli [-h server] [-v] list`

List all the services in the cluster or on the given server, with their status (current server or 'not running'). Using the verbose option will list information about every configured service on each server, including switchover mode, floating address and network device.

Example output:

```
% rsfcli list
oracle  : ipll0
    nfs : ipll0
apachel : oregano
apache2 : not running
```

9.4.1.2 Heartbeats

Format: `rsfcli [-v] heartbeats [server]`

List the number of working heartbeats of each type on every server. Using the verbose option lists the source (device or host name) and destination of each heartbeat. Using the <server> option means only the heartbeats set up on that server will be listed.

Example output:

```
% rsfcli heartbeats
ip11O: net=5 disc=0 serial=0
ipl01: net=5 disc=0 serial=l
oregano: net=6 disc=l serial=l
bayleaf: net=6 disc=l serial=l
```

9.4.1.3 Vname

Format: `rsfcli −h <server> [-F <floatname>] vname`

The vname command can be used to query RSF-1 for the status of floating names currently attached to the server. Run without arguments, it prints a usage message and exits.

The -F switch checks whether the floating name supplied as an argument is currently attached to the server, and indicates the result in the exit code (0=yes, 1=no, 2=error) and output (running|stopped).

9.4.1.4 Start

Format: `rsfcli −s <service> start | rsfcli start <service>`

Start the specified service if it is not already running in the cluster. This will cause the appropriate switchover mode on the server to be set to automatic.

9.4.1.5 Stop

Format: `rsfcli −s <service> stop | rsfcli stop <service>`

Causes the specified service to be shut-down gracefully. If the appropriate switchover mode on a sibling server is set to automatic, this will cause a failover.

9.4.1.6 Auto

Format: `rsfcli −s <service> auto | rsfcli auto <service>`

Sets the switchover mode for the specified service to automatic. If the mode is already automatic, it has no effect.

9.4.1.7 Manual

Format: `rsfcli −s <service> manual | rsfcli manual <service>`

Sets the switchover mode for the specified service to manual. If the mode is already manual, it has no effect.

9.4.1.8 Bounce

Format: `rsfcli -s <service> bounce | rsfcli bounce <service>`

Shut down and restart the specified service on its current server.

9.4.1.9 Restart

Format: `rsfcli -s <service> restart | rsfcli restart <service>`

Cause the start-up sequence for the specified service to be rerun on its current server. This has no effect if the service is not already running.

9.4.1.10 Pass

Format: `rsfcli -s <service> pass`

Shut down the service on the current server and cause it to be restarted on the first available server in the cluster.

9.4.1.11 Shutdown

Format: `rsfcli shutdown | sysdown`

Stop RSF-1 processes and services on the server; `sysdown` is an alias for shutdown.

9.4.1.12 Move

Format: `rsfcli move <service> <server>`

Move (failover) the specified service from its current server to the target server. This will cause the service to be stopped on the current server and then started on the target.

9.4.1.13 Repair

Format: `rsfcli [-h <server>] -s <service> repair |`

`rsfcli [-h <server>] repair <service>`

Reset a service instance currently in the broken state to a safe (retry-able) state. The service must be broken on the local or given server.

9.4.1.14 Dump

Format: `rsfcli dump`

The `dump` command causes `rsfmon` to write the current state of the services to the file `/tmp/rsfmon.stat`. It also resets the internal debugging level of `rsfmon` to zero.

An example of an output file is shown below. The output shows the status and details of all heartbeats between the local server and its siblings, and the status of each service.

```
-----RSFMON, 3.0.24 internal status @ Tue Dec 18 14:11:42 2007
Net_heartbeat (ID=5) from tom to jerry, state = Up
        address:port = jerry:1195
Net_heartbeat (ID=6) from tom to jerry, state = Up
        address:port = jerry-priv:1195
Net_heartbeat (ID=9) from bugs to jerry, state = Up
        address:port = jerry:1195
Serial_heartbeat (ID=7) from tom to jerry, state = Unavailable
        device = /dev/cua/b
Service filestore is running, automatic/unblocked on jerry
-----End Of Dump
```

9.4.1.15 Debug

Format: `rsfcli debug`

The `debug` command progressively increases the debugging level of `rsfmon` from 0 to 9. Debugging output is sent to the log file and `syslogd` at the appropriate priority. At higher levels, more activity is logged, including the contents of heartbeat packets. **Beware** detailed debugging may cause your log files to overflow.

The debugging level can be reset to 0 (disabled) by executing the `dump` command (debugging is mainly intended for use by developers).

9.4.1.16 Is running

Format: `rsfcli isrunning`

The isrunning command checks that RSF-1 is running on the current server. The result is reflected in its return code (0=yes, 1=no). This command is silent.

9.4.1.17 Status

Format: `rsfcli status | rsfcli stat`

The status command prints the status of all nodes, services and heartbeats in the cluster. An example of the output is shown below:

```
Contacted localhost in cluster "HA-Cluster", CRC = 0x2e82

Host test1 (192.168.33.91) UP, service startups enabled,
  RSF-1 release 3.7.2, built on 20-Feb-2013-12:17 "3.7.2".

Host test2 (192.168.33.92) UP, service startups enabled,
  RSF-1 release 3.7.2, built on 20-Feb-2013-12:17 "3.7.2".

2 nodes configured, 2 online.

0 Service tank, IP address vip01, "RSF-1 tank ZFS service":
    running automatic unblocked on test1
    stopped manual unblocked on test2

1 service configured
    1 service instance stopped
    1 service instance running

Heartbeats:
0 net test1 -> test2 [192.168.33.92]: Up, last heartbeat #239 Mon Mar  4 14:21:31
1 disc test1 -> test2 (via /opt/HAC/RSF-1/dev/hb1:518,/opt/HAC/RSF-1/dev/hb1:512) [(20]:
DiscCheck, last heartbeat #0
2 serial test1 -> test2 (via /dev/cua/a): Up, last heartbeat #237 Mon Mar  4 14:21:30
3 net test2 -> test1: Up, last heartbeat #258031 Mon Mar  4 14:21:30
4 disc test2 -> test1 (via /opt/HAC/RSF-1/dev/hb1:512,/opt/HAC/RSF-1/dev/hb1:518) [(20]:
DiscCheck, last heartbeat #0
5 serial test2 -> test1 (via /dev/cua/a): Up, last heartbeat #258031 Mon Mar  4 14:21:30
6 heartbeats configured, 4 up, 2 down
```

9.4.1.18 Modes

Format: `rsfcli modes`

The modes command shows a short summary of the cluster status. This information contains the status of each node (whether it is online) and the status of each service (running, stopped, broken) on each node.

9.4.1.19 RSF Log

Format:

`rsfcli rsflog <severity> <message> | rsfcli msg <severity> <message>`

The rsflog command writes a message to the RSF-1 log file (/opt/HAC/RSF-1/log/rsfmon.log), with the specified severity. Currently supported severity levels are:

emerg, alert, crit, err, warn, notice, info, debug.

E.g. the command ‘`rsfcli rsflog notice "Notice message"` ‘ appends the following line to the log file:

```
[11548 Mar  4 14:46:01] NOTICE: notice message
```

9.4.1.20 Using rsfcli vname in scripts

The RSF-1 command-line interface (`rsfcli`) provides a facility 'vname' (it stands for *virtual name*) which can be used to write scripts whose behaviour is dependent on the state of a service. This command has two main options: the first lists available floating names and the other tests if a host is currently controlling a floating name. Since each service has a floating name associated with it, this allows a script to test for the presence of a particular service.

Listing the floating names

The first example lists the floating names from a test cluster. It connects to a single machine in the cluster `brain` and prints a list of the available names (in this case, there's only one).

```
#
# List the floating names that can live on a host (in this case brain)
#
./rsfcli -f -h jerry vname
pork
bugs
```

Here is another example, but this time there are two possible names as can be seen in the following extract from an `rsfcli list` command:

```
jerry:
       sshd2  running     auto         pork    le0:8
       sshd1  running     auto         bugs    le0:7
```

Testing a host for a floating name

The most powerful use of the **vname** facility is for determining if a given floating name is held by a particular host. The two commands below show the result this produces when run on the host `brain` which currently controls the floating name `nibbles` and its partner `pinky` which does not.

```
#
# Test if a floating name is currently running on a host
#
./rsfcli -F nibbles -h brain vname
running
echo $?
0

#
# Test if a floating name is currently running on a host
#
./rsfcli -F nibbles -h pinky vname
stopped
echo $?
1
```

As you can see `rsfcli` produces different output in the two situations, and also returns different status results. This can be used to make scripts that have different behaviour depending on the state of a service as illustrated below.

```
#
```

```
# Execute a command if (and only if) this host currently has the
# floating IP address 'nibbles'.
#
RSFCLI='/opt/HAC/RSF-1/bin/rsfcli'
${RSFCLI} -h localhost -F nibbles vname > /dev/null
if [$? == 0]; then
    echo 'I have the ball'
else
    echo 'Someone else has the ball'
fi
```

This facility can be used in a number of ways (for example it might be used to make a script operate only when its host is running a particular service).

9.4.2 hacdiag

`hacdiag` gathers diagnostic information about a running RSF-1 server and its environment. The data is archived and compressed for despatch to High-Availability.Com or your reseller. Please run `rsfdiag` on all nodes in the relevant cluster and attach the output file to an email when logging a support call or requested by an engineer.

10 Modifying RSF-1 Operation

10.1 Chapter overview

This chapter explains how to modify certain aspects of RSF-1, and how to make those modifications take effect. You may wish to modify RSF-1 either to tune its operation or to add new applications later.

10.2 Changing Service Parameters

You may edit the configuration file manually with a text editor at any time. (Chapter 10 describes the various parameters in the file). Because RSF-1 only reads the configuration file on start-up, changes will not take effect immediately. For some minor modifications, you need only restart RSF-1. For others, you may have to restart services as well.

Restarting RSF-1 is covered in chapter 8.4 Restarting RSF-1. Note that if you have edited the configuration file, you must stop all running instances of RSF-1 first, then restart them (otherwise it will detect configuration differences).

After editing the configuration file, remember to copy the new version to all RSF-1 servers and perform the required actions on each where necessary. We recommend that you nominate one 'master' RSF-1 server where these changes are initially made, to prevent confusion. Because the configuration file must always be kept in sync on all servers in a cluster, you must also copy it to servers unaffected by the changes and restart those too.

As mentioned previously, you should set all switchover modes to manual prior to taking any action that affects RSF-1 monitoring.

10.2.1 Changing a service name or floating name

Before changing either name, you must first stop the service (to ensure that it will not automatically failover!) You can change the names by editing the `SERVICE` definition in the configuration file. You must also rename the service script directories appropriately on all affected servers if changing the service name.

Restart RSF-1 and then start the service on the server where it was previously running. You may want to initiate a failover from the console to test your changes.

10.2.2 Changing the service description

The service description is shown in the RSF-1 console but is otherwise unused. Therefore, you need only restart RSF-1 if you wish the change to show up immediately. Edit the `SERVICE` definition in the configuration file.

10.2.3 Changing the hostname of a server

If you are changing the real hostname of a server but not the machine itself, simply edit the `MACHINE` definition and any `SERVER` entries in the configuration file and restart RSF-1 on all servers. There may be other changes you must make to the operating system, clients or applications before doing this.

10.2.4 Changing the timeouts

You may wish to reduce the timeouts to make failover occur sooner. Alternatively, you may wish to increase them to avoid false alarms or give servers more time to reboot. Simply edit the appropriate values under the `SERVICE` definition in the configuration file and restart RSF-1 on all servers.

10.2.5 Changing the heartbeats

If you subsequently wish to enable, modify or disable the disk, serial or network heartbeats, you must edit the appropriate `MACHINE` definitions in the configuration file. The following is a quick guide to changing each type of heartbeat:

- **Network** Changing the interface may involve moving the physical connection to the server and altering the system boot configuration to bring up the new interface instead. Changing the hostname will involve updating the naming service and editing all the `NET` heartbeat definitions that mention the hostname in the configuration file.
- **Serial** Changing the port will involve moving the physical connection and editing the `SERIAL` heartbeat parameter for this `MACHINE` in the configuration file.
- **Disk** To change the disk partition or block offset being used, edit the `DISK` heartbeat definitions for both affected `MACHINES`. Any of the heartbeats can be disabled or removed by commenting or removing the relevant heartbeat definitions in the configuration file. Remember that each heartbeat involves two `MACHINES`.

It is unwise to alter every heartbeat from a server at the same time, as there is a risk of disrupting all of them, causing a failover to occur. After making your changes, restart RSF-1 on all servers.

10.3 Changing the service scripts

You can edit or rename service scripts at any time. If you rename a start-up script, be careful to recreate any symbolic links to it. If some aspect of an application has changed, for example, the file-systems have been moved, it is safest to stop the service before editing the scripts.

Scripts can be added, for example, to add a new application to an existing service. If you are adding new disk groups for use by an application, you can create a new script to handle them or modify an existing fstab file for that service.

10.4 Removing or replacing a service

Before removing or changing a service, you must first stop it within RSF-1. A service can be removed by deleting or commenting its `SERVICE` definition in the configuration file.

You do not have to delete or move the service script directory. To add a new service, follow the instructions in chapter 7. After adding the service, restart RSF-1 on all servers.

10.5 Changing the ports used by RSF-1

RSF-1 uses the `rsfreq` and `rsfnet` ports. The numbers associated with these ports are defined in the `/etc/services` file or table. These numbers can be changed to any available pair of numbers between 1 and 65535. Edit the services file or table and restart RSF-1. Ensure that the ports you have chosen are not already used by another application, by using `netstat` or a similar utility.

All RSF-1 servers must use the same port numbers.

10.6 Changing a Server

10.6.1 Hardware

If you change or upgrade the hardware for one of your RSF-1 servers and/or reinstall the operating system, it is easiest to reinstall RSF-1 and then copy or restore the configuration file and `/opt/HAC/RSF-1/etc` directories from the original server.

Remember to recreate all mount points and other disk information on the new system. Providing the networking configuration has been duplicated and the original host has been switched off or reassigned, no further changes are necessary. However, do note that if the machine ID has changed you will need to acquire a new license from High-Availability.Com.

10.6.2 Software

An operating system upgrade (as opposed to a reinstall) should not affect RSF-1 providing the release you have remains compatible with the new OS release. Contact High-Availability.Com if you are unsure.

10.6.3 Network

If network changes have been made to a server - for example, changing its node name or address - you should shutdown RSF-1 on all servers, make the necessary changes to the configuration file, and restart it once the new server is in place.

10.7 Upgrading RSF-1

The procedures for upgrading RSF-1 are covered in a separate document entitled “RSF-1 V2 Upgrade Guide”. This is available from the High Availability WEB site http://www.high-availability.com or alternatively can be located in the docs subdirectory of the RSF-1 installed package.

High-Availability.Com provides professional services to assist customers in performing installs, upgrades and audits should they so require. Please contact the support number available from the High-Availability.Com WEB site for more information.

11 Towards Total Availability

11.1 Introduction

Congratulations, you should now have a working high availability cluster! However, by itself RSF-1 often forms only one element of an overall availability solution. You should consider *all* points of failure that could potentially affect your cluster, and address them as far as is practical, according to cost and impact.

In this chapter, we mention several other considerations to take into account when examining system availability. By looking at these, you can enhance overall availability, protect your investment in RSF-1 and ensure that its effectiveness is not compromised by factors outside its remit.

11.2 Physical security

Are your systems housed in a secure area? Which personnel have access to them, and how is that access controlled and monitored?

11.3 System security

Are you up to date with vendor security patches? Do you follow the well known advisory and alerting lists, such as CERT? Is there a local security policy? Are superuser accounts protected sufficiently well? Do you regularly change account passwords? Are user privileges controlled? Is there a formal backup regime and schedule? Are backups held at an offsite location? Are correct backups taken even if services are running on secondary servers?

11.4 Related components

Is your network resilient to failure? Is data storage sufficiently protected (with mirroring, backups, hot spares, etc?) Are the servers and peripherals connected to a UPS? Are they on separate electrical feeds?

11.5 Maintenance

Empirical data shows that many cluster products become ineffective soon after installation due to system changes that affect their operation; for example, changes to file-system configuration or accounts on one system not being carried through to sibling servers, with the result that one or more services will no longer failover.

Because of this, it is a good idea to schedule regular failover tests at convenient times. The depth of this testing is your decision, but it should at least verify that failed servers are detected properly by siblings and that services are still capable of running correctly on all sibling servers.

After testing RSF-1, remember to reset switchover modes to automatic.

11.6 Training

Many more system failures are caused by operator error than faulty hardware. Are your staff fully conversant with RSF-1 and the cluster servers? Do they understand how to monitor RSF-1 operation, detect that a failover has occurred and recover from it? Are they aware of any special requirements for administering cluster servers (synchronisation of critical files, use of floating addresses, etc.?) Is the documentation in a readily-accessible place?

Are application programmers aware of the issues in running their software on clustered servers (principally that their software must be capable of running on more than one specific host and must survive system crashes?) Do their applications need to interact with RSF-1 in any way (using `rsfcli`?)

You may wish to create a separate 'laboratory' cluster for testing and training purposes.

You should also ensure that visiting field engineers know which servers are part of RSF-1 clusters and how to administer them.

11.7 Disaster planning

If the systems are damaged or destroyed by external events, are you able to recover them fully from backups and other recorded information? Is there a clear description of the cluster installation?

RSF-1 supports the use of geographically-disparate servers. However, because of complex issues such as data replication/synchronisation and communication, more care, effort and expenditure is required in setting up clusters for disaster recovery.

High-Availability.Com can recommend a number of strategies and useful third party products for such purposes; contact us to find out more.

Appendix A. Glossary of Terms

Address Resolution Protocol (ARP) A protocol used by network hosts to map logical (IP) addresses to MAC addresses.

Alias Another term for a *virtual network interface*. Do not confuse these aliases with standard hostname aliases.

Cluster A number of RSF-1 *servers* running a defined set of *services* and monitoring each other for failures.

Disk group, Diskset A logical grouping of disks within disk management software. Access to a disk group is typically granted to only one host at a time. A disk group must contain one or more complete file-systems. For example, you may use separate disk groups to hold the application data for each of your services.

DNS The domain name system is a distributed database arranged hierarchically. Its purpose is to provide a layer of abstraction between other Internet services (web, email, etc). and the numeric addresses (IP addresses) used to uniquely identify any given machine on the Internet.

Floating licence An application licence that is valid on any one of a number of hosts, but permits only a certain number of instances of the application to be run simultaneously, on the network. (RSF-1 licences *are fixed* to particular servers).

Heartbeat A small packet exchanged between two RSF-1 servers to indicate that the sender is alive, which services are running on it and what control requests have been made.

High-Availability (H.A). The provision of an application or service on a near-continuous basis, regardless of certain common types of hardware and software failure, and maintenance requirements. Highly available systems use multiple, redundant components and monitoring software (such as RSF-1) to minimise disruption caused by such failures.

IP or Internet Protocol is the specification that determines where packets are routed to, based on their destination address.

LAN Local area Network.

MAC Address Ethernet Media Access Control.

Master configuration server The RSF-1 server on which all configuration changes are made before being replicated to the other RSF-1 servers. This ensures that the service configurations remain synchronised within the cluster.

Maximum Segment Length (MSL) The longest piece of cable in an Ethernet segment.

Maximum Segment Life (MSL) The maximum time for which a TCP/IP packet can remain on the network before expiring. All TCP socket connections carry a timeout twice as long as this, during which a closed port cannot be reopened. This prevents a new process receiving packets intended for a previous one, that were still in transit at the time it died.

The MSL period is typically four minutes.

Network Naming Service A repository of hostname to IP address mappings and possibly other information, e.g. DNS or NIS.

Primary The RSF-1 *server* that normally runs a particular *service* and has the highest priority.

Secondary A RSF-1 *server* that runs a particular *service* when the *primary* server fails.

Server *A* host running the RSF-1 software. See also *primary* and *secondary*.

Service A transferable unit consisting of application start-up and shutdown code, its network identity and its data. Services can be migrated between RSF-1 *servers* either manually or automatically upon failure of one server.

Sibling A *server* in a *H.A. cluster* sharing one or more *services* with other servers. A server must be a sibling of at least one other server, unless RSF-1 is being run on a single host for use as a management tool only.

Split brain syndrome An erroneous condition occurring when more than one *server* is running the same *service*, leading to confusion at the client end and possible data corruption. This should not occur in normal use with RSF-1. It is usually caused by mis-configuration or failure to follow the correct procedures.

Single Point Of Failure (SPOF) A component or element in a system, the failure of which would disable the entire system and prevent further operation. Most HA clusters aim to eliminate SPOFs through component redundancy and failover.

Subset A group of *sibling* servers configured within a *cluster* that have no *services* or in common with other servers in the same cluster. Subsets confer no benefit and indeed make proper cluster management difficult.

TCP, or Transmission Control Protocol, makes sure that the packets arrive correctly at their destination address. If TCP determines that a packet was not received, it will try to resend the packet until it is received properly.

Virtual network interface (VIP) An additional IP address and hostname bound to an existing physical network interface. The interface will then accept packets to any of the bound addresses.

WAN Wide Area Network.

Appendix B. RSF-1 for System Engineers

B.1 To the Administrator

This appendix provides essential information to visiting system or field engineers who may be working on RSF-1 servers. It covers issues to be aware of and describes procedures for safely shutting down or rebooting servers. Before commencing work, please ensure that the person involved has read this information.

This appendix is not a complete substitute for the detailed information provided elsewhere in this guide.

B.2 To the Engineer

If you are reading this, one or more of the systems you are working on has additional High Availability software installed. Because of this, there are certain factors you must be aware of before commencing maintenance work on these systems, to avoid causing malfunctions in the local applications or network. Therefore, please read the following information carefully before proceeding further.

B.3 About RSF-1 High-Availability software

High availability is here provided by means of a product called RSF-1, which is distributed by High-Availability.Com. In a highly available system, applications are backed up using redundant hardware and a continuous system monitoring process. In the event of failure of a running system, applications and data are automatically migrated to a standby system and restarted.

Note that RSF-1 does *not* distinguish between a failed system and a system that is shut down for maintenance. Furthermore, this *failover* process is not transparent to end users and may cause unforeseen problems if it is not properly managed.

Before shutting down or rebooting one of the RSF-1 servers, you will therefore need to perform one of two actions:

- Disable automatic failover capabilities to prevent confusion. This will result in downtime for the affected applications.

or

- Failover the applications on the server in question, allowing users to continue accessing them while you proceed.

Before continuing, agree an appropriate course of action with the system manager. They may be able to assist you in carrying it out.

B.4 Identifying RSF-1 in operation

A system actively running RSF-1 can be identified by the following means:

- The presence of a directory called `/opt/HAC/RSF-1`.
- One or more `rsfmon` processes running as root.

To discover if the system is currently running RSF-1 *services* (which represent applications,) you must use the administration console, `rsfgui`. `rsfgui` is located in `/opt/HAC/RSF-1/bin` if it is not already in the PATH.

It may also be installed on a remote management station, `rsfgui` is illustrated and discussed more fully in the Cluster Management Guide, but the following information should be sufficient.

To run the console, enter:

```
# rsfgui cluster <servername>
```

where *servername* is the node name of any RSF-1 server in the cluster.

On running the console, you will be presented with the *server view,* a list of RSF-1 servers. To gain a quick overview of running services, select View->Show_Overview from the menu. The overview displays a grid of servers against services in a separate window. Running services are marked with a green bullet. Servers that could potentially run a service should the current server fail are marked with a grey bullet. Find the services running on the server you intend to work on.

You can also click on one of the servers listed in the initial server view to expand it into a list of services. From here, you can select various actions to perform on a selected service.

B.5 Disabling failover

To disable failover, select the affected service(s) under *each* server on which it can run in the server view, then choose Actions->Mode->Manual from the menu. This changes the *switchover modes* for the service to manual and prevents each server from running the service should it become unavailable. You will need a valid RSF-1 username and password combination to do this; the password for root on the machine may not be enough as RSF-1 may not be configured to recognise it.

At this point, failover is disabled, although any running services will be unaffected. However, should a server be rebooted, it will *not* start any services automatically. This will need to be done manually by the administrator using the console, as shown in section **Error! Reference source not found.**.

Note: If the current server has the switchover mode for the service left in automatic, it will restart the service on boot.

B.6 Failing over running services

Although RSF-1 in normal operation will automatically failover services should a server be shut down, it is better to initiate a controlled failover manually, prior to shutdown.

Use `rsfgui` as described in section B.4 to discover which services are currently running on the server you intend to disable. Select each service in turn under the server in the server view, and choose Actions->Move->*server_name* from the menu. If there are several possible servers listed, ask the site administrator which one to use.

RSF-1 will then shut down the service using the correct procedure. Once this has occurred, the chosen sibling server will restart it. Following this, the console will show the service running on the other server. (You may wish to check that any affected applications are now running correctly on the sibling server). It is now safe to proceed with maintenance on the local server.

B.7 Restoring normal operation

Once you have completed your work on a server, repeat whichever of the procedures you followed above to restore the cluster to its original state, setting switchover modes to *automatic* as necessary.

B.8 Other Issues

A host running RSF-1 services may *appear* to behave oddly under certain circumstances. In fact, it is likely that RSF-1 is operating normally. Some common sources of confusion are described below.

'Missing' Disk Mounts

If the servers are connected to a shared disk, it may appear that certain file-systems from the disk that contain applications and data are not mounted. These file-systems are only mounted by RSF-1 when it is running the relevant service. Similarly, logical disk groups created in disk management software may not be available. If you require access to these file-systems, you may manually import and mount them, assuming they are not in use elsewhere. Ensure that you unmount them before restoring cluster operation.

If you do require access to disk groups or file-systems used by RSF-1 services, these services must be shut down on **all** nodes in the cluster first.

The file-system mounts may be specified in the system file-system table (`fstab` or `vfstab` file). Do not modify or remove these entries.

'Missing' file-system table entries

There are also extra file-system table files under the `/opt/HAC/RSF-1/etc/` directory.

Extra or Missing Network Interfaces

RSF-1 performs IP address failover by means of additional virtual or aliased network interfaces. Again, these interfaces are only present if the relevant service is running.

The fixed host address should always be available.

'Missing' Start-up Scripts

Application start-up is handled by RSF-1, via the scripts in `/opt/HAC/RSF-1/etc/rc.`*`servicename`*`.d`. Therefore, the start-up scripts are not installed in the standard system directories; do not replace them there.

Heartbeat Connections

RSF-1 servers monitor each other via several *heartbeat* links, including network, serial and disk connections. Where these are provided, do not alter or disturb them. Ensure that disconnected cables are reconnected in the correct ports.

Disk heartbeats rely on a dedicated partition in a shared disk device. This device is specified in the configuration files located in `/opt/HAC/RSF-1/etc/config` (look for the DISK parameters). If this disk is replaced, you must recreate the partition. (Special procedures apply to Veritas Volume Manager; see the RSF-1 *Applications Guide*).

Disk Device Numbering

The shared disk configuration and RSF-1 services may rely on the disk device configuration for the servers being *identical*. This particularly applies to device instance numbering. If servers are replaced or rebuilt, ensure that device configuration is preserved. See the *RSF-1 Applications Guide* for details.

B.9 In Case Of Difficulties

See Appendix E for troubleshooting advice. For further help, contact High-Availability.Com Limited.

Appendix C. Service Script Environment

C.1 Introduction

This appendix describes the environment information and capabilities available to service scripts running under RSF-1.

Services under RSF control are started & stopped by a set of scripts. These scripts are of the same form as the system start-up & shutdown scripts normally found in `/etc/rc` directories. This allows such scripts to be used in an RSF installation with very little, if any, modification.

Making use of this information will allow you to write scripts that are more useful, robust and portable. For examples, see the template scripts provided with RSF-1.

If you are using existing template or application scripts without modification, you can ignore this appendix.

C.2 Environment

Before executing service scripts, `rsfexec` sets certain environment variables containing service information passed to it by `rsfmon`. Your scripts may be able to use this information. The following table shows the RSF-1 environment variables.

Table 14 RSF-1 Service Script Environment Variables

RSF_FLOATNAME	Floating service network name or address.
RSF_IPDEV	Device name of the network interface to which the floating name is bound.
RSF_MACHINE_NAME	The name by which RSF-1 identifies this machine in the cluster. This is normally the same as the host name , but should be used when logging messages.
RSF_SERVICE	Name of the service being started / stopped.

In addition, the PATH will always contain the standard system executable directories and the RSF-1 bin directory. If you require additional PATH settings, such as application directories, you must add these near the top of your script.

Note, due to the way in which RSF-1 is run, child processes such as service scripts will not find an attached `tty`, as they would if run from a command line. This occasionally causes problems with some commands, notably `stty`. You should avoid use of these commands or use the < redirection symbol to supply a suitable `tty` device (e.g. `< /dev/console`).

C.3 Environment variables for scripts

The environment in which rsfmon is started is passed on to the processes which run the start/stop scripts. Environment variables to be used by these scripts may therefore be set in the init script which starts rsfmon, that is: `/opt/HAC/RSF-1/init.d/rsfrc`.

Service start-up/shutdown timeouts

When rsfmon initiates a start-up or shutdown of a service, the processes running the start or stop scripts is now timed, and signalled if it takes too long. The default timeout is 15 minutes, though this may be changed using the SCRIPT_TIMEOUT parameter in the global section of the configuration file. When a process it timed out, it is sent the TERM signal, and if it is still running 30 seconds later, it is sent the KILL signal.

New service script start-up/shutdown exit conditions

If a service script start/stop script exits because it was killed by a signal (including a signal sent by rsfmon after a timeout,) then the signal used is examined. If the process was killed by a HUP signal, then the scripts are treated as if they had exited with code 2, failure with restart possible. Any other signal is treated as an exit with code 3, scripts fails, can not be restarted.

C.4 Shell Header File

RSF-1 also includes a header file for Bourne shells that can be sourced within scripts to provide additional variable information and functions. This header file may also be useful when writing support (non-service-related) scripts. We recommend that you include this file at the beginning of every script, with a line like:

```
. rsf.sh
```

For a complete understanding of this file, you should refer to it directly. However the following tables provide brief summaries of its features.

Table 15 RSF-1 shell variables

RSF_DIR	RSF-1 installation directory.
RSF_NAME	RSF-1 OEM name.
RSF_OK RSF_WARN RSF_RESTART RSF_ABORT RSf_SAFE RSF_UNSAFE	Script exit codes. See Table 10 Script exit codes
RSF_OS	Platform operating system name.

Table 16 RSF-1 shell functions

dated_echo	Outputs arguments as message in syslog format (i.e. timestamped).
makelinks	Creates K* symbolic links for stop scripts in reverse numeric order, in given script directories.
checksolfs	Check Solaris UFS file-systems and handles return codes from fsck.
mountfs	Mounts given Unix file-system and handles return codes.

C.5 Start-up Parameters

The scripts are started with two parameters, these are:

- The action to take, one of "start", "stop" or "panic"
- The attempt number, counting the number of attempts to do this action, starting at 1.

Appendix D. Troubleshooting RSF-1

This appendix tells you how to go about diagnosing problems with RSF-1, and how to fix the most common ones.

D.1 General Advice

When RSF-1 does not behave as you expect, use the following procedure to identify the cause.

1. Use this guide to verify that the behaviour you are seeing is not normal.
2. Check the `rsfmon` log (`/opt/HAC/RSF-1/log/rsfmon.log`) file for error messages or clues as to what happened.
3. Try connecting to the servers using the console. If it cannot connect, ensure that `rsfmon` has bound successfully to the `rsfreq` port.
4. Run `rsfdump` on all the servers to dump their internal status. Check that the information is consistent across the servers.
5. List the configured network interfaces on each server using `ifconfig` or `netstat`. Check that none of the floating hostnames are duplicated across the servers.
6. Check your system log file to verify that the problem is not more general.

If you discover a problem with RSF-1, you could try one of the following methods to fix it. Be careful when attempting any of these, be sure you understand their effects:

- Verify that the appropriate switchover modes are set to manual and shut down RSF-1, then start it afresh.
- If a service is running on more than one RSF-1 server (a condition known as *split brain syndrome,)* you must stop *all* the running instances. Check any file-systems associated with the service. Restart the service on one server. Save the `rsfmon` log and any other relevant information, and *contact* High-Availability.Com Limited. In normal use, this situation should never occur.
- Stop RSF-1 and restart it using the *restart* argument to `rsfctl`, but only if an error has been identified and corrected (such as a configuration file error). Always set services to manual mode and restart RSF-1 quickly.

Before contacting your vendor or High-Availability.Com, read Appendix G.

D.2 Common Queries About RSF-1

D.2.1 The rsfmon daemon exits immediately

Most likely, the configuration file is not available or contains errors. Alternatively, there may be a problem starting `rsfmon` - look in the log file (`/opt/HAC/RSF-1/log/rsfmon.log`) or run it manually, perhaps using the `-n` switch to verify this.

D.2.2 Some of the heartbeats are not working

Network heartbeat - Check that the interfaces are configured correctly at boot up. Ensure that the naming services on each server have been updated with the hostnames and addresses of the interfaces. Test connectivity between the servers using `ping`.

Serial heartbeat - Test that your serial cable is wired correctly. Check that no other processes are attempting to use the serial port. Use a command like `tip` or `cu` to connect to the serial port on each server simultaneously; characters typed into one session should then be echoed on the other system.

Disk heartbeat - Ensure that your disk partition is dedicated. Check that both hosts can read the disk partition using `dd` or something similar. Ensure that you have used the correct block offsets.

D.2.3 A service isn't starting up or a fail over isn’t occurring

Ensure that the service isn't already active on another RSF-1 server. “Active” states which will prevent a service from starting on another server are: starting, running, stopping, bouncing, aborting, panicking, panicked and broken_unsafe.

Check that the service switchover mode on the server is set to automatic. If it is, look at the RSF-1 log for error messages from the service scripts. If necessary, run each of the scripts manually in the correct sequence (bring up the virtual network interface first). If your application is failing to start, ensure that you have licensed it properly and that it is not confused by the virtual interface. Alternatively, the configuration file and scripts might not have been copied to the sibling(s).

D.2.4 Shared disks are not accessible

Check that no other server(s) are using any shared volumes before attempting to access them locally. Failure to do this may result in the loss of data.

Hardware configuration - Check all cables and connectors and ensure that they are correctly seated. Confirm that the devices are powered up.

Software configuration - Check that the underlying disk devices are visible. Verify that any logical volumes are correctly configured and available to your RSF-1 servers. Use a utility like `fsck` to verify the file-systems. Try mounting them manually.

Once you have fixed any problems, unmount and release the shared volumes and restart the service.

D.2.5 The clients can't access the service

Check that you have updated your network naming service with details of the service floating hostnames. Ensure that the virtual interface has been configured correctly on the RSF-1 server. Test access to the server using a command like `ping` or `telnet` from the clients. Ensure that traffic to the service hostnames is being routed correctly by your network.

D.2.6 Admin console can't connect to a Server

Check that the server is running. Check that `rsfmon` is running on the server. Check that you can `ping` the server address from the local host.

Check that access from your machine isn’t denied in the `/opt/HAC/RSF-1/etc/access` file, remembering that you may appear as the local loopback address (127.0.0.1) – see section 7.6.1.

D.2.7 A service failover occurred for no apparent reason

Service failovers only occur when RSF-1 believes a sibling instance to be down. Therefore either all contact was lost with the sibling (unlikely, assuming you have several heartbeats) or the `rsfmon` daemons on the sibling were killed.

This is most likely to happen when carrying out maintenance on RSF-1 without first setting the relevant switchover mode(s) to manual.

D.2.8 One or more servers rebooted but the services are not available

Assuming the switchover modes were in automatic, the most likely explanation is that a recurrent problem occurred during service start-up and the service was eventually aborted on all servers. When this happens, the switchover modes will change to manual, and the service will be marked “broken_*” on at least one server, preventing any further start-up attempts.

Check the `rsfmon` log file for errors from the service scripts. After diagnosing and fixing the problem, use the console to start the service and observe the start-up (see section **Error! Reference source not found.**).

D.2.9 How can I be alerted by RSF-1 when a failure occurs?

RSF-1 can only log failures to the `rsfmon` log and the syslog daemon. However, there are several freely available[14] or commercial utilities that can scan a log file for particular messages and take

[14] *swatch* is one such package.

user-configurable actions upon finding them. These actions can include sending email, paging administrators or taking corrective steps.

Service start-up and shutdown scripts may also be used to send messages, e.g. by email.

D.2.10 Can I run multiple consoles?

Yes. However, be careful when changing the state of RSF-1. Because there is a lag between display updates, changes may not show immediately in other console displays. We recommend that you only use *one* console to change the cluster state, though there is no problem associated with using multiple consoles for monitoring.

D.2.11 Network attack resistance in RSF-1 version 2

There are 2 open ports, one TCP and one UDP.

1. The TCP port is used for status inquiries and control. Connections to this port are checked against /opt/HAC/RSF-1/etc/access, and will be closed immediately if access is denied.

   If access is granted, status information may be retreived, but commands which control the cluster are subject to cryptographic authentication. Unauthenticated control commands are discarded. Unrecognised requests are also discarded.

2. The UDP port is used for passing network heartbeats and for cluster discovery. Connections are accepted from anywhere, and cluster discovery packets are replied to immediately. These should not affect the operation of RSF-1.

   Heartbeat packets are checked for consistency using a CRC, and some sanity checks. Failing packets are therefore discarded (random data, spoofed packets etc) and only ones passing these checks are used to update the state of the cluster.

D.3 Further Help

For additional help, see Appendix G.

Appendix E. RSF-1 Log File Entries

E.1 Introduction

During its initialisation phase and normal running, RSF-1 updates its log file with information relating to the current state of the cluster. At a bare minimum this will consist of the RSF-1 server processes boot log, and then a simple hourly stamp indicating the status of this node in the cluster.

Aside from the messages logged as part of normal running, RSF-1 will also log messages when some exception occurs (and possibly any resulting action taken). The reasons for these occurrences vary; some indicating minor changes, i.e. simply switching modes from manual to automatic (or visa versa) whilst others indicate major system occurrences, such as a fail over of services from one host to another.

This section is intended to document a variety of scenarios played out in log files. There is no intent to document every possible entry that may occur in the log files; some log entries are self explanatory, others specifically for the use of High-Availability.Com staff for diagnostic purposes, whilst others are from expected occurrences (such as mode switching) or make up part of a bigger scenario.

Each entry in the log file is preceded with the process ID (PID) of the RSF-1 process that has issued the entry. Following this is a time stamp to indicate when the entry was made; note that this stamp is based on the current time of the host machine on which the process is running.

E.2 Heartbeat delays

Should a machine become very heavily loaded for an extended period of time, the RSF-1 heartbeat processes may become so starved of resources that they may miss a single heartbeat. RSF-1 is designed to be as resilient as possible to such occurrences; however, for completeness, a series of log entries will be made. These take the form:

```
2754 Jan 12 11:38:21 NOTICE: rsfmon parent experienced delay in main loop

2756 Jan 12 11:38:21 2 net heartbeat poll signal timeout(s):

       parent=2754, group=2754

2755 Jan 12 11:38:21 1 serial heartbeat poll signal timeout(s):

       parent=2754, group=2754

2754 Jan 12 11:45:35 NOTICE: Select returned 1 second(s) late. Clock change?

2754 Jan 12 11:45:35 NOTICE: net heartbeat (2) from OraServ DOWN

2754 Jan 12 11:46:21 NOTICE: net heartbeat (2) from OraServ UP
```

The first message in the list indicates that the main RSF-1 process has experienced an (unexpected) delay during its normal processing. This is related to the fourth message where it was also noticed that the select() system call returned back after a delay. RSF-1 constantly monitors that time that critical sections of its code take to execute. When dealing with heartbeats from a remote machine it is imperative that internal timers are closely monitored and regulated. In this case, over a period of time, RSF-1 has noticed and logged that it has experienced a delay at several points in the code (this is symptomatic of a machine that is starting to be severely overloaded).

Please also note that changing the system time will also cause a logging of the fourth message above – this does not in any way adversely affect RSF-1, however, the fact that it has happened will cause a log entry to be made.

The second and third entries have been made by child processes of the main RSF-1 process. These processes are responsible for the actual sending of heartbeats to other nodes configured in the cluster. Child processes are instructed when to send a heartbeat by the main RSF-1 process; in this way all heartbeats are orchestrated so they are sent in a synchronous manner. Child

processes thus wait on the master process for this signal; however, child processes also run their own timers that give the master process a generous time frame in which to send this heartbeat signal. Should this time expire, the child processes will log this fact. So, returning to lines two and three above, we can see that both the network and serial child processes[15] have experienced such a delay; in this case two network and one serial heartbeat. The lines also give the process ID of the parent they were expecting the signal from, along with the process group of which they are members (the process group is used as an efficient way of distributing signals amongst child processes). The process group number will correspond to the process ID of the master RSF-1 process.

The final two lines in the log show that the network heartbeat from the host `OraServ` has actually been lost and then regained. This is the likely outcome from the system becoming starved of resources. Note however, that although the network heartbeat has suffered as a consequence of starved resources, the same fate has not befallen the serial heartbeat. The explanation behind this is that for the network heartbeat to actually be delivered to RSF-1 it first has to travel up the TCP/IP stack, with the associated additional work load this involves. Combine this with the fact that RSF-1 uses UDP packets (delivery not guaranteed,) then the likely outcome is that the network packets will suffer delay on a heavily loaded machine. On the other hand, serial packets require far less processing (they are delivered as-is,) and as a consequence, even on a heavily loaded machine, tend to be delivered[16].

Finally note that in the last two lines the network heartbeat has the identifier `(2)` following it. This identifier can be directly related to the actual heartbeat in question by referring back to the configuration file. In this example let us assume the following configuration extract:

```
MACHINE NfsHost
  SERIAL OraHost /dev/ttyb
  NET OraHost OraServ
MACHINE OraHost
  SERIAL NfsHost /dev/ttyb
  NET NfsHost OraServ
```

RSF-1 loads heartbeats for each machine in the order network, disk, and finally serial. Numbering is started from 0 and incremented as each is assigned to a heartbeat. So in the above example, the first network heartbeat for `NfsHost` is given number 0, the serial number 1, the network for `OraHost` number 2 and finally the serial number 3. So in the above example the heartbeat in question was the network one from `OraHost`.

[15] Note that for each *type* of heartbeat configured a child heartbeat process is created. Child processes are responsible for all heartbeats of a particular type on each host, so, for example, a single network heartbeat child will be responsible for as many network heartbeats that are configured.

[16] High-Availability.Com always recommends at least more than one single type of heartbeat is configured.

Appendix F. Memory and CPU allocation by RSF-1

This appendix is provided as useful background for system administrators wishing to understand and tune the way RSF-1 interacts with a systems memory and CPU.

There are two areas in which the memory allocation in RSF-1 differs from an ordinary user application. These are:

- As RSF-1 is a mission critical application, and as such tries to ensure that its use of memory in a system does not cause it to be delayed by the virtual memory system. To do this, on startup, some extra heap and stack are allocated and initialized, and the current set of pages are locked in memory. This avoids RSF-1 being paged out of memory, and then delayed later waiting to be paged back in. This is done by all the `rsfmon` processes, but is not done to the applications which RSF-1 controls (though they may lock themselves in memory if appropriate). The memory locking is done through various system calls on the different systems supported by RSF-1, `lockall(MCL_CURRENT | MCL_FUTURE)` on some systems, and `plock(PROCLOCK)` on others. These locks are not released until RSF-1 exits.
- Shared memory is used to communicate between the various rsfmon processes. The size of the shared memory segment is not fixed, but depends upon the number of services which a server may run, as defined in the RSF-1 configuration file. The shared memory segment is identified by the key 6969, which matches the port number used for network heartbeats in early versions of RSF-1.

CPU time can not be reserved as such, but RSF-1 can take some steps to try to ensure that it is scheduled to run without delays. It can elevate its normal timesharing priority (the so called "nice" value), controlled by the configuration file `PRIORITY` parameter. It can also try to run at real time priority, this being controlled by the configuration file `REALTIME` parameter.

Appendix G. Further Support

This chapter lists other sources of help for RSF-1 and its supported platforms, including contact addresses and references to additional documentation.

G.1 Contacting High-Availability.Com

Head Office Address

High-Availability.Com Ltd
Suite B
Pentland House
Village Way
Wilmslow
Cheshire
SK9 2GH
UK

Phone and Fax

Phone: +44 (0)844 736 1434
Support: +44 (0)844 736 1974

support@high-availability.com

http://www.high-availability.com

G.1.1 Support Calls

Before placing a support call, please ensure that you have run `rsfdiag` and sent the output to support@High-Availability.Com for ALL nodes in the cluster and also have the following details ready:

- Company name and contact details (phone, fax and email preferred).
- Any registration or support contract details, such as product serial numbers.
- A clear description of the problem, together with times or dates and any actions you took.

`rsfdiag` will collate much of the environment information for you, producing an output file we can accept via email.

G.2 RSF-1 Support Mailing List

High-Availability.Com maintains a support list for its current maintenance customers. The list is maintained purely for announcements of new software releases, feature additions and support bulletins. High-Availability.Com respects the privacy of its customers; subscribed email addresses are considered confidential and as such will never be divulged to any outside party. To join to the list please contact the list administrator using the electronic mail address:

support-list-administrator@high-availability.com

G.3 References and Additional Support

G.3.1 Solaris

- Solaris AnswerBooks and manuals
- Your local Sun Service representative

- Sun web site, http://www.sun.com/
- SunSolve CD or Online, http://sunsolve.sun.com/

G.3.2 AIX

- AIX online documentation.
- Your local IBM support representative.
- IBM web site, http://www.ibm.com/

G.3.3 Linux

- HOWTO and other documentation, sometimes installed in `/usr/doc`
- Linux Documentation Projection on the web, http://www.linuxdoc.org/
- Linux web site, http://www.linux.org/
- Linux newsgroups and other Internet resources.
- The vendor of your Linux distribution.
- RPM web site, http://www.rpm.org/
- Linux HA project, http://linux-ha.org/

G.3.4 OpenServer

- OpenServer online documentation
- Your local SCO support representative.
- SCO web site, http://www.sco.com/

G.3.5 HPUX

- HPUX online documentation
- Your local HPUX support representative
- HP web site, http://www.hp.com/
- HPUX FAQ site, http://www.faqs.org/faqs/hp/hpux-faq/

Appendix H. Summary of changes in RSF-1 versions

This appendix surmises the changes and enhancements of features incorporated in the various versions of RSF-1. For users familiar with older versions of RSF-1 and associated manual, this chapter provides a guide to the new features and avoids having to reread the complete manual.

H.1 Changes to Version 2.1

H.1.1 FQDN hostnames

Hostnames appearing in the configuration file were limited to short local names (i.e. they could not contain dots). They can now also be long names (i.e. with the domain part attached,) or the numeric IP addresses themselves. See section 7.4.3.11 for details.

H.1.2 Process locking

RSF-1 now locks itself in memory on those systems which support memory locking. This will improve reliability in cases of excessive system paging or swapping.

H.2 Changes in Version 2.4

H.2.1 Real time scheduling

RSF-1 now uses real time scheduling on system that support it. This provides a much more robust clustering environment. See section 7.4.3.5 for details.

H.2.2 Syslog facility codes

The facility code used by RSF-1 to generate syslog messages may now be specified. See section 7.4.3.9 for details.

H.2.3 Service start-up and shutdown timeouts

Service scripts are timed and killed if they run for an unusual amount of time. See section 7.4.3.8 for details.

H.2.4 A machines name need not match its hostname

RSF-1 can be configured with a machine name which is not the same as the hostname of the machine on which it is running. This allows multiple hosts in a cluster to have the same hostname, which is required by a few applications. See section 7.4.3.11 and **Error! Reference source not found.** for details.

H.2.5 Service script start-up and shutdown exit conditions

Services that time out during running by RSF-1 are now assigned exit codes of 2 and 3 depending on the signal causing the script to exit. See section 7.5.3 for details.

H.3 Changes in Version 2.6

H.3.1 Disabling service script timeouts

A timeout value of 0 disables service script timeouts. See section 7.4.3.8 for details.

H.3.2 Single node clusters

RSF-1 now allows clusters consisting of just a single node so that it can simply be used as a management interface. See section 7.4.3.11 for details.

H.3.3 Directory structure

The directory structure & path has been changed. Instead of `/opt/RSIrsf`, `/opt/HAC` and `/opt/HAC/RSF-1` are now used for RSF-1 files. Some RSF-1 utilities have been moved into `/opt/HAC`, as they are also used by other HAC products. The RSF-1 specific `rsf.sh` script has been moved here. Other files out side this tree, including the rsfmon daemon pid & lock file and status dump file have changed names.

H.3.4 RSF release file

Allow the version number reported by rsfmon to be changed by putting a new version string in the file `/opt/HAC/RSF-1/etc/.rsfrelease`. Primarily aimed at OEM resellers, but also allows a certain amount of on-site customization.

H.3.5 New exit codes

The exit codes RSF_SAFE and RSF_UNSAFE have been added. See section 7.5.3 for details.

H.3.6 Rsf_install

Removed and replaced by package post and pre install scripts.

H.3.7 Licensing on Linux can use any Ethernet card

As Linux does not implement a unique host ID, the host ID used for licensing rsfmon on these systems consists of the last 4 bytes of the Ethernet MAC address. In the past, only the first Ethernet card was checked for a matching MAC address, now all configured cards are checked.

H.3.8 Controlling user name logging

The username of the user who initiates a control action for rsfmon is now logged in the rsfmon.log file. (The username is the one from the `rsfpasswd` file, not the UNIX user ID of the logged on user).

H.4 Changes in 2.7

H.4.1 Opteron support

Support added for Solaris on Opteron in 32 and 64 bit mode. Support for Intel IA64.

H.4.2 RSFCLI list extensions

The list subcommand of rsfcli now lists all heartbeats and services in a cluster.

H.4.3 Broadcast listener

Added broadcast listener and responder on RSF-1 request port. The GUI can now broadcast out for available clusters.

H.4.4 Downloadable configuration file

Configuration file can be downloaded over TCP/IP. Future proofing for broker mechanism.

H.4.5 Mount points

Filesystem mount points introduced into configuration file. These mount points are then checked upon service start up and shut down as part of sanity checking. The distributed example configuration file now includes example mount point entries.

H.4.6 Configuration file fix

MACHINE lines in the configuration file that had an optional address portion were being ignored. This is now fixed.

H.4.7 Rsfcli changes

Rsfcli now incorporates the rsfdump, rsfstat and rsfdebug commands.

H.4.8 Linux mount checks

S20ext3 contains more checking for already mounted partitions.

H.4.9 New port number

All packages now use the IANA assigned port number 1195. Please visit www.iana.org for more details.

H.5 Changes in 2.8

H.5.1 Differing CRC's in heartbeats

When RSF-1 detects differing CRC's in heartbeats (and therefore differences in node configuration files) services are blocked.

H.5.2 Check for RSF-1 running

Before RSF-1 logs are rotated, rsfrc checks to ensure RSF-1 is not running first. Previously, this could cause the RSF-1 log file to be renamed and a new empty one created, which would never be written into.

H.5.3 Rsfcli updates

The rsfcli stat command now indicates if service start-ups are enabled based on correct config file CRC's. The stat command also indicates if rsfmon is shutting down.

H.5.4 RSF-1 minimum service start-up timeout

The minimum service start-up time has been changed to 3*POLL_TIME+1 to avoid possible false failovers.