Heartbeat 3.0

The Heartbeat User's Guide

Publican
BOOK PUBLISHING TOOL

Florian Haas

Heartbeat 3.0 The Heartbeat User's Guide

Author Florian Haas `florian.haas@linbit.com`

The definitive reference guide for users of the Heartbeat cluster messaging layer.

Preface **vii**
1. Document Conventions ... vii
1.1. Typographic Conventions ... vii
1.2. Pull-quote Conventions ... viii
1.3. Notes and Warnings ... ix
2. We Need Feedback! ... x

I. Introduction to Heartbeat **1**

1. Fundamentals **3**
1.1. Heartbeat as a Cluster Messaging Layer ... 3
1.2. Components ... 3
1.2.1. Communication module ... 3
1.2.2. Cluster Consensus Membership ... 4
1.2.3. Cluster Plumbing Library ... 4
1.2.4. IPC Library ... 4
1.2.5. Non-blocking logging daemon ... 5

II. Installing and Configuring Heartbeat **7**

2. Building and installing from source **9**
2.1. Building and installing Cluster Glue from source ... 9
2.1.1. Cluster Glue build prerequisites ... 9
2.1.2. Downloading Cluster Glue sources ... 9
2.1.3. Building Cluster Glue ... 10
2.1.4. Building Packages ... 11
2.2. Building and installing Heartbeat from source ... 11
2.2.1. Heartbeat build prerequisites ... 11
2.2.2. Downloading Heartbeat sources ... 11
2.2.3. Building Heartbeat ... 12
2.2.4. Building Packages ... 13

3. Creating an initial Heartbeat configuration **15**
3.1. The **`ha.cf`** file ... 15
3.2. The **`authkeys`** file ... 15
3.3. Propagating the cluster configuration to cluster nodes ... 16

III. Administrative Tasks **17**

4. Upgrading from previous Heartbeat versions **19**
4.1. Upgrading from Heartbeat 2.1 clusters not using the CRM ... 19
4.1.1. Stopping Heartbeat services ... 19
4.1.2. Upgrade software ... 19
4.1.3. Enabling the Heartbeat cluster to use Pacemaker ... 20
4.1.4. Restarting Heartbeat ... 20
4.2. Upgrading from CRM-enabled Heartbeat 2.1 clusters ... 20
4.2.1. Place the cluster in unmanaged mode ... 21
4.2.2. Stop Heartbeat services ... 21
4.2.3. Upgrade software ... 21
4.2.4. Restarting Heartbeat services ... 22
4.2.5. Upgrading the CIB schema ... 22

4.2.6. Returning the cluster to managed mode 22

A. Manual pages **23**

heartbeat 23
ha.cf 24
authkeys 34
cl_status 35
hb_addnode 37
hb_delnode 37

B. Legacy manual pages **39**

apphbd 39
hb_takeover 40
hb_standby 41

C. Resource agent manual pages **43**

ocf_heartbeat_anything 43
ocf_heartbeat_AoEtarget 44
ocf_heartbeat_apache 45
ocf_heartbeat_AudibleAlarm 46
ocf_heartbeat_ClusterMon 47
ocf_heartbeat_CTDB 48
ocf_heartbeat_db2 49
ocf_heartbeat_Delay 50
ocf_heartbeat_drbd 50
ocf_heartbeat_Dummy 51
ocf_heartbeat_eDir88 52
ocf_heartbeat_Evmsd 53
ocf_heartbeat_EvmsSCC 54
ocf_heartbeat_Filesystem 55
ocf_heartbeat_ICP 56
ocf_heartbeat_ids 56
ocf_heartbeat_IPaddr2 57
ocf_heartbeat_IPaddr 59
ocf_heartbeat_IPsrcaddr 60
ocf_heartbeat_IPv6addr 61
ocf_heartbeat_iSCSILogicalUnit 62
ocf_heartbeat_iSCSITarget 63
ocf_heartbeat_iscsi 64
ocf_heartbeat_jboss 65
ocf_heartbeat_LinuxSCSI 66
ocf_heartbeat_LVM 67
ocf_heartbeat_MailTo 68
ocf_heartbeat_ManageRAID 68
ocf_heartbeat_ManageVE 69
ocf_heartbeat_mysql-proxy 70
ocf_heartbeat_mysql 71
ocf_heartbeat_nfsserver 72
ocf_heartbeat_oracle 73
ocf_heartbeat_oralsnr 75
ocf_heartbeat_pgsql 76
ocf_heartbeat_pingd 77
ocf_heartbeat_portblock 78

ocf_heartbeat_postfix 79
ocf_heartbeat_proftpd 80
ocf_heartbeat_Pure-FTPd 81
ocf_heartbeat_Raid1 82
ocf_heartbeat_Route 82
ocf_heartbeat_rsyncd 83
ocf_heartbeat_SAPDatabase 84
ocf_heartbeat_SAPInstance 86
ocf_heartbeat_scsi2reservation 87
ocf_heartbeat_SendArp 88
ocf_heartbeat_ServeRAID 89
ocf_heartbeat_sfex 89
ocf_heartbeat_SphinxSearchDaemon 90
ocf_heartbeat_Squid 91
ocf_heartbeat_Stateful 92
ocf_heartbeat_SysInfo 93
ocf_heartbeat_syslog-ng 94
ocf_heartbeat_tomcat 94
ocf_heartbeat_VIPArip 96
ocf_heartbeat_VirtualDomain 97
ocf_heartbeat_vmware 98
ocf_heartbeat_WAS6 98
ocf_heartbeat_WAS 99
ocf_heartbeat_WinPopup 99
ocf_heartbeat_Xen 100
ocf_heartbeat_Xinetd 101

Index **103**

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F1`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F7`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

[1] https://fedorahosted.org/liberation-fonts/

> Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This avoids difficult-to-follow phrasing such as 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar'.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in `mono-spaced roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `mono-spaced roman` but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }

}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' won't cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

You should over ride this by creating your own local Feedback.xml file.

Part I. Introduction to Heartbeat

Fundamentals

1.1. Heartbeat as a Cluster Messaging Layer

Heartbeat is a daemon that provides cluster infrastructure (communication and membership) services to its clients. This allows clients to know about the presence (or disappearance!) of peer processes on other machines and to easily exchange messages with them.

In order to be useful to users, the Heartbeat daemon needs to be combined with a cluster resource manager (CRM) which has the task of starting and stopping the services (IP addresses, web servers, etc) that cluster will make highly available. The canonical cluster resource manager typically associated with Heartbeat is Pacemaker, a highly scalable and feature-rich implementation that supports both Heartbeat and, alternatively, the Corosync cluster messaging layer.

Note

Up until Heartbeat release 2.1.3, Pacemaker was developed jointly with Heartbeat as part of the Linux-HA umbrella project. After this release, the Pacemaker project was spun off as a separate project and continues as as such, while maintaining full support for Heartbeat cluster messaging.

1.2. Components

The Heartbeat messaging layer comprises several interlocking but distinct components.

1.2.1. Communication module

The Heartbeat communication module provides strongly authenticated,locally-ordered multicast messaging over basically any media, IP-based or not. Heartbeat supports cluster communications over the following network link types:

- unicast UDP over IPv4;
- broadcast UDP over IPv4;
- multicast UDP over IPv4;
- serial link communications.

Warning

Please consider some important caveats with regard to serial link connectivity (see). As a general rule: when in doubt, serial links should be avoided.

Heartbeat can detect node failure reliably in less than a half-second. It will register with the system watchdog timer if configured to do so.

The heartbeat layer has an API which provides the following classes of services:

- intra-cluster communication - sending and receiving packets to cluster nodes

- configuration queries
- connectivity information (who can the current node hear packets from) - both for queries and state change notifications
- basic group membership services

1.2.2. Cluster Consensus Membership

CCM provides strongly connected consensus cluster membership services. It ensures that every node in a computed membership can talk to every other node in this same membership. CCM Implements both an OCF draft membership API, and the SAF AIS membership API. Typically it computes membership in sub-second time.

1.2.3. Cluster Plumbing Library

The Cluster plumbing library is a collection of very useful functions which provide a variety of services used by many of our main components. A few of the major objects provided by this library include:

- compression API (with underlying compression plugins)
- Non-blocking logging API
- memory management oriented to continuously running services
- Hierarchical name-value pair messaging facility promoting portability and version upgrade compatibility (also provides optional message compression facilities)
- Signal unification - allowing signals to appear as mainloop events
- Core dump management utilities - promoting capture of core dumps in a uniform way, and under all circumstances
- timers (like glib mainloop timers - but they work even when the time of day clock jumps)
- child process management - death of children causes invocation of process object, with configurable death-of-child messages
- Triggers (arbitrary events triggered by software)
- Realtime management - setting and unsetting high priorities, and locked into memory attributes of processes.
- 64-bit HZ-granularity time manipulation (longclock_t)
- User id management for security purposes, for processes which need some root privileges.
- Mainloop integration for IPC, plain file descriptors, signals, etc. This means that all these different event sources are managed and dispatched consistently.

1.2.4. IPC Library

All interprocess communication is performed using a very general IPC library which provides non-blocking access to IPC using a flexible queueing strategy, and includes integrated flow control. This

IPC API does not require sockets, but the currently available implementations use UNIX (Local) Domain sockets.

This API also includes built-in authentication and authorization of peer processes, and is portable to most POSIX-like OSes. Although use of Glib mainloop with these APIs is not required, Heartbeat provides simple and convenient integration with mainloop.

1.2.5. Non-blocking logging daemon

`logd` is Heartbeat's logging daemon, capable of logging to a syslog daemon, to files, or both. `logd` never blocks, instead, it discards messages lagging too far behind. Once it is capable of outputting messages again, `logd` prints a lost message count. Queue sizes are controllable both overall, and on a per-application basis.

Part II. Installing and Configuring Heartbeat

Building and installing from source

Building and installing the Heartbeat cluster messaging layer from source amounts to building the following packages:

- `heartbeat` itself;
- the `cluster-glue` package containing the Heartbeat local resource manager (LRM) and STONITH plugins.

Since `heartbeat` has a build dependency on `cluster-glue`, it is necessary to build and install `cluster-glue` *first*, before one proceeds with building and installing `heartbeat`.

2.1. Building and installing Cluster Glue from source

2.1.1. Cluster Glue build prerequisites

Building Cluster Glue requires the presence of the following tools and libraries on the build system:

- A C compiler (typically `gcc`) and associated C development libraries;
- the `flex` scanner generator and the `bison` parser compiler;
- `net-snmp` development headers, to enable SNMP related functionality;
- `OpenIPMI` development headers, to enable IPMI related functionality;
- Python (just the language interpreter, not library headers).

Note

This list applies to the default software configuration. If you configure the source with non-standard options, other dependencies may apply.

2.1.2. Downloading Cluster Glue sources

Several options are available for retrieving the Cluster Glue source code, for building locally on a target system.

2.1.2.1. Downloading a release tarball

Downloading a released version of Heartbeat as a compressed tarball is equivalent to fetching a *tagged* snapshot from the Mercurial source code repository. Release tags follow the format `glue-`*`x.y.z`*, where *`x.y.z`* is the released version of Cluster Glue you wish to download.

If, for example, one wants to download the `1.0.1` release, the correct sequence of commands would be:

```
# wget http://hg.linux-ha.org/glue/archive/glue-1.0.1.tar.bz2

# tar -vxjf glue-1.0.1.tar.bz2
```

2.1.2.2. Downloading the latest Mercurial snapshot

The latest development code is always available in the Mercurial repository as the `tip` revision.

To download a tarball auto-generated from the `tip`, use this sequence of commands:

```
# wget http://hg.linux-ha.org/glue/archive/tip.tar.bz2

# tar -vxjf tip.tar.bz2
```

2.1.2.3. Checking out sources from Mercurial

This is the method you would apply if you have the Mercurial utilities locally installed. Checking out the sources amounts to *cloning* the repository:

```
$ hg clone http://hg.linux-ha.org/glue cluster-glue
requesting all changes
adding changesets
adding manifests
adding file changes
added 12491 changesets with 34830 changes to 2632 files
updating working directory
356 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

2.1.3. Building Cluster Glue

Building Cluster Glue is an automated process making extensive use of GNU Autotools. When building and installing on the same machine, it usually amounts to just the following sequence of commands:

```
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
```

Note

The **`autogen.sh`** script is a convenience wrapper around **`automake`**, **`autoheader`**, **`autoconf`**, and **`libtool`**.

A number of configuration options are supported, and you may tweak some of them to optimize Heartbeat for your system. To retrieve a list of configuration options, you may invoke **`configure`** with the `--help` option. A customized build may thus comprise these steps:

```
$ ./autogen.sh
$ ./configure --help
$ ./configure configuration-options
$ make
$ sudo make install
```

2.1.4. Building Packages

RPM spec files are provided in the Cluster Glue source tree both for SuSE and Red Hat based distributions:

- **`cluster-glue-suse.spec`** should be used for OpenSUSE and SLES installations.
- **`cluster-glue-fedora.spec`** is for Fedora, Red Hat Enterprise Linux, and CentOS.

The Cluster Glue source tree also contains a **`debian/`** subdirectory, which you may use for building a Debian package. Simply invoke **`dpkg-buildpackage`** from the top of the source tree.

2.2. Building and installing Heartbeat from source

2.2.1. Heartbeat build prerequisites

Building Heartbeat requires the presence of the following tools and libraries on the build system:

- A C compiler (typically `gcc`) and associated C development libraries;
- the `flex` scanner generator and the `bison` parser compiler;
- `net-snmp` development headers, to enable SNMP related functionality;
- `OpenIPMI` development headers, to enable IPMI related functionality;
- Python (just the language interpreter, not library headers)
- the `cluster-glue` development headers. See *Section 2.1, “Building and installing Cluster Glue from source”* for details on how to build these from source.

Note

This list applies to the default software configuration. If you configure the source with non-standard options, other dependencies may apply.

2.2.2. Downloading Heartbeat sources

Several options are available for retrieving the Heartbeat source code, for building locally on a target system.

2.2.2.1. Downloading a release tarball

Downloading a released version of Heartbeat as a compressed tarball is equivalent to fetching a *tagged* snapshot from the Mercurial source code repository. Release tags follow the format `STABLE-x.y.z`, where `x.y.z` is the released version of Heartbeat you wish to download.

If, for example, one wants to download the `3.0.1` release, the correct sequence of commands would be:

```
# wget http://hg.linux-ha.org/dev/archive/STABLE-3.0.1.tar.bz2
# tar -vxjf STABLE-3.0.1.tar.bz2
```

2.2.2.2. Downloading the latest Mercurial snapshot

The latest development code is always available in the Mercurial repository as the `tip` revision.

To download a tarball auto-generated from the `tip`, use this sequence of commands:

```
# wget http://hg.linux-ha.org/dev/archive/tip.tar.bz2

# tar -vxjf tip.tar.bz2
```

2.2.2.3. Checking out sources from Mercurial

This is the method you would apply if you have the Mercurial utilities locally installed. Checking out the sources amounts to *cloning* the repository:

```
$ hg clone http://hg.linux-ha.org/dev heartbeat-dev
requesting all changes
adding changesets
adding manifests
adding file changes
added 12491 changesets with 34830 changes to 2632 files
updating working directory
356 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

2.2.3. Building Heartbeat

Building Heartbeat is an automated process making extensive use of GNU Autotools. When building and installing on the same machine, it usually amounts to just the following sequence of commands:

```
$ ./bootstrap
$ ./ConfigureMe configure
$ make
$ sudo make install
```

Note

The **`bootstrap`** script is a convenience wrapper around **`automake`**, **`autoheader`**, **`autoconf`**, and **`libtool`**.

`ConfigureMe` is a convenience wrapper for the **`autoconf`**-generated **`configure`** script.

A number of configuration options are supported, and you may tweak some of them to optimize Heartbeat for your system. To retrieve a list of configuration options, you may invoke **`configure`** with the `--help` option. A customized build may thus comprise these steps:

```
$ ./bootstrap
$ ./ConfigureMe configure --help
$ ./ConfigureMe configure configuration-options
$ make
$ sudo make install
```

2.2.4. Building Packages

RPM spec files are provided in the Heartbeat source tree both for SuSE and Red Hat based distributions:

- **`heartbeat-suse.spec`** should be used for OpenSUSE and SLES installations.
- **`heartbeat-fedora.spec`** is for Fedora, Red Hat Enterprise Linux, and CentOS.

The Heartbeat source tree also contains a **`debian/`** subdirectory, which you may use for building a Debian package. Simply invoke **`dpkg-buildpackage`** from the top of the source tree.

Creating an initial Heartbeat configuration

For any Heartbeat cluster, the following configuration files must be available:

- **`/etc/ha.d/ha.cf`** — the global cluster configuration file.
- **`/etc/ha.d/authkeys`** — a file containing keys for mutual node authentication.

3.1. The `ha.cf` file

The following example is a small and simple **`ha.cf`** file:

```
autojoin none
mcast bond0 239.0.0.43 694 1 0
bcast eth2
warntime 5
deadtime 15
initdead 60
keepalive 2
node alice
node bob
crm respawn
```

Setting `autojoin` to `none` disables cluster node auto-discovery and requires that cluster nodes be listed explicitly, using the `node` options. This speeds up cluster start-up in clusters with a fixed small number of nodes.

This example assumes that `bond0` is the cluster's interface to the shared network, and that `eth2` is the interface dedicated for DRBD replication between both nodes. Thus, `bond0` can be used for Multicast heartbeat, whereas on `eth2` broadcast is acceptable as `eth2` is not a shared network.

The next options configure node failure detection. They set the time after which Heartbeat issues a warning that a no longer available peer node *may* be dead (`warntime`), the time after which Heartbeat considers a node *confirmed* dead (`deadtime`), and the maximum time it waits for other nodes to check in at cluster startup (`initdead`). `keepalive` sets the interval at which Heartbeat keep-alive packets are sent. All these options are given in seconds.

The `node` option identifies cluster members. The option values listed here must match the exact host names of cluster nodes as given by **`uname -n`**.

`crm respawn` enables the Pacemaker cluster manager, and ensures that Pacemaker is automatically restarted in case of a failure.

3.2. The `authkeys` file

`/etc/ha.d/authkeys` contains pre-shared secrets used for mutual cluster node authentication. It should only be readable by `root` and follows this format:

```
auth num
num algorithm secret
```

num is a simple key index, starting with 1. Usually, you will only have one key in your **authkeys** file.

algorithm is the signature algorithm being used. You may use either `md5` or `sha1`; the use of `crc` (a simple cyclic redundancy check, not secure) is not recommended.

secret is the actual authentication key.

You may create an **authkeys** file, using a generated secret, with the following shell hack:

```
( echo -ne "auth 1\n1 sha1 "; \
  dd if=/dev/urandom bs=512 count=1 | openssl md5 ) \
  > /etc/ha.d/authkeys
chmod 0600 /etc/ha.d/authkeys
```

3.3. Propagating the cluster configuration to cluster nodes

In order to propagate the contents of the **ha.cf** and **authkeys** configuration files, you may use the **ha_propagate** command, which you would invoke using either

```
/usr/lib/heartbeat/ha_propagate
```

or

```
/usr/lib64/heartbeat/ha_propagate
```

This utility will copy the configuration files over to any `node` listed in **/etc/ha.d/ha.cf** using **scp**. It will afterwards also connect to the nodes using **ssh** and issue `chkconfig heartbeat on` in order to enable Heartbeat services on system startup.

Part III. Administrative Tasks

Upgrading from previous Heartbeat versions

This chapter describes upgrading to Heartbeat 3.0 from previous releases.

4.1. Upgrading from Heartbeat 2.1 clusters not using the CRM

For Heartbeat 2.1 clusters not using the CRM (i.e., clusters configured with the **`haresources`** file, an upgrade to 3.0 involves converting the current configuration to one that is suitable for Pacemaker.

Note

This upgrade procedure does incur application down time. However, when the upgrade is properly planned, tested, and executed, this down time amounts to minutes, possibly even seconds (depending on configuration).

4.1.1. Stopping Heartbeat services

You should commence the upgrade procedure on your current *standby* node, that is, the cluster node currently not running any resources. If your cluster is using an active-active configuration (both nodes running resources), select one and issue the following command to transfer all resources to the peer node:

```
# hb_standby
```

Then, *and on that node only*, stop Heartbeat services:

```
# /etc/init.d/heartbeat stop
```

4.1.2. Upgrade software

While upgrading, it is important to recall that the monolithic Heartbeat 2.1 tree has been split up into modular parts. Thus you will replace Heartbeat with three individual pieces of software: Cluster Glue, Pacemaker, and Heartbeat 3 which comprises just the cluster messaging layer.

- **Upgrading from source**

  In the unpacked archive *that you installed Heartbeat 2.1 from*, run **`make uninstall`**. Then, install Cluster Glue and Heartbeat.

- **Upgrading using locally built packages**

  When installing packages manually, uninstall the `heartbeat` package first. Then install `cluster-glue`, the version 3 `heartbeat` package, `resource-agents`, and `pacemaker`.

- **Upgrading using a package repository**

  When upgrading using an APT, YUM, or Zypper repository, you should just be able to run the install command for `heartbeat` version 3 and `pacemaker`, and the dependencies will be resolved automatically.

Do not restart Heartbeat services at this point.

4.1.3. Enabling the Heartbeat cluster to use Pacemaker

The cluster messaging layer must now be instructed to start Pacemaker on cluster start-up. To do so, add

```
crm respawn
```

to your **`ha.cf`** configuration file.

Important

At this point, you should also check your **`ha.cf`** file against the *ha.cf(5)* manual page, and remove any deprecated options.

When your **`ha.cf`** modifications are complete, copy the file to the peer node.

4.1.4. Restarting Heartbeat

Your cluster is now ready to be restarted in Pacemaker-enabled mode. To do so:

1. Run **`/etc/init.d/heartbeat heartbeat stop`** on your still-active node. This will shutdown your cluster resources.
2. Run **`/etc/init.d/heartbeat heartbeat start`** on your standby node (the one where you created your CIB). This will start the local Heartbeat instance and Pacemaker, and wait for other cluster nodes to check in.
3. Run **`/etc/init.d/heartbeat heartbeat start`** on your the other node. This will start the local Heartbeat instance and Pacemaker, fetch the CIB automatically, and start applications.

4.2. Upgrading from CRM-enabled Heartbeat 2.1 clusters

This section outlines the steps necessary to upgrade a Heartbeat 2.1 cluster with the built-in CRM enabled, to Heartbeat 3.0 with Pacemaker.

Note

When properly planned and executed, the upgrade procedure can be completed in a manner of minutes, with no application down time.

It is strongly recommended to read and understand the steps outlined below *before* attempting a production cluster upgrade.

All commands stated below must be run as `root`. *Do not* perform the individual steps in parallel on all of your cluster nodes. Instead, complete the procedure on each node before continuing with the next.

4.2.1. Place the cluster in unmanaged mode

With this step, the cluster temporarily relinquishes control of its resources. This means that the cluster no longer monitors its nodes or resources for the duration of the upgrade, and will not rectify and application or node failures during this time. Currently running resources, however, will continue to run.

```
# crm_attribute -t crm_config -n is_managed_default -v false
```

Note

In most configurations, individual resources do not set the `is-managed` attribute individually, and hence the cluster-wide attribute `is-managed-default` applies to all of them.

If in your specific configuration you do have resources that have this attribute set, you should remove it to make sure the default applies:

```
# crm_resource -t primitive -r resource-id -p is-managed -v false
```

4.2.2. Stop Heartbeat services

You may now stop Heartbeat with **`/etc/init.d/heartbeat stop`** or the preferred command to stop a system service on your distribution (**`service heartbeat stop`**, **`rcheartbeat stop`** etc.).

4.2.3. Upgrade software

While upgrading, it is important to recall that the monolithic Heartbeat 2.1 tree has been split up into modular parts. Thus you will replace Heartbeat with three individual pieces of software: Cluster Glue, Pacemaker, and Heartbeat 3 which comprises just the cluster messaging layer.

- **Upgrading from source**

  In the unpacked archive *that you installed Heartbeat 2.1 from*, run **`make uninstall`**. Then, install Cluster Glue and Heartbeat.

- **Upgrading using locally built packages**

  When installing packages manually, uninstall the `heartbeat` package first. Then install `cluster-glue`, the version 3 `heartbeat` package, `resource-agents`, and `pacemaker`.

- **Upgrading using a package repository**

  When upgrading using an APT, YUM, or Zypper repository, you should just be able to run the install command for `heartbeat` version 3 and `pacemaker`, and the dependencies will be resolved automatically.

Only if this is the last node to upgrade in your cluster, you should now proceed to *Section 4.2.4, “Restarting Heartbeat services”*. Otherwise, you should move to the next node and proceed as outlined in *Section 4.2.2, “Stop Heartbeat services”* and *Section 4.2.3, “Upgrade software”*.

4.2.4. Restarting Heartbeat services

Note

You should only proceed with this step if *all* cluster nodes have been upgraded to new software. If you still have remaining cluster nodes that have not been upgraded, you should continue on those nodes as described in *Section 4.2.2, “Stop Heartbeat services”* and *Section 4.2.3, “Upgrade software”*.

After all cluster nodes have been updated, you may restart Heartbeat on all nodes with **`/etc/init.d/heartbeat start`**. At this time,

- the cluster is still in unmanaged mode (meaning it does not start, stop, or monitor any resources), and
- the cluster is still using the pre-upgrade CIB schema.

4.2.5. Upgrading the CIB schema

Although an upgraded cluster can theoretically operate on the pre-upgrade CIB schema indefinitely, it is strongly recommended to upgrade the CIB to the current schema. To do so, run the following command *after* cluster communications between all nodes have been re-established:

```
#
```

4.2.6. Returning the cluster to managed mode

Once the cluster software and the CIB schema have been upgraded, it is recommended to return the cluster into managed mode:

```
# crm_attribute -t crm_config -n is-managed-default -v true
```

Note

While the pre-upgrade command manipulates the `is_managed_default` cluster property, the post-upgrade command applies to `is-managed-default` (using hyphens "-" in place of underscores "_"). This is due to a change in the CIB schema between Heartbeat 2.1's built-in CRM and Pacemaker.

Alternatively, you may also change the `is-managed-default` cluster property using the **`crm`** shell, which is only available in Pacemaker:

```
# crm configure property is-managed-default true
```

Appendix A. Manual pages

Name

heartbeat — Heartbeat subsystem for High-Availability Linux

Description

heartbeat is a basic heartbeat subsystem for Linux-HA. It will run scripts at initialisation, and when machines go up or down. This version will also perform IP address takeover using gratuitous ARPs. It works correctly for a 2-node configuration, and is extensible to larger configurations.

It implements the following kinds of heartbeats:

- UDP/IP broadcast;
- UDP/IP multicast;
- UDP/IP unicast;
- Bidirectional Serial Rings ("raw" serial ports) — this type is deprecated and should no longer be used;
- special "ping" heartbeats for routers, etc. — this type has been superseded by functionality in pacemaker and should no longer be used.

Comprehensive documentation on heartbeat is available in the Heartbeat User's Guide. If this documentation is not installed on your system, it can be found at http://linux-ha.org/.

Options

The following options are supported by heartbeat:

`-d`
: Increment debugging level. Higher levels are more verbose.

`-r`
: Reload heartbeat. This option is functionally identical to sending a running heartbeat process a HUP signal. If the configuration has not changed, then this option is essentially a no-op. If ha.cf(5) or authkeys(5) has changed, then heartbeat will re-read these files and update its configuration.

  This option may not be used together with `-R`.

`-k`
: Kill (stop) heartbeat.

`-s`
: Report heartbeat status.

`-R`
: Heartbeat restart exec flag (internal use only). May not be used with `-r`.

`-C`
: Heartbeat current resource state for restart (internal use only). Only valid with `-R`.

`-V`
: Print out heartbeat version.

Note that most of these options are used for supporting the heartbeat init script, which provides the conventional start, stop, status and restart options (among others). It is recommended to use this rather than invoking the heartbeat command directly.

See also

ha.cf(5), authkeys(5)

Name

ha.cf — Configuration file for the Heartbeat cluster messaging layer

Description

`/etc/ha.d/ha.cf` is read by heartbeat(8) upon node start-up. It lists the communication facilities enabled between nodes, enables or disables certain features, and optionally lists the cluster nodes by host name.

This file can safely be made world readable, but should be writable only by root.

Global directives

Some directives in **`ha.cf`** are global in nature. The order of these global options is important in configuring the ha.cf file, since each directive is interpreted as it is encountered in ha.cf.

These directives are `use_logd` and `udpport`. It is recommended that these be placed first in the ha.cf file when they are entered.

Other directives in this category are `baud`, `logfacility`, `logfile`, and `debugfile`, but those directives are deprecated and should no longer be used.

Supported directives

The following directives are supported in **`ha.cf`** (listed here in alphabetical order):

`apiauth`
: This directive specifies what users and/or groups are allowed to connect to a specific API group name. The syntax is simple:

```
apiauth apigroupname [uid=uid1,uid2 ...] [gid=gid1,gid2 ...]
```

You can specify either a uid list, or a gid list, or both. However you must specify either a uid list or a gid list. If you include both a uid list and a gid list, then a process is authorized to connect to that API group if if it is either in the uid-list or it is in the gid-list.

The API group name default has special meaning. If it is specified, it will be used for authorizing clients without any API group name, and all client groups not identified by any other apiauth directive.

Unless you specify otherwise in the ha.cf file, certain services will be provided default authorizations as follows:

Service	Default apiauth
ipfail	uid=hacluster
ccm	gid=haclient
ping	gid=haclient
cl_status	gid=haclient
lha-snmpagent	uid=root
crm	uid=hacluster

Table A.1. Default service authorizations

`autojoin`

The autojoin directive enables nodes to join automatically just by communicating with the cluster, hence not requiring node directives in the ha.cf file. Since our communication is normally strongly authenticated, only nodes which know the cluster key can join (automatically or otherwise).

The values you can give for the autojoin directive have the following meanings:

- none: disables automatic joining.
- other: allows nodes other than ourself who are not listed in ha.cf to join automatically. In other words, our node has to be listed in ha.cf, but other nodes do not.
- any: allows any node to join automatically without being listed in ha.cf, even the current node.

Note that the set of nodes currently considered part of the cluster is kept in the **`hostcache`** file. With autojoin enabled, the node directive is no longer authoritative - the hostcache file is.

`bcast`

The bcast directive is used to configure which interfaces Heartbeat sends UDP broadcast traffic on. More than one interface can be specified on the line. The udpport directive is used to configure which port is used for these broadcast communications if the udpport directive is specified before the bcast directive, otherwise the default port will be used. A couple of sample bcast lines are shown below.

```
bcast eth0 eth1  # on Linux systems
bcast le0        # for Solaris systems
```

Note

Broadcast links are not supported in Pacemaker clusters on BSD systems.

`compression`

The compression directive sets which compression method will be used when a message is big and compression is needed.

It could be either zlib or bz2, depending on whether you have the corresponding library in the system. You can check /usr/lib/heartbeat/plugins/HBcompress to see what compression module is available.

If this directive is not set, there will be no compression.

`compression_threshold`
: The compression_threshold directive sets the threshold to compress a message, e.g. if the threshold is 1, then any message with size greater than 1 KB will be compressed. The default is 2 (KB). This directive only makes sense if you have set the compression directive.

`conn_logd_time`
: The conn_logd_time directive specifies the time Heartbeat will reconnect to the logging daemon if the connection between Heartbeat and the logging daemon is broken. The conn_logd_time is specified according to the Heartbeat time syntax, for example:

```
conn_logd_time 60 #60 seconds
```

The default is 60 seconds.

Note

Heartbeat will not automatically reconnect to the logging daemon. It only tries to reconnect when it needs to log a message and conn_logd_time have passed since the last attempt to connect.

`coredumps`
: The coredumps directive tells Heartbeat to do things to enable making core dumps - should it need to dump core.

The allowed values are `true` and `false`.

`crm`
: Enables the Pacemaker cluster manager. For historical reasons, the default for this option is `off`; however, it should always be set to `respawn`.

When set to `respawn`, the directive automatically implies:

```
apiauth stonithd        uid=root
apiauth crmd            uid=hacluster
apiauth cib             uid=hacluster

respawn hacluster       ccm
respawn hacluster       cib
respawn root            stonithd
respawn root            lrmd
respawn hacluster       crmd
```

`deadtime`
: The deadtime directive is used to specify how quickly Heartbeat should decide that a node in a cluster is dead. Setting this value too low will cause the system to falsely declare itself dead. Setting it too high will delay takeover after the failure of a node in the cluster.

`debug`
: The debug directive is used to set the level of debugging in effect in the system. Production systems should have their debug level set to zero (i.e., turned off). This is the default. Legal values of the debug option are between 0-255. The most useful values are between 0 (off) and 3. Setting the debug level greater than 1 can have an adverse effect on the size of your log files, and on the system's ability to send heartbeats at rapid rates, thus affecting the cluster reliability.

  The debug level of the system can also be specified on the command line using the -d option. Additionally, the debug level of the system can be dynamically changed by sending the heartbeat process SIGUSR1 and SIGUSR2 signals. SIGUSR1 raises the debug level, and SIGUSR2 lowers it.

`hbgenmethod` time|file
: The hbgenmethod directive specifies how Heartbeat should compute its current generation number for communications. This is a specialized and obscure directive, used mainly in firewalls which have no local disk, and other devices which do not have a method of storing data persistently across reboots. It defaults to storing the Heartbeat generations in a file. Generation numbers are used by Heartbeat for replay attack protection.

  Warning

  If one specifies the time method, there are certain possible cases where troubles can arise. If a machine restarts Heartbeat and its local time of day clock is less than or equal to than the value of the time of day clock when Heartbeat last started, then that node will be unable to join the cluster.

`initdead`
: The initdead parameter is used to set the time that it takes to declare a cluster node dead when Heartbeat is first started. This parameter generally needs to be set to a higher value, because experience suggests that it sometimes takes operating systems many seconds for their communication systems before they operate correctly. initdead is specified according to the Heartbeat time syntax. A sample initdead value is shown below:

```
initdead 30
```

  In some switched network environments, switches engage in a spanning tree algorithm whenever a NIC connects to a port. This can take a long time to complete, and it is only necessary if the NIC being connected is another switch. If this is the case, you may be able to configure certain NICs as not being switches and shrink the connection delay significantly. If not, you'll need to raise initdead to make this problem go away.

  If this is set too low, you'll see one node declare the other as dead.

`keepalive`
: The keepalive directive sets the interval between heartbeat packets. It is specified according to the Heartbeat time syntax.

`logfacility`

The logfacility is used to tell Heartbeat which syslog logging facility it should use for logging its messages.

The possible values for logfacility vary by operating system, but some of the most common ones are {auth, authpriv, daemon, syslog, user, local0, local1, local2, local3, local4, local5, local6, local7}.

A sample logfacility directive is shown below:

```
logfacility local7
```

If you want to disable logging to syslog:

```
logfacility none
```

`mcast`

The mcast directive is used to configure a multicast communication path. The syntax of an mcast directive is:

```
mcast dev mcast-group udp-port ttl 0
```

- dev - IP device to send/rcv heartbeats on
- mcast-group - multicast group to join (class D multicast address 224.0.0.0 - 239.255.255.255). For most Heartbeat uses, the first byte should be 239.
- port - UDP port to sendto/rcvfrom (set this to the same value as udpport)
- ttl - the ttl value for outbound heartbeats. This affects how far the multicast packet will propagate. (0-255). Set to 1 for the current subnet. Must be greater than zero.

A sample mcast directive is shown below:

```
mcast eth0 239.0.0.1 694 1 0
```

`msgfmt` classic|netstring

The msgfmt directive specifies the format Heartbeat uses in wire.

- classic - Heartbeat will convert a message into a string and transmit in wire. Binary values are converted with a base64 library.
- netstring - Binary messages will be transmitted directly. This is more efficient since it avoids conversion between string and binary values.

When in doubt, leave the default (classic).

`node`

The node directive tells what machines are in the cluster. The syntax of the node directive is simple:

```
node nodename1 nodename2 ...
```

Node names in the directive must match the "uname -n" of that machine.

You can declare multiple node names in one directive. You can also use the directive multiple times. Normally every node in the cluster must be listed in the ha.cf file, including the current node, unless the autojoin directive is enabled.

The node directive is not completely authoritative with regard to nodes heartbeat will communicate with. If a node has ever been added in the past, it will tend to remain in the hostcache file more until it's manually removed.

`realtime` on|off

The realtime directive specifies whether or not Heartbeat should try and take advantage of the operating system's realtime scheduling features. When enabled, Heartbeat will lock itself into memory, and raise its priority to a realtime priority (as set by the rtprio directive). This feature is mainly used for debugging various kinds of loops which might otherwise cripple the system and impair debugging them.

The default is on.

`rtprio`

The rtprio directive is used to specify the priority at which Heartbeat runs. It does not need to be specified unless other realtime priority programs are also running on the system. The minimum and maximum values for this field can be determined from the sched_get_priority_min(SCHED_FIFO) and sched_get_priority_max(SCHED_FIFO) calls respectively. The default value for rtprio is halfway between the minimum and maximum values.

A sample rtprio directive is shown below:

```
rtprio 5
```

`ucast`

The ucast directive configures Heartbeat to communicate over a UDP unicast communications link. The udpport directive is used to configure which port is used for these unicast communications if the udpport directive is specified before the ucast directive, otherwise the default port will be used.

The general syntax of a ucast directive is:

```
ucast dev peer-ip-address
```

Where dev is the device to use when talking to the peer, and peer-ip-address is the IP address we will send packets to.

A sample ucast directive is shown below:

```
ucast eth0 10.10.10.133
```

This directive will cause us to send packets to 10.10.10.133 over interface eth0.

Note that ucast directives which go to the local machine are effectively ignored. This allows the ha.cf directives on all machines to be identical.

`udpport`

The udpport directive specifies which port Heartbeat will use for its UDP intra-cluster communication. There are two common reasons for overriding this value: there are multiple bcast clusters on the same subnet, or this port is already in use in accordance with some locally-established policy.

The default value for this parameter is the the port ha-cluster in /etc/services (if present), or 694 if port ha-cluster is not in /etc/services. 694 is the IANA registered port number for Heartbeat (a.k.a. ha-cluster).

A sample udpport directive is shown below.

```
udpport 694
```

You have to configure udpport (in ha.cf) *before* you configure ucast or bcast, if not heartbeat will use the default port (694).

Note

Due to a specification error in the syntax of the mcast directive, this directive does not apply to mcast communications.

`use_logd` on|off

The use_logd directive specifies whether Heartbeat logs its messages through logging daemon or not.

If the logging daemon is used, all log messages will be sent through IPC to the logging daemon, which then writes them into log files. In case the logging daemon dies (for whatever reason), a warning message will be logged and all messages will be written to log files directly.

If the logging daemon is used, logfile/debugfile/logfacility in this file are not meaningful any longer. You should check the config file for logging daemon (the default is /etc/logd.cf).

If use_logd is not used, all log messages will be written to log files directly.

The logging daemon is started/stopped in heartbeat script.

Setting use_logd to "on" is recommended.

`uuidfrom`

In the normal case, heartbeat generates a UUID for each node in the system as a way of uniquely identifying a node - even if it should change nodenames. This UUID is typically stored in the file /var/lib/heartbeat/hb_uuid.

For certain kinds of installations (those booting from CDs or other read-only media), it is impossible for heartbeat to save a generated to disk as it normally does. In these cases, one can use the uuidfrom directive to instruct heartbeat to use the nodename as though it were a UUID, by specifying uuidfrom nodename.

All possible legal uuidfrom directives are shown below.

```
uuidfrom file
uuidfrom nodename
```

`warntime`

The warntime directive is used to specify how quickly Heartbeat should issue a "late heartbeat" warning.

The warntime value is specified according to the HeartbeatTimeSyntax. A sample warntime specification is shown below.

```
warntime 10    # 10 seconds
```

The warntime directive is important for tuning deadtime

Deprecated directives

The following directives are interpreted by the configuration file parser for historical reasons, but should be considered deprecated and should no longer be used.

`auto_failback`

In legacy Heartbeat clusters, the auto_failback option would determine whether a resource would automatically fail back to its "primary" node, or remain on whatever node is serving it until that node fails, or an administrator intervenes. The possible values for auto_failback were:

- on - enable automatic failbacks
- off - disable automatic failback
- legacy - enable automatic failbacks in systems where all nodes in the cluster do not yet support the auto_failback option.

This option has been replaced the configurable failback policies in Pacemaker, and should no longer be used.

`baud`

The baud directive is used to set the speed for serial communications. Any of the following speeds can be specified, provided they are supported by your operating system: 9600, 19200, 38400, 57600, 115200, 230400, 460800. The default speed is 19200.

This option is obsolete as serial links should not be used in Pacemaker clusters.

`deadping`

The deadping directive is used to specify how quickly Heartbeat should decide that a ping node in a cluster is dead. Setting this value too low will cause the system to falsely declare the ping node dead. Setting it too high will delay detection of communication failure.

This feature has been replaced by the more flexible `pingd` resource agent in Pacemaker, and should no longer be used.

`debugfile`

The debugfile directive specifies the file Heartbeat will write debug messages to.

This directive is ignored when `use_logd` is specified. Enabling `use_logd` is the recommended approach.

`hbaping`

Hbaping directives are given to declare fiber channel devices as ping nodes.

This directive was never fully supported in Heartbeat (requiring manual modifications to the code base) and should not be used.

`hopfudge`

The hopfudge directive controls how many nodes a packet can be forwarded through before it is thrown away in the worst case. However, the hopfudge value is added to the number of nodes in the system. It defaults to 1.

This option applies to serial links only, which are deprecated.

`logfile`

The logfile directive configures a log file. All non-debug messages from Heartbeat will go into this file.

This directive is ignored when `use_logd` is specified. Enabling `use_logd` is the recommended approach.

`ping`

Ping directives are given to declare ping nodes to Heartbeat. The syntax of the ping directive is simple:

```
ping ip-address ...
```

Each IP address listed in a ping directive is considered to be independent. That is, connectivity to each node is considered to be equally important.

In order to declare that a group of nodes are equally qualified for a particular function, and that the presence of any of them indicates successful communication, use the ping_group directive.

This feature has been replaced by the more flexible `pingd` resource agent in Pacemaker, and should no longer be used.

`ping_group`

Ping group directives are given to declare a group ping node to Heartbeat. syntax of the ping_group directive is as follows:

```
ping_group group-name ip-address ...
```

Each IP address listed in a ping_group directive is considered to be related, and connectivity to any one node is considered to be connectivity to the group.

A ping group is considered by Heartbeat to be a single cluster node (group-name). The ability to communicate with any of the group members means that the group-name member is reachable. This is useful when (for example) two different routers may be used to contact the internet, depending on which is up, or when finding an appropriate reliable single ping node is difficult.

This feature has been replaced by the more flexible `pingd` resource agent in Pacemaker, and should no longer be used.

`respawn`

The respawn directive is used to specify a program to run and monitor while it runs. If this program exits with anything other than exit code 100, it will be automatically restarted. The first parameter is the user id to run the program under, and the second parameter is the program to run. Subsequent parameters will be given to the program as arguments.

This functionality was primarily designed for the legacy ipfail program, which has been replaced by the more flexible `pingd` resource agent in Pacemaker. Thus, this directive should no longer be used, except when it is implicitly generated by `crm yes`.

`serial`

The serial directive tells Heartbeat to use the specified serial port(s) for its communication. The parameters to the serial directive are the names of tty devices suitable for opening without waiting for carrier first. On Linux, those ports are typically named /dev/ttySX.

A few sample serial directives are shown below:

```
serial /dev/ttyS0 /dev/ttyS1     # Linux
serial /dev/cuaa0                # FreeBSD
serial /dev/cua/a                # Solaris
```

The baud directive is used to configure the baud rate for the port(s) if the baud directive is specified before the serial directive, otherwise the default baud rate will be used.

Using this option is *strongly* discouraged in Pacemaker clusters, as its CIB updates can easily hit practical message size limits for serial links, with undefined results.

`stonith`

The stonith directive is used to configure Heartbeat's legacy STONITH configuration. It assumes you're going to put in a STONITH configuration file on each machine in the cluster to configure the (single) STONITH device that this node will use to reset the other node in the cluster.

This functionality has been replaced by STONITH agents in Pacemaker.

`stonith_host`

The stonith_host directive is used to configure Heartbeat's (release 1 only), STONITH configuration. With this directive, you put all the STONITH configuration information for the devices in your cluster in the ha.cf file, rather than in a separate file.

This functionality has been replaced by STONITH agents in Pacemaker.

`traditional_compression` on|off

This directive enables traditional compression. It is highly recommended that this be set to off (the default); otherwise heartbeat performance can be significantly negatively impacted.

`watchdog`

The watchdog directive configures Heartbeat to use a watchdog device. In some circumstances, a watchdog device can be used in place of a STONITH device. In any case, it is a reasonable thing to configure if you don't have a STONITH device, or if you wish, in addition to your STONITH device.

It is the purpose of a watchdog device to shut the machine down if Heartbeat does not hear its own heartbeats as often as it thinks it should. This keeps things like scheduler bugs from becoming split-brain configurations.

The general syntax of a watchdog directive is:

```
watchdog watchdog-device-name
```

A sample watchdog directive is shown below:

```
watchdog /dev/watchdog
```

The most common watchdog device currently used with general Linux systems is the softdog device. The softdog device is a software-based watchdog device and is usually referred to as /dev/watchdog - although like most UNIX devices, this is a convention not a rule.

This functionality has been replaced by cluster self-monitoring and STONITH resource agents in Pacemaker. This directive should no longer be used.

Required directives

The following directives must always be present in **`ha.cf`**:

- At least one communication topology directive (`bcast`, `mcast`, or `ucast`);
- Either one or more `node` directives, or `autojoin any`.

Example

Below is an example **`ha.cf`** for a 2-node Pacemaker cluster with redundant network communication paths:

```
use_logd on
mcast eth0 239.0.0.42 694 1 0
bcast eth1
node alice
node bob
crm respawn
```

Name

authkeys — Authentication file for the Heartbeat cluster messaging layer

Description

`/etc/ha.d/authkeys` is read by heartbeat(8). It enables Heartbeat to securely authenticate cluster nodes.

This file must not be readable or writable by any users other than root.

File format

Two lines are required in the authkeys file:

1. A line which says which key to use in signing *outgoing* packets
2. One or more lines defining how *incoming* packets might be being signed.

The file must follow the following format:

```
auth num
num method secret
num method secret
num method secret
...
```

`num` is a numerical identifier, between 1 and 15 inclusive. It must be unique within the file.

`method` is one of the available authentication signature methods (see below for supported methods).

`secret` is an alphanumerical shared secret used to identify cluster nodes to each other.

`auth` *`num`* selects the currently active authentication method and secret.

Supported signature methods

The following signature methods are supported in **`authkeys`** (listed here in alphabetical order):

`md5`
: MD5 hash method. This method requires a shared secret.

`sha1`
: SHA-1 hash method. This method requires a shared secret.

`crc`
: Cyclic Redundancy Check hash method. This method does not require a shared secret and is insecure; it's use is strongly discouraged.

An absolutely up-to-date list of authentication methods supported may be retrieved by running **`ls /usr/lib/heartbeat/plugins/HBauth/*.so`**.

Name

cl_status — Check status of the High-Availability Linux (Linux-HA) subsystem

Synopsis

`cl_status` **`sub-command`** `options` *`parameters`*

Description

`cl_status` is used to check the status of the High-Availability Linux subsystem.

Supported sub-commands

`hbstatus`
: Indicate if heartbeat is running on the local system.

`listnodes`
: List the nodes in the cluster.

`nodetype` ping|normal
: List the nodes of the given type.

> **Note**
> Ping nodes are obsolete in Pacemaker cluster, having been replaced with the pingd resource agent.

`listhblinks` *`node`*
: List the network interfaces used as heartbeat links. *`node`* should be specified as listed in the ha.cf(5) file for the cluster.

`hblinkstatus` *`node`* *`link`*
: Show the status of a heartbeat link. *`node`* should be specified as listed in the ha.cf(5) file for the cluster. *`link`* should be as per the output of the listhblinks subcommand.

`clientstatus` *`node`* *`client`* [*`timeout`*]
: Show the status of heartbeat clients. *`node`* and *`client`* should be specified as listed in the ha.cf(5) file for the cluster. Timeout is in milliseconds, the default is 100ms.

`rscstatus`
: Show the status of cluster resources. Status will be one of: local, foreign, all or none.

> **Note**
> This option is deprecated, it is obsolete in Pacemaker clusters.

`parameter` `-p` *`parameter`*
: Retrieve the value of cluster parameters. The parameters may be one of the following: apiauth, auto_failback, baud, debug, debugfile, deadping, deadtime, hbversion, hopfudge, initdead, keepalive, logfacility, logfile, msgfmt, nice_failback, node, normalpoll, stonith, udpport, warntime, watchdog.

> **Note**
> Some of these options are deprecated; see ha.cf(5)

Options

The following options are supported by heartbeat:

`-m`
: Make the output more human readable. The default output should be easier for scripts to parse. Available with all commands.

`-p`
: List only 'ping' nodes. Available with listnodes sub-command.

Note

Ping nodes are obsolete in Pacemaker cluster, having been replaced with the pingd resource agent.

`-n`
: List only 'normal' nodes. Available with listnodes sub-command.

See also

heartbeat(8), ha.cf(5), authkeys(5)

Name

hb_addnode — sends a message to a Heartbeat cluster to add new nodes

Synopsis

`hb_addnode` *node* [*node*] [*node*]

Description

/usr/share/heartbeat/hb_addnode adds a new node, or multiple nodes, to the cluster configuration. If there is any node in the arguments that is already a cluster member, the command fails and no nodes are added.

Options

The following options are supported:

`--help`
: Issues a brief usage message.

See also

hb_delnode(1), heartbeat(8), cl_status(1)

Name

hb_delnode — sends a message to a Heartbeat cluster to remove one or more nodes

Synopsis

`hb_delnode` *node* [*node*] [*node*]

Description

/usr/share/heartbeat/hb_delnode removes a node, or multiple nodes, from the cluster configuration. If there is any node in the arguments that is currently not a cluster member, the command fails and no nodes are removed.

Options

The following options are supported:

`--help`
: Issues a brief usage message.

See also

hb_addnode(1), heartbeat(8), cl_status(1)

Appendix B. Legacy manual pages

The manual pages in this appendix are of no practical relevance to Heartbeat/Pacemaker clusters. They are listed here for reference purposes for legacy systems only.

Name

apphbd — Application Heartbeat Monitor for High-Availability Linux

Synopsis

apphbd [`-srkdh`] [`-c` *`file`*]

Description

> **Warning**
>
> **apphbd** is deprecated; its use is strongly discouraged. The functionality provided by **apphbd** has been replaced by resource-level monitoring in Pacemaker.

`/usr/lib/heartbeat/apphbd` is a basic application heartbeat monitor daemon for Linux-HA. A group of Application Heartbeat APIs are defined for this heartbeat monitoring service. Applications may register with the daemon in order to be monitored. If an application fails to send a heartbeat within the specified interval, the daemon will log an event.

apphbd may use a watchdog timer to monitor itself.

apphbd extends its functionality by using plugins. A plugin, `recmgr` notifies the recovery manager daemon if certain events occur (e.g. an application fails to heartbeat).

The Recovery Manager daemon (**`/usr/lib/heartbeat/recoverymgrd`**) receives notification from the `recmgr` plugin, then it tries to execute recovery scripts as configured. See the **`recoverymgrd`** default configuration file, **`recoverymgrd.conf`** for details.

`recoverymgrd` registers itself with **apphbd** as a client application. **apphbd** should be started first with the `recmgr` plugin loaded. Then, `recoverymgrd` should be configured and started

Options

The following options are supported:

`-s`
: Show the status of apphbd, running or stopped.

`-k`
: Stop (kill) the daemon.

`-r`
: Restart the daemon. **apphbd** will reload its configuration file when restarted.

`-d` *`level`*
: Set the debug level.

`-h`
: Show a brief usage message.

`-c` *`file`*
: Set an alternate configuration file. The default configuration file is **`./apphbd.cf`**. For details on the file format and supported options, refer to the example **`apphbd.cf`** file found in the documentation directory.

Files

- **`/var/run/apphbd.pid`** – default PID file
- **`apphbd.cf`** – Default configuration file for **`apphbd`**. **`apphbd`** searches the file in its working directory.
- **`recoverymgrd.conf`** – default configuration file for recoverymgrd. recoverymgrd searches the file in its working directory. An alternative configuration file may be specified on the command line.
- **`/usr/lib/heartbeat/plugins/AppHBNotification`** – directory containing plugins for **`apphbd`**.

See also

heartbeat(8)

Name

hb_takeover — issues a failover request to the cluster manager

Synopsis

`hb_takeover` [all|foreign|local|failback]

Description

Warning

This command is deprecated. It is only suitable for legacy Heartbeat clusters without Pacemaker enabled. In Pacemaker-enabled clusters, the crm(8) shell supports switching individual nodes into standby mode, and replaces **`hb_takeover`**.

`/usr/share/heartbeat/hb_takeover` issues a request to the cluster to move resources to the node where it is invoked, from the other node. Issuing **`hb_takeover`** on the current node is equivalent to performing **`hb_standby`** on the other node.

Caveats

`hb_takeover` is only usable in R1-style configurations (i.e., those configured using the **`haresources`** file).

See also

hb_standby(1), heartbeat(8), cl_status(1)

Name

hb_standby — issues a failover request to the cluster manager

Synopsis

`hb_standby` [all|foreign|local|failback]

Description

Warning

This command is deprecated. It is only suitable for legacy Heartbeat clusters without Pacemaker enabled. In Pacemaker-enabled clusters, the crm(8) shell supports switching individual nodes into standby mode, and replaces **`hb_standby`**.

`/usr/share/heartbeat/hb_standby` issues a request to the cluster to move resources from the node where it is invoked, to the other node (if it is currently available). The meaning of the options is relative. This manual assumes the following configuration to be present in **`/etc/ha.d/haresources`**:

```
alice drbddisk::r0 Filesystem::/dev/drbd0::/local/groups::ext3 10.0.0.1 smb
bob drbddisk::r1 Filesystem::/dev/drbd1::/local/ldap::ext3 10.0.0.2 ldap
```

Options

The following options are supported:

local

Migrates any resources that the local node *is* the preferred node for.

When invoked on `alice`, Samba would be shut down, the IP address 10.0.0.1 would be released, **`/local/groups`** would be unmounted, **`/dev/drbd0`** would be placed into the secondary role and `bob` would take all these services over.

When run on `bob`, OpenLDAP would shut down, 10.0.0.2 would be released, **`/local/ldap`** would be unmounted, **`/dev/drbd1`** would be placed into the Secondary role and `alice` would take over all these services.

foreign|failback

Migrates any resources that the local node *is not* the preferred node for.

When run on `alice`, OpenLDAP would shut down, 10.0.0.2 would be released, **`/local/ldap`** would be unmounted, **`/dev/drbd1`** would be placed into the Secondary role and `bob` would take over all these services.

When invoked on `bob`, Samba would be shut down, the IP address 10.0.0.1 would be released, **`/local/groups`** would be unmounted, **`/dev/drbd0`** would be placed into the secondary role and `alice` would take all these services over.

all
: Migrates all resources to the other node.

Invoking **`hb_standby`** without any options is identical to **`hb_standby`** all.

Caveats

`hb_standby` is only usable in R1-style configurations (i.e., those configured using the **`haresources`** file).

See also

hb_takeover(1), heartbeat(8), cl_status(1)

Appendix C. Resource agent manual pages

Name

ocf_heartbeat_anything — Manages an arbitrary service

Synopsis

`anything` [start | stop | monitor | meta-data | validate-all]

Description

This is a generic OCF RA to manage almost anything.

Supported Parameters

`binfile`
: The full name of the binary to be executed. This is expected to keep running with the same pid and not just do something and exit. (optional, string, no default)

`cmdline_options`
: Command line options to pass to the binary (optional, string, no default)

`pidfile`
: File to read/write the PID from/to. (optional, string, default `/var/run//anything_default.pid`)

`logfile`
: File to write STDOUT to (optional, string, no default)

`errlogfile`
: File to write STDERR to (optional, string, no default)

`user`
: User to run the command as (optional, string, default `root`)

`monitor_hook`
: Command to run in monitor operation (optional, string, no default)

`stop_timeout`
: In the stop operation: Seconds to wait for kill -TERM to succeed before sending kill -SIGKILL. Defaults to 2/3 of the stop operation timeout. (optional, string, no default)

Example

The following is an example configuration for a anything resource using the crm(8) shell:

```
primitive example_anything ocf:heartbeat:anything \
  params \
    binfile=string \
  op monitor depth="0" timeout="20" interval="10"
```

See also

http://www.linux-ha.org/wiki/anything_(resource_agent)

Name

ocf_heartbeat_AoEtarget — Manages ATA-over-Ethernet (AoE) target exports

Synopsis

`AoEtarget` [start | stop | monitor | reload | meta-data | validate-all]

Description

This resource agent manages an ATA-over-Ethernet (AoE) target using vblade. It exports any block device, or file, as an AoE target using the specified Ethernet device, shelf, and slot number.

Supported Parameters

`device`
: The local block device (or file) to export as an AoE target. (optional, string, no default)

`nic`
: The local Ethernet interface to use for exporting this AoE target. (optional, string, default `eth0`)

`shelf`
: The AoE shelf number to use when exporting this target. (optional, integer, no default)

`slot`
: The AoE slot number to use when exporting this target. (optional, integer, no default)

`pid`
: The file to record the daemon pid to. (optional, string, default `/var/run/heartbeat/rsctmp/AoEtarget-default.pid`)

`binary`
: Location of the vblade binary. (optional, string, default `/usr/sbin/vblade`)

Example

The following is an example configuration for a AoEtarget resource using the crm(8) shell:

```
primitive example_AoEtarget ocf:heartbeat:AoEtarget \
  params \
    device=string \
    nic="eth0" \
    slot=integer \
  op monitor timeout="15" interval="10" depth="0"
```

See also

http://www.linux-ha.org/wiki/AoEtarget_(resource_agent)

Name

ocf_heartbeat_apache — Manages an Apache web server instance

Synopsis

apache [start | stop | status | monitor | meta-data | validate-all]

Description

This is the resource agent for the Apache web server. Thie resource agent operates both version 1.x and version 2.x Apache servers. The start operation ends with a loop in which monitor is repeatedly called to make sure that the server started and that it is operational. Hence, if the monitor operation does not succeed within the start operation timeout, the apache resource will end with an error status. The monitor operation by default loads the server status page which depends on the mod_status module and the corresponding configuration file (usually /etc/apache2/mod_status.conf). Make sure that the server status page works and that the access is allowed *only* from localhost (address 127.0.0.1). See the statusurl and testregex attributes for more details. See also http://httpd.apache.org/

Supported Parameters

`configfile`
: The full pathname of the Apache configuration file. This file is parsed to provide defaults for various other resource agent parameters. (optional, string, default `/etc/apache2/httpd.conf`)

`httpd`
: The full pathname of the httpd binary (optional). (optional, string, default `/usr/sbin/httpd`)

`port`
: A port number that we can probe for status information using the statusurl. This will default to the port number found in the configuration file, or 80, if none can be found in the configuration file. (optional, integer, no default)

`statusurl`
: The URL to monitor (the apache server status page by default). If left unspecified, it will be inferred from the apache configuration file. If you set this, make sure that it succeeds *only* from the localhost (127.0.0.1). Otherwise, it may happen that the cluster complains about the resource being active on multiple nodes. (optional, string, no default)

`testregex`
: Regular expression to match in the output of statusurl. Case insensitive. (optional, string, default `exists, but impossible to show in a human readable format (try grep testregex)`)

`client`
: Client to use to query to Apache. If not specified, the RA will try to find one on the system. Currently, wget and curl are supported. For example, you can set this paramter to "curl" if you prefer that to wget. (optional, string, no default)

`testurl`
: URL to test. If it does not start with "http", then it's considered to be relative to the Listen address. (optional, string, no default)

`testregex10`
: Regular expression to match in the output of testurl. Case insensitive. (optional, string, no default)

`testconffile`
: A file which contains test configuration. Could be useful if you have to check more than one web application or in case sensitive info should be passed as arguments (passwords). Furthermore, using a config file is the only way to specify certain parameters. Please see README.webapps for examples and file description. (optional, string, no default)

`testname`
: Name of the test within the test configuration file. (optional, string, no default)

`options`
: Extra options to apply when starting apache. See man httpd(8). (optional, string, no default)

`envfiles`
: Files (one or more) which contain extra environment variables. If you want to prevent script from reading the default file, set this parameter to empty string. (optional, string, default `/etc/apache2/envvars`)

Example

The following is an example configuration for a apache resource using the crm(8) shell:

```
primitive example_apache ocf:heartbeat:apache \
  params \

  op monitor depth="0" timeout="20" interval="10"
```

See also

http://www.linux-ha.org/wiki/apache_(resource_agent)

Name

ocf_heartbeat_AudibleAlarm — Emits audible beeps at a configurable interval

Synopsis

`AudibleAlarm` [start | stop | restart | status | monitor | meta-data | validate-all]

Description

Resource script for AudibleAlarm. It sets an audible alarm running by beeping at a set interval.

Supported Parameters

`nodelist`
: The node list that should never sound the alarm. (optional, string, no default)

Example

The following is an example configuration for a AudibleAlarm resource using the crm(8) shell:

```
primitive example_AudibleAlarm ocf:heartbeat:AudibleAlarm \
  params \

  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/AudibleAlarm_(resource_agent)

Name

ocf_heartbeat_ClusterMon — Runs crm_mon in the background, recording the cluster status to an HTML file

Synopsis

`ClusterMon` [start | stop | monitor | meta-data | validate-all]

Description

This is a ClusterMon Resource Agent. It outputs current cluster status to the html.

Supported Parameters

`user`
: The user we want to run crm_mon as (optional, string, default `root`)

`update`
: How frequently should we update the cluster status (optional, integer, default `15`)

`extra_options`
: Additional options to pass to crm_mon. Eg. -n -r (optional, string, no default)

`pidfile`
: PID file location to ensure only one instance is running (optional, string, default `/tmp/ClusterMon_default.pid`)

`htmlfile`
: Location to write HTML output to. (optional, string, default `/tmp/ClusterMon_default.html`)

Example

The following is an example configuration for a ClusterMon resource using the crm(8) shell:

```
primitive example_ClusterMon ocf:heartbeat:ClusterMon \
  params \

  op monitor depth="0" timeout="20" interval="10"
```

See also

http://www.linux-ha.org/wiki/ClusterMon_(resource_agent)

Name

ocf_heartbeat_CTDB — CTDB Resource Agent

Synopsis

`CTDB` [start | stop | monitor | meta-data | validate-all]

Description

This resource agent manages CTDB, allowing one to use Clustered Samba in a Linux-HA/Pacemaker cluster. You need a shared filesystem (e.g. OCFS2) on which CTDB lock and Samba state will be stored. Configure shares in smb.conf on all nodes, and create /etc/ctdb/nodes containing a list of private IP addresses of each node in the cluster. Configure this RA as a clone, and it will take care of the rest. For more information see http://linux-ha.org/wiki/CTDB_(resource_agent)

Supported Parameters

`ctdb_recovery_lock`

The location of a shared lock file, common across all nodes. This must be on shared storage, e.g.: /shared-fs/samba/ctdb.lock (optional, string, no default)

`smb_private_dir`

The directory for smbd to use for storing such files as smbpasswd and secrets.tdb. This must be on shared storage, e.g.: /shared-fs/samba/private (optional, string, no default)

`ctdb_config_dir`

The directory containing various CTDB configuration files. The "nodes" and "notify.sh" scripts are expected to be in this directory, as is the "events.d" subdirectory. (optional, string, default `/etc/ctdb`)

`ctdb_binary`

Full path to the CTDB binary. (optional, string, default `/usr/bin/ctdb`)

`ctdbd_binary`

Full path to the CTDB cluster daemon binary. (optional, string, default `/usr/sbin/ctdbd`)

`ctdb_socket`

Full path to the domain socket that ctdbd will create, used for local clients to attach and communicate with the ctdb daemon. (optional, string, default `/var/lib/ctdb/ctdb.socket`)

`ctdb_dbdir`
: The directory to put the local CTDB database files in. Persistent database files will be put in ctdb_dbdir/persistent. (optional, string, default `/var/lib/ctdb`)

`ctdb_logfile`
: Full path to log file. To log to syslog instead, use the value "syslog". (optional, string, default `/var/log/ctdb/log.ctdb`)

`ctdb_debuglevel`
: What debug level to run at (0-10). Higher means more verbose. (optional, integer, default 2)

`smb_conf`
: Path to default samba config file. (optional, string, default `/etc/samba/smb.conf`)

Example

The following is an example configuration for a CTDB resource using the crm(8) shell:

```
primitive example_CTDB ocf:heartbeat:CTDB \
  params \
    ctdb_recovery_lock=string \
    smb_private_dir=string \
  op monitor timeout="20" interval="10" depth="0"
```

See also

http://www.linux-ha.org/wiki/CTDB_(resource_agent)

Name

ocf_heartbeat_db2 — Manages an IBM DB2 Universal Database instance

Synopsis

db2 [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for db2. It manages a DB2 Universal Database instance as an HA resource.

Supported Parameters

`instance`
: The instance of database. (optional, string, no default)

`admin`
: The admin user of the instance. (optional, string, no default)

Example

The following is an example configuration for a db2 resource using the crm(8) shell:

```
primitive example_db2 ocf:heartbeat:db2 \
  params \
    instance=string \
  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/db2_(resource_agent)

Name

ocf_heartbeat_Delay — Waits for a defined timespan

Synopsis

`Delay` [start | stop | status | monitor | meta-data | validate-all]

Description

This script is a test resource for introducing delay.

Supported Parameters

`startdelay`
: How long in seconds to delay on start operation. (optional, integer, default `30`)

`stopdelay`
: How long in seconds to delay on stop operation. Defaults to "startdelay" if unspecified. (optional, integer, default `30`)

`mondelay`
: How long in seconds to delay on monitor operation. Defaults to "startdelay" if unspecified. (optional, integer, default `30`)

Example

The following is an example configuration for a Delay resource using the crm(8) shell:

```
primitive example_Delay ocf:heartbeat:Delay \
  params \

  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/Delay_(resource_agent)

Name

ocf_heartbeat_drbd — Manages a DRBD resource (deprecated)

Synopsis

drbd [start | promote | demote | notify | stop | monitor | monitor | meta-data | validate-all]

Description

Deprecation warning: This agent is deprecated and may be removed from a future release. See the ocf:linbit:drbd resource agent for a supported alternative. -- This resource agent manages a Distributed Replicated Block Device (DRBD) object as a master/slave resource. DRBD is a mechanism for replicating storage; please see the documentation for setup details.

Supported Parameters

`drbd_resource`
: The name of the drbd resource from the drbd.conf file. (optional, string, default `drbd0`)

`drbdconf`
: Full path to the drbd.conf file. (optional, string, default `/etc/drbd.conf`)

`clone_overrides_hostname`
: Whether or not to override the hostname with the clone number. This can be used to create floating peer configurations; drbd will be told to use node_<cloneno> as the hostname instead of the real uname, which can then be used in drbd.conf. (optional, boolean, default no)

`ignore_deprecation`
: If set to true, suppresses the deprecation warning for this agent. (optional, boolean, default `false`)

Example

The following is an example configuration for a drbd resource using the crm(8) shell:

```
primitive example_drbd ocf:heartbeat:drbd \
  params \
    drbd_resource="drbd0" \
  op monitor depth="0" timeout="20" interval="20" role="Slave"
  op monitor depth="0" timeout="20" interval="10" role="Master"
```

See also

http://www.linux-ha.org/wiki/drbd_(resource_agent)

Name

ocf_heartbeat_Dummy — Example stateless resource agent

Synopsis

Dummy [start | stop | monitor | reload | migrate_to | migrate_from | meta-data | validate-all]

Description

This is a Dummy Resource Agent. It does absolutely nothing except keep track of whether its running or not. Its purpose in life is for testing and to serve as a template for RA writers.

Supported Parameters

`state`
: Location to store the resource state in. (optional, string, default `/var/run/heartbeat/rsctmp/Dummy-{OCF_RESOURCE_INSTANCE}.state`)

Example

The following is an example configuration for a Dummy resource using the crm(8) shell:

```
primitive example_Dummy ocf:heartbeat:Dummy \
  params \

  meta allow-migrate="true" \
  op monitor timeout="20" interval="10" depth="0"
```

See also

http://www.linux-ha.org/wiki/Dummy_(resource_agent)

Name

ocf_heartbeat_eDir88 — Manages a Novell eDirectory directory server

Synopsis

`eDir88` [start | stop | monitor | meta-data | validate-all]

Description

Resource script for managing an eDirectory instance. Manages a single instance of eDirectory as an HA resource. The "multiple instances" feature or eDirectory has been added in version 8.8. This script will not work for any version of eDirectory prior to 8.8. This RA can be used to load multiple eDirectory instances on the same host. It is very strongly recommended to put eDir configuration files (as per the eDir_config_file parameter) on local storage on each node. This is necessary for this RA to be able to handle situations where the shared storage has become unavailable. If the eDir configuration file is not available, this RA will fail, and heartbeat will be unable to manage the resource. Side effects include STONITH actions, unmanageable resources, etc... Setting a high action timeout value is _very_ _strongly_ recommended. eDir with IDM can take in excess of 10 minutes to start. If heartbeat times out before eDir has had a chance to start properly, mayhem _WILL ENSUE_. The LDAP module seems to be one of the very last to start. So this script will take even longer to start on installations with IDM and LDAP if the monitoring of IDM and/or LDAP is enabled, as the start command will wait for IDM and LDAP to be available.

Supported Parameters

`eDir_config_file`
: Path to configuration file for eDirectory instance. (optional, string, default `/etc/opt/novell/eDirectory/conf/nds.conf`)

`eDir_monitor_ldap`
: Should we monitor if LDAP is running for the eDirectory instance? (optional, boolean, default `0`)

`eDir_monitor_idm`
: Should we monitor if IDM is running for the eDirectory instance? (optional, boolean, default `0`)

`eDir_jvm_initial_heap`
: Value for the DHOST_INITIAL_HEAP java environment variable. If unset, java defaults will be used. (optional, integer, no default)

`eDir_jvm_max_heap`
: Value for the DHOST_MAX_HEAP java environment variable. If unset, java defaults will be used. (optional, integer, no default)

`eDir_jvm_options`
: Value for the DHOST_OPTIONS java environment variable. If unset, original values will be used. (optional, string, no default)

Example

The following is an example configuration for a eDir88 resource using the crm(8) shell:

```
primitive example_eDir88 ocf:heartbeat:eDir88 \
  params \

  op monitor timeout="60" interval="30"
```

See also

http://www.linux-ha.org/wiki/eDir88_(resource_agent)

Name

ocf_heartbeat_Evmsd — Controls clustered EVMS volume management (deprecated)

Synopsis

Evmsd [start | stop | monitor | meta-data]

Description

Deprecation warning: EVMS is no longer actively maintained and should not be used. This agent is deprecated and may be removed from a future release. -- This is a Evmsd Resource Agent.

Supported Parameters

`ignore_deprecation`
: If set to true, suppresses the deprecation warning for this agent. (optional, boolean, default `false`)

Example

The following is an example configuration for a Evmsd resource using the crm(8) shell:

```
primitive example_Evmsd ocf:heartbeat:Evmsd \
  params \

  op monitor timeout="20" interval="10" depth="0"
```

See also

http://www.linux-ha.org/wiki/Evmsd_(resource_agent)

Name

ocf_heartbeat_EvmsSCC — Manages EVMS Shared Cluster Containers (SCCs) (deprecated)

Synopsis

EvmsSCC [start | stop | notify | status | monitor | meta-data]

Description

Deprecation warning: EVMS is no longer actively maintained and should not be used. This agent is deprecated and may be removed from a future release. -- Resource script for EVMS shared cluster container. It runs evms_activate on one node in the cluster.

Supported Parameters

`ignore_deprecation`
: If set to true, suppresses the deprecation warning for this agent. (optional, boolean, default `false`)

Example

The following is an example configuration for a EvmsSCC resource using the crm(8) shell:

```
primitive example_EvmsSCC ocf:heartbeat:EvmsSCC \
  params \

  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/EvmsSCC_(resource_agent)

Name

ocf_heartbeat_Filesystem — Manages filesystem mounts

Synopsis

`Filesystem` [start | stop | notify | monitor | validate-all | meta-data]

Description

Resource script for Filesystem. It manages a Filesystem on a shared storage medium. The standard monitor operation of depth 0 (also known as probe) checks if the filesystem is mounted. If you want deeper tests, set OCF_CHECK_LEVEL to one of the following values: 10: read first 16 blocks of the device (raw read) This doesn't exercise the filesystem at all, but the device on which the filesystem lives. This is noop for non-block devices such as NFS, SMBFS, or bind mounts. 20: test if a status file can be written and read The status file must be writable by root. This is not always the case with an NFS mount, as NFS exports usually have the "root_squash" option set. In such a setup, you must either use read-only monitoring (depth=10), export with "no_root_squash" on your NFS server, or grant world write permissions on the directory where the status file is to be placed.

Supported Parameters

`device`

The name of block device for the filesystem, or -U, -L options for mount, or NFS mount specification. (optional, string, no default)

`directory`

The mount point for the filesystem. (optional, string, no default)

`fstype`

The optional type of filesystem to be mounted. (optional, string, no default)

`options`

Any extra options to be given as -o options to mount. For bind mounts, add "bind" here and set fstype to "none". We will do the right thing for options such as "bind,ro". (optional, string, no default)

`statusfile_prefix`

The prefix to be used for a status file for resource monitoring with depth 20. If you don't specify this parameter, all status files will be created in a separate directory. (optional, string, default `.Filesystem_status/`)

Example

The following is an example configuration for a Filesystem resource using the crm(8) shell:

```
primitive example_Filesystem ocf:heartbeat:Filesystem \
  params \
    device=string \
    directory=string \
    fstype=string \
  op monitor depth="0" timeout="40" interval="20"
```

See also

http://www.linux-ha.org/wiki/Filesystem_(resource_agent)

Name

ocf_heartbeat_ICP — Manages an ICP Vortex clustered host drive

Synopsis

`ICP` [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for ICP. It Manages an ICP Vortex clustered host drive as an HA resource.

Supported Parameters

`driveid`
: The ICP cluster drive ID. (optional, string, no default)

`device`
: The device name. (optional, string, no default)

Example

The following is an example configuration for a ICP resource using the crm(8) shell:

```
primitive example_ICP ocf:heartbeat:ICP \
  params \
    driveid=string \
    device=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/ICP_(resource_agent)

Name

ocf_heartbeat_ids — Manages an Informix Dynamic Server (IDS) instance

Synopsis

`ids` [start | stop | status | monitor | validate-all | meta-data | methods | usage]

Description

OCF resource agent to manage an IBM Informix Dynamic Server (IDS) instance as an High-Availability resource.

Supported Parameters

`informixdir`
: The value the environment variable INFORMIXDIR has after a typical installation of IDS. Or in other words: the path (without trailing '/') where IDS was installed to. If this parameter is unspecified the script will try to get the value from the shell environment. (optional, string, no default)

`informixserver`
: The value the environment variable INFORMIXSERVER has after a typical installation of IDS. Or in other words: the name of the IDS server instance to manage. If this parameter is unspecified the script will try to get the value from the shell environment. (optional, string, no default)

`onconfig`
: The value the environment variable ONCONFIG has after a typical installation of IDS. Or in other words: the name of the configuration file for the IDS instance specified in INFORMIXSERVER. The specified configuration file will be searched at '/etc/'. If this parameter is unspecified the script will try to get the value from the shell environment. (optional, string, no default)

`dbname`
: This parameter defines which database to use in order to monitor the IDS instance. If this parameter is unspecified the script will use the 'sysmaster' database as a default. (optional, string, default `sysmaster`)

`sqltestquery`
: SQL test query to run on the database specified by the parameter 'dbname' in order to monitor the IDS instance and determine if it's functional or not. If this parameter is unspecified the script will use 'SELECT COUNT(*) FROM systables;' as a default. (optional, string, default `SELECT COUNT(*) FROM systables;`)

Example

The following is an example configuration for a ids resource using the crm(8) shell:

```
primitive example_ids ocf:heartbeat:ids \
  params \

  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/ids_(resource_agent)

Name

ocf_heartbeat_IPaddr2 — Manages virtual IPv4 addresses (Linux specific version)

Synopsis

`IPaddr2` [start | stop | status | monitor | meta-data | validate-all]

Description

This Linux-specific resource manages IP alias IP addresses. It can add an IP alias, or remove one. In addition, it can implement Cluster Alias IP functionality if invoked as a clone resource.

Supported Parameters

`ip`
: The IPv4 address to be configured in dotted quad notation, for example "192.168.1.1". (optional, string, no default)

`nic`
: The base network interface on which the IP address will be brought online. If left empty, the script will try and determine this from the routing table. Do NOT specify an alias interface in the form eth0:1 or anything here; rather, specify the base interface only. (optional, string, default `eth0`)

`cidr_netmask`
: The netmask for the interface in CIDR format (e.g., 24 and not 255.255.255.0) If unspecified, the script will also try to determine this from the routing table. (optional, string, no default)

`broadcast`
: Broadcast address associated with the IP. If left empty, the script will determine this from the netmask. (optional, string, no default)

`iflabel`
: You can specify an additional label for your IP address here. This label is appended to your interface name. If a label is specified in nic name, this parameter has no effect. (optional, string, no default)

`lvs_support`
: Enable support for LVS Direct Routing configurations. In case a IP address is stopped, only move it to the loopback device to allow the local node to continue to service requests, but no longer advertise it on the network. (optional, boolean, default `false`)

`mac`
: Set the interface MAC address explicitly. Currently only used in case of the Cluster IP Alias. Leave empty to chose automatically. (optional, string, no default)

`clusterip_hash`
: Specify the hashing algorithm used for the Cluster IP functionality. (optional, string, default `sourceip-sourceport`)

`unique_clone_address`
: If true, add the clone ID to the supplied value of ip to create a unique address to manage (optional, boolean, default `false`)

`arp_interval`
: Specify the interval between unsolicited ARP packets in milliseconds. (optional, integer, default `200`)

`arp_count`
: Number of unsolicited ARP packets to send. (optional, integer, default 5)

`arp_bg`
: Whether or not to send the arp packets in the background. (optional, string, default `yes`)

`arp_mac`
: MAC address to send the ARP packets too. You really shouldn't be touching this. (optional, string, default `ffffffffffff`)

Example

The following is an example configuration for a IPaddr2 resource using the crm(8) shell:

```
primitive example_IPaddr2 ocf:heartbeat:IPaddr2 \
  params \
    ip=string \
  op monitor depth="0" timeout="20s" interval="10s"
```

See also

http://www.linux-ha.org/wiki/IPaddr2_(resource_agent)

Name

ocf_heartbeat_IPaddr — Manages virtual IPv4 addresses (portable version)

Synopsis

`IPaddr` [start | stop | monitor | validate-all | meta-data]

Description

This script manages IP alias IP addresses It can add an IP alias, or remove one.

Supported Parameters

`ip`
: The IPv4 address to be configured in dotted quad notation, for example "192.168.1.1". (optional, string, no default)

`nic`
: The base network interface on which the IP address will be brought online. If left empty, the script will try and determine this from the routing table. Do NOT specify an alias interface in the form eth0:1 or anything here; rather, specify the base interface only. (optional, string, default `eth0`)

`cidr_netmask`
: The netmask for the interface in CIDR format. (ie, 24), or in dotted quad notation 255.255.255.0). If unspecified, the script will also try to determine this from the routing table. (optional, string, no default)

`broadcast`
: Broadcast address associated with the IP. If left empty, the script will determine this from the netmask. (optional, string, no default)

`iflabel`
: You can specify an additional label for your IP address here. (optional, string, no default)

`lvs_support`
: Enable support for LVS Direct Routing configurations. In case a IP address is stopped, only move it to the loopback device to allow the local node to continue to service requests, but no longer advertise it on the network. (optional, boolean, default `false`)

`local_stop_script`
: Script called when the IP is released (optional, string, no default)

`local_start_script`
: Script called when the IP is added (optional, string, no default)

`ARP_INTERVAL_MS`
: milliseconds between ARPs (optional, integer, default `500`)

`ARP_REPEAT`
: How many gratuitous ARPs to send out when bringing up a new address (optional, integer, default `10`)

`ARP_BACKGROUND`
: run in background (no longer any reason to do this) (optional, boolean, default `yes`)

`ARP_NETMASK`
: netmask for ARP - in nonstandard hexadecimal format. (optional, string, default `ffffffffffff`)

Example

The following is an example configuration for a IPaddr resource using the crm(8) shell:

```
primitive example_IPaddr ocf:heartbeat:IPaddr \
  params \
    ip=string \
  op monitor depth="0" timeout="20s" interval="5s"
```

See also

http://www.linux-ha.org/wiki/IPaddr_(resource_agent)

Name

ocf_heartbeat_IPsrcaddr — Manages the preferred source address for outgoing IP packets

Synopsis

`IPsrcaddr` [start | stop | stop | monitor | validate-all | meta-data]

Description

Resource script for IPsrcaddr. It manages the preferred source address modification.

Supported Parameters

`ipaddress`
: The IP address. (optional, string, no default)

Example

The following is an example configuration for a IPsrcaddr resource using the crm(8) shell:

```
primitive example_IPsrcaddr ocf:heartbeat:IPsrcaddr \
  params \
    ipaddress=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/IPsrcaddr_(resource_agent)

Name

ocf_heartbeat_IPv6addr — Manages IPv6 aliases

Synopsis

`IPv6addr` [start | stop | status | monitor | validate-all | meta-data]

Description

This script manages IPv6 alias IPv6 addresses,It can add an IP6 alias, or remove one.

Supported Parameters

`ipv6addr`
: The IPv6 address this RA will manage (optional, string, no default)

`cidr_netmask`
: The netmask for the interface in CIDR format. (ie, 24). The value of this parameter overwrites the value of _prefix_ of ipv6addr parameter. (optional, string, no default)

`nic`
: The base network interface on which the IPv6 address will be brought online. (optional, string, no default)

Example

The following is an example configuration for a IPv6addr resource using the crm(8) shell:

```
primitive example_IPv6addr ocf:heartbeat:IPv6addr \
  params \
    ipv6addr=string \
  op monitor timeout="15" interval="15" start-delay="0"
```

See also

http://www.linux-ha.org/wiki/IPv6addr_(resource_agent)

Name

ocf_heartbeat_iSCSILogicalUnit — Manages iSCSI Logical Units (LUs)

Synopsis

`iSCSILogicalUnit` [start | stop | monitor | meta-data | validate-all]

Description

Manages iSCSI Logical Unit. An iSCSI Logical unit is a subdivision of an SCSI Target, exported via a daemon that speaks the iSCSI protocol.

Supported Parameters

`implementation`
: The iSCSI target daemon implementation. Must be one of "iet", "tgt", or "lio". If unspecified, an implementation is selected based on the availability of management utilities, with "iet" being tried first, then "tgt", then "lio". (optional, string, no default)

`target_iqn`
: The iSCSI Qualified Name (IQN) that this Logical Unit belongs to. (optional, string, no default)

`lun`
: The Logical Unit number (LUN) exposed to initiators. (optional, integer, no default)

`path`
: The path to the block device exposed. Some implementations allow this to be a regular file, too. (optional, string, no default)

`scsi_id`
: The SCSI ID to be configured for this Logical Unit. The default is the resource name, truncated to 24 bytes. (optional, string, default `default`)

`scsi_sn`
: The SCSI serial number to be configured for this Logical Unit. The default is a hash of the resource name, truncated to 8 bytes. (optional, string, default `c21f969b`)

`vendor_id`
: The SCSI vendor ID to be configured for this Logical Unit. (optional, string, no default)

`product_id`
: The SCSI product ID to be configured for this Logical Unit. (optional, string, no default)

`additional_parameters`
: Additional LU parameters. A space-separated list of "name=value" pairs which will be passed through to the iSCSI daemon's management interface. The supported parameters are implementation dependent. Neither the name nor the value may contain whitespace. (optional, string, no default)

Example

The following is an example configuration for a iSCSILogicalUnit resource using the crm(8) shell:

```
primitive example_iSCSILogicalUnit ocf:heartbeat:iSCSILogicalUnit \
  params \
    target_iqn=string \
    lun=integer \
    path=string \
  op monitor timeout="10" interval="10" depth="0"
```

See also

http://www.linux-ha.org/wiki/iSCSILogicalUnit_(resource_agent)

Name

ocf_heartbeat_iSCSITarget — iSCSI target export agent

Synopsis

`iSCSITarget` [start | stop | monitor | meta-data | validate-all]

Description

Manages iSCSI targets. An iSCSI target is a collection of SCSI Logical Units (LUs) exported via a daemon that speaks the iSCSI protocol.

Supported Parameters

`implementation`

The iSCSI target daemon implementation. Must be one of "iet", "tgt", or "lio". If unspecified, an implementation is selected based on the availability of management utilities, with "iet" being tried first, then "tgt", then "lio". (optional, string, no default)

`iqn`

The target iSCSI Qualified Name (IQN). Should follow the conventional "iqn.yyyy-mm.<reversed domain name>[:identifier]" syntax. (optional, string, no default)

`tid`

The iSCSI target ID. Required for tgt. (optional, integer, no default)

`portals`

iSCSI network portal addresses. Not supported by all implementations. If unset, the default is to create one portal that listens on . (optional, string, default `0.0.0.0:3260`)

`allowed_initiators`

Allowed initiators. A space-separated list of initiators allowed to connect to this target. Initiators may be listed in any syntax the target implementation allows. If this parameter is empty or not set, access to this target will be allowed from any initiator. (optional, string, no default)

`incoming_username`
: A username used for incoming initiator authentication. If unspecified, allowed initiators will be able to log in without authentication. (optional, string, no default)

`incoming_password`
: A password used for incoming initiator authentication. (optional, string, no default)

`additional_parameters`
: Additional target parameters. A space-separated list of "name=value" pairs which will be passed through to the iSCSI daemon's management interface. The supported parameters are implementation dependent. Neither the name nor the value may contain whitespace. (optional, string, no default)

Example

The following is an example configuration for a iSCSITarget resource using the crm(8) shell:

```
primitive example_iSCSITarget ocf:heartbeat:iSCSITarget \
  params \
    iqn=string \
  op monitor timeout="10" interval="10" depth="0"
```

See also

http://www.linux-ha.org/wiki/iSCSITarget_(resource_agent)

Name

ocf_heartbeat_iscsi — Manages a local iSCSI initiator and its connections to iSCSI targets

Synopsis

`iscsi` [start | stop | status | monitor | validate-all | methods | meta-data]

Description

OCF Resource Agent for iSCSI. Add (start) or remove (stop) iSCSI targets.

Supported Parameters

`portal`
: The iSCSI portal address in the form: {ip_address|hostname}[":"port] (optional, string, no default)

`target`
: The iSCSI target. (optional, string, no default)

`discovery_type`
: Discovery type. Currently, with open-iscsi, only the sendtargets type is supported. (optional, string, default `sendtargets`)

`iscsiadm`
: iscsiadm program path. (optional, string, no default)

`udev`

  If the next resource depends on the udev creating a device then we wait until it is finished. On a normally loaded host this should be done quickly, but you may be unlucky. If you are not using udev set this to "no", otherwise we will spin in a loop until a timeout occurs. (optional, string, default `yes`)

Example

The following is an example configuration for a iscsi resource using the crm(8) shell:

```
primitive example_iscsi ocf:heartbeat:iscsi \
  params \
    portal=string \
    target=string \
  op monitor depth="0" timeout="30" interval="120"
```

See also

http://www.linux-ha.org/wiki/iscsi_(resource_agent)

Name

ocf_heartbeat_jboss — Manages a JBoss application server instance

Synopsis

`jboss` [start | stop | status | monitor | meta-data | validate-all]

Description

Resource script for Jboss. It manages a Jboss instance as an HA resource.

Supported Parameters

`resource_name`

  The name of the resource. Defaults to the name of the resource instance. (optional, string, default `default`)

`console`

  A destination of the log of jboss run and shutdown script. (optional, string, no default)

`shutdown_timeout`

  Timeout for jboss bin/shutdown.sh. We wait for this timeout to expire, then send the TERM and QUIT signals. Finally, the KILL signal is used to terminate the jboss process. You should set the timeout for the stop operation to a value bigger than the sum of the timeout parameters. See also kill_timeout. (optional, integer, default `5`)

`kill_timeout`

  If bin/shutdown.sh doesn't stop the jboss process, then we send it TERM and QUIT signals, intermittently and once a second. After this timeout expires, if the process is still live, we use the KILL signal. See also shutdown_timeout. (optional, integer, default `10`)

`user`
: A user name to start a JBoss. (optional, string, default `root`)

`statusurl`
: URL to test in the monitor operation. (optional, string, default `http://127.0.0.1:8080`)

`java_home`
: Home directory of Java. Defaults to the environment variable JAVA_HOME. If it is not set, then define this parameter. (optional, string, no default)

`jboss_home`
: Home directory of Jboss. (optional, string, no default)

`pstring`
: With this string heartbeat matches for the right process to kill. (optional, string, default `java -Dprogram.name=run.sh`)

`run_opts`
: Start options to start Jboss with, defaults are from the Jboss-Doku. (optional, string, default `-c default -l lpg4j`)

`shutdown_opts`
: Stop options to stop Jboss with. (optional, string, default `-s 127.0.0.1:1099`)

Example

The following is an example configuration for a jboss resource using the crm(8) shell:

```
primitive example_jboss ocf:heartbeat:jboss \
  params \
    jboss_home=string \
  op monitor depth="0" timeout="30s" interval="10s"
```

See also

http://www.linux-ha.org/wiki/jboss_(resource_agent)

Name

ocf_heartbeat_LinuxSCSI — Enables and disables SCSI devices through the kernel SCSI hot-plug subsystem (deprecated)

Synopsis

`LinuxSCSI` [start | stop | methods | status | monitor | meta-data | validate-all]

Description

Deprecation warning: This agent makes use of Linux SCSI hot-plug functionality which has been superseded by SCSI reservations. It is deprecated and may be removed from a future release. See the scsi2reservation and sfex agents for alternatives. -- This is a resource agent for LinuxSCSI. It manages the availability of a SCSI device from the point of view of the linux kernel. It make Linux believe the device has gone away, and it can make it come back again.

Supported Parameters

`scsi`
: The SCSI instance to be managed. (optional, string, no default)

`ignore_deprecation`
: If set to true, suppresses the deprecation warning for this agent. (optional, boolean, default `false`)

Example

The following is an example configuration for a LinuxSCSI resource using the crm(8) shell:

```
primitive example_LinuxSCSI ocf:heartbeat:LinuxSCSI \
  params \
    scsi=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/LinuxSCSI_(resource_agent)

Name

ocf_heartbeat_LVM — Controls the availability of an LVM Volume Group

Synopsis

`LVM` [start | stop | status | monitor | methods | meta-data | validate-all]

Description

Resource script for LVM. It manages an Linux Volume Manager volume (LVM) as an HA resource.

Supported Parameters

`volgrpname`
: The name of volume group. (optional, string, no default)

`exclusive`
: If set, the volume group will be activated exclusively. (optional, string, default `false`)

Example

The following is an example configuration for a LVM resource using the crm(8) shell:

```
primitive example_LVM ocf:heartbeat:LVM \
  params \
    volgrpname=string \
  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/LVM_(resource_agent)

Name

ocf_heartbeat_MailTo — Notifies recipients by email in the event of resource takeover

Synopsis

`MailTo` [start | stop | status | monitor | meta-data | validate-all]

Description

This is a resource agent for MailTo. It sends email to a sysadmin whenever a takeover occurs.

Supported Parameters

`email`
: The email address of sysadmin. (optional, string, no default)

`subject`
: The subject of the email. (optional, string, no default)

Example

The following is an example configuration for a MailTo resource using the crm(8) shell:

```
primitive example_MailTo ocf:heartbeat:MailTo \
  params \
    email=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/MailTo_(resource_agent)

Name

ocf_heartbeat_ManageRAID — Manages RAID devices

Synopsis

`ManageRAID` [start | stop | status | monitor | validate-all | meta-data]

Description

Manages starting, stopping and monitoring of RAID devices which are preconfigured in /etc/conf.d/HB-ManageRAID.

Supported Parameters

`raidname`
: Name (case sensitive) of RAID to manage. (preconfigured in /etc/conf.d/HB-ManageRAID) (optional, string, no default)

Example

The following is an example configuration for a ManageRAID resource using the crm(8) shell:

```
primitive example_ManageRAID ocf:heartbeat:ManageRAID \
  params \
    raidname=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/ManageRAID_(resource_agent)

Name

ocf_heartbeat_ManageVE — Manages an OpenVZ Virtual Environment (VE)

Synopsis

ManageVE [start | stop | status | monitor | validate-all | meta-data]

Description

This OCF complaint resource agent manages OpenVZ VEs and thus requires a proper OpenVZ installation including a recent vzctl util.

Supported Parameters

`veid`
: OpenVZ ID of virtual environment (see output of vzlist -a for all assigned IDs) (optional, integer, no default)

Example

The following is an example configuration for a ManageVE resource using the crm(8) shell:

```
primitive example_ManageVE ocf:heartbeat:ManageVE \
  params \
    veid=integer \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/ManageVE_(resource_agent)

Name

ocf_heartbeat_mysql-proxy — Manages a MySQL Proxy daemon

Synopsis

`mysql-proxy` [start | stop | reload | monitor | validate-all | meta-data]

Description

This script manages MySQL Proxy as an OCF resource in a high-availability setup. Tested with MySQL Proxy 0.7.0 on Debian 5.0.

Supported Parameters

`binary`
: Full path to the MySQL Proxy binary. For example, "/usr/sbin/mysql-proxy". (optional, string, default `/usr/sbin/mysql-proxy`)

`defaults_file`
: Full path to a MySQL Proxy configuration file. For example, "/etc/mysql-proxy.conf". (optional, string, no default)

`proxy_backend_addresses`
: Address:port of the remote backend-servers (default: 127.0.0.1:3306). (optional, string, default `127.0.0.1:3306`)

`proxy_read_only_backend_addresses`
: Address:port of the remote (read only) slave-server (default:). (optional, string, default `127.0.0.1:3306`)

`proxy_address`
: Listening address:port of the proxy-server (default: :4040). You can also specify a socket like "/tmp/mysql-proxy.sock". (optional, string, default `:4040`)

`log_level`
: Log all messages of level (error|warning|info|message|debug|) or higher. An empty value disables logging. (optional, string, no default)

`keepalive`
: Try to restart the proxy if it crashed (default:). Valid values: true or false. An empty value equals "false". (optional, string, no default)

`admin_address`
: Listening address:port of the admin-server (default: 127.0.0.1:4041). (optional, string, default `127.0.0.1:4041`)

`admin_username`
: Username to allow to log in (default:). (optional, string, no default)

`admin_password`
: Password to allow to log in (default:). (optional, string, no default)

`admin_lua_script`
: Script to execute by the admin plugin. (optional, string, no default)

`parameters`
: The MySQL Proxy daemon may be called with additional parameters. Specify any of them here. (optional, string, no default)

`pidfile`
: PID file (optional, string, default `/var/run/heartbeat/rsctmp/mysql-proxy-default.pid`)

Example

The following is an example configuration for a mysql-proxy resource using the crm(8) shell:

```
primitive example_mysql-proxy ocf:heartbeat:mysql-proxy \
  params \

  op monitor depth="0" timeout="20s" interval="60s"
```

See also

http://www.linux-ha.org/wiki/mysql-proxy_(resource_agent)

Name

ocf_heartbeat_mysql — Manages a MySQL database instance

Synopsis

`mysql` [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for MySQL. It manages a MySQL Database instance as an HA resource.

Supported Parameters

`binary`
: Location of the MySQL binary (optional, string, default `/usr/bin/safe_mysqld`)

`config`
: Configuration file (optional, string, default `/etc/my.cnf`)

`datadir`
: Directory containing databases (optional, string, default `/var/lib/mysql`)

`user`
: User running MySQL daemon (optional, string, default `mysql`)

`group`
: Group running MySQL daemon (for logfile and directory permissions) (optional, string, default `mysql`)

`log`
: The logfile to be used for mysqld. (optional, string, default `/var/log/mysqld.log`)

`pid`
: The pidfile to be used for mysqld. (optional, string, default `/var/run/mysql/mysqld.pid`)

`socket`
: The socket to be used for mysqld. (optional, string, default `/var/lib/mysql/mysql.sock`)

`test_table`
: Table to be tested in monitor statement (in database.table notation) (optional, string, default `mysql.user`)

`test_user`
: MySQL test user (optional, string, default `root`)

`test_passwd`
: MySQL test user password (optional, string, no default)

`enable_creation`
: If the MySQL database does not exist, it will be created (optional, integer, default `0`)

`additional_parameters`
: Additional parameters which are passed to the mysqld on startup. (e.g. --skip-external-locking or --skip-grant-tables) (optional, string, no default)

Example

The following is an example configuration for a mysql resource using the crm(8) shell:

```
primitive example_mysql ocf:heartbeat:mysql \
  params \

  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/mysql_(resource_agent)

Name

ocf_heartbeat_nfsserver — Manages an NFS server

Synopsis

`nfsserver` [start | stop | monitor | meta-data | validate-all]

Description

Nfsserver helps to manage the Linux nfs server as a failover-able resource in Linux-HA. It depends on Linux specific NFS implementation details, so is considered not portable to other platforms yet.

Supported Parameters

`nfs_init_script`
: The default init script shipped with the Linux distro. The nfsserver resource agent offloads the start/stop/monitor work to the init script because the procedure to start/stop/monitor nfsserver varies on different Linux distro. (optional, string, default `/etc/init.d/nfsserver`)

`nfs_notify_cmd`
: The tool to send out NSM reboot notification. Failover of nfsserver can be considered as rebooting to different machines. The nfsserver resource agent use this command to notify all clients about the happening of failover. (optional, string, default `/sbin/sm-notify/sbin/sm-notify`)

`nfs_shared_infodir`
: The nfsserver resource agent will save nfs related information in this specific directory. And this directory must be able to fail-over before nfsserver itself. (optional, string, no default)

`nfs_ip`
: The floating IP address used to access the nfs service (optional, string, no default)

Example

The following is an example configuration for a nfsserver resource using the crm(8) shell:

```
primitive example_nfsserver ocf:heartbeat:nfsserver \
  params \
    nfs_shared_infodir=string \
    nfs_ip=string \
  op monitor depth="0" timeout="20" interval="10"
```

See also

http://www.linux-ha.org/wiki/nfsserver_(resource_agent)

Name

ocf_heartbeat_oracle — Manages an Oracle Database instance

Synopsis

`oracle` [start | stop | status | monitor | validate-all | methods | meta-data]

Description

Resource script for oracle. Manages an Oracle Database instance as an HA resource.

Supported Parameters

`sid`
: The Oracle SID (aka ORACLE_SID). (optional, string, no default)

`home`
: The Oracle home directory (aka ORACLE_HOME). If not specified, then the SID along with its home should be listed in /etc/oratab. (optional, string, no default)

`user`
: The Oracle owner (aka ORACLE_OWNER). If not specified, then it is set to the owner of file $ORACLE_HOME/dbs/*${ORACLE_SID}.ora. If this does not work for you, just set it explicitely. (optional, string, no default)

`ipcrm`
: Sometimes IPC objects (shared memory segments and semaphores) belonging to an Oracle instance might be left behind which prevents the instance from starting. It is not easy to figure out which shared segments belong to which instance, in particular when more instances are running as same user. What we use here is the "oradebug" feature and its "ipc" trace utility. It is not optimal to parse the debugging information, but I am not aware of any other way to find out about the IPC information. In case the format or wording of the trace report changes, parsing might fail. There are some precautions, however, to prevent stepping on other peoples toes. There is also a dumpinstipc option which will make us print the IPC objects which belong to the instance. Use it to see if we parse the trace file correctly. Three settings are possible: - none: don't mess with IPC and hope for the best (beware: you'll probably be out of luck, sooner or later) - instance: try to figure out the IPC stuff which belongs to the instance and remove only those (default; should be safe) - orauser: remove all IPC belonging to the user which runs the instance (don't use this if you run more than one instance as same user or if other apps running as this user use IPC) The default setting "instance" should be safe to use, but in that case we cannot guarantee that the instance will start. In case IPC objects were already left around, because, for instance, someone mercilessly killing Oracle processes, there is no way any more to find out which IPC objects should be removed. In that case, human intervention is necessary, and probably _all_ instances running as same user will have to be stopped. The third setting, "orauser", guarantees IPC objects removal, but it does that based only on IPC objects ownership, so you should use that only if every instance runs as separate user. Please report any problems. Suggestions/fixes welcome. (optional, string, default `instance`)

`clear_backupmode`
: The clear of the backup mode of ORACLE. (optional, boolean, default `false`)

`shutdown_method`
: How to stop Oracle is a matter of taste it seems. The default method ("checkpoint/abort") is: alter system checkpoint; shutdown abort; This should be the fastest safe way bring the instance down. If you find "shutdown abort" distasteful, set this attribute to "immediate" in which case we will shutdown immediate; If you still think that there's even better way to shutdown an Oracle instance we are willing to listen. (optional, string, default `checkpoint/abort`)

Example

The following is an example configuration for a oracle resource using the crm(8) shell:

```
primitive example_oracle ocf:heartbeat:oracle \
  params \
    sid=string \
  op monitor depth="0" timeout="30" interval="120"
```

See also

http://www.linux-ha.org/wiki/oracle_(resource_agent)

Name

ocf_heartbeat_oralsnr — Manages an Oracle TNS listener

Synopsis

`oralsnr` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for Oracle Listener. It manages an Oracle Listener instance as an HA resource.

Supported Parameters

`sid`

The Oracle SID (aka ORACLE_SID). Necessary for the monitor op, i.e. to do tnsping SID. (optional, string, no default)

`home`

The Oracle home directory (aka ORACLE_HOME). If not specified, then the SID should be listed in /etc/oratab. (optional, string, no default)

`user`

Run the listener as this user. (optional, string, no default)

`listener`

Listener instance to be started (as defined in listener.ora). Defaults to LISTENER. (optional, string, no default)

Example

The following is an example configuration for a oralsnr resource using the crm(8) shell:

```
primitive example_oralsnr ocf:heartbeat:oralsnr \
  params \
    sid=string \
  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/oralsnr_(resource_agent)

Name

ocf_heartbeat_pgsql — Manages a PostgreSQL database instance

Synopsis

`pgsql` [start | stop | status | monitor | meta-data | validate-all | methods]

Description

Resource script for PostgreSQL. It manages a PostgreSQL as an HA resource.

Supported Parameters

`pgctl`
: Path to pg_ctl command. (optional, string, default `/usr/bin/pg_ctl`)

`start_opt`
: Start options (-o start_opt in pgi_ctl). "-i -p 5432" for example. (optional, string, no default)

`ctl_opt`
: Additional pg_ctl options (-w, -W etc..). Default is "" (optional, string, no default)

`psql`
: Path to psql command. (optional, string, default `/usr/bin/psql`)

`pgdata`
: Path PostgreSQL data directory. (optional, string, default `/var/lib/pgsql/data`)

`pgdba`
: User that owns PostgreSQL. (optional, string, default `postgres`)

`pghost`
: Hostname/IP Addreess where PosrgeSQL is listening (optional, string, no default)

`pgport`
: Port where PosrgeSQL is listening (optional, string, default `5432`)

`pgdb`
: Database that will be used for monitoring. (optional, string, default `template1`)

`logfile`
: Path to PostgreSQL server log output file. (optional, string, default `/dev/null`)

`stop_escalate`
: Number of retries (using -m fast) before resorting to -m immediate (optional, string, default `30`)

Example

The following is an example configuration for a pgsql resource using the crm(8) shell:

```
primitive example_pgsql ocf:heartbeat:pgsql \
  params \
op monitor depth="0" timeout="30" interval="30"
```

See also

http://www.linux-ha.org/wiki/pgsql_(resource_agent)

Name

ocf_heartbeat_pingd — Monitors connectivity to specific hosts or IP addresses ("ping nodes") (deprecated)

Synopsis

`pingd` [start | stop | monitor | meta-data | validate-all]

Description

Deprecation warning: This agent is deprecated and may be removed from a future release. See the ocf:pacemaker:pingd resource agent for a supported alternative. -- This is a pingd Resource Agent. It records (in the CIB) the current number of ping nodes a node can connect to.

Supported Parameters

`pidfile`
: PID file (optional, string, default `/var/run/heartbeat/rsctmp/pingd-default`)

`user`
: The user we want to run pingd as (optional, string, default `root`)

`dampen`
: The time to wait (dampening) further changes occur (optional, integer, default `1s`)

`set`
: The name of the instance_attributes set to place the value in. Rarely needs to be specified. (optional, integer, no default)

`name`
: The name of the attributes to set. This is the name to be used in the constraints. (optional, integer, default `default`)

`section`
: The section place the value in. Rarely needs to be specified. (optional, integer, no default)

`multiplier`
: The number by which to multiply the number of connected ping nodes by (optional, integer, no default)

`host_list`
: The list of ping nodes to count. Defaults to all configured ping nodes. Rarely needs to be specified. (optional, integer, no default)

`ignore_deprecation`
: If set to true, suppresses the deprecation warning for this agent. (optional, boolean, default `false`)

Example

The following is an example configuration for a pingd resource using the crm(8) shell:

```
primitive example_pingd ocf:heartbeat:pingd \
  params \

  op monitor depth="0" timeout="20" interval="10"
```

See also

http://www.linux-ha.org/wiki/pingd_(resource_agent)

Name

ocf_heartbeat_portblock — Block and unblocks access to TCP and UDP ports

Synopsis

`portblock` [start | stop | status | monitor | meta-data | validate-all]

Description

Resource script for portblock. It is used to temporarily block ports using iptables. In addition, it may allow for faster TCP reconnects for clients on failover. Use that if there are long lived TCP connections to an HA service. This feature is enabled by setting the tickle_dir parameter and only in concert with action set to unblock. Note that the tickle ACK function is new as of version 3.0.2 and hasn't yet seen widespread use.

Supported Parameters

`protocol`
: The protocol used to be blocked/unblocked. (optional, string, no default)

`portno`
: The port number used to be blocked/unblocked. (optional, integer, no default)

`action`
: The action (block/unblock) to be done on the protocol::portno. (optional, string, no default)

`ip`
: The IP address used to be blocked/unblocked. (optional, string, default `0.0.0.0/0`)

`tickle_dir`
: The shared or local directory (_must_ be absolute path) which stores the established TCP connections. (optional, string, no default)

`sync_script`
: If the tickle_dir is a local directory, then the TCP connection state file has to be replicated to other nodes in the cluster. It can be csync2 (default), some wrapper of rsync, or whatever. It takes the file name as a single argument. For csync2, set it to "csync2 -xv". (optional, string, no default)

Example

The following is an example configuration for a portblock resource using the crm(8) shell:

```
primitive example_portblock ocf:heartbeat:portblock \
  params \
    protocol=string \
    portno=integer \
    action=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/portblock_(resource_agent)

Name

ocf_heartbeat_postfix — Manages a highly available Postfix mail server instance

Synopsis

`postfix` [start | stop | reload | monitor | validate-all | meta-data]

Description

This script manages Postfix as an OCF resource in a high-availability setup. Tested with Postfix 2.5.5 on Debian 5.0.

Supported Parameters

`binary`
: Full path to the Postfix binary. For example, "/usr/sbin/postfix". (optional, string, default `/usr/sbin/postfix`)

`config_dir`
: Full path to a Postfix configuration directory. For example, "/etc/postfix". (optional, string, no default)

`parameters`
: The Postfix daemon may be called with additional parameters. Specify any of them here. (optional, string, no default)

Example

The following is an example configuration for a postfix resource using the crm(8) shell:

```
primitive example_postfix ocf:heartbeat:postfix \
  params \

  op monitor depth="0" timeout="20s" interval="60s"
```

See also

http://www.linux-ha.org/wiki/postfix_(resource_agent)

Name

ocf_heartbeat_proftpd — OCF Resource Agent compliant FTP script.

Synopsis

`proftpd` [start | stop | monitor | monitor | validate-all | meta-data]

Description

This script manages Proftpd in an Active-Passive setup

Supported Parameters

`binary`
: The Proftpd binary (optional, string, default `/usr/sbin/proftpd`)

`conffile`
: The Proftpd configuration file name with full path. For example, "/etc/proftpd.conf" (optional, string, default `/etc/proftpd.conf`)

`pidfile`
: The Proftpd PID file. The location of the PID file is configured in the Proftpd configuration file. (optional, string, default `/var/run/proftpd.pid`)

`curl_binary`
: The absolut path to the curl binary for monitoring with OCF_CHECK_LEVEL greater zero. (optional, string, default `/usr/bin/curl`)

`curl_url`
: The URL which is checked by curl with OCF_CHECK_LEVEL greater zero. (optional, string, default `ftp://localhost/`)

`test_user`
: The name of the ftp user for monitoring with OCF_CHECK_LEVEL greater zero. (optional, string, default `test`)

`test_pass`
: The password of the ftp user for monitoring with OCF_CHECK_LEVEL greater zero. (optional, string, no default)

Example

The following is an example configuration for a proftpd resource using the crm(8) shell:

```
primitive example_proftpd ocf:heartbeat:proftpd \
  params \

  op monitor depth="0" timeout="20s" interval="60s"
  op monitor depth="10" timeout="20s" interval="120s"
```

See also

http://www.linux-ha.org/wiki/proftpd_(resource_agent)

Name

ocf_heartbeat_Pure-FTPd — Manages a Pure-FTPd FTP server instance

Synopsis

`Pure-FTPd` [start | stop | monitor | validate-all | meta-data]

Description

This script manages Pure-FTPd in an Active-Passive setup

Supported Parameters

`script`
: The full path to the Pure-FTPd startup script. For example, "/sbin/pure-config.pl" (optional, string, default `/sbin/pure-config.pl`)

`conffile`
: The Pure-FTPd configuration file name with full path. For example, "/etc/pure-ftpd/pure-ftpd.conf" (optional, string, default `/etc/pure-ftpd/pure-ftpd.conf`)

`daemon_type`
: The Pure-FTPd daemon to be called by pure-ftpd-wrapper. Valid options are "" for pure-ftpd, "mysql" for pure-ftpd-mysql, "postgresql" for pure-ftpd-postgresql and "ldap" for pure-ftpd-ldap (optional, string, no default)

`pidfile`
: PID file (optional, string, default `/var/run/heartbeat/rsctmp/pure-ftpd-default.pid`)

Example

The following is an example configuration for a Pure-FTPd resource using the crm(8) shell:

```
primitive example_Pure-FTPd ocf:heartbeat:Pure-FTPd \
  params \

  op monitor depth="0" timeout="20s" interval="60s"
```

See also

http://www.linux-ha.org/wiki/Pure-FTPd_(resource_agent)

Name

ocf_heartbeat_Raid1 — Manages a software RAID1 device on shared storage

Synopsis

`Raid1` [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for RAID1. It manages a software Raid1 device on a shared storage medium.

Supported Parameters

`raidconf`
: The RAID configuration file. e.g. /etc/raidtab or /etc/mdadm.conf. (optional, string, no default)

`raiddev`
: The block device to use. (optional, string, no default)

`homehost`
: The value for the homehost directive; this is an mdadm feature to protect RAIDs against being activated by accident. It is recommended to create RAIDs managed by the cluster with "homehost" set to a special value, so they are not accidentially auto-assembled by nodes not supposed to own them. (optional, string, no default)

Example

The following is an example configuration for a Raid1 resource using the crm(8) shell:

```
primitive example_Raid1 ocf:heartbeat:Raid1 \
  params \
    raidconf=string \
    raiddev=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/Raid1_(resource_agent)

Name

ocf_heartbeat_Route — Manages network routes

Synopsis

`Route` [start | stop | monitor | reload | meta-data | validate-all]

Description

Enables and disables network routes. Supports host and net routes, routes via a gateway address, and routes using specific source addresses. This resource agent is useful if a node's routing table needs to be manipulated based on node role assignment. Consider the following example use case: - One cluster node serves as an IPsec tunnel endpoint. - All other nodes use the IPsec tunnel to reach hosts in a specific remote network. Then, here is how you would implement this scheme making use of the Route resource agent: - Configure an ipsec LSB resource. - Configure a cloned Route OCF resource. - Create an order constraint to ensure that ipsec is started before Route. - Create a colocation constraint between the ipsec and Route resources, to make sure no instance of your cloned Route resource is started on the tunnel endpoint itself.

Supported Parameters

`destination`
: The destination network (or host) to be configured for the route. Specify the netmask suffix in CIDR notation (e.g. "/24"). If no suffix is given, a host route will be created. Specify "0.0.0.0/0" or "default" if you want this resource to set the system default route. (optional, string, no default)

`device`
: The outgoing network device to use for this route. (optional, string, no default)

`gateway`
: The gateway IP address to use for this route. (optional, string, no default)

`source`
: The source IP address to be configured for the route. (optional, string, no default)

Example

The following is an example configuration for a Route resource using the crm(8) shell:

```
primitive example_Route ocf:heartbeat:Route \
  params \
    destination=string \
  op monitor timeout="20" interval="10" depth="0"
```

See also

http://www.linux-ha.org/wiki/Route_(resource_agent)

Name

ocf_heartbeat_rsyncd — Manages an rsync daemon

Synopsis

`rsyncd` [start | stop | monitor | validate-all | meta-data]

Description

This script manages rsync daemon

Supported Parameters

`binpath`
: The rsync binary path. For example, "/usr/bin/rsync" (optional, string, default `rsync`)

`conffile`
: The rsync daemon configuration file name with full path. For example, "/etc/rsyncd.conf" (optional, string, default `/etc/rsyncd.conf`)

`bwlimit`
: This option allows you to specify a maximum transfer rate in kilobytes per second. This option is most effective when using rsync with large files (several megabytes and up). Due to the nature of rsync transfers, blocks of data are sent, then if rsync determines the transfer was too fast, it will wait before sending the next data block. The result is an average transfer rate equaling the specified limit. A value of zero specifies no limit. (optional, string, no default)

Example

The following is an example configuration for a rsyncd resource using the crm(8) shell:

```
primitive example_rsyncd ocf:heartbeat:rsyncd \
  params \

  op monitor depth="0" timeout="30s" interval="60s"
```

See also

http://www.linux-ha.org/wiki/rsyncd_(resource_agent)

Name

ocf_heartbeat_SAPDatabase — Manages any SAP database (based on Oracle, MaxDB, or DB2)

Synopsis

`SAPDatabase` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for SAP databases. It manages a SAP database of any type as an HA resource.

Supported Parameters

`SID`
: The unique SAP system identifier. e.g. P01 (optional, string, no default)

`DIR_EXECUTABLE`
: The full qualified path where to find sapstartsrv and sapcontrol. (optional, string, no default)

`DBTYPE`
: The name of the database vendor you use. Set either: ORA,DB6,ADA (optional, string, no default)

`NETSERVICENAME`
: The Oracle TNS listener name. (optional, string, no default)

`DBJ2EE_ONLY`
: If you do not have a ABAP stack installed in the SAP database, set this to TRUE (optional, boolean, default `false`)

`JAVA_HOME`
: This is only needed if the DBJ2EE_ONLY parameter is set to true. Enter the path to the Java SDK which is used by the SAP WebAS Java (optional, string, no default)

`STRICT_MONITORING`
: This controls how the resource agent monitors the database. If set to true, it will use SAP tools to test the connect to the database. Do not use with Oracle, because it will result in unwanted failovers in case of an archiver stuck (optional, boolean, default `false`)

`AUTOMATIC_RECOVER`
: The SAPDatabase resource agent tries to recover a failed start attempt automaticaly one time. This is done by running a forced abort of the RDBMS and/or executing recovery commands. (optional, boolean, default `false`)

`DIR_BOOTSTRAP`
: The full qualified path where to find the J2EE instance bootstrap directory. e.g. /usr/sap/P01/J00/j2ee/cluster/bootstrap (optional, string, no default)

`DIR_SECSTORE`
: The full qualified path where to find the J2EE security store directory. e.g. /usr/sap/P01/SYS/global/security/lib/tools (optional, string, no default)

`DB_JARS`
: The full qualified filename of the jdbc driver for the database connection test. It will be automaticaly read from the bootstrap.properties file in Java engine 6.40 and 7.00. For Java engine 7.10 the parameter is mandatory. (optional, string, no default)

`PRE_START_USEREXIT`
: The full qualified path where to find a script or program which should be executed before this resource gets started. (optional, string, no default)

`POST_START_USEREXIT`
: The full qualified path where to find a script or program which should be executed after this resource got started. (optional, string, no default)

`PRE_STOP_USEREXIT`
: The full qualified path where to find a script or program which should be executed before this resource gets stopped. (optional, string, no default)

`POST_STOP_USEREXIT`
: The full qualified path where to find a script or program which should be executed after this resource got stopped. (optional, string, no default)

Example

The following is an example configuration for a SAPDatabase resource using the crm(8) shell:

```
primitive example_SAPDatabase ocf:heartbeat:SAPDatabase \
  params \
    SID=string \
    DBTYPE=string \
  op monitor depth="0" timeout="60" interval="120"
```

See also

http://www.linux-ha.org/wiki/SAPDatabase_(resource_agent)

Name

ocf_heartbeat_SAPInstance — Manages a SAP instance

Synopsis

`SAPInstance` [start | stop | status | monitor | promote | demote | validate-all | meta-data | methods]

Description

Resource script for SAP. It manages a SAP Instance as an HA resource.

Supported Parameters

`InstanceName`
: The full qualified SAP instance name. e.g. P01_DVEBMGS00_sapp01ci (optional, string, no default)

`DIR_EXECUTABLE`
: The full qualified path where to find sapstartsrv and sapcontrol. (optional, string, no default)

`DIR_PROFILE`
: The full qualified path where to find the SAP START profile. (optional, string, no default)

`START_PROFILE`
: The name of the SAP START profile. (optional, string, no default)

`START_WAITTIME`
: After that time in seconds a monitor operation is executed by the resource agent. Does the monitor return SUCCESS, the start is handled as SUCCESS. This is useful to resolve timing problems with e.g. the J2EE-Addin instance. (optional, string, default `3600`)

`AUTOMATIC_RECOVER`
: The SAPInstance resource agent tries to recover a failed start attempt automaticaly one time. This is done by killing runing instance processes and executing cleanipc. (optional, boolean, default `false`)

`MONITOR_SERVICES`
: (optional, string, default `disp+work|msg_server|enserver|enrepserver|jcontrol|jstart`)

`ERS_InstanceName`
: (optional, string, no default)

`ERS_START_PROFILE`
: (optional, string, no default)

`PRE_START_USEREXIT`
: The full qualified path where to find a script or program which should be executed before this resource gets started. (optional, string, no default)

`POST_START_USEREXIT`
: The full qualified path where to find a script or program which should be executed after this resource got started. (optional, string, no default)

`PRE_STOP_USEREXIT`
: The full qualified path where to find a script or program which should be executed before this resource gets stopped. (optional, string, no default)

`POST_STOP_USEREXIT`
: The full qualified path where to find a script or program which should be executed after this resource got stopped. (optional, string, no default)

Example

The following is an example configuration for a SAPInstance resource using the crm(8) shell:

```
primitive example_SAPInstance ocf:heartbeat:SAPInstance \
  params \
    InstanceName=string \
  op monitor depth="0" timeout="60" interval="120"
```

See also

http://www.linux-ha.org/wiki/SAPInstance_(resource_agent)

Name

ocf_heartbeat_scsi2reservation — scsi-2 reservation

Synopsis

`scsi2reservation` [start | stop | monitor | meta-data | validate-all]

Description

The scsi-2-reserve resource agent is a place holder for SCSI-2 reservation. A healthy instance of scsi-2-reserve resource, indicates the own of the specified SCSI device. This resource agent depends on the scsi_reserve from scsires package, which is Linux specific.

Supported Parameters

`scsi_reserve`
: The scsi_reserve is a command from scsires package. It helps to issue SCSI-2 reservation on SCSI devices. (optional, string, default `/usr/sbin/scsi_reserve`)

`sharedisk`
: The shared disk that can be reserved. (optional, string, default `/dev/sdb`)

`start_loop`
: We are going to try several times before giving up. Start_loop indicates how many times we are going to re-try. (optional, string, default `10`)

Example

The following is an example configuration for a scsi2reservation resource using the crm(8) shell:

```
primitive example_scsi2reservation ocf:heartbeat:scsi2reservation \
  params \

  op monitor depth="0" timeout="20" interval="20"
```

See also

http://www.linux-ha.org/wiki/scsi2reservation_(resource_agent)

Name

ocf_heartbeat_SendArp — Broadcasts unsolicited ARP announcements

Synopsis

`SendArp` [start | stop | monitor | meta-data | validate-all]

Description

This script send out gratuitous Arp for an IP address

Supported Parameters

`ip`
: The IP address for sending arp package. (optional, string, no default)

`nic`
: The nic for sending arp package. (optional, string, no default)

Example

The following is an example configuration for a SendArp resource using the crm(8) shell:

```
primitive example_SendArp ocf:heartbeat:SendArp \
  params \
    ip=string \
    nic=string \
  op monitor depth="0" timeout="20" interval="10"
```

See also

http://www.linux-ha.org/wiki/SendArp_(resource_agent)

Name

ocf_heartbeat_ServeRAID — Enables and disables shared ServeRAID merge groups

Synopsis

`ServeRAID` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for ServeRAID. It enables/disables shared ServeRAID merge groups.

Supported Parameters

`serveraid`
: The adapter number of the ServeRAID adapter. (optional, integer, no default)

`mergegroup`
: The logical drive under consideration. (optional, integer, no default)

Example

The following is an example configuration for a ServeRAID resource using the crm(8) shell:

```
primitive example_ServeRAID ocf:heartbeat:ServeRAID \
  params \
    serveraid=integer \
    mergegroup=integer \
  op monitor depth="0" timeout="20" interval="10"
```

See also

http://www.linux-ha.org/wiki/ServeRAID_(resource_agent)

Name

ocf_heartbeat_sfex — Manages exclusive acess to shared storage using Shared Disk File EXclusiveness (SF-EX)

Synopsis

`sfex` [start | stop | monitor | meta-data]

Description

Resource script for SF-EX. It manages a shared storage medium exclusively .

Supported Parameters

`device`
: Block device path that stores exclusive control data. (optional, string, no default)

`index`
: Location in block device where exclusive control data is stored. 1 or more is specified. Default is 1. (optional, integer, default `1`)

`collision_timeout`
: Waiting time when a collision of lock acquisition is detected. Default is 1 second. (optional, integer, default `1`)

`monitor_interval`
: Monitor interval(sec). Default is 10 seconds (optional, integer, default `10`)

`lock_timeout`
: Valid term of lock(sec). Default is 20 seconds. (optional, integer, default `20`)

Example

The following is an example configuration for a sfex resource using the crm(8) shell:

```
primitive example_sfex ocf:heartbeat:sfex \
  params \
    device=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/sfex_(resource_agent)

Name

ocf_heartbeat_SphinxSearchDaemon — Manages the Sphinx search daemon.

Synopsis

`SphinxSearchDaemon` [start | stop | monitor | meta-data | validate-all]

Description

This is a searchd Resource Agent. It manages the Sphinx Search Daemon.

Supported Parameters

`config`
: searchd configuration file (optional, string, default `/etc/sphinx/sphinx.conf`)

`searchd`
: searchd binary (optional, string, default `/usr/local/bin/searchd`)

`search`
: Search binary for functional testing in the monitor action. (optional, string, default `/usr/local/bin/search`)

`testQuery`
: Test query for functional testing in the monitor action. The query does not need to match any documents in the index. The purpose is merely to test whether the search daemon is is able to query its indices and respond properly. (optional, string, default `Heartbeat_Monitor_Query_Match_string`)

Example

The following is an example configuration for a SphinxSearchDaemon resource using the crm(8) shell:

```
primitive example_SphinxSearchDaemon ocf:heartbeat:SphinxSearchDaemon \
  params \

  op monitor timeout="20" interval="10" depth="0"
```

See also

http://www.linux-ha.org/wiki/SphinxSearchDaemon_(resource_agent)

Name

ocf_heartbeat_Squid — Manages a Squid proxy server instance

Synopsis

`Squid` [start | stop | status | monitor | meta-data | validate-all]

Description

The resource agent of Squid. This manages a Squid instance as an HA resource.

Supported Parameters

`squid_exe`
: This is a required parameter. This parameter specifies squid's executable file. (optional, string, no default)

`squid_conf`
: This is a required parameter. This parameter specifies a configuration file for a squid instance managed by this RA. (optional, string, no default)

`squid_pidfile`
: This is a required parameter. This parameter specifies a process id file for a squid instance managed by this RA. (optional, string, no default)

`squid_port`
: This is a required parameter. This parameter specifies a port number for a squid instance managed by this RA. If plural ports are used, you must specifiy the only one of them. (optional, integer, no default)

`squid_stop_timeout`
: This is an omittable parameter. On a stop action, a normal stop method is firstly used. and then the confirmation of its completion is awaited for the specified seconds by this parameter. The default value is 10. (optional, integer, default `10`)

`debug_mode`
: This is an optional parameter. This RA runs in debug mode when this parameter includes 'x' or 'v'. If 'x' is included, both of STDOUT and STDERR redirect to the logfile specified by "debug_log", and then the builtin shell option 'x' is turned on. It is similar about 'v'. (optional, string, no default)

`debug_log`
: This is an optional and omittable parameter. This parameter specifies a destination file for debug logs and works only if this RA run in debug mode. Refer to "debug_mode" about debug mode. If no value is given but it's requied, it's made by the following rules: "/var/log/" as a directory part, the basename of the configuration file given by "syslog_ng_conf" as a basename part, ".log" as a suffix. (optional, string, no default)

Example

The following is an example configuration for a Squid resource using the crm(8) shell:

```
primitive example_Squid ocf:heartbeat:Squid \
  params \
    squid_exe=string \
    squid_conf=string \
    squid_pidfile=string \
    squid_port=integer \
  op monitor depth="0" timeout="30s" interval="10s"
```

See also

http://www.linux-ha.org/wiki/Squid_(resource_agent)

Name

ocf_heartbeat_Stateful — Example stateful resource agent

Synopsis

`Stateful` [start | stop | monitor | meta-data | validate-all]

Description

This is an example resource agent that impliments two states

Supported Parameters

`state`
: Location to store the resource state in (optional, string, default `/var/run/heartbeat/rsctmp/Stateful-{OCF_RESOURCE_INSTANCE}.state`)

Example

The following is an example configuration for a Stateful resource using the crm(8) shell:

```
primitive example_Stateful ocf:heartbeat:Stateful \
  params \

  op monitor depth="0" timeout="20" interval="10"
```

See also

http://www.linux-ha.org/wiki/Stateful_(resource_agent)

Name

ocf_heartbeat_SysInfo — Records various node attributes in the CIB

Synopsis

`SysInfo` [start | stop | monitor | meta-data | validate-all]

Description

This is a SysInfo Resource Agent. It records (in the CIB) various attributes of a node Sample Linux output: arch: i686 os: Linux-2.4.26-gentoo-r14 free_swap: 1999 cpu_info: Intel(R) Celeron(R) CPU 2.40GHz cpu_speed: 4771.02 cpu_cores: 1 cpu_load: 0.00 ram_total: 513 ram_free: 117 root_free: 2.4 Sample Darwin output: arch: i386 os: Darwin-8.6.2 cpu_info: Intel Core Duo cpu_speed: 2.16 cpu_cores: 2 cpu_load: 0.18 ram_total: 2016 ram_free: 787 root_free: 13 Units: free_swap: Mb ram_*: Mb root_free: Gb cpu_speed (Linux): bogomips cpu_speed (Darwin): Ghz

Supported Parameters

`pidfile`
: PID file (optional, string, default `/var/run/heartbeat/rsctmp/SysInfo-default`)

`delay`
: Interval to allow values to stabilize (optional, string, default `0s`)

Example

The following is an example configuration for a SysInfo resource using the crm(8) shell:

```
primitive example_SysInfo ocf:heartbeat:SysInfo \
  params \

  op monitor timeout="20s" interval="60s"
```

See also

http://www.linux-ha.org/wiki/SysInfo_(resource_agent)

Name

ocf_heartbeat_syslog-ng — Syslog-ng resource agent

Synopsis

`syslog-ng` [start | stop | status | monitor | meta-data | validate-all]

Description

This script manages a syslog-ng instance as an HA resource.

Supported Parameters

`configfile`
: This parameter specifies a configuration file for a syslog-ng instance managed by this RA. (optional, string, no default)

`syslog_ng_binary`
: This parameter specifies syslog-ng's executable file. (optional, string, default `/sbin/syslog-ng`)

`start_opts`
: This parameter specifies startup options for a syslog-ng instance managed by this RA. When no value is given, no startup options is used. Don't use option '-F'. It causes a stuck of a start action. (optional, string, no default)

`kill_term_timeout`
: On a stop action, a normal stop method(pkill -TERM) is firstly used. And then the confirmation of its completion is waited for the specified seconds by this parameter. The default value is 10. (optional, integer, default `10`)

Example

The following is an example configuration for a syslog-ng resource using the crm(8) shell:

```
primitive example_syslog-ng ocf:heartbeat:syslog-ng \
  params \
    configfile=string \
  op monitor depth="0" timeout="60s" interval="60s"
```

See also

http://www.linux-ha.org/wiki/syslog-ng_(resource_agent)

Name

ocf_heartbeat_tomcat — Manages a Tomcat servlet environment instance

Synopsis

tomcat [start | stop | status | monitor | meta-data | validate-all]

Description

Resource script for tomcat. It manages a Tomcat instance as an HA resource.

Supported Parameters

`tomcat_name`
: The name of the resource (optional, string, no default)

`script_log`
: A destination of the log of this script (optional, string, no default)

`tomcat_stop_timeout`
: Time-out at the time of the stop (optional, integer, no default)

`tomcat_suspend_trialcount`
: The re-try number of times awaiting a stop (optional, integer, no default)

`tomcat_user`
: A user name to start a resource (optional, string, no default)

`statusurl`
: URL for state confirmation (optional, string, no default)

`java_home`
: Home directory of the Java (optional, string, no default)

`catalina_home`
: Home directory of Tomcat (optional, string, no default)

`catalina_pid`
: A PID file name of Tomcat (optional, string, no default)

`tomcat_start_opts`
: Tomcat start options (optional, string, no default)

`catalina_opts`
: Catalina options (optional, string, no default)

`catalina_rotate_log`
: Rotate catalina.out flag (optional, string, no default)

`catalina_rotatetime`
: Time span of the rotate catalina.out (optional, integer, no default)

Example

The following is an example configuration for a tomcat resource using the crm(8) shell:

```
primitive example_tomcat ocf:heartbeat:tomcat \
  params \
    java_home=string \
    catalina_home=string \
  op monitor depth="0" timeout="30s" interval="10s"
```

See also

http://www.linux-ha.org/wiki/tomcat_(resource_agent)

Name

ocf_heartbeat_VIPArip — Manages a virtual IP address through RIP2

Synopsis

`VIPArip` [start | stop | monitor | validate-all | meta-data]

Description

Virtual IP Address by RIP2 protocol. This script manages IP alias in different subnet with quagga/ripd. It can add an IP alias, or remove one.

Supported Parameters

`ip`
: The IPv4 address in different subnet, for example "192.168.1.1". (optional, string, no default)

`nic`
: The nic for broadcast the route information. The ripd uses this nic to broadcast the route informaton to others (optional, string, default `eth0`)

`zebra_binary`
: Absolute path to the zebra binary. (optional, string, default `/usr/sbin/zebra`)

`ripd_binary`
: Absolute path to the ripd binary. (optional, string, default `/usr/sbin/ripd`)

Example

The following is an example configuration for a VIPArip resource using the crm(8) shell:

```
primitive example_VIPArip ocf:heartbeat:VIPArip \
  params \
    ip=string \
  op monitor depth="0" timeout="20s" interval="5s"
```

See also

http://www.linux-ha.org/wiki/VIPArip_(resource_agent)

Name

ocf_heartbeat_VirtualDomain — Manages virtual domains through the libvirt virtualization framework

Synopsis

`VirtualDomain` [start | stop | status | monitor | migrate_from | migrate_to | meta-data | validate-all]

Description

Resource agent for a virtual domain (a.k.a. domU, virtual machine, virtual environment etc., depending on context) managed by libvirtd.

Supported Parameters

`config`
: Absolute path to the libvirt configuration file, for this virtual domain. (optional, string, no default)

`hypervisor`
: Hypervisor URI to connect to. See the libvirt documentation for details on supported URI formats. The default is system dependent. (optional, string, default `qemu:///system`)

`force_stop`
: Always forcefully shut down ("destroy") the domain on stop. The default behavior is to resort to a forceful shutdown only after a graceful shutdown attempt has failed. You should only set this to true if your virtual domain (or your virtualization backend) does not support graceful shutdown. (optional, boolean, default `0`)

`migration_transport`
: Transport used to connect to the remote hypervisor while migrating. Please refer to the libvirt documentation for details on transports available. If this parameter is omitted, the resource will use libvirt's default transport to connect to the remote hypervisor. (optional, string, no default)

`monitor_scripts`
: To additionally monitor services within the virtual domain, add this parameter with a list of scripts to monitor. Note: when monitor scripts are used, the start and migrate_from operations will complete only when all monitor scripts have completed successfully. Be sure to set the timeout of these operations to accommodate this delay. (optional, string, no default)

Example

The following is an example configuration for a VirtualDomain resource using the crm(8) shell:

```
primitive example_VirtualDomain ocf:heartbeat:VirtualDomain \
  params \
    config=string \
  meta allow-migrate="true" \
  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/VirtualDomain_(resource_agent)

Name

ocf_heartbeat_vmware — Manages VMWare Server 2.0 virtual machines

Synopsis

`vmware` [start | stop | monitor | meta-data]

Description

OCF compliant script to control vmware server 2.0 virtual machines.

Supported Parameters

`vmxpath`
: VMX configuration file path (optional, string, no default)

`vimshbin`
: vmware-vim-cmd executable path (optional, string, default `/usr/bin/vmware-vim-cmd`)

Example

The following is an example configuration for a vmware resource using the crm(8) shell:

```
primitive example_vmware ocf:heartbeat:vmware \
  params \
    vmxpath=string \
    vimshbin="/usr/bin/vmware-vim-cmd" \
  op monitor timeout="30" interval="300" depth="0" start-delay="0"
```

See also

http://www.linux-ha.org/wiki/vmware_(resource_agent)

Name

ocf_heartbeat_WAS6 — Manages a WebSphere Application Server 6 instance

Synopsis

`WAS6` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for WAS6. It manages a Websphere Application Server (WAS6) as an HA resource.

Supported Parameters

`profile`
: The WAS profile name. (optional, string, default `default`)

Example

The following is an example configuration for a WAS6 resource using the crm(8) shell:

```
primitive example_WAS6 ocf:heartbeat:WAS6 \
  params \

  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/WAS6_(resource_agent)

Name

ocf_heartbeat_WAS — Manages a WebSphere Application Server instance

Synopsis

`WAS` [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for WAS. It manages a Websphere Application Server (WAS) as an HA resource.

Supported Parameters

`config`
: The WAS-configuration file. (optional, string, default `/usr/WebSphere/AppServer/config/server-cfg.xml`)

`port`
: The WAS-(snoop)-port-number. (optional, integer, default `9080`)

Example

The following is an example configuration for a WAS resource using the crm(8) shell:

```
primitive example_WAS ocf:heartbeat:WAS \
  params \

  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/WAS_(resource_agent)

Name

ocf_heartbeat_WinPopup — Sends an SMB notification message to selected hosts

Synopsis

`WinPopup` [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for WinPopup. It sends WinPopups message to a sysadmin's workstation whenever a takeover occurs.

Supported Parameters

`hostfile`
: The file containing the hosts to send WinPopup messages to. (optional, string, no default)

Example

The following is an example configuration for a WinPopup resource using the crm(8) shell:

```
primitive example_WinPopup ocf:heartbeat:WinPopup \
  params \
    hostfile=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/WinPopup_(resource_agent)

Name

ocf_heartbeat_Xen — Manages Xen unprivileged domains (DomUs)

Synopsis

Xen [start | stop | migrate_from | migrate_to | monitor | meta-data | validate-all]

Description

Resource Agent for the Xen Hypervisor. Manages Xen virtual machine instances by mapping cluster resource start and stop, to Xen create and shutdown, respectively. A note on names We will try to extract the name from the config file (the xmfile attribute). If you use a simple assignment statement, then you should be fine. Otherwise, if there's some python acrobacy involved such as dynamically assigning names depending on other variables, and we will try to detect this, then please set the name attribute. You should also do that if there is any chance of a pathological situation where a config file might be missing, for example if it resides on a shared storage. If all fails, we finally fall back to the instance id to preserve backward compatibility. Para-virtualized guests can also be migrated by enabling the meta_attribute allow-migrate.

Supported Parameters

`xmfile`
: Absolute path to the Xen control file, for this virtual machine. (optional, string, no default)

`name`
: Name of the virtual machine. (optional, string, no default)

`shutdown_timeout`
: The Xen agent will first try an orderly shutdown using xm shutdown. Should this not succeed within this timeout, the agent will escalate to xm destroy, forcibly killing the node. If this is not set, it will default to two-third of the stop action timeout. Setting this value to 0 forces an immediate destroy. (optional, boolean, no default)

`allow_mem_management`
: This parameter enables dynamic adjustment of memory for start and stop actions used for Dom0 and the DomUs. The default is to not adjust memory dynamically. (optional, boolean, default `0`)

`reserved_Dom0_memory`
: In case memory management is used, this parameter defines the minimum amount of memory to be reserved for the dom0. The default minimum memory is 512MB. (optional, string, default `512`)

`monitor_scripts`
: To additionally monitor services within the unprivileged domain, add this parameter with a list of scripts to monitor. NB: In this case make sure to set the start-delay of the monitor operation to at least the time it takes for the DomU to start all services. (optional, string, no default)

Example

The following is an example configuration for a Xen resource using the crm(8) shell:

```
primitive example_Xen ocf:heartbeat:Xen \
  params \
    xmfile=string \
  meta allow-migrate="true" \
  op monitor depth="0" timeout="30" interval="10"
```

See also

http://www.linux-ha.org/wiki/Xen_(resource_agent)

Name

ocf_heartbeat_Xinetd — Manages an Xinetd service

Synopsis

`Xinetd` [start | stop | restart | status | monitor | validate-all | meta-data]

Description

Resource script for Xinetd. It starts/stops services managed by xinetd. Note that the xinetd daemon itself must be running: we are not going to start it or stop it ourselves. Important: in case the services managed by the cluster are the only ones enabled, you should specify the -stayalive option for xinetd or it will exit on Heartbeat stop. Alternatively, you may enable some internal service such as echo.

Supported Parameters

`service`

The service name managed by xinetd. (optional, string, no default)

Example

The following is an example configuration for a Xinetd resource using the crm(8) shell:

```
primitive example_Xinetd ocf:heartbeat:Xinetd \
  params \
    service=string \
  op monitor depth="0" timeout="10" interval="10"
```

See also

http://www.linux-ha.org/wiki/Xinetd_(resource_agent)

Index

A

authkeys (Heartbeat configuration file), 15, 15

F

feedback
    contact information for this manual, x

H

ha.cf (Heartbeat configuration file), 15, 15
Heartbeat
    configuration, 15