

High Availability Messaging Solution **Using the AXIGEN Mail Server, Heartbeat and DRBD**



July 8, 2008



Copyright & Trademark Notices

This article applies to version 6.0 or higher of the AXIGEN Mail Server.

Notices

References in this publication to GECAD TECHNOLOGIES S.A. products, programs, or services do not imply that GECAD TECHNOLOGIES S.A. intends to make these available in all countries in which GECAD TECHNOLOGIES S.A. operates. Evaluation and verification of operation in conjunction with other products, except those expressly designated by GECAD TECHNOLOGIES S.A., are the user's responsibility. GECAD TECHNOLOGIES S.A. may have patents or pending patent applications covering subject matter in this document. Supplying this document does not give you any license to these patents. You can send license inquiries, in writing, to the GECAD TECHNOLOGIES S.A. sales department, using: sales@axigen.com.

Copyright Acknowledgement

(c) GECAD TECHNOLOGIES S.A. 2008. All rights reserved.

All rights reserved. This document is copyrighted and all rights are reserved by GECAD TECHNOLOGIES S.A. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage and retrieval system without the permission in writing from GECAD TECHNOLOGIES S.A.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. GECAD TECHNOLOGIES S.A. will not be responsible for any loss, costs or damages incurred due to the use of this documentation.

AXIGEN™ Mail Server is a SOFTWARE PRODUCT of GECAD TECHNOLOGIES S.A. This document is copyrighted and all rights are reserved by GECAD TECHNOLOGIES S.A. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage and retrieval system without prior permission in writing from GECAD TECHNOLOGIES S.A.

GECAD TECHNOLOGIES and AXIGEN™ are trademarks of GECAD TECHNOLOGIES S.A. Other company, product or service names may be trademarks or service marks of others.

GECAD TECHNOLOGIES S.A.

Address: 10A Dimitrie Pompei Blvd., Connect Business Center,
2nd fl., Bucharest 2, ROMANIA

Phone: +40-21-303-2080

Fax: +40-21-303-2081

Sales: sales@axigen.com

Technical support: support@axigen.com

Website: <http://www.axigen.com>

(c) Copyright GECAD TECHNOLOGIES S.A. 2008. All rights reserved.



Summary:

1. Prerequisites	4
1.1 Products Overview	4
1.2 Architecture.....	4
1.3 Hardware & Software Considerations.....	4
1.3.1 O.S. Platform	4
1.3.2 Cluster Nodes	4
1.3.3 Block Devices	5
1.3.4 Power Fencing Devices.....	5
2. DRBD	6
2.1 Installation.....	6
2.2 Configuring DRBD	6
3. AXIGEN Mail Server.....	8
3.1 AXIGEN Overview	8
3.2 AXIGEN Installation	8
4. Heartbeat	10
4.1 Heartbeat v1 Configuration Style (Legacy Approach).....	10
4.1.1 Prerequisites	10
4.1.2 Configuration	10
4.2 Heartbeat v2 Configuration Style (CRM Mode)	12
4.2.1 Cluster Wide Configuration Attributes.....	13
4.2.2 Resources	13
4.2.3 Constraints.....	15



1. Prerequisites

This article is intended as a general overview on designing HA solutions using DRBD and Heartbeat along with the AXIGEN Mail Server.

1.1 Products Overview

[AXIGEN Mail Server](#) – a carrier-class messaging solution, **truly innovative** in several respects, **particularly scalable and configurable**. This messaging solution offers the entire range of email services (SMTP, POP3, IMAP, WebMail), includes a List server, Logging, Reporting and FTP Backup modules and provides various, flexible administration options (including a central Web administration interface - WebAdmin).

[DRBD](#) (Distributed Replicated Block Device) – software solution for mirroring the contents of block devices between servers. DRBD eliminates the need of using a shared storage device between our cluster nodes running AXIGEN.

[Heartbeat](#) (part of the Linux-HA project) – cluster management framework for HA architectures.

1.2 Architecture

We will focus on a two node cluster architecture in an *ACTIVE/PASSIVE* configuration where a designated standby host will be available to relocate our resources in case of a failure encountered in the primary system.

For such a setup, we will assume that single points of failure are eliminated via redundant components such as multiple communication paths between nodes, independent power supplies for the two hosts etc.

Note: As Heartbeat provides two distinct versions of configuration, by using either v1 (legacy) or v2 (CRM mode configuration), both methods will be detailed as each of them presents a number of advantages that can suit different design schemes.

1.3 Hardware & Software Considerations

1.3.1 O.S. Platform

This article focuses on CentOS 5.x, but you can locate precompiled packages for various Linux distributions or make use of the source code available on their respective homepages for both Heartbeat and DRBD projects.

1.3.2 Cluster Nodes

We will consider two identical systems where hardware components are subject to local requirements for the solution performance and resource availability. From now on, we will refer to these nodes as:

Hostname/FQDN
primary.cluster.node
backup.cluster.node

Note: These hosts are considered accessible via the above mentioned identification format, having in place the proper name for IP resolution mechanisms such as DNS entries or updated/etc/hosts file.

Example:

```
/etc/hosts
192.168.8.221      primary.cluster.node
192.168.8.222      backup.cluster.node
```

On both nodes, multiple network adapters are required for DRBD/Heartbeat messages. This traffic should be managed by dedicated connections avoiding the use of actual production links. As a requirement for DRBD, you should consider a back-to-back approach on the communication link (For example, Gigabit Ethernet). If switches are to be used, you should ensure to have designed redundant links for communication purposes.

1.3.3 Block Devices

On both nodes, we will consider two similar lower level devices defined from now on as `/dev/sdb1`

Other types of block devices on the system can also be used as a lower level device, including here software RAID devices, LVM Logical Volume etc.

1.3.4 Power Fencing Devices

The STONITH module provides a number of plugins capable of managing a wide range of devices and methods for isolating cluster nodes in case of a failure. For a full reference on the available plugins, you can make use of the following command:

```
# /usr/sbin/stonith -L
apcmaster
apcmastersnmp
apcsmart
baytech
cyclades
external/ibmrsa
external/ibmrsa-telnet
external/ipmi
external/rackpdu
external/riloe
external/ssh
external/vmware
ibmhmc
meatware
null
nw_rpc100s
rcd_serial
rps10
ssh
suicide
wti_nps
```

Note: The above command assumes that the corresponding `heartbeat-stonith` package is installed on your system. Additional details are provided in the *Heartbeat configuration section*.

As a reference for our setup, we will consider a STONITH device (APC MasterSwitch PDU) accessible via the `apcmastersnmp` STONITH plugin. The coordinates for this sample fencing device are:

```
IP address – 192.168.8.225
SNMP port – 161
Community – private
```

2. DRBD - Distributed Replicated Block Device

2.1. Installation

DRBD is available via two packages providing complementary functionalities:

- DRBD kernel module - implements the core functionalities of DRBD
- DRBD package - containing all user space tools, configuration files, initscripts

For our CentOS platform the installation procedure can consist in using *yum* for this purpose:

```
# yum install drbd82 kmod-drbd82
```

Note: It may also be required to update the kernel tool too, as a dependency of the DRBD kernel module.

2.2 Configuring DRBD

A configuration example of DRBD is provided within the documentation files and can be used as a reference for the sections and parameters supported. For this reason you should copy this file in */etc/* and modify its contents according to your needs.

```
# cp /usr/share/doc/drbd82-*/drbd.conf /etc/
```

A basic sample for the DRBD configuration file consists of several sections:

- *global*
- *common*
- *resource*

All global parameters such as *usage-count* will be defined in the global section.

Example:

```
global {  
    usage-count no;  
}
```

The common section contains policies that will be inherited by all defined resources. For example, we will define here the replication protocol in use. Protocol C is a synchronous replication protocol where local write operations are considered completed only after having being confirmed on both nodes.

Example:

```
common {  
    protocol C;  
    syncer { rate 10M; }  
}
```

The resource section contains a collection of information that characterizes our replicated storage device.

Example:

```
resource axigenStorage {
  on primary.cluster.node {
    device /dev/drbd0;
    disk /dev/sdb1;
    address 192.168.8.221:7788;
    meta-disk internal;
  }

  on backup.cluster.node {
    device /dev/drbd0;
    disk /dev/sdb1;
    address 192.168.8.222:7788;
    meta-disk internal;
  }
}
```

In the above sample our resource has been defined as *axigenStorage* and all details about disk configuration, device definition, peer nodes and network configuration have been provided as parameters.

Besides these coordinates expressed in the DRBD configuration file, the status of this resource on each peer is also characterized by a role: *Primary* or *Secondary*.

Note:

Primary role – the DRBD device is available for read/write operations.

Secondary role – the DRBD device only receives updates from the peer node; other read/write operations on this device are not allowed.

Additional policies are available for definition, including peer authentication mechanisms, synchronization details and actions in case of split-brain detection.

Note: *A split-brain condition is encountered when the cluster nodes end up with having a different set of data. The DRBD documentation provides a number of mechanisms for dealing with and preventing split-brain conditions.*

3. AXIGEN Mail Server

3.1 AXIGEN Overview

Based on our requirements regarding data that needs to be kept synchronized between the two nodes, different parts of the AXIGEN storage and internal data could be placed as our *axigenStorage* resource defined under */etc/drbd.conf*.

On a Linux distribution, the AXIGEN internal structure is formed from:

- various initscripts and initscript configuration data
- /opt/axigen/bin/* - containing all AXIGEN binaries.
/var/opt/axigen/ - containing the main internal data of AXIGEN including:
- *run/* - configuration files
 - *log/* - log files
 - *filters/* - all user filters and also server/domain SMTP Routing and Advanced Message Rules
 - *domains/* - hosted domains
 - [...]

If locations such as the ones for the AXIGEN binaries and initscripts do not raise interest, as this data will only change during upgrade operations, our example will consider that the entire */var/opt/axigen/* location will be our resource (*axigenStorage*).

3.2 AXIGEN Installation

We will consider a fresh installation approach on both nodes. As our main resource, *axigenStorage* (made from the */var/opt/axigen/* directory), will be replicated on both nodes, our steps should comprise from enabling DRBD and then replicate the data between the two nodes.

For this process, we are required to ensure that the *drbd* module is loaded on both nodes:

```
# modprobe drbd
```

Next, we should initialize the device metadata, attach the local device and connect our resource with the counterpart peer:

```
# drbdadm create-md axigenStorage  
# drbdadm attach axigenStorage  
# drbdadm syncer axigenStorage  
# drbdadm connect axigenStorage
```

The initial synchronization process consists in running on our primary node (*primary.cluster.node*):

```
# drbdadm -- --overwrite-data-of-peer primary axigenStorage
```

Note: At this point, we have promoted the *axigenStorage* resource to a Primary role on our main machine.

Synchronization snapshot:

```
# cat /proc/drbd
```

```
version: 8.2.5 (api:88/proto:86-88)
GIT-hash: 9faf052fdae5ef0c61b4d03890e2d2eab550610c build by buildsvn@c5-i386-build, 2008-05-01 03:43:30
0: cs:SyncSource st:Secondary/Secondary ds:UpToDate/Inconsistent C r---
ns:136424 nr:0 dw:0 dr:136928 al:0 bm:53 lo:0 pe:5 ua:16 ap:0
[=====>.....] sync'ed: 53.1% (129952/266240)K
finish: 0:00:11 speed: 11,444 (10,480) K/sec
resync: used:1/31 hits:8519 misses:19 starving:0 dirty:0 changed:19
act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

From this moment on, every write operation on the virtual DRBD device block found on the main cluster node will also be replicated on the backup node where the *axigenStorage* resource has assigned a *Secondary* role.

On our main machine, *primary.cluster.node*, we can now create a filesystem on the virtual block device and mount */dev/drbd0* under */var/opt/axigen/*

Example:

```
mkfs.ext3 /dev/drbd0
mkdir /var/opt/axigen
mount /dev/drbd0 /var/opt/axigen
```

The installation and configuration procedures for the AXIGEN Mail Server are available under the *INSTALL* file provided within the package downloaded from the AXIGEN website.

Note: AXIGEN components that will not be replicated between the two nodes must be manually added on the backup server. This includes the AXIGEN binaries/initscripts/etc. Also, in our approach, the */var/opt/axigen/* location should be available as a mount point for */dev/drbd0*.

For the rest of the article, we will consider that:

- the AXIGEN service is configured and that it has been stopped and disabled from being initialized via *init*:

Example:

```
# chkconfig axigen off
# /etc/init.d/axigen stop
```

- all drbd components are stopped



4. Heartbeat

Based on the setup requirements, you may need to install the following packages:

```
heartbeat  
heartbeat-stonith  
heartbeat-gui  
heartbeat-pils
```

Example:

```
# yum install heartbeat heartbeat-gui
```

Depending on the actual architecture and resource management policies, such as monitoring operations, resource relocation and resource collocation rules, you can make use of the following:

Heartbeat v1 style configuration

The Heartbeat v1 configuration style is considered a legacy approach and comes with a number of characteristics:

- ease of installation and configuration
- limited to two node architectures
- does not provide resource health monitoring capabilities
- resources configuration data is extracted via the *haresources* file
- any changes of the cluster and configuration data must be manually replicated

Heartbeat v2 (CRM style configuration)

- eliminates the R1 two node architecture limit
- configuration details are now part of the CIB (Cluster Information Base) XML structure
- requires an understanding of this XML based configuration structure
- subsequent updates are automatically replicated on all cluster nodes
- resource monitoring is included and, in case of resource failure, restart operations or relocations policies can be defined

4.1 Heartbeat v1 Configuration Style (Legacy Approach)

4.1.1 Prerequisites

With this release of Heartbeat, we will consider the *drbdisk* resource agent as designated to handle our *axigenStorage* DRBD resource. The *drbdisk* only includes promotion/demotion capabilities; as a result, it is required that *init* is configured to automatically start the *drbd* service (Example of mechanisms for this purpose, depending on the OS platform: *chkconfig*, *sysvconfig*, *rc-update*, *update-rc.d*)

Example:

```
# chkconfig drbd on
```

4.1.2 Configuration

For the Heartbeat v1 release style, three configuration files are required:

```
# cp /usr/share/doc/heartbeat-*/ha.cf /etc/ha.d/  
# cp /usr/share/doc/heartbeat-*/haresources /etc/ha.d/  
# cp /usr/share/doc/heartbeat-*/authkeys /etc/ha.d/
```

- **ha.cf**

The main cluster configuration file contains details on the nodes taking part in the cluster system, communication paths (serial/Ethernet adapters) and methods (unicast/broadcast/multicast) based on the network architecture details. Multiple counters and thresholds are also available for configuration in this configuration file.

Example:

```
logfile /var/log/ha-log           # Log file
keepalive 2                       #
deadtime 30                       # Counters and
warntime 10                       # Threshold values
initdead 120                      #
udpport 694                       # UDP port
bcast eth1 eth2                  # communication details
auto_failback off                 # resource transition policy
stonith_host * apcmastersnmp 192.168.8.225 161 private # example stonith device
node primary.cluster.node backup.cluster.node          # cluster nodes
```

Authentication File - *authkeys*

- **authkeys**

The file structure is composed of an authentication mechanism expressed in the format of:

```
auth method-id
method-id method key
```

Example:

```
auth 1
1 sha1 secret
```

Given the nature of the data contained here, you should ensure that this file is not world readable:

```
# chmod 600 /etc/ha.d/authkeys
```

Heartbeat v1 configuration makes use of the *haresources* file for resource definition. The resources handled by Heartbeat using this configuration style will be defined as a group, one per line:

```
cluster-node resource1 resource2 ... resourceN
cluster-node - preferred system for running the group of resources
```

Note: The resources of such a group will be started from the left to the right and stopped from the right to the left.

Example:

```
primary.cluster.node drbddisk::axigenStorage Filesystem::/dev/drbd0::/var/opt/axigen::ext3
192.168.8.226 axigen
```

The order for starting the resources:

- promoting the *axigenStorage* resource to a primary role on the *primary.cluster.node* peer via *drbdisk*
- mount */dev/drbd0* under */var/opt/axigen/*, in our case we have assumed an ext3 filesystem
- start our virtual IP address resource
- startup AXIGEN via available *intiscript*

Note: As the *drbdisk* only implements promotion/demotion facilities, one needs to ensure that the *drbd* service is running on both nodes.

Example:

```
# /etc/init.d/drbd restart
```

By starting Heartbeat on both hosts, all our resources will be started on the primary cluster node and our designated backup system will be ready to relocate the resources in the event of a failure on our main node.

```
/etc/init.d/heartbeat start
```

Note: For resource health monitoring operations, external provisioning tools must be deployed.

Heartbeat should also be initialized by init during system startup. For this process, you can make use of a number of tools: *chkconfig*, *sysvconfig*, *rc-update*, *update-rc.d*

Example:

```
# chkconfig heartbeat on
```

4.2 Heartbeat v2 Configuration Style (CRM Mode)

Configuring Heartbeat v2, as detailed in section 1.2 supports some modifications as to reflect the CRM style method. In the *ha.cf* file we need to append a corresponding entry stating that we will in fact make use of the v2 release style configuration.

Example:

```
crm yes
```

The *haresources* file is no longer necessary as the v2 release makes use of a XML configuration file, usually located under:
/var/lib/heartbeat/crm/cib.xml

Note: This file should not be manually edited. Instead, XML editors and Heartbeat administration tools (such as *cibadmin*, or the *hb_gui* interface) should be used to generate and manage the cluster configuration details.

To facilitate the migration from the legacy mode to this new structure, Heartbeat also provides a tool for conversion to the CIB structure:

```
# /usr/lib/heartbeat/haresources2cib.py
```

The cluster configuration structure is divided into two main sections:

- configuration - containing information about cluster nodes, resources, location policies
- status - maintained by the cluster software itself

The configuration section requires managing operations from the administrator side and is composed from four parts:

```
<configuration>
  <crm_config/>
  <nodes/>
  <resources/>
  <constraints/>
</configuration>
```

4.2.1 Cluster Wide Configuration Attributes

Example:

```
<cib admin_epoch="0" epoch="1" num_updates="1">
  <configuration>
    <crm_config>
      <cluster_property_set id="sample-options">
        <attributes>
          <nvpair id="require_quorum" name="require_quorum" value="true"/>
          <nvpair id="symmetric_cluster" name="symetric_cluster" value="true"/>
          <nvpair id="suppress_cib_writes" name="suppress_cib_writes" value="true"/>
          <nvpair id="stonith_enabled" name="stonith_enabled" value="true"/>
          <nvpair id="stonith_action" name="stonith_action" value="reboot"/>
          [...]
        </attributes>
      </cluster_property_set>
    </crm_config>
```

We can configure attributes that control the overall behavior of our cluster system such as *symmetric_cluster* allowing resources to run on any node of the cluster, enabling STONITH devices and configuring the actions that should be triggered (reboot/poweroff).

The tuple formed via (*admin_epoch*, *epoch*, *num_updates*) provides details on the best configuration to be used by the system. The node with the highest tuple will replace the configuration details on the rest of the peers.

4.2.2 Resources

We will define a group of resources that make up all the elements required for running AXIGEN:

- virtual interface containing details such as IP address/broadcast address/netmask value
- device hosting our */var/opt/axigen/* data including information on the filesystem itself
- AXIGEN initscript

By default, the cluster does not monitor the resource health, this option being available by creating specific monitoring operations:

Example (of such an operation):

```
<operations>
  <op id="op-ip" name="monitor" interval="15s" timeout="5s"/>
</operations>
```

Using this data, we can define our AXIGEN group of resources as also described in the following example:

```
<group id="AXIGEN_Group">
  <primitive class="ocf" id="R_IP_192_168_8_226" provider="heartbeat" type="IPaddr">
    <instance_attributes id="ia-R_IP_192_168_8_226">
      <attributes>
        <nvpair id="attr-ipaddr:0" name="ip" value="192.168.8.226"/>
        <nvpair id="attr-broadcast:0" name="broadcast" value="192.168.8.255"/>
        <nvpair id="attr-cidr:0" name="cidr_netmask" value="24"/>
      </attributes>
    </instance_attributes>
    <operations>
      <op id="op-ip:0" name="monitor" interval="15s" timeout="5s"/>
    </operations>
  </primitive>
  <primitive class="ocf" id="R_Fileystem_AXIGEN" provider="heartbeat" type="Filesystem">
    <instance_attributes id="ia:R_Fileystem_AXIGEN">
      <attributes>
        <nvpair id="attr-dev:0" name="device" value="/dev/drbd0"/>
        <nvpair id="attr-filesys:0" name="directory" value="/var/opt/axigen"/>
        <nvpair id="attr-type:ext3_0" name="fstype" value="ext3"/>
      </attributes>
    </instance_attributes>
  </primitive>
  <primitive class="lsb" id="R_AXIGEN" provider="heartbeat" type="axigen">
    <operations>
      <op id="op-axigen:0" name="monitor" interval="30s" timeout="15s"/>
    </operations>
  </primitive>
</group>
```

The DRBD resource is configured via a complex *master_slave* object having the following sample format:

```
<master_slave id="ms:drbd-axigen">
  <meta_attributes id="ma:drbd-axigen">
    <attributes>
      <nvpair id="ma:drbd-axigen-0" name="notify" value="yes"/>
    </attributes>
  </meta_attributes>
  <primitive id="drbd-axigen:0" class="ocf" provider="heartbeat" type="drbd">
    <instance_attributes id="ia-drbd">
      <attributes>
        <nvpair id="ia-drbd:0" name="drbd_resource" value="axigenStorage"/>
      </attributes>
    </instance_attributes>
    <operations>
      <op id="mon-drbd:0" name="monitor" interval="29s" timeout="10s" role="Master"/>
      <op id="mon-drbd:1" name="monitor" interval="30s" timeout="10s" role="Slave"/>
    </operations>
  </primitive>
</master_slave>
```

Note: The use of drbd as a RA requires disabling the drbd service from being started via init (Ex: chkconfig, sysvconfig, rc-update, update-rc.d).

Example:

```
# chkconfig drbd off
```

- **Configuring a STONITH device**

For the APC MasterSwitch device used in our scenario, a new object will be created using our predefined configuration details:

```
<clone id="fencing">
  <instance_attributes id="ia-fencing">
    <attributes>
      <nvpair id="nv-clone-max" name="clone_max" value="2"/>
      <nvpair id="nv-clone-node-max" name="clone_node_max" value="1"/>
    </attributes>
  </instance_attributes>
  <primitive id="child_fencing" class="stonith" type="apcmastersnmp" provider="heartbeat">
    <operations>
      <op id="mon-fencing:0" name="monitor" interval="5s" timeout="20s" prereq="nothing"/>
      <op id="mon-fencing:1" name="start" timeout="20s" prereq="nothing"/>
    </operations>
    <instance_attributes id="ia-child_fencing">
      <attributes>
        <nvpair id="attr-fencing-ip" name="ipaddr" value="192.168.8.225"/>
        <nvpair id="attr-fencing-port" name="port" value="161"/>
        <nvpair id="attr-fencing-community" name="community" value="private"/>
      </attributes>
    </instance_attributes>
  </primitive>
</clone>
```

4.2.3 Constraints

- **Resource location**

For our particular setup, we will assign the *primary.cluster.node* as the main location for running our resource.

Example:

```
<rsc_location id="rl:AXIGEN_Group" rsc="AXIGEN_Group">
  <rule id="rl:main_AXIGEN_Group" score="500">
    <expression id="rl:1_AXIGEN_Group" operation="eq" attribute="#uname"
value="primary.cluster.node"/>
  </rule>
</rsc_location>
```

Additional order and collocation policies are available. In this particular case, we need to express the need of running the AXIGEN service only on a node for which the DRBD resource is designated as having a primary role.

Example:

```
<rsc_order id="AXIGEN_Group_after_drbd" from="AXIGEN_Group" action="start" to="ms:drbd-axigen" to_action="promote" type="after"/>
<rsc_colocation id="axigenStorage_on_drbd" to="ms:drbd-axigen" to_role="master" from="AXIGEN_Group" score="INFINITY"/>
```

Starting Heartbeat and enabling this service to be started automatically by init can be enforced via:

```
# /etc/init.d/heartbeat start
# chkconfig heartbeat on
```

- **Monitoring the CRM Style Cluster System With crm_mon**

```
# crm_mon
```

```
=====
Last updated: Mon Jun 2 00:00:45 2008
Current DC: backup.cluster.node (b6000d3e-958b-4126-a7c2-3cac039f8f08)
2 Nodes configured.
3 Resources configured.
=====
```

```
Node:backup.cluster.node (b6000d3e-958b-4126-a7c2-3cac039f8f08): online
Node: primary.cluster.node (69a34e96-592a-4462-8943-972af6ec6c50): online
```

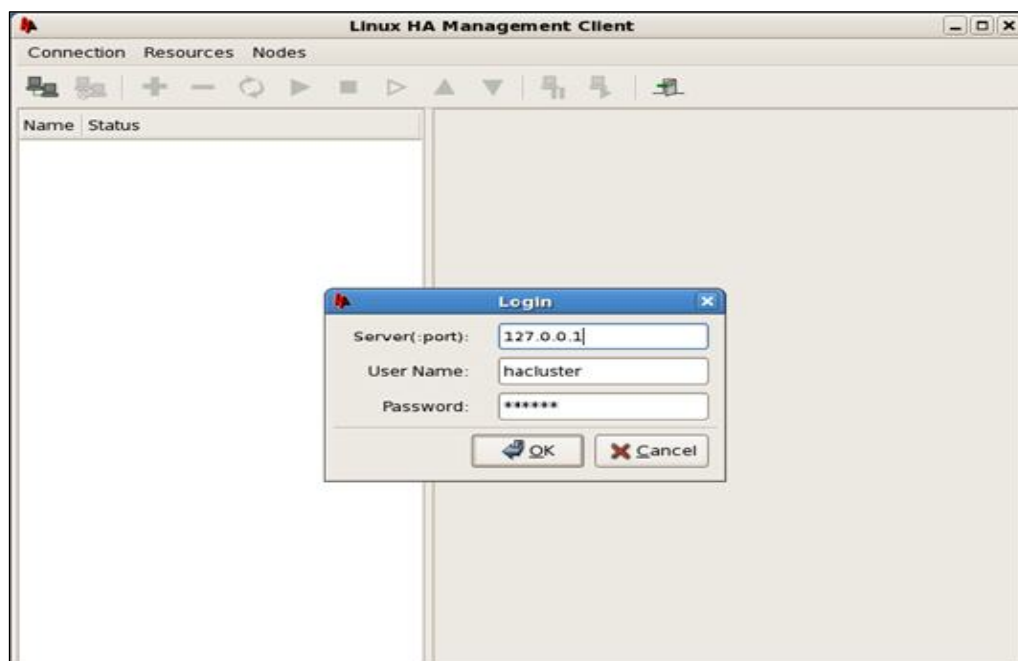
```
Resource Group: AXIGEN_Group
R_IP_192_168_8_226 (heartbeat::ocf:IPAddr): Started primary.cluster.node
R_Filesystem_AXIGEN (heartbeat::ocf:Filesystem): Started primary.cluster.node
R_AXIGEN (lsb:axigen): Started primary.cluster.node
Master/Slave Set: ms:drbd-axigen
drbd-axigen:0:0 (heartbeat::ocf:drbd): Master primary.cluster.node
drbd-axigen:0:1 (heartbeat::ocf:drbd): Started backup.cluster.node
Clone Set: fencing
child_fencing:0 (stonith:apcmastersnmp): Started backup.cluster.node
child_fencing:1 (stonith:apcmastersnmp): Started primary.cluster.node
```

- **hb_gui**

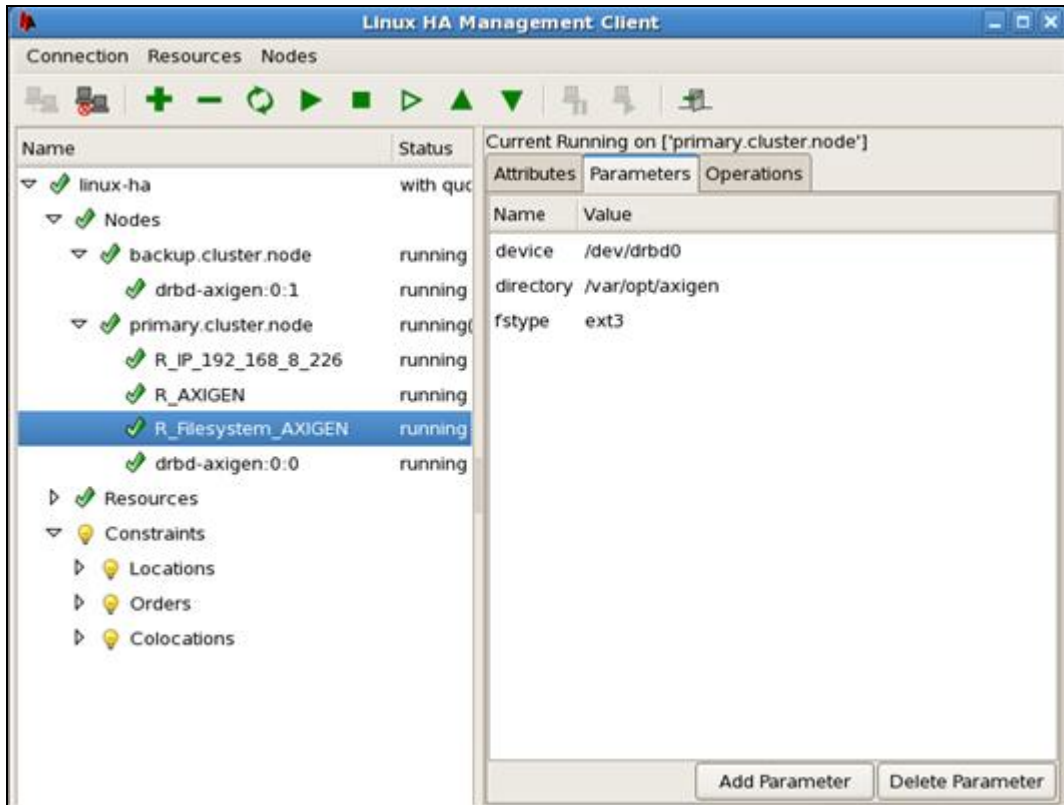
An alternative for creating the cluster CIB configuration consists in making use of the provided *hb_gui* tool. Note that the two methods of configuring the cluster resources (manually vs *hb_gui*) are considered to be incompatible.

Examples of hb_gui configuration contexts

Login screen:



Normal operation:



Resource relocation example in case of primary system failure:

