Introduction and History
In its 20+ year history, PowerHA SystemMirror (HACMP) has lacked a consistent, well-documented, thoroughly supported CLI. Configuration and control of PowerHA SystemMirror has been left to SMIT. Attempts were made at graphical utilities which were mostly wrappers for the SMIT functions to perform the task at hand. For the command-line oriented administrator, looking to automate operations, including configuration tasks, the best that could be done was to find the command in SMIT and hope that all the right environment variables were set.

In 2006, HACMP 5.4 introduced a utility called clvt that provided a glimmer of hope. Developed as a tool for use by the Smart Assists to perform configuration tasks at the command level, its capabilities were limited to those needed by the Smart Assists. Very little was done to update the clvt utility beyond that needed for the Smart Assists in later releases.

This lack of a robust CLI changes with the PowerHA SystemMirror 7.1 plug-in for IBM System Director. Now all of PowerHA SystemMirror tasks need to be executable using a command level utility, not just the ones required by the Smart Assist framework. A substantial amount of development effort was expended to “polish” up the interface and complete the capabilities. Although a huge improvement has been made in 7.1, be sure to check the man page for the task you wish to perform because not all tasks are there, yet. And of course, be sure to test thoroughly, prior to final deployment.

The cluster used to develop this paper is running on a pair of LPARs, one on each of two Power 570 systems. The storage and network are provided via VIO, that is, the network traffic is bridged and the disks are VSCSI. One Ethernet adapter will be used for the cluster. The repository disk has been identified. See the figures below for more details.
Figure 1. The cluster infrastructure

Figure 2. The resource group configuration
Basics

There is online help. The man page for clmgr is available with PowerHA SystemMirror 7.1 SP3. Prior to that code level, the man page is available using man clvt.

If you know what you want to do, but you aren’t quite sure what options are available or their format, simply type as much of the command as you know, followed by a \texttt{-h}. This will produce only the help relevant to the typed command. For example, if you want to activate a node, you can type \texttt{clmgr on node -h} which produces the following output:

\begin{verbatim}
# clmgr start node -h
# Available options for "clvt start node":
start_node
<_NAME>
WHEN
MANAGE
BROADCAST
CLINFO
FORCE
FIX
TIMEOUT
\end{verbatim}

The output above provides all the options for starting a node.

The format of the command is:
\texttt{clmgr <action> <class> [<name>] [<attributes>]}

The action is what you want to do to the class (like node or resource group). The name parameter identifies the specific class object.

The most common actions are:
add, query, delete
online, offline
move
sync

The most common classes are:
cluster
node
resource_group
service_ip
application_controller

Specify the \texttt{-a} parameter when using the query action to filter out only the desired attribute.
Specify the `–v` parameter when using the query action to produce verbose output. This will also produce verbose output for all objects of the specified class. Verbose output is the default when specifying a named object on the query, that is, no `–v` is necessary.

**The tasks to be performed – using only clmgr**

- Build a two-node PowerHA SystemMirror 7.1 cluster
- Define persistent addresses for each node
- Define two service addresses for each resource group
- Define two application controllers, one for each resource group
- Define two resource groups, with resources, each having a different home node
- Synchronize the cluster definition
- Start cluster services on each node
- Verify resource groups are online as expected
- Move a resource group from one node to another
- Stop cluster services, moving resource groups to the other node
- Restart cluster services on down node
- Move resource groups back to home nodes
- Generate a snapshot of this validated cluster configuration

**Step 1 – Build the cluster**

The command:

```
# clmgr add cluster pha_cluster repository=hdisk2 nodes=node1,node2
```

The result:

```
# cltopinfo
Cluster Name: pha_cluster
Cluster Connection Authentication Mode: Standard
Cluster Message Authentication Mode: None
Cluster Message Encryption: None
Use Persistent Labels for Communication: No
Repository Disk: hdisk2
Cluster IP Address:
There are 2 node(s) and 1 network(s) defined
NODE node1:
    Network net_ether_01
        node1  10.6.51.121
NODE node2:
    Network net_ether_01
        node2  10.6.51.221

No resource groups defined
```
Step 2 – Define Persistent Addresses for each node

The command:
```bash
# clmgr add persistent_ip 192.168.3.1 network=net_ether_01 node=node1
# clmgr a pe 192.168.3.2 network=net_ether_01 node=node2
```

The result:
```bash
# clmgr -v q pe
NAME="node1admin"
IPADDR="192.168.3.1"
NODE="node1"
SITE=""
NETWORK="net_ether_01"
INTERFACE=""
NETTYPE="ether"
TYPE="persistent"
ATTR="public"
GLOBALNAME=""
HADDR=""

NAME="node2admin"
IPADDR="192.168.3.2"
NODE="node2"
SITE=""
NETWORK="net_ether_01"
INTERFACE=""
NETTYPE="ether"
TYPE="persistent"
ATTR="public"
GLOBALNAME=""
HADDR=""
```

Step 3 – Define two Service Addresses, one for each node’s resource group

The command:
```bash
# clmgr add service_ip appAsvc network=net_ether_01 netmask=255.255.255.0
# clmgr a se appBsvc network=net_ether_01 netmask=255.255.255.0
```

The result:
```bash
# clmgr -v q se
NAME="appAsvc"
IPADDR="192.168.3.10"
NODE=""
SITE="ignore"
NETWORK="net_ether_01"
INTERFACE=""
NETTYPE="ether"
TYPE="service"
ATTR="public"
GLOBALNAME=""
HWADDR=""
```
Step 4 – Define two Application Controllers, one for each node’s resource group

The command:
```
# clmgr add application_controller appActrl \
startscript=/lab/config/startA stopscript=/lab/config/stopA
# clmgr a ac appBctrl startscript=/lab/config/startB \
stopscript=/lab/config/stopB
```

The result:
```
# clmgr -v q ac
NAME="appActrl"
MONITORS=""
STARTSCRIPT="/lab/config/startA"
STOPSCRIPT="/lab/config/stopA"

NAME="appBctrl"
MONITORS=""
STARTSCRIPT="/lab/config/startB"
STOPSCRIPT="/lab/config/stopB"
```

Step 5 – Define two resource groups with resources, one for each node

The command:
```
# clmgr add resource_group appAgroup nodes=node1,node2 startup=OHN \ 
fallover=FNPN fallback=NFB service_label=appAsvc \ 
applications=appActrl volume_group=vgA fs_before_ipaddr=true

# clmgr add resource_group appBgroup nodes=node2,node1 startup=OHN \ 
fallover=FNPN fallback=NFB service_label=appBsvc \ 
applications=appBctrl volume_group=vgB
```

Note: the policy parameters must be uppercase.

The result:
```
# cltopinfo
Cluster Name: pha_cluster
Cluster Connection Authentication Mode: Standard
Cluster Message Authentication Mode: None
Cluster Message Encryption: None
```
Use Persistent Labels for Communication: No
Repository Disk: hdisk2
Cluster IP Address:
There are 2 node(s) and 1 network(s) defined
NODE node1:
  Network net_ether_01
    appBsvc 192.168.3.20
    appAsvc 192.168.3.10
    node1  10.6.51.121
NODE node2:
  Network net_ether_01
    appBsvc 192.168.3.20
    appAsvc 192.168.3.10
    node2  10.6.51.221
Resource Group appAgroup
  Startup Policy   Online On Home Node Only
  Fallover Policy  Fallover To Next Priority Node In The List
  Fallback Policy  Never Fallback
  Participating Nodes      node1 node2
  Service IP Label                 appAsvc
Resource Group appBgroup
  Startup Policy   Online On Home Node Only
  Fallover Policy  Fallover To Next Priority Node In The List
  Fallback Policy  Never Fallback
  Participating Nodes      node2 node1
  Service IP Label                 appBsvc
Step 6 – Synchronize the cluster definitions

The command:

```
# clmgr sync cluster
```

The result (some output was pruned):

```
# clmgr sync cluster
Verification to be performed on the following:
  Cluster Topology
  Cluster Resources

Retrieving data from available cluster nodes. This could take a few minutes.

  Start data collection on node node1
  Start data collection on node node2

Verifying Cluster Topology...

  Completed 10 percent of the verification checks

  node1                                net_ether_01
  node2                                net_ether_01

Verifying Cluster Resources...

  Completed 40 percent of the verification checks

  appActrl                          appAgroup
  appBctrl                          appBgroup

  Completed 100 percent of the verification checks

Remember to redo automatic error notification if configuration has changed.

Verification has completed normally.

Committing any changes, as required, to all available nodes...
Adding any necessary PowerHA SystemMirror for AIX entries to
/etc/inittab and /etc/rc.net for IP Address Takeover on node node1.
Adding any necessary PowerHA SystemMirror for AIX entries to
/etc/inittab and /etc/rc.net for IP Address Takeover on node node2.

Verification has completed normally.

WARNING: Multiple communication interfaces are recommended for networks that
use IP aliasing in order to prevent the communication interface from
becoming a single point of failure. There are fewer than the recommended
number of communication interfaces defined on the following node(s) for
the given network(s):

<table>
<thead>
<tr>
<th>Node:</th>
<th>Network:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WARNING: Network option "nonlocsrcroute" is set to 0 and will be set to 1 on during HACMP startup on the following nodes:

   node1
   node2

WARNING: Application monitors are required for detecting application failures in order for HACMP to recover from them. Application monitors are started by HACMP when the resource group in which they participate is activated. The following application(s), shown with their associated resource group, do not have an application monitor configured:

- Step 7 – Check state of cluster services
  The command:
  
  # clmgr -a state query cluster
  # clmgr -cv -a name,state query node

  The result:
  
  # clmgr -a state query cluster
  STATE="OFFLINE"
  # clmgr -cv -a name,state query node
  # NAME:STATE
  node1:OFFLINE
  node2:OFFLINE

- Step 8 – Start cluster services on both nodes
  The command:
  
  # clmgr start cluster

  The result:
  
  # clmgr -a state q cluster
  STATE="STABLE"
  # clmgr -cv -a name,state,raw_state q node
  # NAME:STATE:RAW_STATE
  node1:NORMAL:ST_STABLE
  node2:NORMAL:ST_STABLE

- Step 9 – Check the status of the resource groups
  The command:
  
  # clmgr -cv -a name,state,current_node q rg

  The result:
  
  # clmgr -cv -a name,state,current_node q rg
  # NAME:STATE:CURRENT_NODE
  appAgroup:ONLINE:node1
  appBgroup:ONLINE:node2
At this point, checking the resources in the resource group must be done using AIX commands. Although there is state information for many of the cluster objects via clmgr, it does not provide state information for objects like the service address or application controller scripts.

Let’s move on to some testing.

**Step 10 – Move the appAgroup resource group to node2**

The command:

```
# clmgr mv rg appAgroup node=node2
```

The result:

```
# clmgr mv rg appAgroup node=node2
Attempting to move resource group appAgroup to node node2.

Waiting for the cluster to process the resource group movement request....

Waiting for the cluster to stabilize....

Broadcast message from root@node1 (tty) at 16:29:04 ...

appA stopping

.......... 

Resource group movement successful. 
Resource group appAgroup is online on node node2.

Cluster Name: pha_cluster

Resource Group Name: appAgroup
Primary instance(s):
The following node temporarily has the highest priority for this instance:
node2, user-requested rg_move performed on Wed Dec  1 16:28:56 2010

<table>
<thead>
<tr>
<th>Node</th>
<th>Group State</th>
</tr>
</thead>
<tbody>
<tr>
<td>node1</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>node2</td>
<td>ONLINE</td>
</tr>
</tbody>
</table>

Resource Group Name: appBgroup

<table>
<thead>
<tr>
<th>Node</th>
<th>Group State</th>
</tr>
</thead>
<tbody>
<tr>
<td>node2</td>
<td>ONLINE</td>
</tr>
<tr>
<td>node1</td>
<td>OFFLINE</td>
</tr>
</tbody>
</table>
Step 11 – Stop cluster services on node2 with “Move Resource Groups” option

The command:

```
# clmgr stop node node2 manage=move
```

The result:
```
# clmgr -cv -a name,state,raw_state q node
# NAME:STATE:RAW_STATE
node1:NORMAL:ST_STABLE
node2:OFFLINE:ST_INIT
```
```
# clmgr -cv -a name,state,current_node q rg
# NAME:STATE:CURRENT_NODE
appAgroup:ONLINE:node1
appBgroup:ONLINE:node1
```

Step 12 – Restart cluster services on node2

The command (using online alias instead of start):

```
# clmgr on node node2
```

The result:
```
# clmgr -cv -a name,state,raw_state q node
# NAME:STATE:RAW_STATE
node1:NORMAL:ST_STABLE
node2:NORMAL:ST_STABLE
```
```
# clmgr -cv -a name,state,current_node q rg
# NAME:STATE:CURRENT_NODE
appAgroup:ONLINE:node1
appBgroup:ONLINE:node1
```
Step 13 – Move the appBgroup resource group back to node2
The command:
# clmgr mv rg appBgroup node=node2

The result:
# clmgr -cv -a name,state,current_node q rg
appAgroup:ONLINE:node1
appBgroup:ONLINE:node2

Step 14 – Make a snapshot of all this good work
The command:
# clmgr mk sn clmgr_snap description="Snap of clmgr example cluster"

The result:
# clmgr mk sn clmgr_snap description="Snap of clmgr example cluster"
clsnapshot: Creating file
/usr/es/sbin/cluster.snapshots/clmgr_snap.odm.
clsnapshot: Creating file
/usr/es/sbin/cluster.snapshots/clmgr_snap.info.
clsnapshot: Executing clsnapshotinfo command on node: node1...
clsnapshot: Executing clsnapshotinfo command on node: node2...
clsnapshot: Succeeded creating Cluster Snapshot: clmgr_snap
**Usability features**

As shown above, many aliases or shortcuts are available in the utility. Here are some examples from the man page:

**ACTION** a verb describing the operation to be performed
- add (Aliases: a)
- query (Aliases: q, ls, get)
- modify (Aliases: mod, ch, set)
- delete (Aliases: de, rm, er)
- online (Aliases: on, start)
- offline (Aliases: off, stop)
- move (Aliases: mv)

**CLASS** the type of object upon which the ACTION will be performed.
- cluster (Aliases: cl)
- site (Aliases: si)
- node (Aliases: no)
- interface (Aliases: in, if)
- network (Aliases: ne, nw)
- resource_group (Aliases: rg)
- service_ip (Aliases: se)
- persistent_ip (Aliases: pe)
- application_controller (Aliases: ac, app)
- application_monitor (Aliases: am, mon)

Misspell something or get a parameter wrong, it tells you clearly what is wrong.

```bash
# clmgr add service_ip appAsvc network=net_ether01 netmask=255.255.255.0
ERROR: failed to create "appAsvc".
```

Network net_ether01 cannot be found in the configuration.
Do you want to do a query but you’re not sure what object names were used in PowerHA SystemMirror for a given class? Provide any name and consult the very user-friendly error message.

```bash
# clmgr q rg what_group
ERROR: "what_group" does not appear to exist!
```

Available Resource Groups:

```
   appAgroup
   appBgroup
```

Notice, the problem value is clearly shown, *and* all possible correct values are also shown.

To find all resource groups that have the option to mount filesystems before the IP addresses are acquired:

```bash
# clmgr q rg fs_before_ipaddr=true
appAgroup
```
Troubleshooting and logging
Troubleshooting a utility like clmgr is arguably not all that important. If it doesn’t work, switch to SMIT and get the task done. But, if you are trying to get clmgr to do something and you are planning to use it extensively in the future, it does produce some very good log information. The logging is done to /var/hacmp/log/clutils.log (or wherever clutils.log is kept on your system).

Further, the clutils.log contains the output of the action, so if the output has overrun the scroll buffer, it’s all safely in the log.

An excerpt of the log is shown here. This is the result of starting cluster services on node2.

CLMGR USER (9153:10354698:5177392): ::root::system
CLMGR COMMAND (9153:7471358:10354698): clmgr -T 9153 online node node2
CLMGR ACTUAL (9153:7471358:10354698): start_node node2
CLMGR RETURN (9153:7471358:10354698): 0
CLMGR STDERR -- BEGIN (9153:10354698:5177392): Wed Dec 1 17:00:10 CET 2010
CLMGR STDERR -- END (9153:10354698:5177392): Wed Dec 1 17:00:10 CET 2010
CLMGR ENDED (9153:10354698:5177392): Wed Dec 1 17:00:10 CET 2010

Some useful bits of information are contained in the output, including the user executing the command, the command entered by the user (CLMGR COMMAND) and the actual command executed. The return code is also displayed, which may be helpful if a failure had occurred.

And of course, this is just the beginning of troubleshooting. If the problem is with the executed command, not with clmgr, then other logs or the AIX configuration will have to be consulted. For example, the failure to configure the repository disk during the initial synchronization will not be solved by looking in clutils.log. That will most likely be solved using syslog.
Conclusion: clmgr is a viable cli

While earlier versions of clvt were limited, the updated clmgr is ready for mainstream use. As for the common administrative tasks in PowerHA SystemMirror, I think I’ve shown that this utility covers them all. What has been demonstrated in this paper is that there is now a viable CLI utility for PowerHA SystemMirror administration.

Many more features are available than were shown in this paper. For example, to execute a DARE operation, one could change an attribute (like adding a volume group to a resource group) using the modify action and then synchronize as shown earlier.

Additionally, since this is the utility used by the IBM System Director Plug-in for PowerHA SystemMirror, you are assured that the underlying infrastructure for the Plug-in functions well.

In conclusion, will this be the right tool for every administrative situation? Is this going to replace SMIT for PowerHA SystemMirror administration? The answer to both questions is “no”. But as you’ve seen, tasks like starting or stopping services, or performing a synchronization can be simplified requiring only a few keystrokes.

Bill Miller is currently a course developer and instructor in IBM Systems and Technology Group. He focuses on PowerHA SystemMirror courseware. In addition to his work in IBM Training, he has held various positions in IBM including development and consulting services. He can be contacted at bmiller4@us.ibm.com. He would like to thank Gary Lowther and Alex Abderrazag for their reviews and input to this paper.

Disclaimer: All of the commands shown in this paper worked as shown during the development of the paper. Neither IBM nor the author make any guarantee that the same or similar results will be seen in other cluster environments. Use this document as a reference and guide to using the clmgr utility.