Chapter 19
-
MC/ServiceGuard

INDEX

Introduction **3**
Example for a Cluster Configuration (Hardware) 3
Example for a Package Configuration 4
Useful Procedures and Commands **5**
Check if ServiceGuard is installed 5
Check if ServiceGuard is active 6
View the status of the Cluster 6
Starting the Cluster 7
Halting the Cluster 7
Starting a Node 8
Halting a Node 8
Starting Packages 8
Stopping Packages 9
Modifying the Switching Attributes of a Package 9
Create configuration templates 9
Validating a Cluster Configuration 10
Generation and Distribution of a Cluster Configuration 10
LVM Modifications of Cluster Volume Groups 11
Troubleshooting **13**
Where to find Logfiles 13
What happens during Package Start 13
What happens during Package Stop 14
Cluster Lock Disk Initialization 14
What about ServiceGuard TOCs 16
How to track down Networking Problems 16
Changing ServiceGuard’s Log Level 20
Overview of ServiceGuard Files **23**
MC/ServiceGuard Commands Overview **24**
Info Commands 24
Config Commands 24
Admin Commands 25
ServiceGuard Common issues **26**
Problems during cmcheckconf / cmapplyconf 26
SGeSAP Common issues 29
Additional Information **32**

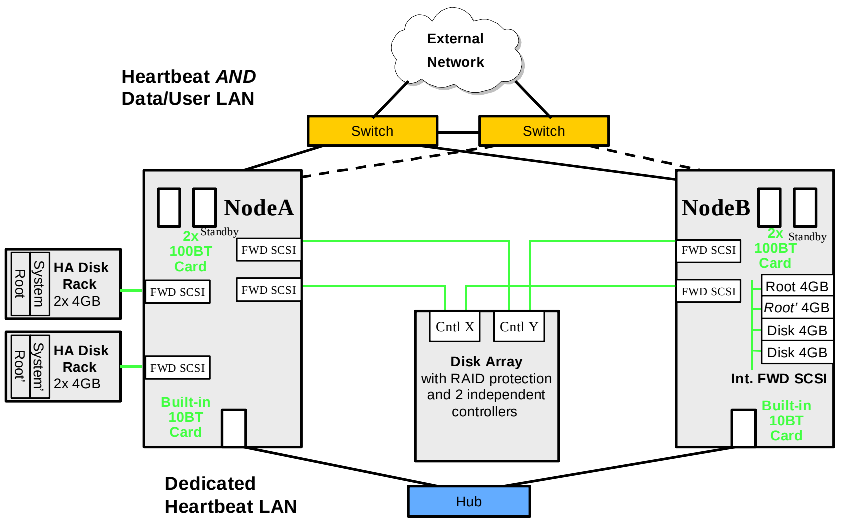
This chapter wants to briefly introduce High Availalability solutions using MC/ServiceGuard and how to setup, maintain and troubleshoot them. It is *not* suppoosed to replace the official documentation, especially the manual *Managing MC/ServiceGuard* (B3936-90065), downloadable from http://docs.hp.com/hpux/ha/index.html#ServiceGuard. In fact, several parts are extracted and concentrated from that and from other internal and external documents.

Introduction

MC/ServiceGuard allows you to create high availability clusters of HP 9000 Series 800 computers. A high availability computer system allows application services to continue in spite of a hardware or software failure. Highly available systems protect users from software failures as well as from failure of a system processing unit (SPU), disk, or local area network (LAN) component. In the event that one component fails, the redundant component takes over. MC/ServiceGuard and other high availability subsystems coordinate the transfer between components.

The central goal in cluster design is to **prevent any Single Point of Failure**. To make an application highly available, it must not rely on any single resource. This of course requires hardware redundancy in conjunction with a carefully designed concept.
An MC/ServiceGuard **cluster** is a networked grouping of HP 9000 series 800 servers (host systems known as nodes) having sufficient redundancy of software and hardware that a single point of failure will not significantly disrupt service. Application services (individual HP-UX processes) are grouped together in **packages**; in the event of a single service, node, network, or other resource failure, MC/ServiceGuard can automatically transfer control of the package to another node within the cluster, allowing services to remain available with minimal interruption.

Example for a Cluster Configuration (Hardware)



Key features of such a hardware configurations are:

- Redundant networking connectivity (more than one LAN interface).
- Redundant networking infrastructure (in this case one redundant network using two switches and one dedicated network for inter-node communication).
- Redundant mass storage access (more than one SCSI or FC interface).
- Redundant mass storage infrastructure (e.g. using hardware features of disk arrays or software solutions like MirrorDisk/UX or Veritas Volume Manager).
- Redundant power supply (using UPS, more than one power circuit).

Note that both nodes are physically connected to the same disk array. However, only one node at a time may access the data for a given group of disks. RAID technology provides redundancy in case of disk failures. In addition, two data buses are shown for the disk array that is connected to Node A and Node N, each of them connected to a different disk array controller of sufficient redundancy.

Note that the network hardware is cabled to provide redundant LAN interfaces on each node. MC/ServiceGuard uses TCP/IP network services for reliable communication among nodes in the cluster, including the transmission of heartbeat messages, signals from each functioning node which are central to the operation of the cluster. TCP/IP services also are used for other types of inter-node communication.

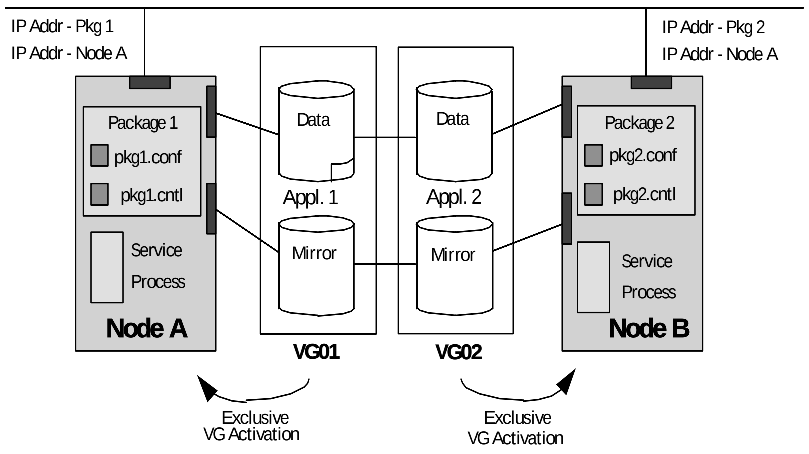
Another important point in cluster design are power connections. In order to remove all single points of failure from the cluster, you should provide as many separate power circuits as needed to prevent a single point of failure of your nodes, disks and disk mirrors. Each power circuit should be protected by an uninterruptible power source.

MC/ServiceGuard is designed to work in conjunction with other high availability products, such as MirrorDisk/UX or Veritas Volume Manager, which provide disk redundancy to eliminate single points of failure in the disk subsystem; Event Monitoring Service (EMS), which lets you monitor and detect failures that are not directly handled by MC/ServiceGuard; disk arrays, which use various RAID levels for data protection; and HP-supported uninterruptible power supplies (UPS), such as HP PowerTrust, which eliminates failures related to power outage. These products are highly recommended along with MC/ServiceGuard to provide the greatest degree of availability.

Example for a Package Configuration

Under normal conditions, a fully operating MC/ServiceGuard cluster simply monitors the health of the cluster's components while the packages are running on individual nodes. Any host system running in the MC/ServiceGuard cluster is called an active node. When you create the package, you specify a primary node and one or more adoptive nodes. When a node or its network communications fails, MC/ServiceGuard can transfer control of the package to the next available adoptive node.

After this transfer, the package typically remains on the adoptive node as long the adoptive node continues running. If you wish, however, you can configure the package to return to its primary node as soon as the primary node comes back online. Alternatively, you may manually transfer control of the package back to the primary node at the appropriate time.



Key feature of such a package configuration include:

- Each package has exclusive access to at least one LVM volume group or VxVM device group. To allow multiple nodes to share applications they all need to have access to the disk drives on which the applications reside. This is accomplished by HP-UX LVM feature called a **Cluster Volume Group.** Cluster VGs are Volume Groups which had been initially configured on one node and then imported to other nodes which share the same bus.
  Veritas also provides cluster disk groups via VxVM and CVM (Cluster Volume Manager). This is supported as of MC/ServiceGuard revision 11.09 with PHSS_23511. See also the Whitepaper *Integrating VERITAS VolumeManager (VxVM) with MC/ServiceGuard A.11.09*, http://docs.hp.com/hpux/pdf/B3936-90048.pdf. Beginning with MC/Serviceguard revision 11.13 full support is included without patching.
- Each package may provide one or more so called **relocatable IP addresses**. These adresses are always configured on that node that curretly runs the package. Using them from the outer world provides a some kind of transparent view to the cluster. Talking to a relocatable address means talking to the node that runs the package, whatever node of the cluster that may be. In case of a package failover the address fails over to the adoptive node, too.

Useful Procedures and Commands

In the following section some useful procedures and commands are listed. Please not that this is only a brief overview. The complete description of all used commands and all their command line options can be retrieved from their reference manual pages, section 1M.

Check if ServiceGuard is installed

Use `swlist` to check if a ServiceGuard installation is present on that machine:

```
# swlist ServiceGuard
...
```

```
     # ServiceGuard                         A.11.14         Service Guard
       ServiceGuard.CM-SG                   A.11.14         Service Guard
```

You may use the command `cmquerycl` to search existing ServiceGuard clusters in the Network. Please note that this command may take its time.

```
# cmquerycl [-v]
```

Check if ServiceGuard is active

If a node is active as part of a ServiceGuard cluster then at least the process cmcld is running, which is the central cluster daemon. This process must not be killed from the command line!

```
# ps –ef | grep cmcld
```

View the status of the Cluster

To view information about the current high availability cluster use `cmviewcl`. It displays the current status information of a cluster. Output can be displayed for the whole cluster or it may be limited to particular nodes or packages.

```
# cmviewcl –v

CLUSTER      STATUS
GP2          up

  NODE         STATUS       STATE
  ux_aps_a     up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           8/8/1/0      lan1
    STANDBY      up           8/12/1/0     lan3
    PRIMARY      up           10/12/6      lan0

  NODE         STATUS       STATE
  ux_dbs_a     up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           8/8/1/0      lan1
    PRIMARY      up           10/12/6      lan0
    STANDBY      up           8/12/1/0     lan3

    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    dbci         up           running      enabled     ux_dbs_a

      Script_Parameters:
      ITEM         NAME                         STATUS     MAX_RESTARTS   RESTARTS
      Subnet       192.125.195.0                up

      Node_Switching_Parameters:
      NODE_TYPE    STATUS       SWITCHING    NAME
      Primary      up           enabled      ux_dbs_a     (current)
      Alternate    up           enabled      ux_aps_a
```

Starting the Cluster

The command `cmruncl` causes all nodes in a configured cluster or all nodes specified to start their cluster daemons and form a new cluster. This command should only be run when the cluster is not active on any of the configured nodes.

If a cluster (a cluster daemon) is already running on a subset of nodes, the `cmrunnode` command needs to be used to start the remaining nodes and force them to join the existing cluster. Automatic cluster start during reboot can be configured in file `/etc/rc.config.d/cmcluster.` In this case `cmrunnode` is issued automatically.

```
# cmruncl

cmruncl:  Waiting for cluster to form ...
cmruncl:  Cluster successfully formed.
cmruncl:  Check the syslog file for any warnings found during startup.
```

To run a cluster on a subset of nodes you need to use the `–n` option of `cmruncl`. In this case you have to explicitely confirm that you intent to override ServiceGuard’s internal integrity protection:

```
# cmruncl –n ux_dbs_a

WARNING:
Performing this task overrides the data integrity protection normally
provided by ServiceGuard.  You must be certain that no package applications
or resources are running on the other nodes in the cluster:
       ux_dbs_b

To ensure this, these nodes should be rebooted (i.e. /usr/sbin/shutdown -r)
before proceeding.

Are you sure you want to continue (y/[n])? y

cmruncl  : Waiting for cluster to form....
cmruncl  : Cluster successfully formed.
cmruncl  : Check the syslog files on all nodes in the cluster
cmruncl  : to verify that no warnings occurred during startup.
```

Halting the Cluster

To halt a high availability cluster use the command `cmhaltcl`. This causes all nodes in a configured cluster to stop their cluster daemons, optionally halting all packages or applications in the process. This command will halt all the daemons on all currently running systems. If the user only wants to shutdown a subset of daemons, the `cmhaltnode` command should be used instead. The -f option causes all packages to be halted first automatically.

```
# cmhaltcl –f

Disabling package switching to all nodes being halted.
Warning:  Do not modify or enable packages until the halt operation is completed.

Halting Package dbci
This operation may take some time.
Halting cluster services on node ux_dbs_a
Halting cluster services on node ux_dbs_b
```

```
..
cmhaltcl  : Successfully halted all nodes specified.
Halt operation completed.
```

Starting a Node

To run a node in a high availability cluster use the command `cmrunnode`. This causes a node to start its cluster daemon to join the existing cluster. Starting a node will not cause any active packages to be moved to the new node. However, if a package is down, has its switching enabled, and is able to run on the new node, that package will automatically run there.

Please note that `cmrunnode` only joins to an existing cluster! If no such cluster is found, it retries usually for 10 minutes (`AUTOSTART_TIMEOUT`). This is especially useful for the automatic cluster start during bootup. If **all** nodes reach the cmrunnode step within 10 minutes then a new cluster is formed.

```
# cmrunnode

cmrunnode   : Waiting for cluster to form.....
cmrunnode   : Cluster successfully formed.
cmrunnode   : Check the syslog files on all nodes in the cluster
cmrunnode   : to verify that no warnings occurred during startup.
```

Halting a Node

To halt a node in a high availability cluster use the command `cmhaltnode`. This causes a node to halt its cluster daemon and remove itself from the existing cluster. When `cmhaltnode` is run on a node, the cluster daemon is halted and, optionally, all packages that were running on that node are moved to other nodes if possible. If no node name is specified, the cluster daemon running on the local node will be halted and removed from the existing cluster.

```
# cmhaltnode

Disabling package switching to all nodes being halted.
Warning:  Do not modify or enable packages until the halt operation is completed.

Halting cluster services on node ux_dbs_a
..
cmhaltnode  : Successfully halted all nodes specified.
Halt operation completed.
```

Starting Packages

To run a high availability package use the command `cmrunpkg` runs a high availability package that was previously halted.This command may be run on any node within the cluster and may operate on any package within the cluster. If a node is not specified, the node on which the command is run will be used. This will result in an error if the current node is not able to run the package or is not in the list of possible owners of the package. When a package is started on a new node, the package’s run script is executed with argument *start*.

```
# cmrunpkg dbci
```

```
cmrunpkg  : Completed successfully on all packages specified.
```

Stopping Packages

To halt a high availability package use the command `cmhaltpkg`. This performs a manual halt of high availability package(s) running on ServiceGuard clusters. The command may be run on any node within the cluster and may operate on any package within the cluster.

```
# cmhaltpkg dbci
cmhaltpkg  : Completed successfully on all packages specified.
```

Modifying the Switching Attributes of a Package

To enable or disable switching attributes for a high availability package use the command `cmmodpkg`. First it enables or disables the ability of a package to switch to another node upon failure of the package (global switching ability of the package), and - second - it enables or disables a particular node from running specific packages (node ability to run the package).

For example, if a globally disabled package fails, it will not switch to any other node, and if a globally enabled package fails, it will attempt to switch to the first available node on which it is configured to run.

`# cmmodpkg –e dbci`	Globally enable switching for a package.
`# cmmodpkg –d dbci`	Globally disable switching for a package.
`# cmmodpkg –n ux_dbs_a –e dbci`	Enable a node to run a package.
`# cmmodpkg –n ux_dbs_a –d dbci`	Disable a node to run a package.

```
# cmmodpkg –e dbci
cmmodpkg  : Completed successfully on all packages specified.
```

Create configuration templates

There are two types of configuration files, a cluster configuration file and for each package a package configuration file.

A template for a cluster configuration file is create with the command `cmquerycl`, which retrieves the needed configuration information from all nodes to be used in the cluster. It searches all specified nodes for cluster configuration and Logical Volume Manager (LVM) information. Cluster configuration information includes network information such as LAN interface, IP addresses, bridged networks and possible heartbeat networks. LVM information includes volume group (VG) interconnection and file system mount point information. This command should be run as the first step in preparing for cluster configuration. It may also be used as a troubleshooting tool to identify the current configuration of a cluster, since it prints out a pretty overview of its discovery results:

```
# cmquerycl -C cmclconfig.ascii -n ux_dbs_a -n ux_dbs_b

Node Names:     ux_dbs_a
                ux_dbs_b
```

```
Bridged networks:
...
IP subnets:
...
Possible Heartbeat IPs:
...
Possible Cluster Lock Devices:
...
LVM volume groups:
...
```

Package configuration templates are created using cmmakepkg. . The output file needs to be customized for a specific cluster environment. If no output filename is provided, output will be directed to stdout.

```
# cmmakepkg -p dbci.conf
Package template is created.
This file must be edited before it can be used.
```

The command is also responsible for creating a package control script template:

```
# cmmakepkg -s dbci.cntl
Package control script is created.
This file must be edited before it can be used.
```

Validating a Cluster Configuration

To check a high availability cluster configuration and/or package configuration files use the command `cmcheckconf.` This verifies the cluster configuration as specified by the cluster ASCII file and/or the package configuration files specified by each package ASCII file in the command.
If the cluster has already been configured previously, the `cmcheckconf` command will compare the configuration in the cluster ASCII file against the previously configuration information stored in the binary configuration file and validates the changes. The same rules apply to the package ASCII file.

```
# cmcheckconf -C cmclconfig.ascii -P dbci/dbci.conf

Begin cluster verification...
Verification completed with no errors found.
Use the cmapplyconf command to apply the configuration.
```

Generation and Distribution of a Cluster Configuration

To verify and apply ServiceGuard cluster configuration and package configuration files use the command `cmapplyconf`. It verifies the cluster configuration and package configuration specified in the cluster ASCII file and the associated package ASCII file(s), creates or updates the binary configuration file, called `/etc/cmclconfig`, and distributes it to all nodes. This binary configuration file contains the cluster configuration information as well as package configuration information for all packages specified.
If changes to either the cluster configuration or to any of the package configuration files are needed, first update the appropriate ASCII file(s) (cluster or package), then validate the

changes using the cmcheckconf command and then use cmapplyconf again to verify and redistribute the binary file to all nodes.
The cluster ASCII file only needs to be specified if configuring the cluster for the first time, or if adding or deleting nodes to the cluster. The package ASCII file only needs to be specified if the package is being added, or if the package configuration is being modified. It is recommended that the user run the cmgetconf command to get either the cluster ASCII configuration file or package ASCII configuration file whenever changes to the existing configuration are required.

```
# cmapplyconf -C cmclconfig.ascii -P dbci/dbci.conf

Begin cluster verification...
Modifying configuration on node ux_dbs_a
Modifying configuration on node ux_dbs_b

Modify the cluster configuration ([y]/n)? y
Modifying node ux_dbs_a in cluster GP2.
Modifying node ux_dbs_b in cluster GP2.
Completed the cluster creation.
```

LVM Modifications of Cluster Volume Groups

Changes to the configuration of cluster volume groups needs some specific attention. Typically only that node knows obout the changes where it was actually done. There is nothing like an automatic configdistribution to other nodes.

The easiest way to make config changes visible on other nodes is to re-import the chnaged VG from there. The following section describes three typical config changes and the steps that are needed for them. All config changes are done on that node that currently runs the associated package. It is assumed that the affected VG is activated in *exclusive* mode here (use vgdisplay to verify that). The so-called *Re-Import VG Procedure* below contains the most important steps.

1. **Changing the size of an lvol or file system only.**
   No additional ServiceGuard specific steps are needed. The change is only done in the LVM structures contained on the shared disks. Adoptive nodes *see* that change during activation time automatically.
2. **Adding or removing a physical volume to/from a cluster volume group.**
   Perform the *Re-Import VG Procedure* below from all other cluster nodes. Make sure that the new physical volume is visible in ioscan and accessible through its device special files. The command `insf –Cdisk` may be needed to create that files.
3. **Adding or removing an lvol or file system to/from a cluster VG.**
   The device special files of new lvols need to be created on all other cluster nodes, those of removed lvols need to be deleted. This may be done with mknod/rmsf, however it is recommended to perform the *Re-Import VG Procedure* below. Additionally the package control scripts need to be adapted on all nodes to reflect the new configuration. Mount points need to be created for new file systems.

The Re-Import VG Procedure

The following steps are needed to perform the re-import of an LVM cluster VG. We assume that a cluster volume group *vg01* with minor number *0x010000* needs to be re-imported.

- Steps for the node that has the VG active:

1. Create a mapfile:

```
node1# vgexport –v –p –s –m /tmp/vg01.map vg01
```

   Warning messages indicating that the VG is active should be ignored.

2. Copy mapfile to each other node (rcp, ftp, etc.)

```
node1# rcp /tmp/vg01.map node2:/tmp/vg01.map
...
```

- Steps to be performed on each other node:

3. Note VG minor number and permissions/ownership:

```
node2# ll /dev/vg01
total 12
drwxr-xr-x   2 root       root          1024 Apr 16 12:04 ./
dr-xr-xr-x  19 bin        bin           5120 Jun 20 06:44 ../
crw-r-----   1 root       sys         64 0x010000 Apr  4 13:32 group
brw-r-----   1 root       sys         64 0x010001 Apr  4 13:32 lvol1
crw-r-----   1 root       sys         64 0x010001 Apr  4 13:32 rlvol1
```

4. Export the VG:

```
node2# vgexport vg01
```

5. Re-create VG directory:

```
node2# mkdir /dev/vg01
```

6. Re-create VG group special file, use minor number noted above.

```
node2# mknod /dev/vg01/group c 64 0x010000
```

7. Run vgimport using the copied mapfile:

```
node2# vgimport –v –s –m /tmp/vg01.map vg01
```

   Messages indicating that no backup for this VG may exist should be ignored.

   Note:
   On systems using physical data replication products like BusinessCopy/XP, ContinousAccess/XP, EMC SRDF or EMC Timefinder it may be impossible to reliably identify the correct list of PVs using vgimport's –s option. You should specify the list of PVs explicitely then. The newly introduced `–f` option for vgimport helps to specify large PV lists on the command line (see man page). The `–f` Option is only available as of UX 11.X. For UX 11.00 you need LVM commands patch PHCO_20870 or later.

8. Change permissions and ownership of the VG directory and its device special files according to the information noted above.

9. Test the activation of the VG in read-only mode and perform vgcfgbackup:

```
node2# vgchange –a r vg01
node2# vgcfgbackup vg01
node2# vgchange –a n vg01
```

Troubleshooting

Where to find Logfiles

- ServiceGuard logs lots of information to `/var/adm/syslog/syslog.log`. If nothing is found there you should check if syslogd still works as exspected (e.g. `inetd –l`; `inetd –l` should cause messages like *Connection logging enabled/disabled*).
- The starting and stopping of packages and services is done from package control scripts, usually `/etc/cmcluster/`*`package`*`/`*`package`*`.cntl` on the node that performs the starting or stopping.
  These scripts log their messages to `/etc/cmcluster/`*`package`*`/`*`package`*`.cntl.log`.
- Logging is always done locally, so the logfiles of all affected nodes need to be checked.
- See section Changing ServiceGuard's Log Level.

What happens during Package Start

During package start the corresponding package control script is call with the argument *start*. The script is usually called `/etc/cmcluster/`*`package`*`/`*`package`*`.cntl`. The command cmviewconf can be used to have a look at the script configuration:

```
# cmviewconf | egrep ‘package (name|run|halt)’
      package name:                    dbci
      package run script:              /etc/cmcluster/dbci/dbci.cntl
      package run timeout:             (No Timeout)
      package halt script:             /etc/cmcluster/dbci/dbci.cntl
      package halt timeout:            (No Timeout)
```

The package start in general consists of several steps, summarized in the next section.

- **activate_volume_group**
  All volume groups of the package are activated with exclusive option (as specified in the package control script). The message
  Activation mode requested for the volume group conflicts with configured mode
  appears if the volume group is not flagged as cluster aware (`vgchange –c y VG`). The command cmapplyconf sets this flag autmatically for all VGs listed in the cluster ASCII file and it clears it from all VGs that are missing (if no `–k` option is used).
- **check_and_mount**
  The file systems are checked using fsck and mounted, using the mount options specified in the package control script. Missing (*No such file or directory*) or busy (*already mounted, is busy, or allowable number of mount points exceeded*) mount point make package start fail in this stage. Use `fuser <`*`directory`*`>` to find

responsible processes.

- **add_ip_address**
  The ServiceGuard command cmmodnet is used to configure all relocatable IP addresses associated with this package. The command can be also used from command line, e.g. `cmmodnet –a –I 192.10.10.120 192.10.10.0`.
- **customer_defined_run_cmds**
  This is the place where the customer's HA application is started. Failures in this area should be trouble shooted from the application side. Commenting out the faulty command may be a good strategy to get a minimum trouble shooting environment running (otherwise the complete package start fails, causing all file systems to be umounted, etc).
- **start_services**
  The ServiceGuard command cmrunserv is used to start the so called services. These are typically shell scripts monitoring the health of the HA application. An exiting service script means *Monitoring Failed*, causing the corresponding package to failover to an adoptive node. The cmrunserv command is not allowed to be used manually from the command line.
- **start_resources**
  The ServiceGuard command cmstartres is used to start monitoring of configured EMS resources.

What happens during Package Stop

Stopping of a package performs in general the opposite steps as described above. If a package halt fails some kind of cleanup may is needed to get the system manually back to a defined "package halted state".

This includes:

- The services are stopped, is usually the case.
- The application is halted.
- All relocatable IP adresses are de-configured, can be done manually using e.g. `cmmodnet –r –I 192.10.10.120 192.10.10`.
- All associated file systems are umounted. All processes keeping them busy need to be terminated first (use fuser and umount).. If the file system was exported via NFS it may be required to kill/restart the `rpc.statd` and `rpc.lockd` processes also.
- Deactivate all VGs of the package (`vgchange –a n VG`). This is only possible if all file systems were successfully umounted before. Otherwise this fails with a *Device busy* error.

Cluster Lock Disk Initialization

The cluster lock is used as a tie-breaker only for situations in which a running cluster fails and, as MC/ServiceGuard attempts to form a new cluster, the cluster is split into two sub-clusters of equal size.

Every hour ServiceGuard checks the availability of configured cluster lock disks. A failed check is logged to `syslog.log`:

```
WARNING: Cluster lock on disk /dev/dsk/cXtYdZ is missing!
Until it is fixed, a single failure could cause all nodes in the cluster to crash.
```

Either there is a hardware problem (which needs to be fixed as soon as possible) or the cluster lock disk is not initialized correctly, which is typically done with cmapplyconf.

The cmviewconf comand can be used to retrieve the current cluster lock configuration:

```
# cmviewconf | grep -e "Node name" -e lock
   flags:                               12      (single cluster lock)
   first lock vg name:                  /dev/vgito
   second lock vg name:                 (not configured)
      Node name:                        ux_dbs_a
      first lock pv name:               /dev/dsk/c0t4d4
      first lock disk interface type:   c720
      Node name:                        ux_dbs_b
      first lock pv name:               /dev/dsk/c0t5d4
      first lock disk interface type:   c720
```

The cluster lock information is backed up and restored with LVM's vgcfgbackup and vgcfgrestore commands. If no vgcfgbackup was done after cluster lock initialization then a new cmapplyconf needs to be done to get this fixed. Another (inoffical) method is to use the unsupported cminitlock tool, to be retrieved from http://wtec.cup.hp.com/~hpuxha/tools/index.html (HP internal).

How to initialize the cluster lock disk(s) using cmapplyconf:

- Halt the entire cluster.

  ```
  # cmhaltcl –f
  ```

- Perform the following command from all nodes in the cluster to remove the cluster flag from cluster lock VG(s).

  ```
  # vgchange –c n <VG>
  ```

- Activate cluster lock VG(s) on one node only:

  ```
  # vgchange –a y <VG>
  ```

- Perform cmapplyconf on the node where you activated the cluster lock VG(s). The cluster flag is added back to the VG automatically.

  ```
  # cmapplyconf –C <cluster-ascii
  ```

- Perform vgcfgbackup to backup the cluster lock information:

  ```
  # vgcfgbackup <VG>
  ```

- Deactivate cluster lock VG(s):

  ```
  # vgchange –a n <VG>
  ```

- Run vgcfgbackup on all other cluster nodes also:

  ```
  # vgchange –a r <VG>
  # vgcfgbackup <VG>
  # vgchange –a n <VG>
  ```

- Restart the cluster:

```
# cmruncl
```

What about ServiceGuard TOCs

ServiceGuard uses the safety timer to ensure that a cluster node NOT communicating heartbeats with the majority of the active nodes in the cluster should NOT be running HA services, such as those defined in ServiceGuard packages, as this can result in data corruption (the *"split-brain"* syndrome).

Whenever the ServiceGuard daemon, cmcld, successfully receives a heartbeat message, which implies that it can communicate with another cluster node, it adjusts the safety timer to an explicit time in the future based on the current system clock and the cluster's calculated failover time.This explicit time specifies until when a cluster node is allowed to wait for the next heartbeat message before concluding that the node is not communicating to other cluster nodes and thus have to fail over its HA services.

In this context, the failover time essentially governs how long a node can tolerate the absence of heartbeat messages whereas the safety timer specifies an explicit deadline for the next heartbeat. The calculation of the failover time depends on the ServiceGuard revision, the configured node timeout, the heartbeat interval and on the type of hardware used. A calculation tool can be found on http://haweb.cup.hp.com/Support/SG_Failover.html (HP internal).

If cmcld does not keep on advancing the safety timer, the system clock will eventually take over the safety timer. Once the system clock is equal to or beyond the safety timer, we say that the safety timer expires. To ensure that the node will stop its HA services once the safety timer expires, the node triggers a ServiceGuard TOC to take itself out of the cluster. So in essence it is not cmcld that initiates the TOC, it is cmcld that prevents the TOC from happening.
Before initiating the TOC the following message is logged to the kernel's message buffer and to the system's console:

```
MC/ServiceGuard: Unable to maintain contact with cmcld daemon.
Performing TOC to ensure data integrity.
```

Having understood the mechanism of the safety timer, the next question to ask in analysing a ServiceGuard TOC dump is really why cmcld was not receiving and processing heartbeats during the last failover time period, also referred to as the last safety timer window, before the TOC. An excellent starting point for 1st pass dump analysis can be found on http://wtec.cup.hp.com/~hpux/crash/FirstPassWeb (HP internal).

How to track down Networking Problems

While discovering the network (e.g. during cmcheckconf of cmapplyconf), MC/ServiceGuard tries to separate it into so called **Bridged Nets**. Interfaces that can talk to each other on link level are associated to the same brigded net.

The result of a successful discovery can be viewed using the cmviewonf tool:

```
# cmviewconf

Cluster information:

   cluster name:                        alwaysup
   version:                             0
   flags:                               12      (single cluster lock)
   heartbeat interval:                  1.00    (seconds)
   node timeout:                        8.00    (seconds)
   heartbeat connection timeout:        16.00   (seconds)
   auto start timeout:                  600.00  (seconds)
   network polling interval:            2.00    (seconds)
   first lock vg name:                  /dev/vg01
   second lock vg name:                 (not configured)

Cluster Node information:

   Node ID 2:
      Node name:                        grcdg238
      first lock pv name:               /dev/dsk/c0t4d4
      first lock disk interface type:   c720

      Network ID 1:
        mac addr:                       0x080009fd4375
        hardware path:                  8/16/6
        network interface name:         lan0
        subnet:                         15.140.8.0
        subnet mask:                    255.255.248.0
        ip address:                     15.140.10.236
        flags:                          1       (Heartbeat Network)
        bridged net ID:                 1
```
[…]

Network interfaces with the same ‘bridged net ID’ are considered to be on the same bridged net. A good starting point for trouble shooting networking problems is `cmquerycl –l net –v`. To check the link level connectivity between interfaces one may use the `linkloop(1M)` command. The `cmscancl(1M)` script does this checking automatically for all interfaces on all nodes of the cluster.

Example:

```
# lanscan
Hardware Station        Crd Hdw   Net-Interface  NM  MAC       HP-DLPI DLPI
Path     Address        In# State NamePPA        ID  Type      Support Mjr#
8/16/6   0x080009FD4375 0   UP    lan0 snap0     1   ETHER     Yes     119
8/8/2/0  0x00108318AFEE 2   UP    lan2 snap2     2   ETHER     Yes     119
8/8/1/0  0x00108318AFED 1   UP    lan1 snap1     3   ETHER     Yes     119
```

The following linkloop command checks the local link level connectivity from lan2 to lan1. Please note that yYou need to specify the outgoing PPA (NMIDfor 10.x) with the `–i` option.

```
# linkloop –i 2 0x00108318AFED
Link connectivity to LAN station: 0x00108318AFED
--- OK
```

Sometimes ServiceGuard’s discovery does not match the results achieved with linkloop, because it uses a slightly differerent method. There is also a pretty tool ‘dlpiping’ (internally downloadable from http://wtec.cup.hp.com/~hpuxha/tools) which can be used in such cases. The trick is that this tools uses exactly the same mechanisms that ServiceGuard uses. So the results are more representative than simply using linkloop.

To test connectivity from local PPA 1 to local PPA 2:

```
# dlpiping 2 1
Bound PPA 2, Ethernet, address 0x00108318afee
Bound PPA 1, Ethernet, address 0x00108318afed
Send from PPA 2 to PPA 1 0x00108318afedaa080009167f
Send from PPA 1 to PPA 2 0x00108318afeeaa080009167f
Recv from 0x00108318afedaa080009167f on PPA 2 0x00108318afeeaa080009167f
Recv from 0x00108318afeeaa080009167f on PPA 1 0x00108318afedaa080009167f
```

To test connectivity from local PPA 1 to remote PPA 2:

```
# dlpiping -n <remote node> 2 1
Bound PPA 2, Ethernet, address 0x00108318afee
Bound remote PPA 1, Ethernet, address 0x00108318cff8
Send from PPA 2 to remote PPA 1 0x00108318cff8aa080009167f
Send from remote PPA 1 to PPA 2 0x00108318afeeaa080009167f
Recv from 0x00108318cff8aa080009167f on PPA 2 0x00108318afeeaa080009167f
Recv from 0x00108318afeeaa080009167f on remote PPA 1 0x00108318cff8aa080009167f
```

Tracing SG heartbeat and network polling

Often we want to observe what MC/SG really does when it polls LAN interfaces or when it transmits heartbeats for inter-cluster communication. This can be done using nettl tracing with specific filters.

- Start tracing, e.g. with:

  ```
  # nettl -start
  # nettl -tn 0xff800000 -e all -f /tmp/mytrace -tm 99999 -m 128 -size 1024
  ```

- Generate the traffic you want to trace, e.g. run cmcheckconf or wait some time while the cluster is up and running.

- Stop tracing:

  ```
  # nettl -tf -e all
  ```

- Generate a filter file for nettl (see next sections for details), e.g.:

  ```
  # echo "filter type 0x167f" >/tmp/myfilter
  ```

- Format and look at the trace, e.g. with:

  ```
  # netfmt -nNlc /tmp/myfilter /tmp/mytrace.TRC0 |more
  ```

There are several types of traffic generated by MC/ServiceGuard and its daemons.

1. **Polling packets used by cmclconfd** to explore the networking infrastructure at configuration time (e.g. cmquerycl, cmcheckconf und cmapplyconf). cmclconfd sends packets using the dlpi API and adds a SNAP encapsulation (0xaa) of type config_snap (080009167f). Filtering can be done using the filter file entry:

```
filter type 0x167f
```

sample result:

```
Source : 00-60-b0-6e-fb-ac [I] [ ] LENGTH: 48
Dest : 08-00-09-d4-71-d8 [I] [HP ] TRACED LEN: 128
Date : Wed Jan 26 13:48:35.202359 MET 2000
=================================== 802.2 ================================
DSAP : 0xaa SSAP : 0xaa CONTROL : 0x03[U-FORMAT]
=================================== SNAP =================================
OUI : 08-00-09 TYPE: 0x167f
---------------------------------------------------------------------------
.0: 00 00 00 01 00 00 00 01 00 00 00 02 00 00 00 01 ................
16: 00 00 00 04 ff 00 00 00 00 00 00 01 00 00 00 01 ................
32: 00 00 00 04 ff 00 00 00 fa d2 00 00 00 00 00 00 ................
48: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
64: 00 00 00 00 00 00 00 00 00 00 01 39 20 0a 08 00 ..............9
80: 09 f8 9c 9b e8 85 00 02 03 0c 30 30 36 30 42 30 ..........0060B0
96: 34 35 33 42 37 34 38 33 43 54 -- -- -- -- -- -- 453B7483CT......
```

2. **Polling packets used by cmcld** to check the health of monitored LAN interfaces during normal cluster operation. cmcld also sends packets using the dlpi API and adds a SNAP encapsulation (0xaa) of type cl_comm_snap (080009167e). Filtering can be done using the filter file entry:

```
filter type 0x167e
```

sample result:

```
Source : 00-60-b0-6e-fb-ac [I] [ ] LENGTH: 116
Dest : 08-00-09-26-7c-8e [I] [HP ] TRACED LEN: 128
Date : Wed Jan 26 12:27:20.645972 MET 2000
=================================== 802.2 ================================
DSAP : 0xaa SSAP : 0xaa CONTROL : 0x03[U-FORMAT]
=================================== SNAP =================================
OUI : 08-00-09 TYPE: 0x167e
---------------------------------------------------------------------------
 0: 00 00 00 04 00 00 00 02 00 00 00 01 00 00 00 00 ................
16: 40 02 22 c0 00 00 00 3c 00 00 00 30 38 73 4e ac @.“....<...08sN.
32: 00 00 00 01 38 8e d9 e0 00 00 00 02 00 00 00 00 ....8...........
48: 40 04 35 7e 00 00 00 1c 00 04 00 00 00 00 00 01 @.5~............
64: 00 00 00 01 00 00 00 02 00 00 00 02 00 00 00 18 ................
80: 00 00 00 18 00 00 00 01 00 00 00 01 00 00 00 01 ................
96: 00 00 00 01 00 00 00 00 00 00 -- -- -- -- -- -- ................
```

3. **Heartbeat traffic** for inter-node communication using socket connections on port hacl-hb (usually 5300). This traffic can be filtered e.g. with this filter file:

```
filter tcp_dport hacl-hb
filter tcp_sport hacl-hb
```

sample result:

```
================================ IP Header (outbound – pid: 16002)========
Source: 15.140.8.40(A) Dest: 15.140.8.58(A)
len: 128 ttl: 64 proto: 6 cksum: 0x9661 id: 0x7499
flags: DF tos: 0x0 hdrlen: 20 offset: 0x0 optlen: 0
-------------------------------- TCP Header ------------------------------
sport: 3547 → dport: 5300 flags: PUSH ACK
```

```
seq: 0x3a0cfd91 urp: 0x0 chksum: 0x97b8 data len: 88
ack: 0x7e5864bd win: 0x8000 optlen: 0
-------------------------------- HACL-HB --------------------------------
 0: 00 00 00 02 00 00 00 02 00 00 00 01 40 44 04 20 .......... ..@D.
16: 40 79 6a 38 00 00 00 3c 00 00 00 20 38 73 4e ac @yj8...<... 8sN.
32: 00 00 00 02 38 8e f3 9c 00 00 00 01 00 00 00 00 ....8...........
48: 00 00 01 c8 00 00 00 00 00 00 00 00 00 00 00 07 ................
64: 00 00 00 03 00 00 00 01 00 05 ef 6f 00 00 00 01 ...........o....
80: 03 93 87 00 03 93 87 00 -- -- -- -- -- -- -- -- ................
```

Changing ServiceGuard's Log Level

MC/ServiceGuard uses configurable internal functions to log messages, warnings, notices and errors. A ServiceGuard message is classified by the module that issued the message, by the reason (category) the message is being logged, and by the level of detail of information.

These are the different logging categories:

CAT	SG internal category	syslog(3C) Level
INT	LOG_INTERNAL	LOG_INFO
EXT	LOG_EXTERNAL	LOG_NOTICE
PER	LOG_PERIODIC	LOG_INFO
XER	LOG_EXT_ERROR	LOG_ERR
ERR	LOG_INT_ERROR	LOG_ERR
DTH	LOG_DEATH	LOG_EMERG
TRC	LOG_TRACE	LOG_INFO

ServiceGuard can be divided into several modules. Each message has its origin in one of the following modules:

MOD	SG module
CLM	Cluster management
PKG	Package management
NET	Network interface
SRV	Service monitoring
LOC	Local communications
REM	Remote communications
SDB	Status database
SYN	Synchronization
CSV	Command server
COM	Communications server
DLM	Distributed lock manager
GMS	OPS group membership service
LVM	Shared LVM
UTL	General support
CDB	Configuration database
STA	Status database API
DEV	Storage devices
ATS	Shared tape devices

The cmsetlog command

The cmsetlog command enables users to obtain a more verbose output of SG. This is extremely useful if a problem should be reproduced. cmsetlog allows to set the log level and to restrict logging to specific categories and modules. cmsetlog is used to enable and disable debug logging. cmsetlog only works if the SG cluster is already running and can therefore not be used to obtain debug logs if the cluster startup fails. Refer to the `-T` option below.

Warning: cmsetlog should only be used for debugging purposes and all debug logging should be disabled once the debugging has been finished.

cmsetlog is located in the `/usr/contrib/bin/` directory, to stress that this program is basically an unsupported tool that should not be used by customers.

Enable SG debug logging with cmsetlog

cmsetlog can enable the debug logging only on the local node. Note that cmsetlog operates only on the local daemon. It must be run on each node you wish to change logging on. With high log levels greater 3, the log files can **grow very fast** and can fill up the filesystem. Here are various examples that show how cmsetlog can be used to obtain debug logging:

Log SG debug information to syslog.log at log level 5 for all categories (except PER (frequent actions)) and modules:

```
# cmsetlog 5
```

Redirect the SG logging in the file /tmp/SG.log. No further logs go into syslog.log. This is recommended for the verbose log levels greater 3:

```
# cmsetlog –f /tmp/SG.log
```

Use the most verbose log level for SG, but restrict logging to the modules ‘network interface’ and ‘remote communications’:

```
# cmsetlog –M NET –M REM 6
```

Use the most verbose logging level, but restrict logging to the NET module. Include all categories, specifically the PER (frequent action) to obtain logging of the network polling that is used to monitor the health of the LAN interfaces:

```
# cmsetlog –M NET –C PER –C ERR –C XER –C INT –C EXT –C DTH –C TRC 6
```

The most complete log that is possible:

```
# cmsetlog –C PER –C ERR –C XER –C INT –C EXT –C DTH –C TRC 6
```

Disable SG debug logging with cmsetlog

The debug logging is automatically stopped and reset to default once the cluster halted. To reset the debug logging to default modules, categories and loglevel on a running cluster, simply use the command:

```
# cmsetlog –r
```

If the '-f <file>' option has been used with cmsetlog to redirect logging to another file, you can re-direct it back to syslog.log with the command:

```
# cmsetlog –s
```

Debug logging for Advanced Tape Services (ATS) with stsetlog

stsetlog is an undocumented command that enables debug logging for the ATS feature of SG. The usage is

```
# stsetlog <level>
```

Level can be in the range of 0 to 6, where 6 is the most verbose level that also logs the messages sent by ATS. To disable debug logging use "stsetlog 0". The default logfile for ATS debug logging is `/var/adm/cmcluster/sharedtape/cmtaped.log`.

Note: The command "`cmsetlog –M ATS <level>`" does not enable ATS debug logging.

Debug logging using the '-T' option

Starting with SG 11.03 (and with patches PHSS_15531 for SG 10.10) a new option to some of the SG commands has been added to pass cmsetlog log levels to cmcld without invoking cmsetlog itself.

For some commands like cmruncl and cmrunnode there is an additional option `–F <file>` that allows to write the debug logs into a file. This option is not supported by cmcheckconf, cmapplyconf, cmquerycl and others. This is useful especially if the node is currently not running as a cluster member, because cmsetlog wouldn't work in this case. Here are some examples:

Run cmruncl with debug level 5 output in the file `/tmp/cmruncl.out`:

```
# cmruncl –v –T 5 –F /tmp/cmruncl.out
```

Run cmquerycl with debug level 3 output in the file `cmquery.debug`:

```
# cmquerycl –n nodeA –nodeB –T 3 >cmquery.debug
```

Debug logging of cmclconfd

The `-T` option described above can also be used to instrument the cmclconfd. This daemon is used to gather and send configuration data from the local and the remote nodes and is therefore started with many SG commands.

To enable cmclconfd debug logging modify the following lines in `/etc/inetd.conf` file:

```
hacl-cfg dgram udp wait root /usr/lbin/cmclconfd cmclconfd –p
hacl-cfg stream tcp nowait root /usr/lbin/cmclconfd cmclconfd –c
```

to

```
    hacl-cfg dgram udp wait root /usr/lbin/cmclconfd cmclconfd –p –T 5
    hacl-cfg stream tcp nowait root /usr/lbin/cmclconfd cmclconfd –c –T 5
```

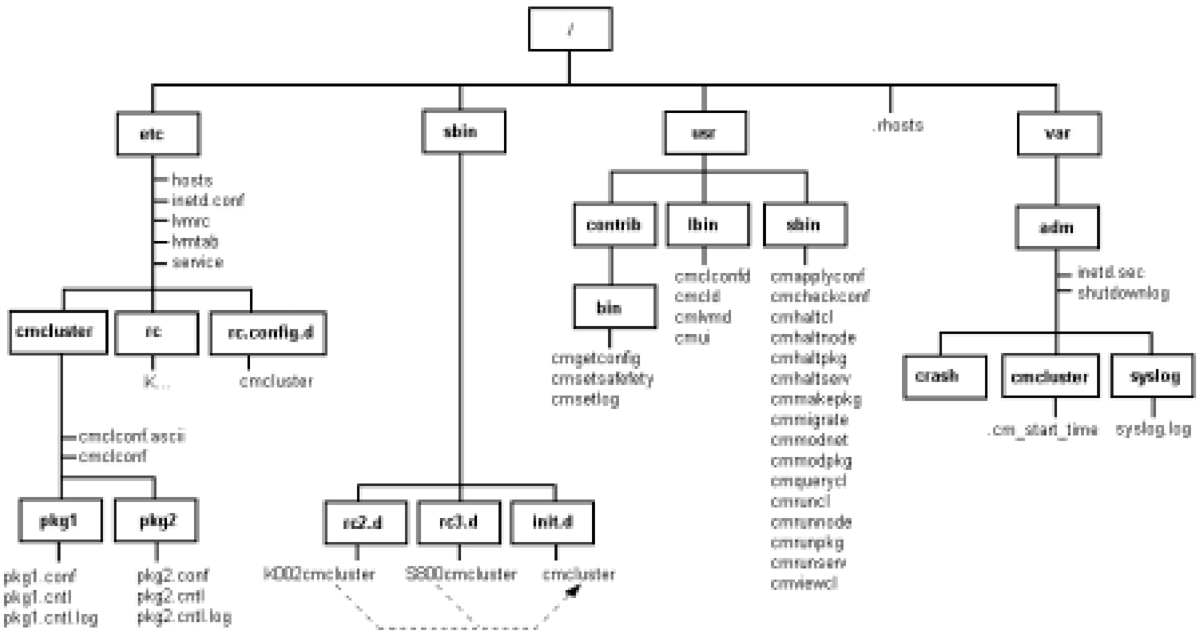
and run

```
    # inetd –c
```

This change has to be performed on all nodes that are required to be debugged. The logs will go into `syslog.log`. Note that this file can grow quickly.

To disable the logging of the cmclconfd, undo the changes in `/etc/inetd.conf` and run `inetd -c` again.

Overview of ServiceGuard Files



MC/ServiceGuard Commands Overview

This section should be used as short reference to ServiceGuard commands. They are grouped into **Info Commands** for getting cluster information, **Admin Commands** for cluster administration taks and **Config Commands** for performing cluster configuration changes. Please note that these reference is not supposed to replace the official documentation or the man pages!

Info Commands

Info Command	Options	Comment
`cmviewcl`		View cluster information
	`–n nodename`	View cluster information for a specific node
	`–p package`	View package information
	`-l {package|cluster|node|group}`	View information about cluster restricted to a certain level of information.
	`-v`	Be more verbose
`cmquerycl`		Get information about e. g. lvm, file systems and lan config. Use this tool to create the cluster_ASCII_file.
	`-n {nodename}`	Get above infos for specific node
	`-c {clustername}`	Get above infos for specific cluster
	`-C cluster_ASCII_file`	Write info into ASCII File
`cmscancl`		Gather system configuration info from nodes with MC/ServiceGuard.
	`-n {nodename}`	Get above infos for specific node
	`-s`	Write info to screen
	`-o output_file`	Write info to output_file
`cmviewconf`	`-o {filename}`	Get information about cluster settings, network and packages using the binary configuration file.
`cmgetconf`		Read the binary cluster configuration file and write it to an ASCII file.
	`-c {clustername}`	Name of cluster for which to query
	`-p {packagename}`	Name of package for which to query
	`filename`	File to where the output will be copied

Config Commands

Config Command	Options	Comment
`cmcheckconf`		Check cluster configuration and/or package configuration files
	`-C cluster_ascii_file`	Check cluster_ascii_file
	`-P pkg_ascii_file`	Check pkg_ascii_file
	`-p pkg_reference_file`	Check multiple packages listed in file

Config Command	Options	Comment
cmapplyconf		Apply cluster configuration and package configuration files
	-C cluster_ascii_file	Apply cluster_ascii_file
	-P pkg_ascii_file	Apply pkg_ascii_file
	-p pkg_reference_file	Apply multiple packages listed in file
	-f	Force the distribution even if a binary configuration file exists on any nodes
cmdeleteconf		Delete either the cluster or the package configuration
	-c cluster_name	Name of the cluster to delete
	-p package_name	Name of an existing package to delete from the cluster
cmmakepkg		Create package template files
	-p | -s	Create configuration file {.conf} or create control script {.cntl}

Admin Commands

Admin Command	Options	Comment
cmmodpkg		Enable or disable switching attributes for a package and run a package.
	-e|-d	Enable or disable
	-n node_name	Run a package on a specific node and overrun the order of the package .conf file
	-R –s {SER_NAME PKG_NAME}	Reset the restart counter for service SER_NAME in package PKG_NAME
cmruncl		Run a high availability cluster
	-n node_name	Start the cluster daemon on the specified subset of node(s)
cmhaltcl		Halt a high availability cluster
	-f	Force halt, even if packages are running. This causes all packages to be halted.
cmrunnode	node_name	Run a node in a high availability cluster. If node_name is not specified, the cluster daemon will be started on the local node and will join the existing cluster.
cmhaltnode		Halts cluster daemon and remove itself from the existing cluster.
	-f	Force halt, even if packages are running. This causes all packages to be switched to another node.
	node_name	If node_name is not specified, the cluster daemon running on the local node will be halted and removed from the existing cluster.
cmrunpkg		Run a high availability package

Admin Command	Options	Comment
	`-n node_name`	If a node is not specified, the node on which the command is run will be used. If the package was previously halted, the global switching for the package will still be disabled until a cmmodpkg -e is done on the package.
	`package_name`	Name of a package
`cmhaltpkg`		Halt a high availability package
	`-n node_name`	If the -n option is not specified, the package will be halted regard-less of where it is currently running. When a package is halted,package switching is disabled for that package.
	`package_name`	Name of a package
`cmrunserv`		Run a service. Only to be used from the package control script
	`service_name`	Name of service as it exists in the package configuration information
	`service_command_string`	Process string to be started
	`-r {no. of restarts}`	Number of times that the service is restarted until the package is halted
	`-R`	The service should be restarted an unlimited number of times if it fails.
`cmhaltserv`	`service_name`	Halt a service from the high package run script
`cmstartres`	`-p package_name resource_name`	cmstartres starts resource monitoring for an EMS resource on the local node. This resource must be configured in the specified package_name
	`-u`	Wait for the EMS resource to be available before starting resource monitoring
`cmstopres`	`-p package_name resource_name`	cmstopres stops resource monitoring for an EMS resource on the local node. This resource must be configured in the specified package_name

ServiceGuard Common issues

Problems during cmcheckconf / cmapplyconf

- Errors:
  `Unable to determine the nodes on the current cluster.`
  `Unable to communicate with node %s`

  ServiceGuard's cmclconfd does authorization checks when being connected. This is done by searching the requesting machine/user name in /etc/cmcluster/cmclnodelist. If this file does not exist then the root user's .rhosts file is checked.
  The cmclconfd needs to be triggered by inetd, so it's a good idea to also check /etc/inetd.conf and /var/adm/inetd.sec. Check this from every node to every other node:

```
# telnet localhost hacl-cfg
# telnet <own hostname> hacl-cfg
# telnet <other hostname> hacl-cfg
```

The telnets should simply hang, which is the normal behaviour if you reach the remote cmclconfd. Messages like *connection refused* are not allowed.

- Error:
  `The local node %s appears to belong to a different cluster.`

  The node's /etc/cmcluster/cmclconfig file already contains a configuration with a different cluster ID in it. You should use cmdeleteconf to remove that configuration first. As a last resort you can remove the file from that node.

- Errors:
  `Volume group %s currently belongs to another cluster.`
  `First cluster lock volume group %s belongs to another cluster.`
  `Second cluster lock volume group %s belongs to another cluster.`

  The LVM header of that volume group contains the cluster ID of a different cluster. If you are sure that your cluster should own these disks you can perform

  ```
  # vgchange -c n VG
  ```

  to clear the ID. This command should be performed on all nodes of the cluster.

- Warning:
  `Volume group %s is configured differently on node %s than on node %s.`

  This message may be caused by a differnent number of PV links for some of the physical volumes of that volume group. However the command should complete successfully.

- Warning:
  `The disk at %s on node %s does not have an ID.`

  ServiceGuard prints warning messages for all disks that do not contain a valid header headers. This is normal and can be ignored for all disks that are not part of a LVM volume group or VxVM disk group.

- Error:
  `Unable to determine a unique identifier for physical volume %s on node %s.`

  The Physical Volume ID found of that disk was not found in /etc/lvmtab of that node. You should make sure that the volume group is configured correctly. Try to export/import the VG to get the lvmtab updated.

- Problem:
  cmcheckconf/cmapplyconf hangs or needs very long to complete.

  When performing cmcheckconf/cmapplyconf a cmclconfd helper process is launched on each node for gathering configuration information. Most likely reasons for a hang

```
Warning: No possible heartbeat networks found.

Non-uniform connections detected

Network interface %s on node %s couldn't talk to itself.

cmcheckconf hangs or needs very long to complete

Unable to recv initialize VG message to %s: %s

Failed to evaluate network

Failed to release volume group %s
```

SGeSAP Common issues

The *MC/ServiceGuard Extension for SAP R/3* is one of the most important, but unfortunately also one of the most complicated. To get detailed step-by-step instructions and other important information you should consult the manual Managing MC/ServiceGuard Extension for SAP R/3, downloadable from http://docs.hp.com/hpux/ha. In the following section the most common issues with integration are listed.

NFS export configuration

- Wrong usage of `exportfs` and `nfs.server start`:
  NFS exports done by the package may disappear or may get configured differently if you use `exportfs(1M)` or re-start the NFS server. This typically causes '`stale NFS file handle`' errors and trouble with access permission.
- Wrong confguration of `-access` and `–root` options:
  If a directory is exported with an access restriction (`-access` option used) then all nodes in the root access list (`-root` option) need also to be listed in the access list.
- Hostnames and IP addresses missing in export list:
  The NFS server authenticates clients by searching the IP source address in its exports list. That source address can be affected by the routing from the client to the server, so it is a good idea to add all names and addresses (all networking interfaces) of the clients to the configuration. Otherwise changed routing may lead to serious permission trouble.

NFS/Automounter

- autofs: `automountd` restart while relocatable IPs are configured (JAGad36252):
  SGeSAP mounts filesystems from the relocatable IP address. If autofs is used and the NFS server is restarted on the node that has the relocatable IP locally configured, the mounts may be changed into loopback mounts (lofs mounts). autofs tries to optimize these mounts, which is from a SGeSAP viewpoint an unwanted behaviour. Subsequent package halts will fail due to "`cannot umount <filesystem>, Device busy`" messages. Before any of `nfs.client`, `nfs.server`, `autofs` is restarted the relocatable IP address has to be deconfigured.
- No NFS read access for root to directory `/usr/sap/trans`:
  The package control script tries to access `/usr/sap/trans/bin` as root. If this fails “`tranport directory invalid`” error during package start are issued.
- (auto-)mounting into shared file systems:
  If other file systems are mounted into package file systems (e.g. by a DBA who adds space to the database without taking care about cluster issues) “`cannot umount, Device busy`” errors are the result during package halts.

Package mount point trouble

- `/usr/sap/SID` mount point used instead of `/usr/sap/SID/instance`:
  If the (db)ci package uses `/usr/sap/SID` instead of `/usr/sap/SID/InstDir` application server shutdown problems on failover nodes may be the result.
  The reason is a little bit tricky: The `sapstop` executable, which is actually called by the `stopsap` script of the application server, basically does the following: It executes

the script `kill.sap` which was stored inside its own `/usr/sap/SID/InstDir/` working directory during startup. This script sends a `kill -2` to the `sapstart` process that finally handles the complete shutdown of the instance. If you use a directory above `/usr/sap/SID/InstDir` as mountpoint for a shared disk, you risk making the instance directory of internal application servers unreachable if a switchover takes place (the directory tree gets covered). This is no problem as long as the AS is shutdowned before the CI mounts its directories (i.e. the case of normal package halt).

- `/usr/sap` used instead of `/usr/sap/SID/InstDir`:
  Causes '`tranport directory invalid`' during package start. The transport directory `/usr/sap/trans` is shared between all SAP nodes using the automounter. If the package is mounted on `/usr/sap`, instead of `/usr/sap/SID/InstDir` the transport directory can become unavailable after package start if it was mounted first.
- `/usr/sap` used instead of `/usr/sap/SID/InstDir`:
  Causes '`cannot umount`' and'`Device busy`' during package halt.
- The same as above can happen during package halt when the transport directory keeps the filesystem busy that was mpounted to `/usr/sap`, instead of `/usr/sap/SID/DVEBMGS<instanceno>`.
- `/oracle/SID` used for AS on failover node:
  Causes mount point collision with shared `/oracle/SID`. Because `/oracle/SID` is needed as a free mountpoint by the package, it must not be a mountpoint for a logical volume containing the client ORACLE files. Alternatives include using `/oracle` as a mountpoint for the local filesystem or using a link.

General NFS

- Inconsistent LVM minor numbers for cluster VGs:
  Causes '`stale NFS file handle`'. The minor numbers of the `/dev/*/group` volume group device files must be consistent across all nodes in a SG cluster since this device information is part of the file handle that NFS uses.
- Wrong order of functions `add_ip_address()` and `export_fs()`:
  May cause '`stale NFS file handle`' during failover. The problem with this order is that the IP address is added before the exportfs occurs. During package failover this can cause problems with already mounted clients. The clients are hanging trying to reach the relocatable IP address. As soon as the relocatable IP address is available again, the client tries an NFS operation to, for example, read a file. However, since the filesystem is not exported yet, the client gets back a "`stale NFS file handle`" error. This is a fairly small timing window, but it has been seen as common problem with SGeSAP 3.00.09 and 3.01 installations. See also HPUX HA Newsletter #11.
- No local SAP executables configured:
  For SGeSAP installations it is required to use local SAP executables. This is also a requirement for appropriate performance. Also SAP recommends storing executables locally if the host systems in your network have adequate disk storage. The sapstart exectuable performs an automatic copy of executables from the central NFS directory to the local directory if you set up SAP for local executables. The procedure is known as SAPCPE and documented in SAP OSS note 4375 and in the SGeSAP manual.
- NFS always puts a risk on package halts:
  `NODE_FAIL_FAST_ENABLED` may be an option to guarantee that the package is downed

and can be started on the failover node.

- Hang at NFS level (`NFS server not responding`):
  SGeSAP uses NFS loopback mounts which may cause trouble under certain circumstances, especially during memory pressure:
  JAGad88815, fixed with PHKL_25237 (11.00), PHKL_25238 (11.11)
  JAGad56173, fixed with PHKL_25525 (11.00), PHKL_27266 (11.11)
- Problems with NFS soft mounts:
  Do not NFS mount `/sapmnt` with the soft option (including local mount on relocatable address) as batches loose access on `/sapmnt/SID/global`. This is normally implemented on all sites. Not mandatory for `/usr/sap/trans`.

Shared Memory limits for 32bit

- The maximum of 1.75GB shared memory (2.75GB with SHMEM_MAGIC) that can be configured on a 32bit HP-UX system often leads to problems during package startup, when a considerable amount of shared memory is already in use or the available shared address space is fragmented.
- Shared memory shortage is often caused by failed instance shutdowns in conjunction with failed resource cleanup. Later revisions of SGeSAP offer enhanced ‘strict’ resource cleanup options.
- Never exhaust all space with your SAP R/3 configuration!
- Failover nodes may have less space available, testing is a must!
- Using 64bit configuration is highly recommended if possible.

SGeSAP Patching

- SGeSAP patches only patch the file `/opt/cmcluster/sap/sap.functions`. This needs to be copied manually to `/etc/cmcluster/`.
- Proper NFS/ONC+ patches are crucial for SGeSAP.
- The SAP Application Server handling will not work correctly with newer revisions of the r-commands (e.g. >= PHNE_17030 for 11.00). The problem is fixed with SGeSAP 3.03. For older revision SGeSAP patches are available.
- SAP package start hangs on a ping command if an unpatched ServiceGuard 11.12 is in use. You need to apply SG 11.12 patch PHSS_22541 or later to fix this issue.

Additional Information

http://docs.hp.com/hpux/ha/#doc, Manuals and Release Notes

http://haweb.cup.hp.com/Support (HP internal), ACSL Division

http://wtec.cup.hp.com/~hpuxha (HP internal), HA WTEC

http://wtec.cup.hp.com/~hpuxha/newsletter/index.htm (HP internal), HA WTEC Newsletter

ftp://wtec.cup.hp.com/dist/HPUXHA/TRAINING (HP internal), SG Internals Training