# Chapter 19
# -
# Serviceguard

# INDEX

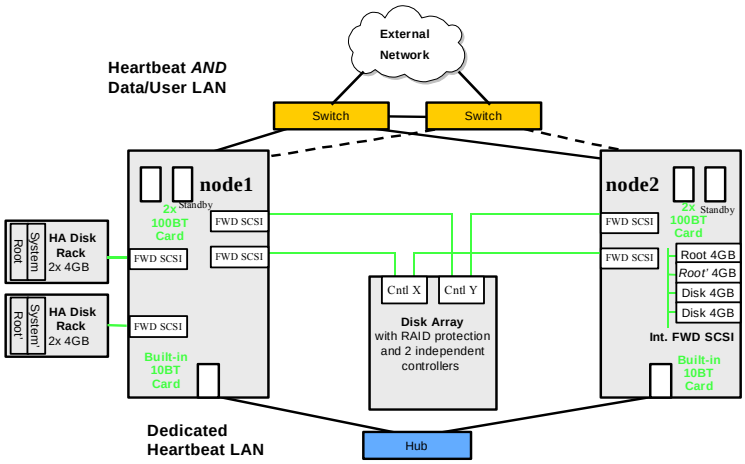
This chapter wants to briefly introduce High Availability solutions using Serviceguard and how to setup, maintain and trouble-shoot them. It is *not* supposed to replace the official documentation, especially the manual *Managing Serviceguard* (B3936-90065), downloadable from http://docs.hp.com/hpux/ha/index.html#Serviceguard.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

# Introduction

Serviceguard allows you to create high availability clusters of servers running HP-UX[1] or Linux[2]. 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. 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 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, 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)




[1] HP-UX on PA-RISC and IA64 , only selected configurations supported. Not all features available on IA64.
[2] Only selected distributions on selected configurations. Not all features available on Linux.

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 B, 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. 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.

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 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 Serviceguard to provide the greatest degree of availability.

The Cluster Lock (either a Cluster Lock Disk or a Quorum Server) function is used to remediate a cluster breakdown (arbitration). Cluster vitality is determined by all nodes passing heartbeat packets to the cluster manager by way of the heartbeat (HB) LAN. If a HB network failure occurs which isolates some nodes from the rest, cluster communication (and status synchronization) is lost. Serviceguard endeavors to form a reduced cluster of known functioning nodes to adopt packages from nodes whose status is no longer known. To determine the members of the new cluster, Serviceguard uses the **Quorum Rule**:

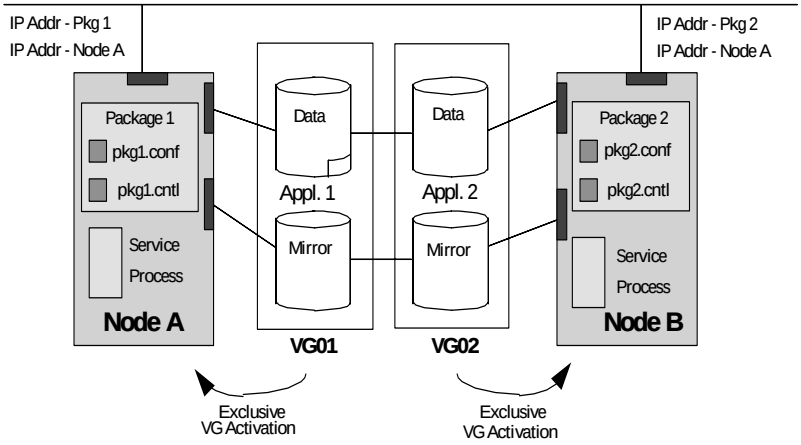
- **Greater than 50%** of nodes can interact:
  These nodes reform a new cluster.
- **Exactly 50%** of nodes can interact:
  The Cluster Lock Disk or Quorum Server is approached to negotiate the new cluster.
- **Less than 50%** of nodes can interact:
  These node are forced to TOC to preserve data integrity.

The white paper *Arbitration For Data Integrity in Serviceguard Clusters* describes some basic cluster membership concepts, then discusses some different ways of determining cluster membership. Several types of arbitration are shown with example configurations that illustrate each type. Then the arbitration methods used by Serviceguard and related products are presented in some detail. See http://docs.hp.com/hpux/pdf/B3936-90070.pdf.

## Example for a Package Configuration

Under normal conditions, a fully operating 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 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, 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 disk 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 Serviceguard revision A.11.09 with PHSS_23511. See also the Whitepaper *Integrating VERITAS VolumeManager (VxVM) with Serviceguard A.11.09,* http://docs.hp.com/hpux/pdf/B3936-90048.pdf. Beginning with Serviceguard revision A.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 currently 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 note 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 an 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 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
testcluster  up

  NODE         STATUS       STATE
  node1        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
  node2        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
    pkg1         up           running      enabled      node1

      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      node1
      Alternate    up           enabled      node2     (current)
```

## 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 node1

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:
       node2

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. The -f option causes all packages to be halted first automatically. If the user only wants to shutdown a subset of daemons, the `cmhaltnode` command should be used instead.

```
# 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 pkg1
This operation may take some time.
Halting cluster services on node node1
Halting cluster services on node node2
..
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 node1
..
cmhaltnode  : Successfully halted all nodes specified.
Halt operation completed.
```

## Starting Packages

To run a high availability package use the command `cmrunpkg`. `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 pkg1

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 pkg1
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`. Firstly 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 secondly 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 pkg1` | Globally enable switching for a package. |
| `# cmmodpkg –d pkg1` | Globally disable switching for a package. |
| `# cmmodpkg –n node1 –e pkg1` | Enable a node to run a package. |
| `# cmmodpkg –n node1 –d pkg1` | Disable a node to run a package. |

```
# cmmodpkg –e pkg1
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 trouble-shooting tool to identify the current configuration of a cluster, since it prints out an overview of its discovery results:

```
# cmquerycl -C cmclconfig.ascii -n node1 -n node2

Node Names:    node1
               node2

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 pkg1.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 pkg1.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 pkg1/pkg1.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/cmcluster/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 runs 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 pkg1/pkg1.conf

Begin cluster verification...
Modifying configuration on node node1
Modifying configuration on node node2

Modify the cluster configuration ([y]/n)? y
Modifying node node1 in cluster testcluster.
Modifying node node2 in cluster testcluster.
Completed the cluster creation.
```

## LVM Modifications of Cluster Volume Groups

Changes to the LVM configuration of cluster volume groups on the shared bus need some specific attention. Only the local node, where the LVM configuration changes were actually done, knows about the changes. There is nothing like an automatic configuration distribution to other cluster nodes.

The easiest way to make LVM configuration changes visible on other cluster nodes is to re-import the changed 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 the 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 logical volume (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* the change automatically, when they activate the VG.
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 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 logical volume (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 `/dev/`*`vg01`* with VG group special file 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 of the other cluster nodes (using for example rcp, ftp, etc.)

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

- Steps to be performed on each of the other nodes:

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`. 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 available as of HP-UX 11.00 with LVM commands patch PHCO_20870.

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
```

## Replacing a failed Quorum Server system

The Quorum Server is only used in the 50% scenarios described above. Therefore an unavailable Quorum Server will neither impact a running cluster nor the cluster determination in the other two scenarios. If the Quorum Server is offline when needed however, all nodes in that cluster will TOC.

**Procedure for replacing a defective Quorum Server system:**

1. Remove the old Quorum Server system from the network.
2. Bring in the new system and install the OS and configure it for the old Quorum Server's IP address. If the Quorum Server was configured using the old system's name, configure the new system with the same name.
3. Install and configure the Quorum Server software on the new system.
4. Start the Quorum Server.

**Notes and Warnings:**

1. Prevent the old Quorum Server from using the old IP on the network.
2. While the old Quorum Server is down and new one is being set up:
   - cmquerycl/cmapplyconf commands will not work.
   - cmruncl/cmhaltcl/cmrunnode/cmhaltnode will work.
   - If there is a node or a network failure which creates a 50/50 membership split, Quorum Server will not be available as a tie-breaker and all nodes in that cluster TOC.

3. cmhaltnode does not induce Serviceguard to consult the Quorum Server in order to reform the cluster.

If the procedure described above is used for replacing the Quorum Server, no changes will be necessary on the cluster nodes dependent on it.

For customers concerned about the TOC issue if the Quorum Server fails, another system can be loaded and configured with the Quorum Server software, but running on a different IP address (call this the standby Quorum Server). Then, if the original Quorum Server fails, disconnect it from the network, give it's IP address to the standby Quorum Server, and then reboot the standby server to take over the role of the original Quorum Server.

# Troubleshooting

## Where to find Logfiles

- Serviceguard logs lots of information to `/var/adm/syslog/syslog.log`. If nothing is found in this file, you should check if syslogd still works as expected (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 called with the argument *start*. The script is typicallly 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:                   pkg1
      package run script:             /etc/cmcluster/pkg1/pkg1.cntl
      package run timeout:            (No Timeout)
      package halt script:            /etc/cmcluster/pkg1/pkg1.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 cluster awareVGs 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 by fsck and are then 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 points 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`. Extreme caution should be exercised when executing this command outside the context of the package control script.
- **customer_defined_run_cmds**
  This is the place where the customer's HA application is started. Failures analysis in this area should be started 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 package's 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` and `cmhaltserv` commands *must not* be used manually from the command line.
- **start_resources**
  The Serviceguard command `cmstartres` is used to start monitoring of deferred 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 be needed to get the system back to a defined "package halted state" manually.

This includes:

- The services are stopped, which is usually the case.
- The application is halted.
- All relocatable IP adresses are de-configured. This can be done manually using e.g. `cmmodnet -r -i 192.10.10.120 192.10.10`.
- All associated filesystems are umounted. All processes keeping them busy need to be terminated first (use fuser and umount).. If the filesystem 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 filesystems 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 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 done by 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/vglock
   second lock vg name:                (not configured)
      Node name:                       node1
      first lock pv name:              /dev/dsk/c0t4d4
      first lock disk interface type:  c720
      Node name:                       node2
      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, that can 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
  ```

## 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 failover 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:

```
Serviceguard: Unable to maintain contact with cmcld daemon.
Performing TOC to ensure data integrity.
```

Beginning with HP-UX 11.22 this kind of information is also logged to the dumps INDEX and to /etc/shutdownlog to make it easier to tell that a TOC was initiated by Serviceguard. For HP-UX 11.00 and 11.11. kernel patches have been released to enable this supportability

feature:

- HP-UX 11.00: PHKL_27851.
- HP-UX 11.11: PHKL_28113 and PHKL_28114.

With the feature enabled the dump's INDEX file will show:

```
panic     SafetyTimer expired, pcsq.pcoq = 0.25ea68, isr.ior = 0.ffd0000
```

and the /etc/shutdownlog will show:

```
18:23   Thu Apr 24 2003.  Reboot after panic: SafetyTimer expired, ...
```

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` or `cmapplyconf`), 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 `cmviewconf` tool:

```
# cmviewconf

Cluster information:

   cluster name:                        testcluster
   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/vglock
   second lock vg name:                 (not configured)

Cluster Node information:

   Node ID 1:
      Node name:                        node1
      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 you need to specify the outgoing PPA (NMID for 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. The unsupported tool `dlpiping` (http://wtec.cup.hp.com/~hpuxha/tools, HP internal) can be used in such cases. Other than `linkloop` this tools uses exactly the same communcation mechanism that Serviceguard uses. Therefore 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 Serviceguard and its daemons.

1. **Polling packets used by cmclconfd** to explore the networking infrastructure at configuration time (e.g. `cmquerycl, cmcheckconf` and `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

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 |

| CAT | SG internal category | syslog(3C) Level |
|---|---|---|
| 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 Serviceguard. 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 Serviceguard 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 Serviceguard 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 Serviceguard, 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 should 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 Serviceguard A.11.03 (and with patches PHSS_15531 for Serviceguard A.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 Serviceguard commands.

To enable cmclconfd debug logging add the –T 5 option to the hacl-cfg lines in `/etc/inetd.conf`:

```
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.

## Using the Serviceguard Fight Recorder

Beginning with Serviceguard A.11.15 the Flight Recorder (SGFR) feature is available to extract extended logging information for trouble-shooting purposes.

### Obtaining recent Flight Recorder Logs online

The dumpfiles automatically created by SGFR are formatted and placed in `/var/adm/cmcluster/frdump.cmcld.x`, where x is a number, incremented from 0 to 9.

To retrieve a detailed log from dumped SFGR log, apply the cmfmtfr command:

```
# /usr/contrib/bin/cmfmtfr /var/adm/cmcluster/frdump.cmcld.x
```

This converts the specified SGFR binary file into readable form. The output goes to standard output.

**Obtaining Flight Recorder Logs from an HPUX crash dump**

To retrieve SGFR log from a system crash dump, use q4, the HP-UX crash dump analyzer.

Initialize q4 for debugging, e.g. using the Q4 located in /usr/contrib/Q4:

```
# . /usr/contrib/Q4/bin/set_env          'Source' the set_env script. Note the '.'!
# cd crashdir                                        e.g. /var/adm/crash/crash.0
# q4 -p
```

Load the cmfr.pl script in q4, using the include command.

```
q4> include /usr/contrib/lib/Q4/cmfr.pl
```

Still in q4, execute DumpFRB to extract the SGFR log buffer from the crash dump and puts it into an SGFR binary file, dumpfile.

```
q4> run DumpFRBin dumpfile
```

Quit q4 and apply the cmfmtfr command to convert the SGFR binary file, dumpfile, into readable form. The output goes to standard output.

```
# /usr/contrib/bin/cmfmtfr dumpfile
```

**Obtaining Flight Recorder Logs from a cmcld core**

To retrieve SGFR log from a cmcld core apply the cmcorefr command:

```
# /usr/contrib/bin/cmcorefr -o dumpfile /var/adm/cmcluster/core
```

where *-o dumpfile* extracts the SGFR log buffer from cmcld's core (usually located under /var/adm/cmcluster). This extracts a SGFR log buffer from the core, and it outputs a SGFR binary file. Apply the `cmfmtfr` command to convert the SGFR binary file, dumpfile, into readable form. The output goes to standard output.

```
# /usr/contrib/bin/cmfmtfr dumpfile
```

# Common Serviceguard Issues

## LVM related Problems

- 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. 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.**

  An attempt to read a valid PVID from the listed device failed.Check if the device is readable and contains a PVID ≠ 0 (see the LVM chapter how to check this).

- Error:
  **Unable to recv initialize VG message to %s: %s**
  or:
  **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 hangs or long runtimes are cmclconfd processes being blocked while trying to access disk devices. By default cmclconfd expects *every* claimed disk that is visible in ioscan to be readable wihtout problems. This assumption is not true in some cases, especially for devices associated to physical data replication (ContinousAccess/XP, BusinessCopy/XP, EMC SRDF, EMC Timefinder). Making that devices accessible (e.g. by splitting pairs) should solve related problems.

To sort out responsible devices watch out for cmclconfd processes runnnig with priority 148 (BIO). Use e.g. `ps -el | grep cmclconfd` on every cluster node. The cmclconfd's *Open Files* screen in GlancePlus or the public domain tool *lsof* should show you what device we are waiting for.

Another workaround is to temporarily remove all `VOLUME_GROUP` statements (except for cluster lock volume groups) form the cluster ASCII file (comment them out by prepending a '#' sign) **and** use the -k option with `cmcheckconf` or `cmapplyconf`.

- Syslog Warning:
  **`Failed to release volume group %s.`**

  This warning is logged by cmclconfd to syslog while probing the node's LVM configuration, because it is unable to close a opened volume group and release it from kernel memory. The problem has some undesired LVM related side effects, e.g. cluster lock initialization and volume group activation failures (**`Couldn't set the unique id`**) after using `vgchgid(1M)`. Make sure to have the following patches (or their successors) installed:
  - SG 11.09: PHSS_24850
  - SG 11.12: PHSS_24537
  - SG 11.13: PHSS_24678

- Error:
  **`Found two volume groups with the same name %s but different ids`**
  Warning:
  **`The volume group %s is activated on more than one node`**

  This problem typically occures when Serviceguard considers "privat" volumes groups to be "shared", which happens often to the root volume group /dev/vg00.

  Serviceguard uses LVM's VGID as unique differentiator to find shared volume groups. In general there are two scenarios why Serviceguard may get misleading information about volume groups, complaining about their configuration:

  - The systems were cloned, meaning that their private disks were copied using tools like dd(1). This causes the VGIDs to be identical in their LVM headers, although they should be unique. See the LVM Chapter for information how to check the VGID of a disk.
  - The VGID in the /etc/lvmtab file does not match the VGID of attached disks. Copying the lvmtab file to another system might cause this problem; vgscan(1M) can be used to fix the file.

- Symptom:
  **Some LVM volume groups are not automatically activated during bootup.**

  This is not really a Serviceguard problem, but typically happens on clusters only. By default HP-UX tries to activate all LVM volume groups during bootup by running the /sbin/lvmrc script, which is customizable by editing /etc/lvmrc. Sometimes `AUTO_VG_ACTIVATE=0` is configured in /etc/lvmrc to get rid of error messages caused by failed activation attempts (*vgchange –a y* vs. *–a e*). In this case all non-cluster volume

groups need to be added manually to the function `custom_vg_activation()` in /etc/lvmrc.

## Networking related Problems

- 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 hostname and 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 is a good idea to also check `/etc/inetd.conf`, `/etc/services` and `/var/adm/inetd.sec`.

  Correct `/etc/inetd.conf` entries look like this:

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

  These are the valid `/etc/services` entries:

```
clvm-cfg    1476/tcp            # HA LVM configuration
hacl-hb     5300/tcp            # High Availability (HA) Cluster heartbeat
hacl-gs     5301/tcp            # HA Cluster General Services
hacl-cfg    5302/tcp            # HA Cluster TCP configuration
hacl-cfg    5302/udp            # HA Cluster UDP configuration
hacl-probe  5303/tcp            # HA Cluster TCP probe
hacl-probe  5303/udp            # HA Cluster UDP probe
hacl-local  5304/tcp            # HA Cluster Commands
hacl-test   5305/tcp            # HA Cluster Test
hacl-dlm    5408/tcp            # HA Cluster distributed lock manager
```

  Check this from every node to every other node of the cluster:

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

  The telnet commands should simply hang, which is the normal behaviour if you reach the remote `cmclconfd`. Messages like *connection refused* or *hacl-cfg: bad port number* indicate a problem.

- Error:
  **`Non-uniform connections detected`**

  This error is usually the result of a misconfiguration of a network component. The dominant errors are mismatching speed/duplex of a LAN interface, sometimes even assigning the same link level address to more than one interface.

  The problem occures while Serviceguard probes the link level connectivity between all interfaces that are part of the cluster configuration. The probing result needs to be always symmetrical. Traffic needs to pass either in both or no direction! For trouble-

shooting see section How to track down Networking Problems. Please note that often `linkloop(1M)` is not able to catch such problems since its checking differs from Serviceguard's.

As of Serviceguard 11.09 patch PHSS_21425 the probing is much more robust, epsecially important for large configurations with heavy network traffic (see also JAGad05639).

- Error:
  **`Network interface %s on node %s couldn't talk to itself.`**

  This message is usually a result of a failed LAN interface check, either on link or on IP level. Each interface being part of the cluster configuration needs to be either configured (UP and ping'able through its IP adress) or unconfigured (unplumbed standby interface, without having any IP adress).

  If the interface is supposed be configured with `HEARTBEAT_IP` or `STATIONARY_IP` you should check with e.g. `ping(1M)` if it is up and running. Otherwise, for standby interfaces, you should check with `ifconfig(1M)` if it is really unplumbed. It should return *no such interface*. E.g., to unplumb lan3 use (valid for HP-UX ≥10.30):

  ```
  # ifconfig lan3 unplumb
  # ifconfig lan3
  ifconfig: no such interface
  ```

  It's also important that every interface is able to communicate on link level. Verify this for each reported interface using the linkloop(1M) command:

  ```
  # linkloop –i <PPA> <MAC address>
  ```

  e.g. for lan2 assuming the following lanscan output:

  ```
  Hardware Station        Crd  Hdw   Net-Interface   NM   MAC     HP-DLPI DLPI
  Path     Address        In#  State NamePPA         ID   Type    Support Mjr#
  10/4/8   0x080009DC6151 0    UP    lan0 snap0      1    ETHER     Yes   119
  10/12/6  0x080009F08A5A 2    UP    lan2 snap2      2    ETHER     Yes   119

  # linkloop –I 2 0x080009F08A5A
  ```

  The –i 2 option specifies lan2 as outgoing interface. For testing lan2 with itself 0x080009F08A5A must be specified as destination MAC address.

  **Note:** Some Ignite revisions are known to recover a bogus "0.0.0.0" configuration for unused interfaces to `/etc/rc.config.d/netconf`. Remove that entry and unplumb the interface as described above.

- Error:
  **`Detected a partition of IP subnet %s.`**

  The error indicates that some lan interfaces are unable to talk to each other on link level, although they should be able to do so. Verify if the configuration information in the cluster ASCII file is correct. Check the network's physical connections. A failure with the `linkloop` command means there is no connectivity on link level, maybe

because some network component such as a switch does not pass that type of traffic.

- DLPI errors:

```
DLPI ack error for primitive %d, errno %d, unix errno %d
Unexpected DLPI primitive %d
Unable to send DLPI attach request to ppa %d, %d: %s
Unable to get DLPI attach ack from ppa %d, %d: %s
Unable to add DLPI interface for %s
Unable to send DLPI message, %s
Failed to send over DLPI
unexpected DLPI failure
DLPI send error! dl_errno: %d, dl_unix_errno: %d
DLPI error! dl_errno: %d, dl_unix_errno: %d
...
```

  Serviceguard uses DLPI (Datalink Provider Interface) to perform network polling in order to check the health of the lan cards in the cluster. Link level packets are sent and received which allows cmcld to gather statistical information to ensure data is being transmitted and received and to check for errors returned by the network drivers.

  Discussing all possible DLPI error conditions would exceed this document's scope. It is usually best practice to check affected interfaces with tools like `linkloop(1M)` and `lanadmin(1M)`. In the past many of those problems were tracked down to defective hardware components. Some of those problem were caused by Serviceguard defects, so it is also advisable to install a current Serviceguard patch to address known problems in this area. See also section How to track down Networking Problems.

- DLPI warnings:

```
DLPI message too small (%i < %i + %i).  Ignoring the message.
DLPI message too big (%i + %i).  Ignoring the message.
Length of DLPI header (%d) is too small. Ignoring the message.
Net_id %d received a DLPI message with an incorrect checksum.
Ignoring the message.
...
```

  Beginning with Serviceguard A.10.12 (with PHSS_21995) and A.11.09 (with PHSS_22540) DPLI traffic is protected against corruption using checksums. Earlier revisions could abort under such conditions resulting in a node TOC. The affected hardware (interfaces, network switch or hub, other hardware on that bridged net, etc.) should be checked and replaced if needed. See also HA Newsletter #15 (HP internal) for details.

- Problem:
  **Crossover cables for a Heartbeat LAN of 2-node clusters**

  When either LAN card fails, or the crossover cable is disconnected, both LAN cards go down. This is because the electrical signals necessary for the cards to determine that a valid LAN connection exists are not present. The result is that since both nodes appear to have a bad LAN card, Serviceguard may TOC the wrong node. On multi-speed cards, such as 10/100Base-T, the cards should *not* negotiate which speed will be used when the system boots up. Otherwise, if only one system is booted and the remote system is down, the negotiation would fail, and the card would not be enabled at all. So when the second node eventually comes up, it's LAN would also be down.

For the reasons listed above, it is *not* recommend (but supported) to use crossover cables for Serviceguard configurations. At least a hub should be added to isolate the interfaces electrically and auto-negotiation should be disabled by setting a fixed speed/duplex configuration.

## Intermittent Cluster Reformations

- Problem:
  **Intermittent cluster reformations with possible node TOC**

  If a node does not receive a heartbeat from a remote node within the `NODE_TIMEOUT` interval, then that node will be *timed out*. At that time all cluster nodes enter the cluster reformation process.

  If the heartbeat interval is one second, and the node timeout interval is two seconds, it takes two consecutive missed heartbeats to cause the node to time out, and a cluster reformation to start. Cluster reformation involves informing all nodes of the reformation (including the node which missed the heartbeat), voting for a new cluster coordinator, and reforming the cluster (the new cluster is based upon the number of nodes which responded during the reformation.

  Note, if the node which missed the heartbeat is able to respond during the reformation, then the reformation will end up with the same number of nodes in the cluster and your packages will not be effected).

  Example:
  Node node1 is missing heartbeats from node2. It starts reformation and goes for the cluster lock disk, before node2 comes back:

```
Aug 5 16:08:29 node1 cmcld: Timed out node node2. It may have failed.
Aug 5 16:08:29 node1 cmcld: Attempting to form a new cluster
Aug 5 16:08:36 node1 cmcld: Obtaining Cluster Lock
Aug 5 16:08:36 node1 vmunix: SCSI: Reset requested from above --
                                lbolt:25093597, bus:0
Aug 5 16:08:37 node1 vmunix: SCSI: Resetting SCSI -- lbolt:25093697, bus:0
Aug 5 16:08:37 node1 vmunix: SCSI: Reset detected -- lbolt:25093697, bus:0
Aug 5 16:08:54 node1 cmcld: Attempting to adjust cluster membership
Aug 5 16:08:56 node1 cmcld: Enabling safety time protection
Aug 5 16:08:56 node1 cmcld: Clearing Cluster Lock
Aug 5 16:08:57 node1 cmcld: 2 nodes have formed a new cluster, sequence #7
Aug 5 16:08:57 node1 cmcld: The new active cluster membership is:
                                node1(id=1), node2(id=2)
```

  Often the factory-default setting for the cluster parameter `NODE_TIMEOUT` of 2 seconds (2000000 microseconds) is too small for many configurations. The general recommendation is to set `NODE_TIMEOUT` in the range of 5000000-8000000 microseconds. See also the Section About Serviceguard TOCs.

- Problem:
  **SCSI reset messages logged to syslog and the kernel's message buffer**

  If Serviceguard performs a cluster reformation, SCSI reset messages appear in the

`dmesg` output and `syslog.log`, if the reformation requires a race to the cluster lock disk. To ensure that the SCSI bus is available for the server to grab the lock, a reset is performed, (please note the "*Reset requested from above"* message), e.g.:

```
SCSI: Reset requested from above -- lbolt: 400804081, bus: 0
SCSI: Resetting SCSI -- lbolt: 400804381, bus: 0
SCSI: Reset detected -- lbolt: 400804381, bus: 0
```

The SCSI messages are only seen when the cluster lock disk is handled by the `sdisk` driver. The `disc3` driver does not output information when a SCSI reset is issued or detected.

- 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.

## Cluster daemon abort with possible node TOC

- Active cmcld aborts with syslog messages like:
  **`cmcld: Aborting! %s (file: %s, line: %d)`**
  **`cmcld: Aborting: %s %d (%s)`**
  **`cmcld: Service Guard Aborting!`**
  **`cmcld: Aborting Serviceguard Daemon to preserve data integrity.`**

  These messagesa are logged by cmcld before it *actively* aborts due to some fatal error condition, that may be also part of the error message. Typically the `syslog.log` looks similar to this:

```
Aug  5 11:05:31 node1 cmcld: Aborting: cl_rwlock.c 1030 (reader/writer lock
not locked)
Aug  5 11:05:35 node1 cmlvmd: Could not read messages from /usr/lbin/cmcld:
Software caused connection abort
Aug  5 11:05:35 node1 cmlvmd: CLVMD exiting
Aug  5 11:05:35 node1 cmsrvassistd[8688]: The cluster daemon aborted our
connection.
Aug  5 11:05:35 node1 cmsrvassistd[8688]: Lost connection with Serviceguard
cluster daemon (cmcld): Software caused connection abort
Aug  5 11:05:35 node1 cmtaped[8691]: The cluster daemon aborted our
connection.
Aug  5 11:05:35 node1 cmtaped[8691]: cmtaped terminating. (ATS 1.14)
...
```

  The exact error message should be checked against known problems. Often also a cmcld core file is written to the directory `/var/adm/cmcluster`, which may be helpful for further investigation.

- Passive `cmcld` aborts without `cmcld syslog.log` message.

  There are known problems where `cmcld` crashes without logging additional information to `syslog.log` (e.g. due to receiving `SIGSEGV` or `SIGBUS`). In this case you may only find messages from `cmlvmd, cmsrvassistd` etc. indicating that the

connection to `cmcld` was lost. Please note that DLPI corruption may also cause such aborts on earlier Serviceguard revisions, see above.

However, there should be a `cmcld` core file in `/var/adm/cmcluster`, which may be helpful for further investigation. It is advisable to install a current Serviceguard patch to address all known issues in this area.

There is a known kernel defect affecting HP-UX 11.11 multiprocessor systems (see `clock.c` issues JAGad94281, JAGae25547) which causes unexpected cmcld aborts. In this case a `cma_dump.log` file is written to the directory `/var/adm/cmcluster`. The first lines contain:

```
%Internal DCE Threads problem (version CMA BL10+), terminating execution.
% Reason: cma__io_available: unexpected select error
```

The patch PHKL_25869 fixes the problem for 64bit systems. A superseding patch including the final fix also for 32bit systems is expected to be released soon.

- Node TOC after tuning TCP parameters using ndd(1M):

```
Oct 17 10:10:02 node1 cmlvmd: Could not read messages from /usr/lbin/cmcld:
Connection reset by peer
Oct 17 10:10:02 node1 cmlvmd: CLVMD exiting
...
Oct 17 10:10:08 node1 vmunix: Halting node1 to preserve data integrity
Oct 17 10:10:08 node1 vmunix: Reason: LVM daemon failed
```

It is officially unsupported to change the TCP settings on a Serviceguard system. The specific `ndd(1M)` parameters involved here are `tcp_keepalive_interval` and `tcp_ip_abort_interval`. The same applies for the 10.X `nettune(1M)` parameters `tcp_keepstart`, `tcp_keepfreq` and `tcp_keepstop`.

The reason is that Serviceguard needs all ports to behave the same way, and if `ndd(1M)` is run after starting the cluster, then some ports will use the original TCP behavior, while all other ports established after running `ndd(1M)` use the new behavior. It has been found that usually no problems arise if the `ndd(1M)` tuning is done *before* the cluster is started, but nevertheless, this neither tested nor supported.

## SGeSAP related Problems

The *Serviceguard Extension for SAP R/3* (SGeSAP) is one of the most important integrations, 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 Serviceguard Extension for SAP R/3, downloadable from http://docs.hp.com/hpux/ha. In the following section the most common issues with this 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. Otherwise access is not granted resulting in *permission denied* errors.
- 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` or `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 an error "`tranport directory invalid`" 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.

- 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: PHKL_25237 (11.00), PHKL_25238 (11.11)
  JAGad56173, fixed: PHKL_25525 + PHNE_24909 (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.

### Hanging package start due to remsh problems

- SGeSAP uses the remsh command to start and stop remote application server instances. The implementation up to SGeSAP 3.06 depends on a specific remshd behaviour introduced with JAGba73645 for HP-UX 11.11 (initial release), 11.00 (with PHNE_17030) and 10.20/800 (with PHNE_20748).

  Beginning with PHNE_23087 (11.11) and PHNE_23003 (11.00) this behaviour is disabled by default (see JAGad84516). A new option `-m` for remshd is available to get the behaviour needed by SGeSAP. The change needs to be done to `/etc/inetd.conf`. Remember to re-read the config using `inetd -c`.

  ```
  shell        stream tcp nowait root /usr/lbin/remshd   remshd -m
  ```

  The problem is fixed with SGeSAP 3.07. (see JAGae13065 for details).

### Problems with application server packages

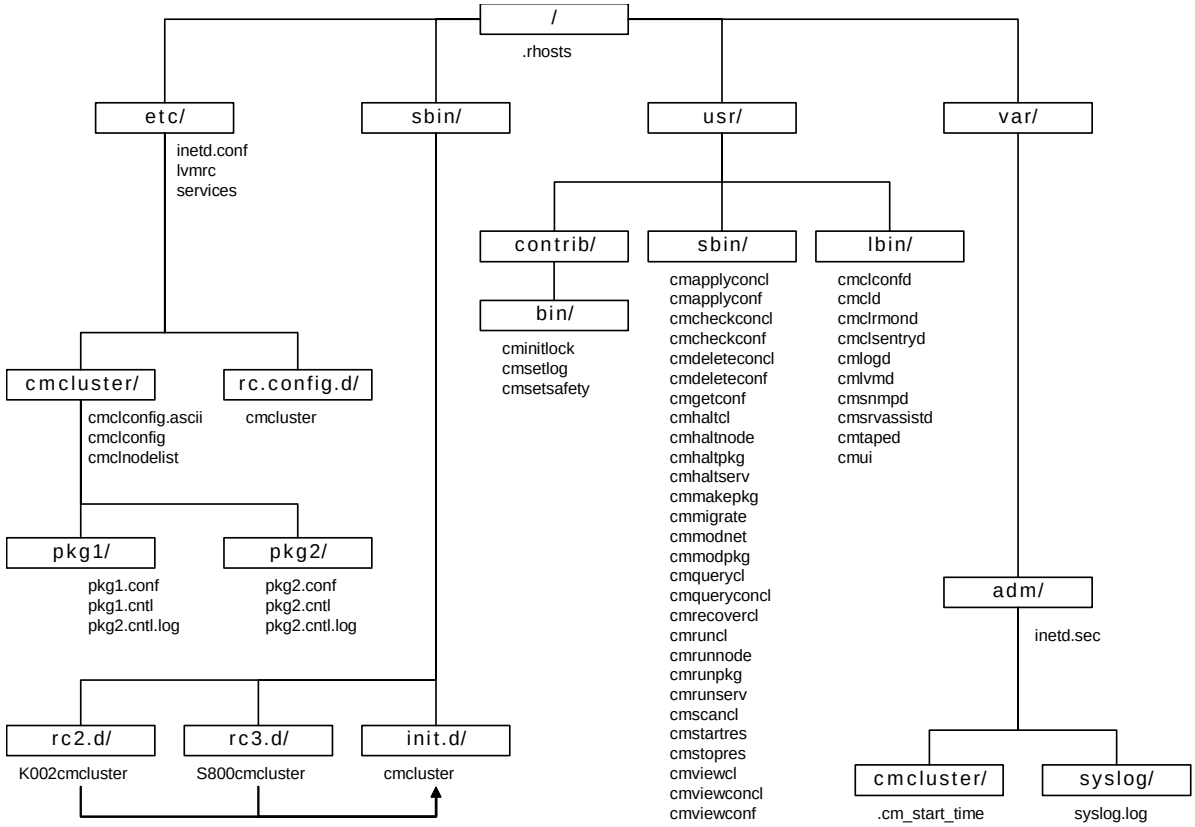
- Beginning with SGeSAP 3.03 application server (AS) packages are supported. However, it is recommended to use at least SGeSAP 3.08, to be released with AR0603. (see JAGae58065, JAGae58067, JAGae58074). Typical problems are AS instances being being halted using stopsap instead of cmhaltpkg, deadlocks during AS package start (if `WAIT_OWN_AS=2`), incorrect restart and failover behaviour of AS packages.
- The `AS_HOST[@]` variable of AS packages need to be set to the stationary hostname of the node where the AS package usually resides. Do *not* include the relocatable name here, as indicated by older revisions of the manual Managing Serviceguard Extension for SAP R/3. If desired, the primary node's `sap.conf` may point to the standby node, while the standby's `sap.conf` points to the primary node (symmetrical configuration). In this case individual `sap.conf` files need to be maintained on the cluster nodes.

### SGeSAP Patching

- SGeSAP patches only change the file `/opt/cmcluster/sap/sap.functions`. This needs to be copied manually to `/etc/cmcluster`.
- Current NFS/ONC+ patches are crucial for SGeSAP.

- SAP package start hangs on a ping command if an unpatched Serviceguard A.11.12 is in use. You need to apply Serviceguard A.11.12 patch PHSS_22541 or later to fix this issue.

## Serviceguard related Files



# 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 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 |

## Additional Information

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

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

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

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

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