Chapter 16
-
LVM

INDEX

Terminology 3

LVM Structural Information 3
- PVRA, VGRA and BDRA ... 3
- LIF Header and LIF Volume ... 4
- PV-ID and VG-ID ... 4
- vgcfgbackup(1M) ... 5
- /etc/lvmtab and vgscan(1M) ... 6

LVM Related Parameters and Limitations 7
- LVM parameters ... 7
- Supported file and file system sizes ... 7

Display Commands 8
- Information on VGs ... 8
- Information on PVs ... 9
- Information on LVs ... 9

LVM Basic Functionality 10
- Adding a new PV / VG / LV ... 10
- Modifying a PV / VG / LV ... 12
- Removing a PV / VG / LV ... 13
- Moving physical extents ... 14
- Importing and exporting VGs ... 14

MirrorDisk/UX 16
- Basic functionality ... 16
- Physical Volume Groups - PVGs ... 17
- Root Mirror ... 18

PV Links 18
- What is it? ... 18
- Configuring PV Links ... 18
- Changing PV Link order ... 20
- Utility cmpdisks ... 20

Offline Diagnostics (ODE) 21

Changing the HW Address of a Disk 23

LVM and MC/ServiceGuard (Cluster LVM) 24

Replacing a Failed Disk 26
- Identifying the failed disk ... 26
- Standard (non-hot-swappable) disk ... 27
- Hot-swappable disks ... 30

Removing a „ghost disk“ From a VG - the PV Key 33
- What is a ghost disk ... 33
- The PV key ... 35

Increasing the Root LVs 38
- using Ignite-UX ... 38
- using the unofficial procedure ... 38

SUBPROCEDURE 1: create LIF and BDRA on the disk 41

Commands Overview 42

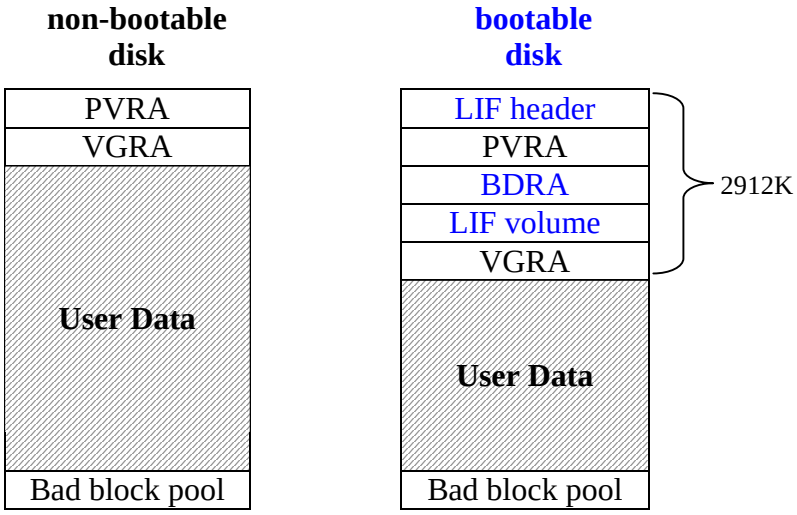
Terminology

The following abbreviations are common in LVM and will be used during this chapter:

VG	=	Volume Group
LV	=	Logical Volume
PV	=	Physical Volume
PVG	=	Physical Volume Group
PE	=	Physical Extent
LE	=	Logical Extent
FS	=	Filesystem
VGRA	=	Volume Group Reserved Area
PVRA	=	Physical Volume Reserved Area
BDRA	=	Boot Data Reserved Area

LVM Structural Information

The LVM structural information resides on reserved areas (PVRA, VGRA) at the beginning of any disk in the VG and is called the *LVM header*. The following image shows the *on disk structure* of an LVM disk:



NOTE: the LVM header of a bootable disk is always **2912 KB**. The header size of a non-bootable disk is not fixed. It depends on the number of PVs, size of PEs, etc. but it is usually smaller. The LVM header has to fit in one PE.

PVRA, VGRA and BDRA

The **PVRA** is unique for every PV in the VG. It contains:

PV-ID, VG-ID, number of this PV, PE size,
start and length of: VGRA, BDRA (if any), User Data and bad block pool
and in case of a ServiceGuard Cluster: Cluster ID, Cluster lock area start and cluster lock

flag.

The **VGRA** is identical for any PV in the VG. It contains:

VG-ID, max. number of PVs/LVs in VG, number of PVs in VG, max. number of PEs per PV, timestamp (time of last write to VGRA).
per PV information: PV-ID, number of PEs, PV flags, ...
per LV information: max. size of LV, LV flags, LV schedule strategy, max. number of mirrors, number of stripes and stripe size, ...

The **BDRA** contains boot relevant information:

number of PVs in root VG, list of PVs in root VG, root VG major/minor number, root LV major/minor numbers, what are the root, boot, swap and dump LVs, status information, timestamp (time of last write to BDRA).

LIF Header and LIF Volume

LIF stands for *Logical Interchange Format*. The *LIF header* resides in the first 8 KB of any LVM boot disk. It contains the directory to the *LIF volume* that begins after the BDRA. It can be displayed using lifls(1M):

```
# lifls -l /dev/rdsk/c1t6d0
volume ISL10 data size 7984 directory size 8
filename   type   start   size     implement  created
================================================================
ISL        -12800 584     306      0          00/11/08 20:49:59
AUTO       -12289 896     1        0          00/11/08 20:49:59
HPUX       -12928 904     848      0          00/11/08 20:50:00
PAD        -12290 1752    1580     0          00/11/08 20:50:00
LABEL      BIN    3336    8        0          99/10/08 02:48:02
```

The LIF volume contains files necessary to boot: ISL, HPUX, LABEL and AUTO (for automatical boot). Look at the Boot Chapter in order to get a detailed explanation of each LIF file.

PV-ID and VG-ID

Any PV has a unique 8 byte long identifier - the PV-ID. The VG-ID is a unique identifier for the VG that this PV belongs to. It is also 8 byte long. Their values are stored in the PVRA.

The (contributed) utility lvm11 displays the LVM header:

```
# lvm11 -p -d /dev/rdsk/c1t2d0 | more
...
...
/* The physical volume ID.      */ 2000252410 965817345i.e.
pvcreate(1m) was run on CPU with ID 2000252410 at Wed Aug  9 12:35:45
2000
/* The volume group ID.         */ 2000252410 965817462i.e.
vgcreate(1m) was run on CPU with ID 2000252410 at Wed Aug  9 12:37:42
2000
...
```

Since the lvm11 tool may not always be available you can also read out PV-ID and VG-ID

using “in core” commands that are available on any HP-UX system.

For example using xd(1):

```
# xd -j8200 -N16 -tu /dev/rdsk/c1t2d0
0000000    2000252410     965817345    2000252410     965817462
           PV CPU-ID       PV time      VG CPU-ID       VG time
```

i.e:
pvcreate and vgcreate were running on CPU with systemID 2000252410 (see uname -i).
pvcreate was running at timestamp 965817345 (this is in seconds after 1st Jan.1970)
vgcreate was running at timestamp 965817462 (117 seconds later)

or using adb(1):

PV-ID:

```
# echo "0d8200?DY" | adb /dev/dsk/c1t2d0
2008:           2000252410      2000 Aug  9 12:35:45
```

VG-ID:

```
# echo "0d8208?DY" | adb /dev/dsk/c1t2d0
2010:           2000252410      2000 Aug  9 12:37:42
```

vgcfgbackup(1M)

A copy of the LVM header is held within the filesystem in the LVM backup file (/etc/lvmconf/*.conf). Any modification of the LVM structure, e.g. through LVM commands like lvcreate, lvchange, vgextend, etc. will be automatically saved in the VGs config file through vgcfgbackup(1M).

NOTE (<= UX 10.01):
The backup will not automatically take place on systems running HP-UX 10.01 and before.

You can run vgcfgbackup(1M) manually at any time:

```
# vgcfgbackup vg00
Volume Group configuration for /dev/vg00 has been saved in
/etc/lvmconf/vg00.conf
```

The content of the backup file is binary but you can use the -l option of vgcfgrestore(1M) to display at least the disks belonging to the VG:

```
# vgcfgrestore -l -n vg00
Volume Group Configuration information in "/etc/lvmconf/vg00.conf"
VG Name /dev/vg00
 ---- Physical volumes : 1 ----
    /dev/rdsk/c1t6d0 (Bootable)
```

If the LVM header has been accidently overwritten or became corrupted on the disk you can recover it from this backup file using vgcfgrestore.

You usually use vgcfgrestore in case of a disk failure in order to write the LVM header from this backup file to the new disk:

```
# vgcfgrestore -n vg00 /dev/rdsk/c1t6d0
```

```
Volume Group configuration has been restored to /dev/rdsk/c1t6d0
```

NOTE: If you modify the LVM configuration but do not want the backup file to be updated, use the option "`-A n`" with the LVM command. Anyway - the previous configuration can be found in /etc/lvmconf/*.conf**.old**.

/etc/lvmtab and vgscan(1M)

The file /etc/lvmtab contains information about all VGs known to the system. The display commands above print their data based on this file. lvmtab is a binary file but you can display the printable characters in that file using the strings(1M) command:

```
# strings /etc/lvmtab
/dev/vg00
/dev/dsk/c2t0d0
/dev/vgsap
/dev/dsk/c4t0d0
/dev/dsk/c5t0d0
/dev/dsk/c4t1d0
/dev/dsk/c5t1d0
/dev/vg01
/dev/dsk/c6t0d0
```

NOTE: this is only the "visible" part of the lvmtab. It does also furthermore contain the VG-IDs, the total number of VGs, the number of PVs per VG and status information. Additional garbage characters printed by strings are not a problem as long as no important data is missing.

All VGs listed in lvmtab are automatically activated during system startup. This is done in the script /sbin/lvmrc, based upon configuration in /etc/lvmrc.
If you do not trust the information in the lvmtab anymore because it may have become corrupt somehow you can easily recreate it from PVRA and VGRA on the disks through the vgscan(1M) command. But be sure to save a copy before:

```
# mv /etc/lvmtab /etc/lvmtab.old
# vgscan -v
```

Warnings can usually be ignored.

NOTE: It's recommended not to copy but to move the lvmtab away, otherwise vgscan would try to repair the existing lvmtab which may fail under certain circumstances. If there is no /etc/lvmtab vgscan recreates it from the scratch.

ATTENTION: On a ServiceGuard system vgscan may fail. This is a known problem that is solved by LVM commands cumulative patches. The workaround is easy, just remove the file /dev/slvmvg before running vgscan.

NOTE: vgscan does not take care about the order of alternate links! It may be necessary to switch the links afterwards (see section PV Links below).

LVM Related Parameters and Limitations

LVM parameters

There are several parameters that limit the sizes of certain structures. Here are the most important ones:

Parameter	Default	Maximum	set by
max. number of VGs	10	256	kernel tunable **maxvgs**
number of PVs per VG	16	255	vgcreate -p <max_pv>
number of LVs per VG	255	255	vgcreate -l <max_lv>
PE size (2^X)	4 MB	256 MB	vgcreate -s <pe_size>
LV size	0 MB	16 TB	lvcreate -L <MB>
max. number of PE per PV	1016 PEs	65535 PEs	vgcreate -e <max_pe>

Supported file and file system sizes

Although it may be possible to create files or file systems larger than these documented limits, such files and file systems are not supported and the results of using them may be unpredictable.

JFS (VxFS) supported sizes

HP-UX Release	HP JFS Version	Veritas Disk Layout Version	Maximum File Size	Maximum File System Size
UX 10.01	JFS 2.0	**Version 2**	2 GB	4 GB
UX 10.10	JFS 2.0	**Version 2**	2 GB	128 GB
UX 10.20	JFS 3.0	Version 2	2 GB	128 GB
		Version 3	128GB	128 GB
UX 11.00	JFS 3.1	Version 2	2 GB	128 GB
		Version 3	1 TB	1 TB
	JFS 3.3	Version 2	2 GB	128 GB
		Version 3	1 TB	1 TB
		Version 4	1 TB	1 TB
UX 11.11	JFS 3.3	Version 2	2 GB	128 GB
		Version 3	2 TB	2 TB
		Version 4	2 TB	2 TB

bold default disk layouts for particular HP-UX Release/JFS version.

NOTE: for UX 11.00 with disk layout version 3 PHKL_22719 (or newer) is needed to avoid mount problems if extending or creating file systems beyond 128 GB.

and just to be complete

HFS supported sizes

HP-UX Release	Maximum File Size	Maximum File System Size
UX 10.01	2 GB	4 GB
UX 10.10	2 GB	128 GB
UX 10.20	128 GB	128 GB
UX 11.00	128 GB	128 GB
UX 11.11	128 GB	128 GB

NOTE: As of UX 10.20 it is possible to exceed the 128 GB limit to 256 GB, but it is not supported.

Display Commands

To display information about VGs, LVs or PVs there is a set of commands available. Each of the commands provides an option -v to display detailed (verbos) output.

Information on VGs

```
# vgdisplay -v vg01

--- Volume groups ---
VG Name                     /dev/vg01
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      1
Open LV                     1
Max PV                      16
Cur PV                      1
Act PV                      1
Max PE per PV               1016
VGDA                        2
PE Size (Mbytes)            4
Total PE                    508
Alloc PE                    508
Free PE                     0
Total PVG                   0
Total Spare PVs             0
Total Spare PVs in use      0

   --- Logical volumes ---
   LV Name                     /dev/vg01/lvol1
   LV Status                   available/syncd
   LV Size (Mbytes)            2032
   Current LE                  508
   Allocated PE                508
   Used PV                     1


   --- Physical volumes ---
   PV Name                     /dev/dsk/c10t6d0
```

```
   PV Status                 available
   Total PE                  508
   Free PE                   0
   Autoswitch                On
```

vgdisplay is useful to check wether the LVM configuration in memory is clean or not. First of all there should be no error messages. The status should be available or available/exclusive for ServiceGuard VGs. Cur PV should equal Act PV and Cur LV should be equal to Open LV.

Information on PVs

```
# pvdisplay -v /dev/dsk/c0t6d0 | more

--- Physical volumes ---
PV Name                     /dev/dsk/c0t6d0
VG Name                     /dev/vg00
PV Status                   available
Allocatable                 yes
VGDA                        2
Cur LV                      9
PE Size (Mbytes)            4
Total PE                    1023
Free PE                     494
Allocated PE                529
Stale PE                    0
IO Timeout (Seconds)        default

--- Distribution of physical volume ---
LV Name            LE of LV  PE for LV
/dev/vg00/lvol1    25        25
/dev/vg00/lvol2    25        25
/dev/vg00/lvol3    50        50

--- Physical extents ---
PE   Status   LV                  LE
0000 current  /dev/vg00/lvol1     0000
0001 current  /dev/vg00/lvol1     0001
0002 current  /dev/vg00/lvol1     0002
.
.
1021 free                         0000
1022 free                         0000
```

Stale PE should be 0.

Information on LVs

```
# lvdisplay -v /dev/vg00/lvol1 | more

--- Logical volumes ---
LV Name                     /dev/vg00/lvol1
VG Name                     /dev/vg00
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               0
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            100
```

```
Current LE                  25
Allocated PE                25
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   off
Allocation                  strict/contiguous

--- Distribution of logical volume ---
PV Name            LE on PV  PE on PV
/dev/dsk/c0t6d0    25        25

--- Logical extents ---
LE   PV1                PE1  Status 1
0000 /dev/dsk/c0t6d0    0000 current
0001 /dev/dsk/c0t6d0    0001 current
0002 /dev/dsk/c0t6d0    0002 current
...
```

None of the LEs/PEs should have a stale status.

LVM Basic Functionality

Adding a new PV / VG / LV

Adding a new PV

A disk has to be initialized before LVM can use it. The pvcreate command writes the PVRA to the disk and such a disk is called a PV:

```
# pvcreate /dev/rdsk/c0t5d0
```

If there is a valid PVRA already on the disk (it could have been used wit LVM before) you will get the following error message:

```
# pvcreate: The Physical Volume already belongs to a Volume Group
```

If you are sure the disk is free you can force the initialization using the -f option:

```
# pvcreate -f /dev/rdsk/c0t5d0
```

NOTE: For bootable disks you have to use the -B option additionally. This preserves the fixed 2912KB space for the LVM header (see section LVM structural information). You can find the procedure how to make a disk bootable in the section Mirroring the root disk later in this chapter.

To add the PV to an existing VG do:

```
#vgextend vg01 /dev/dsk/c0t5d0
#vgdisplay -v vg01
```

Adding a new VG

Here's how to create a new VG with 2 disks:

1) initialize the disk if not yet done:

```
# pvcreate [-f] /dev/rdsk/c0t5d0
```

```
# pvcreate [-f] /dev/rdsk/c0t6d0
```

2) select a unique minor number for the VG

```
# ll /dev/*/group
crw-r--r--   1 root   sys   64 0x000000 Apr  4  2001 /dev/vg00/group
crw-r--r--   1 root   sys   64 0x010000 Oct 26 15:52 /dev/vg01/group
crw-r--r--   1 root   sys   64 0x020000 Aug  2 15:49 /dev/vgsap/group
```

3) create the *VG control file* (*group file*):

```
# mkdir /dev/vgnew
# mknod /dev/vgnew/group c 64 0x030000
```

NOTE: Starting with LVM commands patch PHCO_24645 (UX 11.00) or PHCO_25814 (UX 11.11) vgcreate and vgimport will check for the uniqueness of the group file's minor number.

4) create and display the VG:

```
# vgcreate vgnew /dev/dsk/c0t5d0 /dev/dsk/c0t6d0
# vgdisplay -v vg01
```

NOTE: One of the VG's parameters is **max_pe**, i.e the maximum number of physical extents this VG can handle per disk. The default value is 1016. Multiplying this with the default PE size of 4MB results in approx. 4GB disk space that can be handled by this VG. Adding a larger disk to this VG later is not possible. Believe me - there are absolutely no options to do this other than vgcreate! Anyway - vgcreate automatically adjusts max_pe in order be able to handle the largest PV given in the arguments. Its always a good idea to set max_pe explicitely to a value large enough to allow for future expansions. This can be done with the **-e** option of vgcreate.

Adding a new LV

The following commands create a 500MB large LV named lvdata on any disk(s) of the VG vg01:

```
# lvcreate -n lvdata -L 500 vg01
```

You cannot specify a PV with lvcreate. If you like to put the LV on a certain PV use the following:

```
# lvcreate -n lvdata vg01
```

this creates a LV of 0MB. It has no extents - it just exists. Now extend the LV onto a certain disk:

```
# lvextend -L 500 /dev/vg01/lvdata /dev/dsk/c4t2d0
```

Now you can use newfs to put a FS onto the LV:

```
# newfs -F <fstype> /dev/vg01/rlvdata
```

where fstype is either hfs or vxfs.

NOTE: Nowadays it is always recommended to use a VxFS (=JFS) filesystem.

Modifying a PV / VG / LV

Modifying a PV

There are certain PV parameters that can be changed (see pvchange man page). A frequently used parameter is the IO timeout parameter. This parameter tells LVM how long to wait for disk transactions to complete before taking the device offline. This is accompanied by POWERFAILED messages on the console. Certain disk arrays need a higher timeout value than simple disks. To specify e.g. a timeout of 120 seconds do:

```
# pvchange -t 120 /dev/dsk/c#t#d#
```

The device driver's default is usually 30 seconds. Setting the IO timeout to 0 seconds restores this default:

```
# pvchange -t 0 /dev/dsk/c#t#d#
```

Modifying a LV

The most common modifiaction task is the modification of the size of a LV. To increase a LV from 500MB to 800MB do:

```
# lvextend -L 800 /dev/vg01/lvdata [/dev/dsk/c5t0d0]
```

NOTE: You may get the following error:

```
lvextend: Not enough free physical extents available.
Logical volume "/dev/vg01/lvdata" could not be extended.
Failure possibly caused by contiguous allocation policy.
Failure possibly caused by strict allocation policy
```

The reason for that is exactly one of the above.

If the LV has been extended successfully you need to increase the FS that resides on that LV:

Without OnlineJFS you have to umount the FS first:

```
# umount /dev/vg01/lvdata
# extendfs /dev/vg01/rlvdata
# mount /dev/vg01/lvdata <mountpoint>
```

With OnlineJFS you do not need to umount. Use fsadm instead:

```
# fsadm -b <new size in KB> <mountpoint>
```

NOTE: Reducing a LV without OnlineJFS is not possible. You have to backup the data, remove and recreate the LV, create a new FS and restore the data from the backup into that FS.

With OnlineJFS you can try to reduce the FS using fsadm specifying the new size in KB. Due to some design limitations this often fails with JFS 3.1 and older. **After** fsadm successfully reduced the FS you can use lvreduce to reduce the underlying LV:

```
# lvreduce -L <new size in MB> /dev/vg01/lvdata
```

For details regarding JFS and OnlineJFS consult the JFS Chapter.

To change the **name of a LV** you can simply rename the LV devicefiles:

```
# umount /dev/vg01/lvol1
# mv /dev/vg01/lvol1 /dev/vg01/lvdata
# mv /dev/vg01/rlvol1 /dev/vg01/rlvdata
# mount /dev/vg01/lvdata <mountpoint>
```

There are several other characteristics of an LV that can be modified. Most commonly used are allocation policy, bad block relocation and LV IO-timeout. For details look at the lvchange man page.

Modifying a VG

The vgchange command can be used to (de)activate a VG. Certain parameters like max_pe (see above) cannot be changed without recreating the VG.

In order to rename a VG you have to export and re-import it:

```
# umount /dev/vg01/lvol1
# umount /dev/vg01/lvol2
...

# vgchange -a n vg01
# vgexport -m /tmp/mapfile vg01
# ll /dev/*/group                    (choose a unique minor no.)
# mkdir /dev/vgnew
# mknod /dev/vgnew/group c 64 0x010000
# vgimport -m /tmp/mapfile vgnew /dev/dsk/c4t0d0 /dev/dsk/c5t0d0 ...
```

NOTE: If you are dealing with a large amount of disks i recommend to use the "-f outfile" option with vgexport/vgimport. See section Importing and exporting VGs for details.

NOTE: Starting with LVM commands patch PHCO_24645 (UX 11.00) or PHCO_25814 (UX 11.11) vgcreate and vgimport will check for the uniqueness of the group file's minor number.

```
# vgcfgbackup vgnew
```

For details regarding vgchange look at the man page. vgexport/vgimport is described below in greater detail.

Removing a PV / VG / LV

Remove a LV

```
# umount /data
# lvremove /dev/vg01/lvsap
```

Remove a PV from a VG

```
# vgreduce vg01 /dev/dsk/c5t0d0
```

Remove a VG

umount any LV of this VG, deactivate and export it:

```
# umount /dev/vg01/lvol1
```

```
# umount /dev/vg01/lvol2
...

# vgchange -a n vg01
# vgexport vg01
```

NOTE: vgremove is not recommended because you need to remove all LVs and PVs from the VG before you could use vgremove. This is not necessary with vgexport. Additionally vgexport leaves the LVM structures on the disks untouched which could be an advantage if you like to re-import the VG later.

Moving physical extents

It is only possible to move PEs within a VG. In order to move data across VGs you need to use commands like dd, cp, mv, tar, cpio, ...
There is a command available that allows you to move LVs or certain extents of a LV from one PV to another - pvmove(1M). It is usually used to “free” a PV, i.e. to move all LVs from that PV in order to remove it from the VG. There are several forms of usage:

In order to move all PEs from c0t1d0 to the PVs c0t2d0 and c0t3d0:

```
# pvmove /dev/dsk/c0t1d0 /dev/dsk/c0t2d0 /dev/dsk/c0t3d0
```

In order to move all PEs of lvol4 that are located on PV c0t1d0 to PV c1t2d0:

```
# pvmove -n /dev/vg01/lvol4 /dev/dsk/c0t1d0 /dev/dsk/c0t2d0
```

ATTENTION:
pvmove is slow and unsave and therefore not recommended. It is unsafe because it only moves PE by PE, i.e it can be easyly interrupted before completion. Interrupting a pvmove command usually results in a corrupted LV. It is slow because the LVM header gets updated any time a single PE has been moved.

If MirrorDisk/UX is installed it is highly recommended to use mirroring as an alternative to pvmove. In order to move lvol4 from PV c0t1d0 to c0t2d0 just mirror it to c0t2d0 and remove the mirror from c0t1d0 afterwards:

```
# lvextend -m 1 /dev/vg01/lvol4 /dev/dsk/c0t2d0
# lvreduce -m 0 /dev/vg01/lvol4 /dev/dsk/c0t1d0
```

NOTE: Be careful when moving one of the first 3 LVs of vg00 (boot, swap, root). A successful boot depends on the correct setup of these LVs.

Importing and exporting VGs

The functionality of exporting VGs allows you to remove all data concerning a dedicated VG from the system without touching the data on the disks. The disks of an exported VG can be physically moved to another system and the VG can be imported there. Exporting a VG means the following: remove the VG and corresponding PV entries from /etc/lvmtab and remove the VG directory with their device files in /dev. Again - the data on the disks is left unchanged.
Since the structural layout of the LVM information on disk has not changed throughout the HP-UX releases you can import a VG that has been created on a UX 10.20 system e.g. on a

UX 11.11 system.

vgexport has a -m option to create a so called *mapfile*. This ascii file simply contains the LV names because they are not stored on the disks. You need a mapfile if you do not have the standard names for the LV device files (lvol1, lvol2, ...).

Here's the procedure to export a VG on system A and import it on system B:

on system A:
Umount all LVs that belong to the VG and deactivate it:

```
# vgchange -a n vg01
```

Export the VG:

```
# vgexport -v -m /tmp/vg01.map vg01
```

Now all information about vg01 has been removed from system A. The disks can now be moved to system B and the VG can be imported there:

on system B:
Create the directory for the LV device files and the group file. It is important to choose a minor number that is unique on system B.

```
# ll /dev/*/group
# mkdir /dev/vg01                (you could also choose another VG name)
# mknod /dev/vg01/group c 64 0x##0000
```

NOTE: Starting with LVM commands patch PHCO_24645 (UX 11.00) or PHCO_25814 (UX 11.11) vgcreate and vgimport will check for the uniqueness of the group file's minor number.

Now copy the mapfile from system A and import the VG:

```
# vgimport -v vg01 -m /tmp/vg01.map /dev/dsk/c1t0d0 /dev/dsk/c1t1d0
```

NOTE: The PV device files may be different on system B compared to system A.

If you have a bunch of disks in the VG you may not want to specify each of them within the argument list of vgimport. Using the -s option with vgexport/vgimport lets you get around this:

```
# vgexport -v -s -m /tmp/vg01.map vg01
```

If you specify -s in conjunction with the -m option vgexport simply adds the VG-ID to the mapfile:

```
# cat /tmp/vg01.map
VGID bfb13ce63a7c07c4
1 lvol1
2 lvol2
3 lvsap
4 lvdata
```

When using the -s option with the vgimport command on system B all disks that are connected to the system are scanned one after another. If the VG-ID listed in the mapfile is found on the header of a disk this disk is included automatically into the VG
Here's the appropriate vgimport command:

```
# vgimport -v -s -m /tmp/vg01.map vg01
```

So you do not have to specify the PVs anymore.

ATTENTION:
On systems using 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 this VG-ID mechanism. You should specify the list of PVs explicitely here. 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.

MirrorDisk/UX

Basic functionality

To be able to mirror LVs you need to purchase the product MirrorDisk/UX. Its important to remember that LVs are mirrored - not PVs. Especially the LVM header is not mirrored because it does not belong to the LV. You can have 1 or 2 mirror copies.

Here's how to mirror an existing LV to a specific PV:

```
# lvextend -m 1 /dev/vg01/lvol1 /dev/dsk/c1t0d0
```

NOTE: lvextend allows either to specify the size of a LV (-L or -l) OR the number of mirror copies (-m). You cannot specify both within one command.

lvdisplay shows a mirrored LV like this:

```
# lvdisplay -v /dev/vg01/lvol1 | more
...
...
--- Logical extents ---
LE   PV1              PE1  Status 1   LE    PV2              PE2   Status 2
0000 /dev/dsk/c0t6d0 0000 current    0000  /dev/dsk/c1t6d0  0000  current
0001 /dev/dsk/c0t6d0 0001 current    0001  /dev/dsk/c1t6d0  0001  current
...
```

To reduce the mirror (from PV c1t6d0):

```
# lvreduce –m 0 /dev/vg01/lvol1 /dev/dsk/c1t6d0
```

ATTENTION: If the LV uses the *distributed allocation policy* (aka *extent based striping*) you need to specify **all** PVs that you want to remove the mirror copy from. There is not (yet) an option that lets you specify the PVG as argument to lvreduce but there will be a LVM commands patch (maybe mid 2002). To check if the LV uses distributed allocation policy:

```
# lvdisplay /dev/vgXY/lvXY | grep Allocation
```

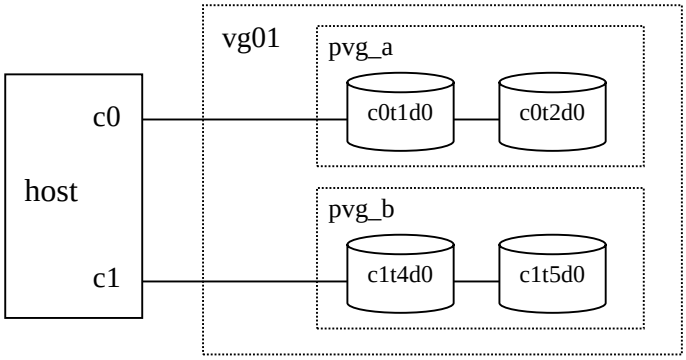
should show "distributed".

NOTE: Extending a mirrored LV works exactly like extending a non-mirrered LV. lvextend enlarges both mirror copies. The LV allocation policies *strict* or *PVG-strict* ensure that the mirrors reside on independent disks.

Physical Volume Groups - PVGs

If there are multiple host bus adapters (SCSI or fibre channel) available on the system it is useful in terms of high availablility to have mirror copies located on different adapters. The strict allocation policy for mirrored LVs guarantees that the mirror copy will not be placed on the same disk but it could be placed on a disk that is on the same adapter. The latter case can be avoided by using *physical volume groups*. A PVG is a subset of PVs within a VG that can be defined using -p option of vgcreate/vgextend or simply by creating an ascii file called /etc/lvmpvg.

Here's an example configuration:



If you want to be sure that the mirrors of LVs on e.g. c0t1d0 are not placed on c0t2d0 you need a lvmpvg file like the following:

```
# cat /etc/lvmpvg
VG  /dev/vg01
PVG pvg_a
/dev/dsk/c0t1d0
/dev/dsk/c0t2d0
PVG pvg_b
/dev/dsk/c1t4d0
/dev/dsk/c1t5d0
```

As soon as this file is saved the configuration is active and vgdisplay will look like this:

```
# vgdisplay -v vg01
...
...
--- Physical volume groups ---
PVG Name                pvg_a
PV Name                 /dev/dsk/c0t1d0
PV Name                 /dev/dsk/c0t2d0

PVG Name                pvg_b
PV Name                 /dev/dsk/c1t4d0
PV Name                 /dev/dsk/c1t5d0
```

Before mirroring a LV you need to set it's allocation policy to *PVG-strict*, e.g:

```
# lvchange -s g /dev/vg01/lvol1

# lvdisplay /dev/vg01/lvol1 | grep Allocation
Allocation                  PVG-strict
```

For details look at the lvmpvg man page.

Root Mirror

To set up a mirrored root config you need to add an additional disk (e.g. `c1t6d0`) to the root VG mirror all the LVs and make it bootable.

Initialize the disk and add it to vg00:

```
# pvcreate [-f] -B /dev/rdsk/c1t6d0
# vgextend vg00 /dev/dsk/c1t6d0
```

mirror the LVs:

```
# for i in lvol1 lvol2 ... lvol8        (specify any LV in the VG you like to mirror)
> do lvextend -m 1 /dev/vg00/$i /dev/dsk/c1t6d0
> done
```

go to SUBPROCEDURE 1: create LIF and BDRA to make the disk bootable

specify mirror disk as alternate boot path:

```
# setboot –a <HW-Path of mirror>
```

If you like to reduce the mirror from `c1t6d0` again, do the following:

```
# for i in lvol8 lvol7 ... lvol1              (specify any LV in the VG)
> do lvreduce -m 0 /dev/vg00/$i /dev/dsk/c1t6d0
> done
```

PV Links

What is it?

Physical Volume Links (or Alternate Links) is a High Availability Feature of LVM. LVM allows to configure multiple links (HW paths) to the same PV. One of them is considered as the primary link and the others act as alternate links. If LVM detects the primary link beeing unavailable as a consequence of a failure of a SCSI/FC card/cable it switches IO to the first available alternate link.

NOTE: It is the order in /etc/lvmtab that defines the default primary and alternate links.

Configuring PV Links

The following example shows how to create a VG with a disk having an alternate link. First check the available disk devices using ioscan:

```
# ioscan -fnkCdisk | more
Class     I  H/W Path       Driver S/W State   H/W Type     Description
=============================================================================
disk      0  0/0/2/0.6.0    sdisk CLAIMED     DEVICE       SEAGATE ST39102LC
```

```
                                /dev/dsk/c1t6d0   /dev/rdsk/c1t6d0
disk      1  0/0/2/1.6.0         sdisk CLAIMED     DEVICE       SEAGATE ST39102LC
                                /dev/dsk/c2t6d0   /dev/rdsk/c2t6d0
disk      2  0/12/0/0.0.0        sdisk CLAIMED     DEVICE       SEAGATE ST118202LC
                                /dev/dsk/c4t0d0   /dev/rdsk/c4t0d0
disk      3  0/12/0/0.1.0        sdisk CLAIMED     DEVICE       SEAGATE ST118202LC
                                /dev/dsk/c4t1d0   /dev/rdsk/c4t1d0
disk     13  0/12/0/0.2.0        sdisk CLAIMED     DEVICE       SEAGATE ST118202LC
                                /dev/dsk/c4t2d0   /dev/rdsk/c4t2d0
disk      6  0/12/0/1.0.0        sdisk CLAIMED     DEVICE       SEAGATE ST118202LC
                                /dev/dsk/c5t0d0   /dev/rdsk/c5t0d0
disk      7  0/12/0/1.1.0        sdisk CLAIMED     DEVICE       SEAGATE ST118202LC
                                /dev/dsk/c5t1d0   /dev/rdsk/c5t1d0
disk     12  0/12/0/1.2.0        sdisk CLAIMED     DEVICE       SEAGATE ST118202LC
                                /dev/dsk/c5t2d0   /dev/rdsk/c5t2d0
disk     19  0/12/0/1.3.0        sdisk CLAIMED     DEVICE       SEAGATE ST118202LC
...
...
```

From cabling or cmpdisks utility (see below) we know that c4t1d0 and c5t1d0 identify the same disk.

Extend vg01 using one of the device files:

```
# pvcreate [-f] /dev/rdsk/c4t1d0
# vgextend vg01 /dev/dsk/c4t1d0
```

NOTE: Do not run pvcreate on the other devicefile. Remember that it points to the same disk and this disk has already been pvcreated.

This is how vgdisplay and pvdisplay report alternate links:

```
# vgdisplay -v vg01
...
...
   --- Physical volumes ---
   PV Name                     /dev/dsk/c4t0d0
   PV Name                     /dev/dsk/c5t0d0  Alternate Link
   PV Status                   available
   Total PE                    542
   Free PE                     99
   Autoswitch                  On

   PV Name                     /dev/dsk/c4t1d0
   PV Name                     /dev/dsk/c5t1d0  Alternate Link
   PV Status                   available
   Total PE                    542
   Free PE                     0
   Autoswitch                  On


# pvdisplay /dev/dsk/c4t1d0
--- Physical volumes ---
PV Name                     /dev/dsk/c4t1d0
PV Name                     /dev/dsk/c5t1d0     Alternate Link
VG Name                     /dev/vg01
PV Status                   available
Allocatable                 yes
VGDA                        2
Cur LV                      2
```

```
PE Size (Mbytes)            32
Total PE                    542
Free PE                     0
Allocated PE                542
Stale PE                    0
IO Timeout (Seconds)        default
Autoswitch                  On
```

Autoswitch:
With autoswitch flag on (default) LVM always switches back to the primary link if it becomes available again. Otherwise the same link is used until the next failure.

IO Timeout:
The time that LVM retrys a link failed link is called *PV timeout* and can be specified using pvchange:

```
# pvchange -t 120
```

sets the timeout to 2 minutes. The default is 0, which causes LVM to use the device driver's default (usually 30 sec).

Changing PV Link order

To make an alternate link become the primary link (manual switch) use pvchange:

```
# pvchange -s <alternate>
```

To change it permanently (across VG deacivation) you have to change the order in /etc/lvmtab:

```
# vgreduce vg01 <primary>
Device file path "/dev/dsk/c0t1d0" is a primary link. Removing
primary link and switching to an alternate link.

# vgextend vg01 <primary>
```

Utility cmpdisks

cmpdisks is an unofficial shell script that collects information about all disks that can be seen on a system. It displays a sorted list of disks and their corresponding HW paths. cmpdisks works across multiple systems and is therefore very useful for ServiceGuard environments. Here's an example output for two nodes connected to shared storage:

```
# cmpdisks hprtdd32 grcdg319

Scanning host hprtdd32 ......................
Scanning host grcdg319 ..........................


***** LVM-VG: 0557706517-0986307905
1 hprtdd32:c2t0d0 0557706517-0986307878 0/0/2/0.0.0 SEAGATE/ST39103LC (0x00/vg00)

***** LVM-VG: 0630309352-0976295069
1 grcdg319:c2t6d0 0630309352-0976295068 0/0/2/1.6.0 SEAGATE/ST39102LC (0x00/vg00)
```

```
***** LVM-VG: 0557706517-0986205681
1 grcdg319:c4t1d0 0630309352-0968061502 0/12/0/0.1.0 HP/C5447A (0x02/vgsap)
  grcdg319:c5t1d0 0630309352-0968061502 0/12/0/1.1.0 HP/C5447A (0x02/vgsap)
  hprtdd32:c4t1d0 0630309352-0968061502 0/6/0/0.1.0 HP/C5447A (0x02/vgsap)
  hprtdd32:c5t1d0 0630309352-0968061502 0/6/0/1.1.0 HP/C5447A (0x02/vgsap)
2 grcdg319:c4t0d0 0630309352-0968061503 0/12/0/0.0.0 HP/C5447A (0x02/vgsap)
  grcdg319:c5t0d0 0630309352-0968061503 0/12/0/1.0.0 HP/C5447A (0x02/vgsap)
  hprtdd32:c4t0d0 0630309352-0968061503 0/6/0/0.0.0 HP/C5447A (0x02/vgsap)
  hprtdd32:c5t0d0 0630309352-0968061503 0/6/0/1.0.0 HP/C5447A (0x02/vgsap)

***** LVM-VG: 0630309352-1002790984
1 grcdg319:c4t2d0 0630309352-1002790983 0/12/0/0.2.0 HP/C5447A (n/a)
  hprtdd32:c4t2d0 0630309352-1002790983 0/6/0/0.2.0 HP/C5447A (n/a)
```

In the output above you can see:

- One non-shared disk in vg00 for each node.
- Two shared disks, each having one alternate link in shared VG vgsap on each node.
- One shared disk without alternate link that is not part of a VG.

Offline Diagnostics (ODE)

You need ODE to be able to do HW troubleshouting in the case the system is not able to boot. The ODE files are LIF files that should be installed in the LIF volume on any bootable disk.

If Online Diagnostics installed (check with `swlist OnlineDiag` or simply type `sysdiag`) you can find the ODE files in a regular file:

```
# lifls -l /usr/sbin/diag/lif/updatediaglif2
volume OFFLIN data size 67748 directory size 8
filename   type   start   size     implement  created
================================================================
ODE        -12960 16      848      0          00/10/24 11:32:30
MAPFILE    -12277 864     128      0          00/10/24 11:32:30
SYSLIB     -12280 992     353      0          00/10/24 11:32:30
CONFIGDATA -12278 1352    218      0          00/10/24 11:32:30
SLMOD2     -12276 1576    140      0          00/10/24 11:32:30
SLDEV2     -12276 1720    134      0          00/10/24 11:32:30
SLDRV2     -12276 1856    168      0          00/10/24 11:32:30
SLSCSI2    -12276 2024    116      0          00/10/24 11:32:30
MAPPER2    -12279 2144    142      0          00/10/24 11:32:30
IOTEST2    -12279 2288    89       0          00/10/24 11:32:30
PERFVER2   -12279 2384    125      0          00/10/24 11:32:30
PVCU       -12801 2512    64       0          00/10/24 11:32:30
SSINFO     -12286 2576    2        0          00/10/24 11:32:30
```

ATTENTION: the file **updatediaglif2** is only for pure 64bit systems (e.g. N-Class). For 32bit systems or systems that support both CPU types (e.g. K-Class) use the file updatediaglif.

The Online Diagnostics bundle can be found on the Support Plus Media. You can write the ODE files to the LIF volume as follows:

```
# cd /usr/sbin/diag/lif
```

```
# getconf HW_CPU_SUPP_BITS                    (the result is either 32, 32/64 or 64)
```

```
# mkboot -b updatediaglif -p ISL -p AUTO -p HPUX -p LABEL
```

```
  /dev/rdsk/c#t#d#                                          (if 32 or 32/64)
# mkboot -b updatediaglif2 -p ISL -p AUTO -p HPUX -p LABEL
  /dev/rdsk/c#t#d#                                          (if 64)
```

(the -p option preserves the specified file so that it is not overwritten)

If you are setting up a mirrored root config you need to install the ODE files also on the mirror disk else you don't have ODE utilities like MAPPER2 if you booted there.

In the case that no Online Diagnostics are installed but the ODE files are already on another boot disk you can copy them using lifcp(1M):

```
# lifcp -T <type> <sourcedevice:file> <targetdevice:file>
```

The procedure:

1) List the existing LIF files on the original disk

```
# lifls -l /dev/rdsk/c0t6d0
volume ISL10 data size 7984 directory size 8
filename   type   start   size     implement  created
=============================================================
ODE        -12960 584     848      0          00/10/24 11:32:30
MAPFILE    -12277 1432    128      0          00/10/24 11:32:30
SYSLIB     -12280 1560    353      0          00/10/24 11:32:30
CONFIGDATA -12278 1920    218      0          00/10/24 11:32:30
SLMOD2     -12276 2144    140      0          00/10/24 11:32:30
SLDEV2     -12276 2288    134      0          00/10/24 11:32:30
SLDRV2     -12276 2424    168      0          00/10/24 11:32:30
SLSCSI2    -12276 2592    116      0          00/10/24 11:32:30
MAPPER2    -12279 2712    142      0          00/10/24 11:32:30
IOTEST2    -12279 2856    89       0          00/10/24 11:32:30
PERFVER2   -12279 2952    125      0          00/10/24 11:32:30
PVCU       -12801 3080    64       0          00/10/24 11:32:30
SSINFO     -12286 3144    2        0          00/10/24 11:32:30
ISL        -12800 3152    306      0          00/11/08 20:49:59
AUTO       -12289 3464    1        0          01/10/16 10:18:16
HPUX       -12928 3472    848      0          00/11/08 20:50:00
LABEL      BIN    4320    8        0          01/10/16 10:18:14
```

2) List the existing LIF files on the mirror:

```
# lifls -l /dev/rdsk/c0t5d0
volume ISL10 data size 7984 directory size 8
filename   type   start   size     implement  created
=============================================================
ISL        -12800 584     306      0          00/11/08 20:49:59
AUTO       -12289 896     1        0          00/11/08 20:49:59
HPUX       -12928 904     848      0          00/11/08 20:50:00
PAD        -12290 1752    1580     0          00/11/08 20:50:00
LABEL      BIN    3336    8        0          01/10/19 16:29:30
```

NOTE: the PAD file is of no use. It contains just nulls and acts as a padding for alignment reasons.

3) copy the missing ODE LIF files to the mirror:

```
# lifcp -r -T-12960 /dev/rdsk/c0t6d0:ODE /dev/rdsk/c0t5d0:ODE
```

```
# lifcp -r -T-12277 /dev/rdsk/c0t6d0:MAPFILE /dev/rdsk/c0t5d0:MAPFILE
# lifcp -r -T-12280 /dev/rdsk/c0t6d0:SYSLIB /dev/rdsk/c0t5d0:SYSLIB
# lifcp -r -T-12278 /dev/rdsk/c0t6d0:CONFIGDATA
/dev/rdsk/c0t5d0:CONFIGDATA
# lifcp -r -T-12276 /dev/rdsk/c0t6d0:SLMOD2 /dev/rdsk/c0t5d0:SLMOD2
# lifcp -r -T-12276 /dev/rdsk/c0t6d0:SLDEV2 /dev/rdsk/c0t5d0:SLDEV2
# lifcp -r -T-12276 /dev/rdsk/c0t6d0:SLDRV2 /dev/rdsk/c0t5d0:SLDRV2
# lifcp -r -T-12276 /dev/rdsk/c0t6d0: SLSCSI2 /dev/rdsk/c0t5d0:SLSCSI2
# lifcp -r -T-12279 /dev/rdsk/c0t6d0:MAPPER2 /dev/rdsk/c0t5d0:MAPPER2
# lifcp -r -T-12279 /dev/rdsk/c0t6d0:IOTEST2 /dev/rdsk/c0t5d0:IOTEST2
# lifcp -r -T-12279 /dev/rdsk/c0t6d0:PERFVER2 /dev/rdsk/c0t5d0:PERFVER2
# lifcp -r -T-12801 /dev/rdsk/c0t6d0:PVCU /dev/rdsk/c0t5d0:PVCU
# lifcp -r -T-12286 /dev/rdsk/c0t6d0:SSINFO /dev/rdsk/c0t5d0:SSINFO
```

now a `lifls` of the mirror should be identical to the root disk.

Changing the HW Address of a Disk

If you like to move a disk to another physical location this will change the HW path and therefore the device file. You need to update the LVM configuration to reflect these changes. This is done by exporting the VG and then importing it using the new device files.
Here's how to do that:

Note the HW paths and devicefiles of all disks in the affected VG:

```
# vgdisplay -v vg##
# ioscan -fnkCdisk
```

Remove the device files you won't need anymore and shutdown the system:

```
# rmsf -H <old_hw_path>
# shutdown -h 0
```

Now carry out the hardware modifications and boot the system from the new HW path to maintenance mode:

```
ISL> hpux –lm
```

The new device files should be automatically created during bootup. If this is not the case you can easily create them using insf(1M):

```
# insf -H <new_hw_path> -e
```

Now note the VG's minor number and export it:

```
# ll /dev/*/group
# vgexport -v -m /tmp/vg##.map vg##
```

Import the VG again

```
# mkdir /dev/vg##
# mknod /dev/vg##/group c 64 0xXY0000
# vgimport -v -m /tmp/vg##.map vg## <new_device_file> ...
```

NOTE: If you are dealing with a large amount of disks i recommend to use the "-f outfile" option with vgexport/vgimport. See section Importing and exporting VGs for details.

If this is the root VG you need to update the information contained in BDRA:

```
# lvlnboot -r /dev/<rootVG>/lvol3          (lvol1 for <= UX 10.10)
# lvlnboot -b /dev/<rootVG>/lvol1          (not for <= UX 10.10)
# lvlnboot -s /dev/<rootVG>/lvol2
# lvlnboot -d /dev/<rootVG>/lvol2

# lvlnboot –v                                        (to ckeck it)
```

Now activate the VG and check the results:

```
# vgchange -a y vg##
# vgdisplay -v vg##
```

Because it is not allowed to change from maintenance mode to a higher run level you need to reboot:

```
# shutdown -r 0
```

LVM and MC/ServiceGuard (Cluster LVM)

In a ServiceGuard environment you have one or more VGs that have disks on the shared bus which can be accessed from multiple systems in the cluster. So it is very important to guarantee that a VG is active only on one node at a time or you will easily end up with inconsistant or corrupted data.

A VG that should be accessable from multiple nodes needs special treatment. You have to ensure that each node has current information about the VG, i.e:

- /etc/lvmtab
- /dev/vgXY/*
- /etc/lvmconf/vgXY.conf

Any changes to the VG that would affect these files need to be updated to all other nodes that could potentially activate the VG.

The following table shows which configuration changes affect which files:

configuration change	affects		
	/etc/lvmtab	/dev/vgXY/	/etc/lvmconf/
adding/removing a PV from the VG	Yes	No	Yes
adding/removing a LV from the VG	No	Yes	Yes
changing LV/PV characteristics (like size)	No	No	Yes

Example: add a disk to a shared VG

On the node where the VG is activated add the PV and generate the mapfile:

```
# pvcreate [-f] /dev/rdsk/c#t#d#
# vgextend vgXY /dev/dsk/c#t#d#
# vgexport -p -s -m /tmp/vgXY.map vgXY
```

Use ftp or rcp to distribute the mapfile (/tmp/vgXY.map) to the other nodes.
On all other nodes perform the following steps:

Remember the VG minor number:

```
# ll /dev/vgXY/group
```

If the VG does already exist, export it first and then import it:

```
# vgexport vgXY
# mknod /dev/vgXY/group c 64 0xXY0000
# vgimport -s -m /tmp/vgXY.map vgXY
```

NOTE: You may also use the "-f outfile" option of vgexport/vgimport where outfile contains a list of all devicefiles belonging to the VG. See section Importing and exporting VGs for details.

Backup the LVM configuration:

```
# vgchange -a r vgXY
# vgcfgbackup vgXY
# vgchange -a n vgXY
```

See ServiceGuard Chapter for details.

Replacing a Failed Disk

In order to replace a failed disk you have to recover the original LVM header onto the new media. The command `vgcfgrestore(1M)` writes the backup of the LVM header from the filesystem (`/etc/lvmconf/vgXY.conf`) to the disk. If data was mirrored you can easily sync it to the new disk. If not you have to figure out the LVs that had extents on the disk and recover the data from your backup.
Replacing a disk in a **ServiceGuard environment** makes no difference. Even replacing a cluster lock disk is no problem if the LVM configfile (`/etc/lvmconf/vgXY.conf`) contains the information about the cluster lock disk. Consult the ServiceGuard Chapter if you are unsure.

In the following you find special procedures for mirrored and un-mirrored PVs, for root and for data VGs and for hot-swap disks.

Identifying the failed disk

First of all you have to figure out which disk actually failed. Do not rely on the output of LVM's display commands only. Especially in mirrored configurations you have to be very careful.

Use the commands ioscan and diskinfo to determine the type of the failed disk. Is it an internal Seagate Disk or a Lun in a XP Ddisk array?

```
# ioscan -fnCdisk
Class     I  H/W Path       Driver   S/W State   H/W Type     Description
==========================================================================
disk      0  2/0/1/0/0.0.0  sdisk    CLAIMED     DEVICE       SEAGATE ST318404LC
                           /dev/dsk/c0t0d0   /dev/rdsk/c0t0d0
disk      1  2/0/1/0/0.1.0  sdisk    CLAIMED     DEVICE       SEAGATE ST318404LC
                           /dev/dsk/c0t1d0   /dev/rdsk/c0t1d0
disk      3  2/0/2/0/0.4.0  sdisk    CLAIMED     DEVICE       DGC     C1300WD
                           /dev/dsk/c1t4d0   /dev/rdsk/c1t4d0
disk     10  2/0/2/0/0.4.1  sdisk    CLAIMED     DEVICE       DGC     C1300WD
                           /dev/dsk/c1t4d1   /dev/rdsk/c1t4d1
disk     93  2/0/4/0/0.8.0.2.0.0.0   sdisk    CLAIMED     DEVICE       HP
DISK-SUBSYSTEM
                           /dev/dsk/c10t0d0   /dev/rdsk/c10t0d0
disk     94  2/0/4/0/0.8.0.2.0.1.0   sdisk    CLAIMED     DEVICE       HP
DISK-SUBSYSTEM
                           /dev/dsk/c10t1d0   /dev/rdsk/c10t1d0
disk     99  2/0/4/0/0.8.0.2.0.6.0   sdisk    CLAIMED     DEVICE       HP
OPEN-3
                           /dev/dsk/c10t6d0   /dev/rdsk/c10t6d0
disk    100  2/0/4/0/0.8.0.2.0.6.1   sdisk    CLAIMED     DEVICE       HP
OPEN-3
                           /dev/dsk/c10t6d1   /dev/rdsk/c10t6d1
...
...

# diskinfo /dev/rdsk/c10t6d0
SCSI describe of /dev/rdsk/c10t6d0:
             vendor: HP
         product id: OPEN-3
               type: direct access
```

```
             size: 2403360 Kbytes
bytes per sector: 512
```

NOTE: If you are dealing with a disk that is connected over fibre channel read the section "*How to Replace Disks at Hosts with TachLite HBAs*" in the Fibre Channel chapter.

It is always a good idea to try to read from the disk in suspect. Try to read a few MB at first:

```
# dd if=/dev/rdsk/cXtXdX of=/dev/null bs=256k count=10 &
```

NOTE: 256K is the largest IO request that LVM can handle. Using smaller blocksizes will make dd slower. Larger Blocksizes are no problem.

If this works then you ensured that the disk itself is at least accessable. But to be sure there is no media problem you have to read the whole disk:

```
# dd if=/dev/rdsk/cXtXdX of=/dev/null bs=256k
```

Additionaly you can use MESA diagnostics (`mstm` command).

Standard (non-hot-swappable) disk

In order to replace a "standard" disk you have to shutdown the system.

Root disk, not mirrored, non-hot-swappable

Assuming that disk `/dev/dsk/c0t6d0` in RootVG `vg00` is the one that failed.
Follow these steps to replace it:

1. Halt the system
2. Replace the disk
3. Boot from the Ignite-UX make_recovery tape and wait until the system is fully recovered.

In the case that there is no make_recovery tape available **:-(** but you have a full backup of vg00 please proceed to the System Recovery Chapter.

Root disk, mirrored, non-hot-swappable

NOTE: For HP-UX 11.0 be sure to have LVM kernel patch PHKL_20419 or later installed. If not you should reduce the mirror first (see) in order to avoid problems during resynchronization. Here's an example how to reduce the mirror:

```
# for i in lvol8 lvol7 ... lvol1        (specify any mirrored LV in the VG)
> do lvreduce –m 0 /dev/vg00/$i /dev/dsk/c0t6d0
> done
```

where c0t6d0 is the disk that failed.

Assuming that disk c0t6d0 in RootVG vg00 is the one that failed and it is properly mirrored to another bootable disk c0t5d0. Properly mirrored means that there are at least root, boot, primary swap and a valid LIF header/volume on that disk.

Follow these steps to replace it:

1. Halt the system
2. Replace the disk
3. Boot from the mirror disk into quorum mode

   ```
   ISL> hpux –lq
   ```

4. restore the PV header to the new disk, activate the VG and mount the filesystems:

   ```
   # vgcfgrestore -n vg00 /dev/rdsk/c0t6d0
   # vgchange -a y vg00        (this may take a while until all data is synced)
   ```

5. recreate LIF and BDRA on the new disk (refer to SUBPROCEDURE 1 at the end of this chapter)
6. locate the LVs that have extents on this disk:

   ```
   # pvdisplay -v /dev/dsk/c0t6d0
   ```

   any LV that is not mirrored is lost and needs to be recovered from data backup. You need to create a FS (newfs(1M)) before recovering data.

7. Synchronize the mirrored LVs

   although the LVs should all be synced after the activation of the VG its a good idea to check for stale extents. Verify that all extents are current:

   ```
   # vgdisplay –v vg00 | grep stale
   ```

   should not report anything. If it does:

   ```
   # vgsync vg00
   ```

   A single LV can be synced like this:

   ```
   # lvsync /dev/vg00/lvol1
   ```

   NOTE: Based on timestamps on the disks LVM decides which is the good copy and which is the bad one so you do not have to specify devicefiles here.

8. Check the results

   ```
   lvdisplay –v <LV>
   vgdisplay –v vg00
   ```

Non-root disk, not mirrored, non-hot-swappable

Assuming that disk /dev/dsk/c1t1d0 in VG vg01 is the one that failed.

Follow these steps to replace it:

1. Halt the system
2. Replace the disk
3. Boot up to multi-user mode as usual
4. restore the PV header to the new disk and activate the VG:

```
# vgcfgrestore -n vg01 /dev/rdsk/c1t1d0
# vgchange -a y vg01
```

5. locate the LVs that have extents on this disk:

```
# pvdisplay -v /dev/dsk/c1t1d0
```

Create filesystems (newfs(1M)) for each LV (there may be LVs that do not have a FS such as ORACLE datafiles) and recover them from backup.

Non-root disk, mirrored, non-hot-swappable

NOTE: For HP-UX 11.0 be sure to have LVM kernel patch PHKL_20419 or later installed. If not you should reduce the mirror first in order to avoid problems during resynchronization. Here's an example how to reduce the mirror:

```
# for i in lvol1 lvol2 ... lvolX        (specify any mirrored LV in the VG)
> do lvreduce –m 0 /dev/vgXY/$i /dev/dsk/c0t6d0
> done
```

where c0t6d0 is the disk that failed.

Assuming that disk c1t1d0 in a non-root VG vg01 is the one that failed and some or all of the LVs are mirrored to other disks.

Follow these steps to replace it:

1. Halt the system
2. Replace the disk
3. Boot up to multi-user mode as usual
4. restore the PV header to the new disk and activate the VG:

```
# vgcfgrestore -n vg01 /dev/rdsk/c1t1d0
# vgchange -a y vg01        (this may take a while until all data is synced)
```

5. locate the LVs that have extents on this disk:

```
# pvdisplay -v /dev/dsk/c1t1d0
```

any LV that is not mirrored is lost and needs to be recovered from backup. You need

to create a FS (newfs(1M)) before recovering data.

6. Synchronize the mirrored LVs

   although the LVs should all be synced after the activation of the VG its a good idea to check for stale extents. Verify that all extents are current:

   ```
   # vgdisplay –v vg01 | grep stale
   ```

   should not report anything. If it does:

   ```
   # vgsync vg01
   ```

   A single LV can be synced like this:

   ```
   # lvsync /dev/vg01/lvdata
   ```

   NOTE: Based on timestamps on the disks LVM decides which is the good copy and which is the bad one so you do not have to specify devicefiles here.

7. Check the results

   ```
   lvdisplay –v <LV>
   vgdisplay –v vgXY
   ```

Hot-swappable disks

Hot swappable disks are disks that can be plugged while the system is running which makes replacing a failed disk easy. The replacement disk has to be of the same type as the failed disk because the disk driver's internal information about the replaced disk may not be updated. The name of the vendor (HP, DEC, SEAGATE) is of no consideration. The new disk has to have the same capacity and blocksize (bytes per sector). Check this with diskinfo(1M).

NOTE: Be aware that not every disk module supports hot-swap. The procedure is not allowed e.g. for single ended (SE) disks.

Turn immediate reporting off

To avoid the loss of data you should verify that immediate reporting is disabled for the disk:

```
# scsictl –a /dev/rdsk/c#t#d#
immediate_report = 0
```

If it is not 0, which is the default for servers, change the setting:

```
# scsictl –m ir=0 /dev/rdsk/c#t#d#
```

SCSI Resets during the hot swap procedure

After inserting a hot swappable disk SCSI bus reset may occure. You will see a message like this in the message buffer:

```
# dmesg
...
...
 SCSI: Reset detected -- bus: 0
          lbp->uPhysScript: c6c000
          lbp->state: 44
```

```
        lbp->offset: 108
        PtCmd [9350d0]: 0a00000a
lsp: 0
lbp->owner: 0
Register values from most recent chip interrupt:
        istat: 02, sist0: 02, sist1: 00, dstat: 80
        dsps: ffffff38
Register values now:
        istat: 0
        dsp: c6c148, dcmddbc: 54000000, dsps: ffffff38
        dnad: c6c148
        dsa: 800, temp: c6c368, scratch: ffff007c
        scntl3: 1b, sxfer: 8, sbcl: 0, sfbr: 8
        dfifo: 0, sstat0/x: 2, sstat1/y: f, sstat2/z: a
        ctest7: 0, ctest3/x: 40
        sien0: 8f, sien1: 87, dien: 7f
scratch_lsp: 0
Pre-dcmddbc script dump [935130]:
        60000040 00000000 48000000 00000000
Script dump [935140]:
        54000000 ffffff38 740a0f00 00000000
        PtCmd [9350d0]: 0a00000a
```

Do I have to reduce the mirror?

Unplugging a disk online and replugging the same mech is not a problem but inserting a different disk instead is not always supported without reducing the mirror because LVM may not recognize that the PV was exchanged by a new one or the new disk may already have a valid LVM header.

Anyhow – under certain circumstances you do not have to reduce the mirrors.
You do not have to reduce the mirror if the PV is ***unattached***. A PV is unattached if the VG is not active or LVM already found the PV to be defective when the VG was activated. In the latter case vgchange would have printed the following message on the console:

```
vgchange: Warning: Couldn't attach to the volume group physical volume
"/dev/dsk/c#t#d#":
the path of the physical volume refers to a device that does not
exist, or is not configured into the kernel.
```

If the status at vgchange time is unknown, you may check if this occurred using vgdisplay. If the disk was defective at vgchange time, the following messages will be printed one or more times:

```
# vgdisplay <VG name>
vgdisplay: Warning: Couldn’t query physical volume “/dev/dsk/c#t#d#"
the specified path does not correspond to physical volume attached to
the volume group.
vgdisplay: Warning: Couldn’t query all of the physical volumes
```

If the above does not apply the PV is considered to be attached.

You do not necessarily have to reduce the mirrors if the PV is attached but:

vgdisplay –v reports the PV as `unavailable`., i.e LVM recognized that the disk has gone

and

you’re 100% sure that the new disk does not have a valid LVM header

and

LVM kernel patch PHKL_23612 (or newer) is installed (for UX 10.20)
LVM kernel patch PHKL_20419 (or newer) is installed (for UX 11.00)

try the following to get an attached PV to an unavailable state:
Pull the disk out and run `vgchange -a y vgXY` (use `-a e` for exclusive activation in ServiceGuard environments). This should force LVM to consider the missing PV to be unavailable (check with pvdisplay!) which ensures that the newly replaced disk is recognised as fresh and stale (after being vgvfgretore'd). In this case the mirror does not have to be reduced.

Assuming that disk cXtYdZ in VG vgXY is the one that failed.

Procedure 1: not mirrored, hot-swappable

If it is the root disk that failed follow the procedure for "standard" disks (root, not mirrored) above, else follow these steps:

1. replace the disk module and verify that it is recognized by the OS with ioscan(1M)
2. restore the PV header to the new disk and reactivate the VG:

```
# vgcfgrestore -n vgXY /dev/rdsk/cXtYdZ
# vgchange -a y vgXY
```

3. locate the LVs that have extents on this disk:

```
# pvdisplay -v /dev/dsk/cXtYdZ
```

   Create filesystems (newfs(1M)) for each LV (there may be LVs that do not have a FS such as Informix chunks) and recover them from backup.

Procedure 2: mirrored, hot-swappable

1. if the PV is attached, reduce the mirror, e.g. with:

```
# for i in lvol1 lvol2 ...      (specify any mirrored LV in the VG)
> do lvreduce -m 0 /dev/vgXY/$i /dev/dsk/cXtYdZ
> done
```

   check with `lvdisplay -v <LV>`

   ATTENTION:
   If the LV uses the distributed allocation policy (aka extent based striping) you need to specify ALL PVs that you want to remove the mirror copy from. There is not (yet) an option that lets you specify the PVG as argument to lvreduce but there will be a LVM commands patch (maybe end of 2001). To check if the LV uses distributed allocation policy: `lvdisplay /dev/vgXY/lvXY | grep Allocation` should show `distributed`.

2. replace the disk module and verify that it is recognized by the OS with ioscan(1M)
3. restore the LVM header to the new disk and reactivate the VG:

```
# vgcfgrestore -n vgXY /dev/rdsk/cXtYdZ
# vgchange -a y vgXY   (if you did not reduce the mirror wait until data is synced)
```

4. if it is a root disk, recreate LIF and BDRA on the new disk (refer to SUBPROCEDURE 1)

5. locate the LVs that have extents on this disk:

   ```
   # pvdisplay -v /dev/dsk/cXtYdZ
   ```

   any LV that is not mirrored is lost and needs to be recovered from data backup. Don't forget to create a FS (newfs(1M)) before recovering data.

6. Recreate the mirror if you reduced it before, else run vgsync vgXY, e.g.:

   ```
   # for i in lvol1 lvol2 ...      (specify LV in the VG you want to mirror)
   > do lvextend -m 1 /dev/vgXY/$i /dev/dsk/cXtYdZ
   > done
   ```

7. Check the results

   ```
   lvdisplay –v <LV>
   vgdisplay –v vgXY
   ```

Removing a „ghost disk“ From a VG - the PV Key

What is a ghost disk

You may come into a situation where you have to remove a PV from a VG that is failed or not even physically connected but still recorded in the lvmtab. Such a PV is called a “ghost disk” or “phantom disk”. You can get a ghost disk if the disk failed and the VG was activated again, maybe the system was rebooted.

If you cannot use vgcfgrestore to write the original LVM header back to the new disk because a valid LVM configuration backup file (/etc/lvmconf/vgXY.conf[.old]) is missing or corrupted you have to remove that PV from the VG (vgreduce), create a new LVM header (pvcreate) and add it to the VG again (vgextend).

NOTE: In such situations the vgcfgrestore command may fail to restore the LVM header, complaining about a ‘Mismatch between the backup file and the running kernel’. If you are 100% sure that your backup is valid you may override this check using the –R option.

In order to remove a PV from a VG you have to free it first, i.e. remove all logical extents from it. If the LVs on such a disk is not mirrored data is lost anyway. If it is mirrored you need to reduce the mirror, remove the PV run pvcreate and add it again.

A “ghost disk” can be identified if vgdisplay reports more current PVs than active ones. Additionally LVM commands may complain about the missing PVs:

```
# vgdisplay vg01
vgdisplay: Warning: couldn't query physical volume "/dev/dsk/c0t11d0":
```

```
The specified path does not correspond to physical volume attached to
 this volume group
vgdisplay: Couldn't query the list of physical volumes.
--- Volume groups ---
VG Name                     /dev/vg01
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      3
Open LV                     3
Max PV                      16
Cur PV                      2          number of PVs recorded in the lvmtab
Act PV                      1          number of PVs recorded in the kernel
Max PE per PV               1016
VGDA                        2
PE Size (Mbytes)            4
Total PE                    511
Alloc PE                    38
Free PE                     473
Total PVG                   0
```

The PV c0t11d0 is still recorded in lvmtab so it's a ghost disk:

```
# strings /etc/lvmtab
/dev/vg01
/dev/dsk/c0t0d2
/dev/dsk/c1t2d2
/dev/dsk/c0t11d0
```

Running vgreduce with the -f option would remove all PVs that are "free", i.e there is no LV having extents on that PV. Otherwise - if the PV is not free vgreduce -f reports an extent map to identify the associated LVs:

```
# vgreduce -f vg01
skip alternate link /dev/dsk/c1t2d2
vgreduce: Couldn't query physical volume "/dev/dsk/c0t11d0":
The specified path does not correspond to physical volume attached to
 this volume group
Not all extents are free. i.e. Out of 508 PEs, only 500 are free.
You must free all PEs using lvreduce/lvremove before the PV can be
removed.
Example: lvreduce -A n -m 0 /dev/vg01/lvol1.
         lvremove -A n /dev/vg01/lvol1.
Here's the map of used PEs

         --- Logical extents ---
         LE      LV              PE     Status 1
         0000    lvol1           0000     ???
         0001    lvol1           0001     ???
         0002    lvol1           0002     ???
         0003    lvol1           0003     ???
         0004    lvol1           0004     ???
         0005    lvol1           0005     ???
         0006    lvol1           0006     ???
         0007    lvol1           0007     ???
```

In this case there is lvol1 having extents on c0t11d0. You have to remove these extents from the PV. If the LV is mirrored use lvreduce to remove the mirrored extents, if the LV is not mirrored, data is lost anyway and you have to use lvremove to delete the LV:

Check the LV state:

```
# lvdisplay -v /dev/vg01/lvol1
lvdisplay: Warning: couldn't query physical volume "/dev/dsk/c0t11d0":
The specified path does not correspond to physical volume attached to
 this volume group
lvdisplay: Couldn't query the list of physical volumes.
--- Logical volumes ---
LV Name                     /dev/vg01/lvol1
VG Name                     /dev/vg01
LV Permission               read/write
LV Status                   available/stale
Mirror copies               1
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            32
Current LE                  8
Allocated PE                16
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   on
Allocation                  strict
IO Timeout (Seconds)        default

   --- Distribution of logical volume ---
   PV Name            LE on PV  PE on PV
   /dev/dsk/c0t0d2    8         8

   --- Logical extents ---
   LE    PV1          PE1   Status 1  PV2               PE2   Status 2
   00000 ???          00000 stale     /dev/dsk/c0t0d2   00000 current
   00001 ???          00001 stale     /dev/dsk/c0t0d2   00001 current
   00002 ???          00002 stale     /dev/dsk/c0t0d2   00002 current
   00003 ???          00003 stale     /dev/dsk/c0t0d2   00003 current
   00004 ???          00004 stale     /dev/dsk/c0t0d2   00004 current
   00005 ???          00005 stale     /dev/dsk/c0t0d2   00005 current
   00006 ???          00006 stale     /dev/dsk/c0t0d2   00006 current
   00007 ???          00007 stale     /dev/dsk/c0t0d2   00007 current
```

You can see, that the LV is mirrored.

Since the disk is not available anymore the PV is not accessable by it's decivefile /dev/dsk/c0t11d0 as usual. Alternatively you can access it by it's *PV key*.

The PV key

The PV key of a disk indicates it's order in the VG. The first PV has the key 0, the second has the key 1, etc. This does not necessarily have to be the order of appearance in lvmtab altough it is usually like that, at least when a VG is initially created.

The PV key can be used to address a PV that is not attached to the VG, e.g. because it is no longer available through it's device file due to a HW problem. If a LV has extents on that PV the key can be obtained using the -k option of lvdisplay:

```
# lvdisplay –v –k /dev/vg01/lvol1
...
```

```
...
  --- Logical extents ---
   LE    PV1          PE1   Status 1  PV2              PE2   Status 2
   00000      0       00000 stale         1             00000 current
   00001      0       00001 stale         1             00001 current
   00002      0       00002 stale         1             00002 current
   00003      0       00003 stale         1             00003 current
   00004      0       00004 stale         1             00004 current
   00005      0       00005 stale         1             00005 current
   00006      0       00006 stale         1             00006 current
   00007      0       00007 stale         1             00007 current
```

Compared to the output above the ??? have been replaced with the PV key (= 0).

NOTE: You can use the xd(1) command to display the PV key because it is stored at a fixed position in the LVM header, exactly 8222 bytes from the beginning of the disk:

```
# xd –j8222 -N2 /dev/rdsk/c1t6d0
```

NOTE: Sometimes you see messages like **PV[*X*] is POWERFAILED** in syslog. In this case ***X*** is the PV key.

Now reduce the mirror with the obtained key as argument:

```
# lvreduce –k –m 0 /dev/vg01/lvol1 0
```

After that the PV can be removed from the VG:

```
# vgreduce -f vg01
skip alternate link /dev/dsk/c1t2d2
vgreduce: Couldn't query physical volume "/dev/dsk/c0t11d0":
The specified path does not correspond to physical volume attached to
 this volume group
PV with key 0 sucessfully deleted from vg vg01
Repair done, please do the following steps.....:
1.  save /etc/lvmtab to another file
2.  remove /etc/lvmtab
3.  use vgscan(1m) -v to re-create /etc/lvmtab
4.  NOW use vgcfgbackup(1m) to save the LVM setup
```

Now do exactly what the output above indicates in order to remove the PV from the lvmtab:

```
# mv /etc/lvmtab /etc/lvmtab.org
# vgscan –v
...
...
Scan of Physical Volumes Complete.
*** LVMTAB has been created successfully.
*** If PV links are configured in the system.
*** Do the following to resync information on disk.
*** #1.   vgchange -a y
*** #2.  lvlnboot -R
```

Check it:

```
# strings /etc/lvmtab
/dev/vg01
/dev/dsk/c0t0d2
/dev/dsk/c1t2d2
```

Reactivate the VG and backup the LVM config:

```
# vgchange -a y vg01
# vgcfgbackup vg01
```

Now you can run pvcreate on the PV, add it to the VG again and recreate the mirror:

```
# pvcreate [-f] /dev/rdsk/c0t11d0
# vgextend vg01 /dev/dsk/c0t11d0
# lvextend –m 1 /dev/vg01/lvol1 /dev/dsk/c0t11d0
```

If the LV was not mirrored, recreate the LV (lvcreate), create a FS on it (newfs(1M)) and recover the data from the backup.

Increasing the Root LVs

Usually you cannot easily add space to the root LVs (/ or /stand) because they are contiguous.

using Ignite-UX

The recommended procedure to add space to the root LVs is to use an bootable Ignite-UX `make_recovery` Tape (refer to the Ignite-UX chapter for details).
The use of an Ignite tape is recommended because it is officially supported, easy to do, save and fast. To create a recovery tape containg the entire root VG just insert a DDS tape into the drive an run:

```
# make_tape_recovery –vA [ -d /dev/rmt/Xm ]
```

If for some reason the above does not apply you can use this **unsupported** procedure:

using the unofficial procedure

NOTE: this procedure is for HP-UX 10.20 and greater.

Since the root LV has to be contiguous it is not possible to increase it because it is not the last LV on the root disk. Anyway - it is possible to do it without using Ignite-UX if there is an additional free disk available - c1t1d0 in the following example:

Create a new VG vgroot with c1t1d0:

```
# pvcreate -B /dev/rdsk/c1t1d0
# mkdir /dev/vgroot
# ll /dev/*/group              (check for unused minor)
# mknod /dev/vgroot/group c 64 0x010000
# vgcreate vgroot /dev/dsk/c1t1d0
```

Create LIF and BDRA on c1t1d0 for vgroot (refer to SUBPROCEDURE 1)

Create LVs for boot, swap and root (in that order):

```
# lvcreate -C y -r n vgroot
# lvextend -L 100 /dev/vgroot/lvol1

# lvcreate -C y -r n vgroot
# lvextend -L 100 /dev/vgroot/lvol2

# lvcreate -C y -r n vgroot
# lvextend -L 300 /dev/vgroot/lvol3
```

Create LVs for `/usr`, `/opt`, `/var`, `/tmp`, `/etc`, `/home`, etc using sizes equal or larger than the ones in vg00 respectively:

```
# lvcreate vgroot
# lvextend -L 500 /dev/vgroot/lvol4
```

```
...
```

Create the filesystems:

```
# newfs -F hfs /dev/vgroot/rlvol1
# newfs -F vxfs /dev/vgroot/rlvol3
# newfs -F vxfs /dev/vgroot/rlvol4
...
```

Mount the filesystems:

```
# mkdir /new_root /new_usr /new_stand ...
# mount /dev/vgroot/lvol1 /new_stand
# mount /dev/vgroot/lvol3 /new_root
# mount /dev/vgroot/lvol4 /new_usr
...
```

Copy the data:

```
# cd /
# find . -xdev -depth | cpio -pvdlmax /new_root
# cd /stand
# find . -xdev -depth | cpio -pvdlmax /new_stand
# cd /usr
# find . -xdev -depth | cpio -pvdlmax /new_usr
...

# bdf
```

Modify the fstab. Replace vg00 with vgroot:

```
# vi /new_root/etc/fstab

/dev/vgroot/lvol1  /stand  hfs  defaults  0  0    # new boot LV
/dev/vgroot/lvol3  /       vxfs delaylog  0  0    # new root LV
/dev/vgroot/lvol4  /usr    vxfs delaylog  0  0    # new /usr LV
...
```

Change the device files for the root disk in /stand/bootconf:

```
# cat /stand/bootconf
l  /dev/dsk/c1t6d0

replace it with c1t1d0
```

Reboot from the disk c1t1d0:

```
# setboot -b <HW path of c1t1d0>
# shutdown -r 0
```

Backup the LVM Config:

```
# vgcfgbackup vgroot
```

Remove the old root VG:

```
# vgchange -a n vg00
# vgexport vg00
```

If you like to rename vgroot to vg00:

Boot to maintenance mode:

```
ISL> hpux -lm
```

Export vgroot and import it as vg00:

```
# /sbin/vgexport vgroot
# /sbin/mkdir /dev/vg00
# /sbin/mknod /dev/vg00/group c 64 0x000000
# /sbin/vgimport vg00 /dev/dsk/c1t1d0
```

Activate vg00 and mount the filesystems:

```
# /sbin/vgchange -a y vg00
# /sbin/mount /dev/vg00/lvol3 /
# /sbin/mount /dev/vg00/lvol1 /stand
# /sbin/mount /dev/vg00/lvol4 /usr
...
```

Modify the fstab. Replace vgroot with vg00 again:

```
# vi /etc/fstab
```

Reboot:

```
# shutdown -r 0
```

SUBPROCEDURE 1: create LIF and BDRA on the disk

REMARK: disk does not need to be pvcreate'd yet.

1. Write LIF header and LIF files (ISL, AUTO, HPUX, LABEL):

```
# mkboot -l /dev/rdsk/c#t#d#
# lifls –l /dev/rdsk/c#t#d#                                    (to ckeck it)
```

2. Write content of AUTO File (may be skipped):

```
# mkboot -a hpux /dev/rdsk/c#t#d#
# lifcp /dev/rdsk/c#t#d#:AUTO -                                (to ckeck it)
```

3. Install ODE files (may be skipped):

```
# cd /usr/sbin/diag/lif

# getconf HW_CPU_SUPP_BITS          (the result is either 32, 32/64 or 64)

# mkboot -b updatediaglif -p ISL -p AUTO -p HPUX -p LABEL
  /dev/rdsk/c#t#d#                                          (if 32 or 32/64)
# mkboot -b updatediaglif2 -p ISL -p AUTO -p HPUX -p LABEL
  /dev/rdsk/c#t#d#                                                (if 64)
```

   (the -p option preserves the specified file so that it is not overwritten)

   Refer to section Offline Diagnostics (ODE) if you have problems with this.

4. Write content of LABEL file,i.e set root, boot, swap and dump device:

   NOTE: This step can be omitted if you replace a failed mirror disk. Then this information has already been restored by vgcfgrestore. To be sure to have the latest information on the disk just do the following steps.

```
# lvlnboot -r /dev/<rootVG>/lvol3        (lvol1 for <= UX 10.10)
# lvlnboot -b /dev/<rootVG>/lvol1          (not for <= UX 10.10)
# lvlnboot -s /dev/<rootVG>/lvol2
# lvlnboot -d /dev/<rootVG>/lvol2

# lvlnboot –v                                          (to ckeck it)
```

Commands Overview

command	desciption
vgcreate	create a new VG
vgdisplay	display information about the VG
vgchange	activate/deactive a VG or change parameter of a VG
vgextend	add a new PV to the VG
vgreduce	remove a PV from the VG
vgremove	remove a VG (better use vgexport)
vgcfgbackup	backup the LVM header of a disk to a file
vgcfgrestore	restore the LVM header from a file to a disk
vgexport	remove the info of a VG from the system
vgimport	create a previously exported VG on this system
vgscan	recontruct /etc/lvmtab from the LVM headers on disk
vgchgid	change the VG-ID of a VG (needed for XPs)
vgsync	synchronize all mirrored LVs in the VG
lvcreate	create a new LV
lvdisplay	display information about LV
lvchange	change characteristics of a LV
lvextend	increase the size of LV / add a mirror copy to LV
lvreduce	decrease the size of LV / remove a mirror copy from LV
lvremove	remove the LV
lvspit	split a mirror copy of a LV (results in a separate LV)
lvmerge	merge a splitted mirror copy of a LV
lvsync	synchronize a mirrored LV
lvlnboot	set info about root, boot, swap, dump LVs in the BDRA
lvrmboot	delete info about root, boot, swap, dump LVs from BDRA
pvcreate	initialize a new disk for LVM
pvdisplay	display information about PV within VG
pvchange	change characteristics of a PV
pvmove	move PEs of a LV from one PV to another

NOTE: All LVM commands are hard-linked to the same single executable:

```
# ll /usr/sbin/vgchange
-r-sr-xr-x  31 root     sys     548864 Oct 18 20:17 /usr/sbin/vgchange

# ll /usr/sbin/pvmove
-r-sr-xr-x  31 root     sys     548864 Oct 18 20:17 /usr/sbin/pvmove

# ll /usr/sbin/lvextend
-r-sr-xr-x  31 root     sys     548864 Oct 18 20:17 /usr/sbin/lvextend
```