Chapter 16
-
LVM

INDEX

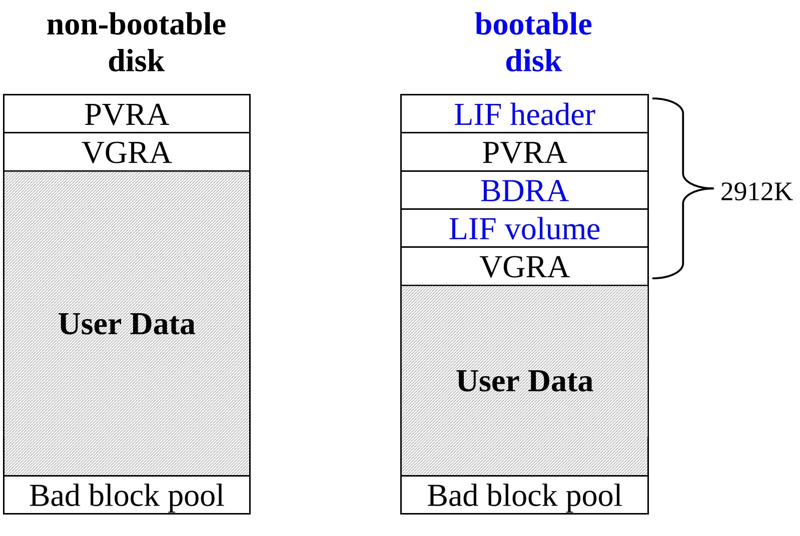
Terminology

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	=	File System
VGRA	=	Volume Group Reserved Area
VGDA	=	Volume Group Descriptor Area
VGSA	=	Volume Group Status Area
MCR	=	Mirror Consistency Record
PVRA	=	Physical Volume Reserved Area
BDRA	=	Boot Data Reserved Area

LVM Structural Information

The LVM structural information resides in reserved areas (PVRA, VGRA) at the beginning of any LVM disk and is also 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 VG configuration parameters PVs/VG (-p max_pv), PEs/PV (-e max_pe) and LVs/VG (-l max_lv), but it is usually smaller. The VG's VGRA must not be larger than a single extent..

NOTE: Itanium systems (UX 11.20, 11.22, 11.23) have a 100MB EFI partition at the beginning of the disk. Refer to the <u>Itanium Chapter</u> for details.

PVRA, BDRA and VGRA

1. The **PVRA** is unique for every PV in the VG. It contains:
   - LVMREC describing the PV with e.g. PV-ID, VG-ID, PV number in VG, PE size; start and length of: VGRA, BDRA (if any), BBDIR, User Data and the Bad Block Pool; in case of a ServiceGuard Cluster the Cluster ID and information about the Cluster Lock Area.
   - BBDIR (Bad Block Directory, maintaining the Bad Block Pool).

2. The **BDRA** (only created with *pvcreate –B*) contains boot relevant information, e.g.:
   - Information about PVs in root VG
   - Information about Boot/Swap/Root LVs (major/minor numbers, etc.)

3. The **VGRA** is identical for any PV of the VG. It contains:
   - The VGDA describing the VG, with e.g.:
     - VG-ID, configured max_lv, max_pv, max_pe.
     - per LV information: LV flags, size, schedule strategy, number of mirrors, stripes, stripe size, etc.
     - per PV information: PV-ID, PV size, PV flags, Extent mapping, etc.
   - The VGSA containing information about missing PVs and stale extents.
   - The MCRs for Mirror Write Cache handling.

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 **lvm** displays the complete LVM header:

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

Since the lvm tool may not always be available you can also read out PV-ID and VG-ID using standard commands that are available on any HP-UX system.

- How to use xd(1) to extract PV-ID and VG-ID:

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

The above information translates to:

- pvcreate and vgcreate was run on the sytem with systemID (uname –i) 2000252410
- pvcreate was run at timestamp 965817345 (seconds after Jan 1st 1970 0:00 UTC)
- vgcreate was run at timestamp 965817462 (117 seconds later)

or using adb(1):

- PV-ID:

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

- VG-ID:

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

vgcfgbackup(1M)

A copy of the LVM header is held within the file system 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).

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

```
# vgcfgbackup vgXY
Volume Group configuration for /dev/vgXY has been saved in
/etc/lvmconf/vgXY.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 vgXY
Volume Group Configuration information in "/etc/lvmconf/vgXY.conf"
VG Name /dev/vgXY
 ---- 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 vgXY /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 "`-A n`" with the LVM command. Anyway - the previous configuration can be found in /etc/lvmconf/*.conf**.old**.

NOTE: vgcfgrestore does not restore the LIF volume. This is done by mkboot.

/etc/lvmtab and vgscan(1M)

The file /etc/lvmtab contains information about all known VGs and their PVs. It is mainly used by vgchange(1M) at VG activation time. lvmtab is a binary file but you can display the printable strings 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 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:

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

Warnings can usually be ignored.

NOTE: If you leave the original file in place then vgscan uses its contents for creating a new one. This may fail depending on the file's contents. You may then try to move the lvmtab away. If there is no /etc/lvmtab, then vgscan recreates it from the scratch. In this case information about currently deactivated VGs may be missing in the new file!

ATTENTION: On a ServiceGuard systems 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.

ATTENTION: On systems using data replication products like BusinessCopy/XP, ContinousAccess/XP, EMC SRDF or EMC Timefinder vgscan may accidently add undesired PVs to VGs.

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).

Parameters and Limitations

LVM parameters

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>
max. number of PE per PV	1016 PEs	65535 PEs	vgcreate -e <max_pe>
max. number of PE per LV	0 PEs	65535 PEs	lvcreate -l <le_number>
LV size	0 MB	16 TB	lvcreate -L <MB>

How the size of the VGRA is calculated

The VGRA size of any non-bootable disk must fit into the size of a single PE. For a bootable disk the VGRA needs to start at offset 2144K while user data always starts at offset 2912K. Due to these constraints the maximum VGRA size of bootable disks is even more restricted as for regular disks.

However, it is good to know how the size of the VGRA depends on the VG's configuration at creation time. The folowing set or formulas calculates vgra_len in KB.

```
vgda_len = (ROUNDUP (16 * max_lv, 1024) +
            (max_pv * ROUNDUP (16 + 4 * max_pe, 1024)) ) / 1024 + 2;

vgsa_len = ROUNDUP (36 + 12 * ROUNDUP (max_pv, 32) +
           ROUNDUP(max_pe,8) * max_pv / 8, 1024) / 1024;

mcr_len  = 8;

vgra_len = 2 * (ROUNDUP (vgda_len + vgsa_len, 8) + mcr_len);
```

The `ROUNDUP()` function used above rounds up arg1 to a multiple of arg2.

The lvmcompute tool can be used to easily calculate the VGRA size and provides also table outputs like those shown in the following section. It is available from the HP internal site ftp://einstein.grc.hp.com/TOOLS/LVM.

Maximum max_pe values for non-boot disks

The following table lists the maximum allowed *max_pe* (-e) values depending on *max_pv* (-p) and *pe_size* (-s) along with their resulting PV sizes in GB. Since the *lv_max* parameter has a lower impact on the results, the table is caculated for *lv_max*=255, which is default and also the worst-case. The fields for the default settings *–s 4 –p 16* are shaded. Light shading indicates that the only restriction is the max. 65535 PE barrier for any given PV.

PVs/VG (vgcreate –p *max_pv*) \ PE size (vgcreate –s *pe_size*) in MB	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
1	65535 64.0G	65535 128.0G	65535 256.0G	65535 512.0G	65535 1024.0G	65535 2048.0G	65535 4095.9G	65535 8191.9G	65535 16383.8G
2	61692 60.2G	65535 128.0G	65535 256.0G	65535 512.0G	65535 1024.0G	65535 2048.0G	65535 4095.9G	65535 8191.9G	65535 16383.8G
4	30716 30.0G	62460 122.0G	65535 256.0G	65535 512.0G	65535 1024.0G	65535 2048.0G	65535 4095.9G	65535 8191.9G	65535 16383.8G
8	15356 15.0G	31228 61.0G	62972 246.0G	65535 512.0G	65535 1024.0G	65535 2048.0G	65535 4095.9G	65535 8191.9G	65535 16383.8G
16	7676 7.5G	15612 30.5G	31484 123.0G	63228 494.0G	65535 1024.0G	65535 2048.0G	65535 4095.9G	65535 8191.9G	65535 16383.8G
32	3836 3.7G	7676 15.0G	15612 61.0G	31484 246.0G	63228 987.9G	65535 2048.0G	65535 4095.9G	65535 8191.9G	65535 16383.8G
48	2556 2.5G	5116 10.0G	10492 41.0G	20988 164.0G	42236 659.9G	65535 2048.0G	65535 4095.9G	65535 8191.9G	65535 16383.8G
64	1788 1.7G	3836 7.5G	7676 30.0G	15612 122.0G	31484 491.9G	63484 1983.9G	65535 4095.9G	65535 8191.9G	65535 16383.8G
80	1532 1.5G	3068 6.0G	6140 24.0G	12540 98.0G	25340 395.9G	50684 1583.9G	65535 4095.9G	65535 8191.9G	65535 16383.8G
96	1276 1.2G	2556 5.0G	5116 20.0G	10492 82.0G	20988 327.9G	42236 1319.9G	65535 4095.9G	65535 8191.9G	65535 16383.8G
112	1020 1.0G	2044 4.0G	4348 17.0G	8956 70.0G	17916 279.9G	36092 1127.9G	65535 4095.9G	65535 8191.9G	65535 16383.8G
128	764 0.7G	1788 3.5G	3836 15.0G	7676 60.0G	15612 243.9G	31740 991.9G	63484 3967.8G	65535 8191.9G	65535 16383.8G
144	764 0.7G	1532 3.0G	3324 13.0G	6908 54.0G	14076 219.9G	28156 879.9G	56316 3519.8G	65535 8191.9G	65535 16383.8G
160	764 0.7G	1532 3.0G	3068 12.0G	6140 48.0G	12540 195.9G	25340 791.9G	50684 3167.8G	65535 8191.9G	65535 16383.8G
176	508 0.5G	1276 2.5G	2812 11.0G	5628 44.0G	11516 179.9G	23036 719.9G	46076 2879.8G	65535 8191.9G	65535 16383.8G
192	508 0.5G	1276 2.5G	2556 10.0G	5116 40.0G	10492 163.9G	20988 655.9G	42236 2639.8G	65535 8191.9G	65535 16383.8G
208	508 0.5G	1020 2.0G	2300 9.0G	4860 38.0G	9724 151.9G	19452 607.9G	38908 2431.8G	65535 8191.9G	65535 16383.8G
224	508 0.5G	1020 2.0G	2044 8.0G	4348 34.0G	8956 139.9G	17916 559.9G	36092 2255.8G	65535 8191.9G	65535 16383.8G
240	504 0.5G	1020 2.0G	2044 8.0G	4092 32.0G	8444 131.9G	16892 527.9G	33788 2111.8G	65535 8191.9G	65535 16383.8G
255	252 0.2G	764 1.5G	1788 7.0G	3836 30.0G	7932 123.9G	15868 495.9G	31740 1983.8G	63740 7967.5G	65535 16383.8G

Maximum max_pe values for boot disks

The following table lists the maximum allowed *max_pe* (-e) values depending on *max_pv* (-p) and *pe_size* (-s) along with their resulting PV sizes in GB. Since the *lv_max* parameter has a lower impact on the results, the table is caculated for *lv_max*=255, which is default and also a the worst-case. The fields for the default settings *–s 4 –p 16* are shaded.

	PE size (vgcreate –s *pe_size*) in MB								
PVs/VG (vgcreate –p *max_pv*)	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
1	65535 64.0G	65535 128.0G	65535 256.0G	65535 512.0G	65535 1024.0G	65535 2048.0G	65535 4095.9G	65535 8191.9G	65535 16383.8G
2	43772 42.7G	43772 85.5G	43772 171.0G	43772 342.0G	43772 683.9G	43772 1367.9G	43772 2735.8G	43772 5471.5G	43772 10943.0G
4	21756 21.2G	21756 42.5G	21756 85.0G	21756 170.0G	21756 339.9G	21756 679.9G	21756 1359.8G	21756 2719.5G	21756 5439.0G
8	10748 10.5G	10748 21.0G	10748 42.0G	10748 84.0G	10748 167.9G	10748 335.9G	10748 671.8G	10748 1343.5G	10748 2687.0G
16	5372 5.2G	5372 10.5G	5372 21.0G	5372 42.0G	5372 83.9G	5372 167.9G	5372 335.8G	5372 671.5G	5372 1343.0G
32	2556 2.5G	2556 5.0G	2556 10.0G	2556 20.0G	2556 39.9G	2556 79.9G	2556 159.8G	2556 319.5G	2556 639.0G
48	1788 1.7G	1788 3.5G	1788 7.0G	1788 14.0G	1788 27.9G	1788 55.9G	1788 111.8G	1788 223.5G	1788 447.0G
64	1276 1.2G	1276 2.5G	1276 5.0G	1276 10.0G	1276 19.9G	1276 39.9G	1276 79.8G	1276 159.5G	1276 319.0G
80	1020 1.0G	1020 2.0G	1020 4.0G	1020 8.0G	1020 15.9G	1020 31.9G	1020 63.8G	1020 127.5G	1020 255.0G
96	764 0.7G	764 1.5G	764 3.0G	764 6.0G	764 11.9G	764 23.9G	764 47.8G	764 95.5G	764 191.0G
112	764 0.7G	764 1.5G	764 3.0G	764 6.0G	764 11.9G	764 23.9G	764 47.8G	764 95.5G	764 191.0G
128	508 0.5G	508 1.0G	508 2.0G	508 4.0G	508 7.9G	508 15.9G	508 31.8G	508 63.5G	508 127.0G
144	508 0.5G	508 1.0G	508 2.0G	508 4.0G	508 7.9G	508 15.9G	508 31.8G	508 63.5G	508 127.0G
160	508 0.5G	508 1.0G	508 2.0G	508 4.0G	508 7.9G	508 15.9G	508 31.8G	508 63.5G	508 127.0G
176	252 0.2G	252 0.5G	252 1.0G	252 2.0G	252 3.9G	252 7.9G	252 15.8G	252 31.5G	252 63.0G
192	252 0.2G	252 0.5G	252 1.0G	252 2.0G	252 3.9G	252 7.9G	252 15.8G	252 31.5G	252 63.0G
208	252 0.2G	252 0.5G	252 1.0G	252 2.0G	252 3.9G	252 7.9G	252 15.8G	252 31.5G	252 63.0G
224	252 0.2G	252 0.5G	252 1.0G	252 2.0G	252 3.9G	252 7.9G	252 15.8G	252 31.5G	252 63.0G
240	252 0.2G	252 0.5G	252 1.0G	252 2.0G	252 3.9G	252 7.9G	252 15.8G	252 31.5G	252 63.0G
255	252 0.2G	252 0.5G	252 1.0G	252 2.0G	252 3.9G	252 7.9G	252 15.8G	252 31.5G	252 63.0G

Supported JFS (VxFS) file and file system sizes

HP-UX Release	HP JFS Version	Veritas Disk Layout	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 JFS 3.5	Version 2	2 GB	128 GB
		Version 3	2 TB	2 TB
		Version 4	2 TB	2 TB

Bold font: 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.

NOTE: 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.

Supported HFS file and file system 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 vgnew
```

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 creates 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 place the LV on a specific PV, then first create an LV of 0MB. It has no extents - it just exists.

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

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 recommended to use a VxFS (=JFS) file system.

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/cXtXdX
```

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

```
# pvchange -t 0 /dev/dsk/cXtXdX
```

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 and 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 an 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 not an atomic operation. Furthermore the command moves data extent by extent and is easily interruptable. If this happens, then the configuration is left in some weird inconsistent state showing an additional pseudo mirror copy for the extents in question. This can be cleaned up **only** using the lvreduce command (Use *lvreduce -m 0 LV* if the LVs were unmirrored and *lvreduce -m 1 LV* if they were mirrored before starting the pvmove; there's no need to specify a PV here).

If MirrorDisk/UX is installed it is usually saver and faster 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
```

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 vgXX
```

Export the VG:

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

Now all information about vgXX 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/vgXX                (you could also choose another VG name)
# mknod /dev/vgXX/group c 64 0xXX0000
```

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 vgXX -m /tmp/vgXX.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/vgXX.map vgXX
```

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

```
# cat /tmp/vgXX.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/vgXX.map vgXX
```

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:

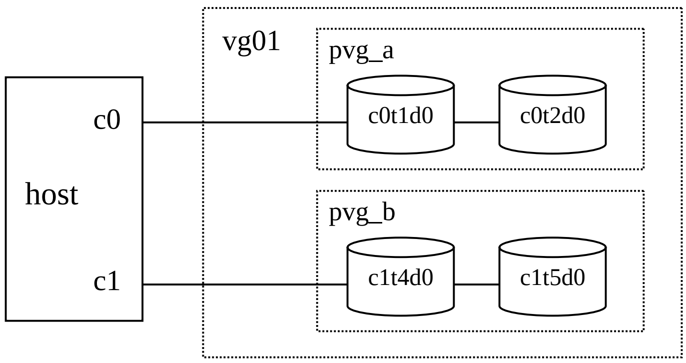
```
# lvdisplay /dev/vgXX/lvXX | grep Allocation
```

should show "distributed".

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

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.

1. Initialize the disk and add it to vg00:

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

2. Mirror the LVs using lvextend:

   ```
   # lvextend –m 1 /dev/vg00/lvolX /dev/dsk/c1t6d0
   ```

   If you want to use a shell loop to extend automatically, use e.g.:

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

3. **Important:** Configure LIF/BDRA, according to the LIF/BDRA Configuration Procedure at the end of this chapter.

4. Specify the mirror disk as alternate boot path in stable storage:

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

To determine the hardware path use e.g. ioscan:

```
# ioscan -fnk /dev/dsk/c1t6d0
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

# setboot –a 0/0/2/0.6.0
```

5. Add the new mirror boot device to /stand/bootconf, e.g.:

`l /dev/dsk/c0t6d0` *(original boot device)*
`l /dev/dsk/c1t6d0` *(new mirror boot device)*

If you like to remove the mirror again, you need to use lvreduce:

```
# lvreduce -m 0 /dev/vg00/lvolX /dev/dsk/c1t6d0
```

Of course, this can also be done automatically using a shell loop, e.g.:

```
# for lvol in lvol8 lvol7 ... lvol1
> do
> lvreduce -m 0 /dev/vg00/$lvol /dev/dsk/c1t6d0
> done
```

(specify all LVs you need to reduce)

PV Links (Alternate Paths)

Physical Volume Links (aka PV Links or Alternate Links) are a High Availability Feature of LVM which allows to configure multiple links (HW paths) to the same PV for redundancy. One of them is considered as the primary link while the others act as alternate links. If LVM detects the primary link beeing unavailable as a consequence of a failure (e.g. of a SCSI/FC card/cable) it re-routes IO traffic to the first available alternate link.

NOTE: It is the order in /etc/lvmtab that defines the default order in which the links are used.

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
```

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

```
# pvchange -t 120 /dev/dsk/cXtXdX
```

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

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.

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/reactivation) 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. It recognizes LVM and VxVM devices, also on Itanium systems.

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 Diagnostic Environment (ODE)

You need the 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.

With 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 updatediaglif**2** 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/cXtXdX                                      (if 32 or 32/64)
# mkboot -b updatediaglif2 -p ISL -p AUTO -p HPUX -p LABEL
  /dev/rdsk/cXtXdX                                      (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.

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/vgXX/*
- /etc/lvmconf/vgXX.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/vgXX/**	**/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: Adding a disk to a cluster VG

- On the node where the VG is activated:
  1. Add the PV to the VG as usual:

     ```
     # pvcreate [-f] /dev/rdsk/cXtXdX
     # vgextend vgXX /dev/dsk/cXtXdX
     ```

  2. Generate a map file:

     ```
     # vgexport -p -s -m /tmp/vgXX.map vgXX
     ```

  3. Use ftp or rcp to distribute the mapfile (/tmp/vgXX.map) to the other nodes.

- On all other nodes where the VG is not activated:
  1. Remember the VG minor number:

     ```
     # ll /dev/vgXX/group
     ```

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

     ```
     # vgexport vgXX
     # mknod /dev/vgXX/group c 64 0xXX0000
     # vgimport -s -m /tmp/vgXX.map vgXX
     ```

     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.

  3. Backup the LVM configuration:

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

See the ServiceGuard Chapter for details.

Replacing a Failed LVM Disk

In order to replace a failed disk you have to recover the original LVM header onto the new media. The command `vgcfgrestore(1M)` recovers the backup of the LVM header from the file system (`/etc/lvmconf/vgXX.conf`) to the disk. If data was mirrored you can easily sync it to the new disk. Otherwise you need to figure out which LVs have extents residing on that disk and recover the data from your backup.

NOTE: The replacement disk must be the same product ID as the replaced one. HP often uses different manufacturers for disks having the same product number. The hotswap procedures will not update the disk driver's internal information to that of the replaced disk. The replacement disk will have the same capacity and blocksize as the defective disk because they have the same product number. The only field that could be incorrect is the string specifying the vendor's name. This will not affect the behavior of the LVM. If it is desired to update the manufacturers' name, then the disk's volume group must be deactivated and reactivated.

Replacing a disk in a **ServiceGuard environment** makes no real difference. Even replacing a cluster lock disk is no problem, since the LVM configuration backup contains all needed information about it. This is true as long as vgcfgbackup was run after configuring the cluster. Consult the ServiceGuard Chapter if you are unsure.

ATTENTION: If this is an **Itanium** system (UX 11.20, UX 11.22, UX 11.23) you need to take care of the new disk partitioned layout. The first partition (cXtXdXs1) contains the EFI (100MB). The former LVM disk is now located at partition 2 of the disk (cXtXdXs2). For details on how to replace an Itanium root disk refer to the Itanium Chapter.

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.

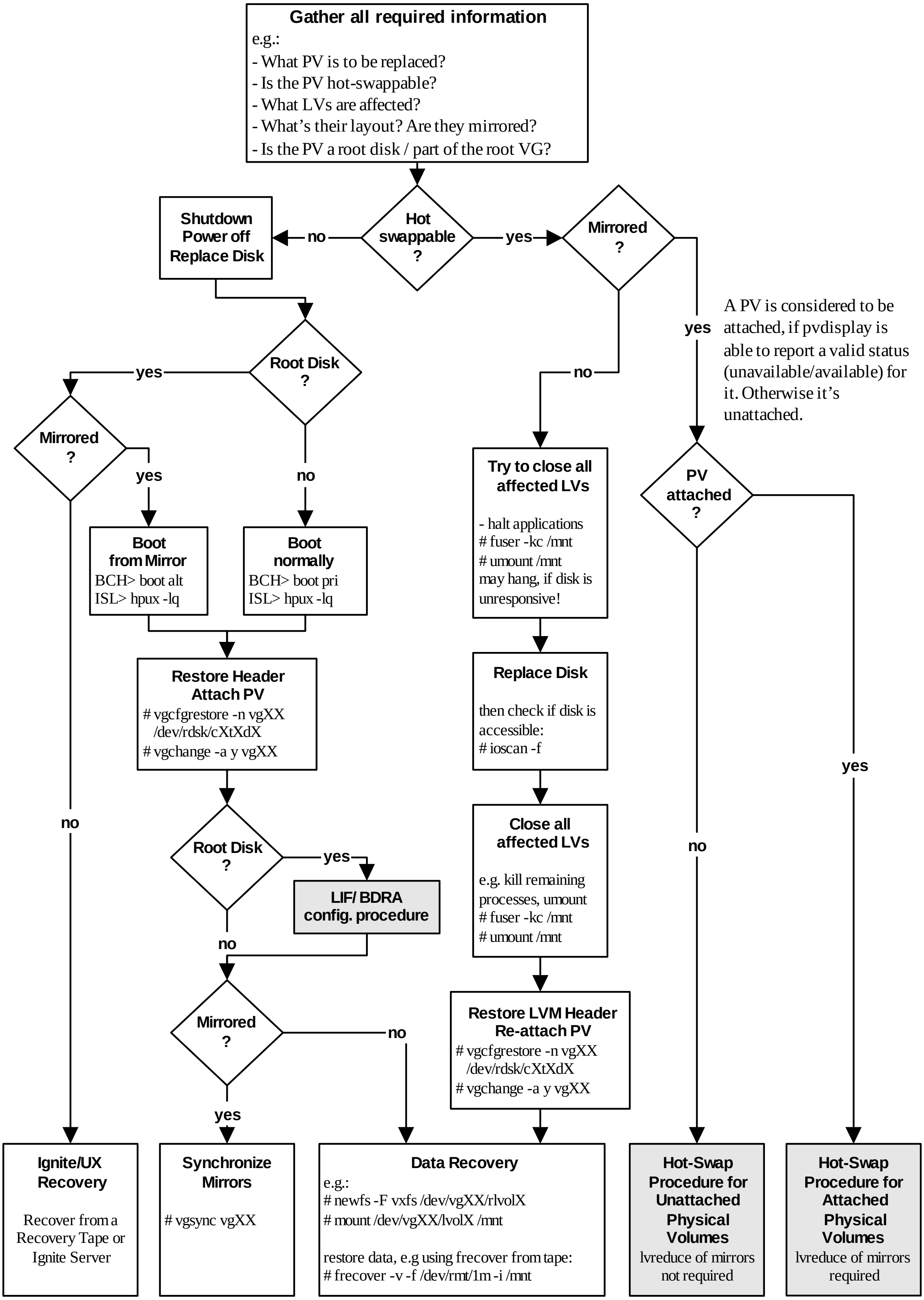
Here are some approaches how to check for typical symptoms of failed disks.

- Use the ioscan(1M) command (`ioscan –fCdisk`) to have a look at the disk's S/W state. Only disks in state `CLAIMED` are currently accessible by the system. Disks in other states like `NO_HW` are of course suspicious. This is also true for disks that are completely missing in ioscan's result. If the disk is `CLAIMED` then at least its contoller is responding.
- The next step could be a test with diskinfo(1M) (`diskinfo /dev/rdsk/cXtXdX`). The reported size must be >0, otherwise the device is not ready for some reason.
- Although being more time consuming, trying to read the disk with dd(1) completely (`dd if=/dev/rdsk/cXtXdX of=/dev/null bs=256K`) or partially (`dd if=/dev/rdsk/cXtXdX of=/dev/null bs=256K count=100`) is also a useful indicator. No I/O errors must be reported here.
- Use hardware diagnostic tools (like MESA diagnostics, `mstm/cstm` commands) to get detailed diagnostic information about the disk. These tools offer the most conclusive information.

Last, but not least, **You must be sure about what disk is the defective one**! Starting any replacement procedure based on wrong assumptions can cause loss and corruption of data.

Disk Replacement Flow Chart

The following flow chart is supposed to provide an overview about possible LVM disk replacement scenarios. The procedures for the most important case in HA environments (hot-swappable and mirrored) are presented in detail later (Hot-Swap Procedure for Attached Physical Volumes, Hot-Swap Procedure for Unattached Physical Volumes).


Gather all required information
e.g.:
- What PV is to be replaced?
- Is the PV hot-swappable?
- What LVs are affected?
- What's their layout? Are they mirrored?
- Is the PV a root disk / part of the root VG?
Hot swappable ?
no
yes
Shutdown Power off Replace Disk
Mirrored ?
yes
no
A PV is considered to be attached, if pvdisplay is able to report a valid status (unavailable/available) for it. Otherwise it's unattached.
Root Disk ?
yes
no
Mirrored ?
yes
no
Boot from Mirror
BCH> boot alt
ISL> hpux -lq
Boot normally
BCH> boot pri
ISL> hpux -lq
Try to close all affected LVs
- halt applications
fuser -kc /mnt
umount /mnt
may hang, if disk is unresponsive!
PV attached ?
yes
no
Restore Header Attach PV
vgcfgrestore -n vgXX /dev/rdsk/cXtXdX
vgchange -a y vgXX
Replace Disk
then check if disk is accessible:
ioscan -f
Root Disk ?
yes
no
LIF/ BDRA config. procedure
Close all affected LVs
e.g. kill remaining processes, umount
fuser -kc /mnt
umount /mnt
Mirrored ?
no
yes
Restore LVM Header Re-attach PV
vgcfgrestore -n vgXX /dev/rdsk/cXtXdX
vgchange -a y vgXX
Ignite/UX Recovery
Recover from a Recovery Tape or Ignite Server
Synchronize Mirrors
vgsync vgXX
Data Recovery
e.g.:
newfs -F vxfs /dev/vgXX/rlvolX
mount /dev/vgXX/lvolX /mnt
restore data, e.g using frecover from tape:
frecover -v -f /dev/rmt/1m -i /mnt
Hot-Swap Procedure for Unattached Physical Volumes
lvreduce of mirrors not required
Hot-Swap Procedure for Attached Physical Volumes
lvreduce of mirrors required

Hot-Swap Procedure for Attached Physical Volumes

NOTE: A physical volume is considered to be *attached*, if the pvdisplay command is able to report a valid status (unavailable/available) for it. Otherwise it's called *unattached*.

ATTENTION: As of today (HP-UX 11.23) the offical replacement procedure stipulates to reduce mirror copies from attached PVs before performing the actual replacement. However, there are other (unofficial) cookbooks allowing the replacement without this safety measure.
In this LVM chapter **we only consider the official procedure to be supported**. The reason is that there are potentially serious problems with replacing an attached device. Although the pvdisplay indicates the device is unavailable, LVM could still be trying to recover it. There is a possibility that a device that pvdisplay shows to be unavailable one moment could immediately appear to be available again just as the new device is being initialized in-place with vgcfgrestore. The consequences can be data corruption or obscure problems that can be difficult to track down, due to the LVM metadata on the device being improperly written.
A seemingly plausable but also unsupported solution to this problem is to initialize (vgcfgrestore) the replacement disk in a different location (e.g. an unused slot in the storage system). Then replace the unavailable disk with the new one. This will work safely as long as LVM recognizes that the device is unavailable before the disk is replaced.

Follow these steps to replace a hot-swap disk module for *attached* Physical Volumes, which means that the disk was defective/unaccessible at the time the volume group was activated. Hot-swapping a disk which was defective during activation *(unattached)* requires a different sequence of commands. Skip to the alternative procedure, Hot-Swap Procedure for Unattached Physical Volumes.

1. Reduce any logical volumes that have mirror copies on the faulty disk so that they no longer mirror onto that disk.

   NOTE: Be advised to check first, *what* LVs have mirror extents allocated on the faulty disk (to be checked with pvdisplay –v /dev/dsk/cXtXdX). Then you should check for each found LV *how* it is mirrored (use lvdisplay –v /dev/vgXX/lvolX). If the mirror extents span more than one PV then it is *highly recommended* to specifiy all PVs with the lvreduce command that are in the "same mirror set of disks" as the faulty one. Otherwise LVM may pick the "wrong" disks for reduction, leading to undesired results (e.g. asymmetrical layouts). Take a note of this PV list, since you need this information later when you re-establish the mirror using lvextend.

   ```
   # lvreduce -m 0 -A n /dev/vgXX/lvolX <list of PVs>      (for 1 way mirroring)
   ```
   or
   ```
   # lvreduce -m 1 -A n /dev/vgXX/lvolX <list of PVs>      (for 2 way mirroring)
   ```

   where *list of PVs* is the the list of devices determined according to the note above. We use the `–A n` option to prevent the lvreduce command from performing an automaticic vgcfgbackup operation, which is likely to get stuck on accessing a defective disk.

2. Replace the faulty disk. Please refer to the appropriate administration guide for instructions on how to replace the disk.

   Do an ioscan on the replaced disk to insure that it is accessible (`CLAIMED`) and also as a double check that it is a proper replacement (see note above).

   ```
   # ioscan -f /dev/dsk/cXtXdX
   ```

3. For fibre channel disks perform the `replace_dsk` steps described in the section "*How to Replace Disks at Hosts with TachLite HBAs*" in the Fibre Channel chapter.

4. Restore the LVM configuration/headers onto the replaced disk from your backup of the LVM configuration.

```
# vgcfgrestore -n VG /dev/rdsk/cXtXdX
```

5. Attach the new disk to the active volume group with the vgchange command.

```
# vgchange -a y vgXX
```
or
```
# vgchange -a e vgXX
```
(for exclusively activated Cluster VGs)

6. **Important:** If the disk is the mirror of a root disk, then you must configure the LIF/BDRA according to the LIF/BDRA Configuration Procedure at the end of this chapter.

7. Lvextend the mirrors back onto the replaced disk. This may take several minutes as it will have to copy all the data from the original copy of the data to the mirrored extents. The logical volume(s) are still accessible to users' applications during this command.

```
# lvextend -m 1 /dev/vgXX/lvolX /dev/dsk/cXtXdX &
```
(for 1 way mirroring)

or
```
# lvextend -m 2 /dev/vgXX/lvolX /dev/dsk/cXtXdX &
```
(for 3 way mirroring)

To check the progress of the synchronization you could use:

```
# lvdisplay -v $(find /dev/vgXY -type b) | grep stale | wc -l
```

A shell loop like this could be used to extend a bunch of lvols automatically:

```
# for lvol in lvol1 lvol2 lvol3 ...
```
(specify any LV you need to mirror)

```
> do
> lvextend –m 1 /dev/vgXX/$lvol /dev/dsk/cXtXdX
> done
```

Hot-Swap Procedure for Unattached Physical Volumes

Follow these steps to replace a hot-swap disk module for unattached physical volumes.

6. Replace the faulty disk. Please refer to the appropriate administration guide for instructions on how to replace the disk.

   Do an ioscan on the replaced disk to insure that it is accessible (`CLAIMED`) and also as a double check that it is a proper replacement (see note above).

```
# ioscan -f /dev/dsk/cXtXdX
```

7. For fibre channel disks perform the `replace_dsk` steps described in the section "*How to Replace Disks at Hosts with TachLite HBAs*" in the Fibre Channel chapter.

8. Restore the LVM configuration/headers onto the replaced disk from your backup of the LVM configuration.

```
# vgcfgrestore -n VG /dev/rdsk/cXtXdX
```

9. Attach the new disk to the active volume group with the vgchange command.

```
# vgchange -a y vgXX
```
or
```
# vgchange -a e vgXX
```
(for exclusively activated Cluster VGs)

10. **Important:** If the disk is the mirror of a root disk, then you must configure the LIF/BDRA according to the LIF/BDRA Configuration Procedure at the end of this chapter.

11. Resynchronize the mirrors of the replaced disk. This may take several minutes as it will have to copy all the data from the original copy of the data to the mirrored extents. The logical volume(s) are still accessible to users' applications during this command.

```
# vgsync vgXY &
```

To check the progress of the synchronization you could use:

```
# lvdisplay -v $(find /dev/vgXY -type b) | grep stale | wc -l
```

Removing a Ghost Disk using 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 has failed or not even physically connected but still recorded in the lvmtab. Such a PV is sometimes called a "ghost disk" or "phantom disk". You can get a ghost disk if the disk has failed before VG activation, maybe because the system has been rebooted after the failure.

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/vgXX.conf[.old]) is missing or corrupted you have to remove that PV from the VG (vgreduce) to get a clean configuration.

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 before removing the PV.

A *ghost disk* is usually indicated by vgdisplay reporting 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
```

Note that the PV c0t11d0 is still recorded in lvmtab:

```
# 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     ???
...
```

In this case lvol1 is having extents on device c0t11d0. You have to remove these extents from the PV before you are allowed to actually remove the PV from the VG. If the LV is mirrored use the command lvreduce to remove its mirrored extents. If the LV is unmirrored, 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
```

In this example you can see, that the LV in question is mirrored. One of its PVs is not attached to the VG, so its device file is unknown to LVM and displayed as “???”. Addressing this PV is no longer possible using the device file name

Removing a PV using its PV key

The PV key of a disk indicates its 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. This usually happens if it was not accessible during activation, e.g. due to a hardware or configuration problem.

NOTE: The PV may be unattached due to some temporary problem during VG activation which is no longer present. In this case you should try to re-activate the VG to force LVM to re-scan the devices listed in lvmtab:

```
# vgchange -a y vgXX
```
or
```
# vgchange -a e vgXX
```
(for exclusively activated Cluster VGs)

If the problem persists follow these steps to clear the situation:

1. Obtain the PV key 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.

2. Reduce the mirror with the obtained key as argument:

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

3. 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
```

4. Perform the above steps indicated above 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
```

5. Check the results:

```
# strings /etc/lvmtab

/dev/vg01
/dev/dsk/c0t0d2
/dev/dsk/c1t2d2
```

6. Re-activate the VG and backup the LVM config:

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

If the LV was not mirrored, re-create the LV (lvcreate), create a FS on it (newfs) and recover your data from backup.

Increasing the Root LV's size

Usually you cannot easily add space to the root LVs (/ or /stand) because they need to be contiguous. The following procedures work around this.

Using Ignite/UX

The recommended and only supported procedure to add space to the root LVs is to use Ignite/UX, e.g. a *make_tape_recovery* Medium (refer to the Ignite-UX chapter for details). To create a recovery tape with Ignite/UX containg the entire root VG just insert a medium into the drive an run:

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

If for some reason the above does not apply you may use the unofficial (and also unsupported) procedure below.

Using the Unofficial Procedure

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:

1. Create a new VG vgroot with c1t1d0:

```
# pvcreate -B /dev/rdsk/c1t1d0                    (don't forget the –B option!)
# mkdir /dev/vgroot
# ll /dev/*/group                                 (check for unused minor number)
# mknod /dev/vgroot/group c 64 0x010000
# vgcreate vgroot /dev/dsk/c1t1d0
```

2. Create LVs for boot, swap and root (in that order). Use at least the same size as in your original root VG:

```
# lvcreate -C y -r n vgroot
# lvextend -L 100 /dev/vgroot/lvol1               (e.g. 100 MB for /stand)

# lvcreate -C y -r n vgroot
# lvextend -L 512 /dev/vgroot/lvol2               (e.g. 512 MB pri. swap)

# lvcreate -C y -r n vgroot
# lvextend -L 200 /dev/vgroot/lvol3               (e.g. 200 MB for /)
```

3. Configure LIF and BDRA on c1t1d0 (see the LIF/BDRA Configuration Procedure).

4. Create LVs for `/usr`, `/opt`, `/var`, `/tmp`, `/etc`, `/home`, etc. Use at least the same size as in your original root VG:

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

5. Create the file systems:

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

6. Mount the file systems:

```
# mkdir /new_root /new_usr /new_stand …           (Create mount points)
# mount /dev/vgroot/lvol1 /new_stand
# mount /dev/vgroot/lvol3 /new_root
# mount /dev/vgroot/lvol4 /new_usr
...
```

7. Copy the data, e.g. using find(1) with cpio(1):

```
# 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
...
```

8. Modify the fstab in /new_root/etc. Replace occurences of *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)
```

9. Change the device files for the root disk in /stand/bootconf to c1t1d0:

```
# vi /stand/bootconf

l  /dev/dsk/c1t1d0
```

10. Configure disk c1t1d0 as boot path in stable storage and boot from it:

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

11. When the system comes up again, backup vgroot's LVM Configuration:

```
# vgcfgbackup vgroot
```

12. And finally remove the old root VG if desired:

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

If you like to rename vgroot to vg00:

1. Boot to LVM maintenance mode:

```
ISL> hpux –lm
```

2. Export vgroot and import it as vg00:

```
# vgexport vgroot
# mkdir /dev/vg00
# mknod /dev/vg00/group c 64 0x000000        (we import vg00 with minor 0)
# vgimport vg00 /dev/dsk/c1t1d0
```

3. Activate vg00 and mount the files ystems:

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

4. Modify the fstab. Replace vgroot with vg00 again:

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

5. Reboot:

```
# shutdown -r 0
```

LIF/BDRA Configuration Procedure

This subprocedure installs/updates information on disk that is **mandatory** for boot support. Therefore it is referenced from several other parts of this chapter.

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

```
# mkboot -l /dev/rdsk/cXtXdX
# lifls –l /dev/rdsk/cXtXdX
```
(to ckeck it)

2. Write content of AUTO File: *(if autoboot is desired)*

```
# mkboot -a hpux /dev/rdsk/cXtXdX
```
(autoboot with qurom enforced)
```
# mkboot -a 'hpux –lq' /dev/rdsk/cXtXdX
```
(autoboot without qurom enforced)
```
# lifcp /dev/rdsk/cXtXdX:AUTO -
```
(to ckeck it)

NOTE: By default, LVM enforces a quorum of >50% of a VG's PVs being available at activation time. If e.g. the root VG contains 2 PVs, then the system rejects to boot unless you disable the quorum check using the –lq option.

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/cXtXdX
```
(if 32 or 32/64)

```
# mkboot -b updatediaglif2 -p ISL -p AUTO -p HPUX -p LABEL
  /dev/rdsk/cXtXdX
```
(if 64)

(the -p option preserve the specified file so that it is not overwritten in LIF)

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
# lvlnboot -b /dev/<rootVG>/lvol1
# 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
`pvremove`	remove LVM data structure from a PV
`pvck`	check or repair a physical volume in a VG

NOTE: All LVM commands are hard-linked to the same single executable, e.g:

```
# ll -iF vg* lv* pv* nomwc*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvchange*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvcreate*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvdisplay*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvextend*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvlnboot*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvmerge*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvreduce*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvremove*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvrmboot*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvsplit*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 lvsync*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 nomwcsyncd*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 pvchange*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 pvck*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 pvcreate*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 pvdisplay*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 pvmove*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 pvremove*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgcfgbackup*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgcfgrestore*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgchange*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgchgid*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgcreate*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgdisplay*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgexport*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgextend*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgimport*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgreduce*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgremove*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgscan*
   241 -r-sr-xr-x  31 root       sys         557056 Jan 30 11:03 vgsync*
```