Chapter 13

LVM

HP-UX Handbook
Revision 13.00

TERMS OF USE AND LEGAL RESTRICTIONS FOR THE HP-UX RECOVERY HANDBOOK

ATTENTION: PLEASE READ THESE TERMS CAREFULLY BEFORE USING THE HP-UX HANDBOOK. USING THESE MATERIALS INDICATES THAT YOU ACCEPT THESE TERMS. IF YOU DO NOT ACCEPT THESE TERMS, DO NOT USE THE HP-UX HANDBOOK.

THE HP-UX HANDBOOK HAS BEEN COMPILED FROM THE NOTES OF HP ENGINEERS AND CONTAINS HP CONFIDENTIAL INFORMATION.

THE HP-UX HANDBOOK IS NOT A PRODUCT QUALITY DOCUMENT AND IS NOT NECESSARILY MAINTAINED OR UP TO DATE. THE HP-UX HANDBOOK IS HERE MADE AVAILABLE TO HP CONTRACT CUSTOMERS FOR THEIR INTERNAL USE ONLY AND ON THE CONDITION THAT NEITHER THE HP-UX HANDBOOK NOR ANY OF THE MATERIALS IT CONTAINS IS PASSED ON TO ANY THIRD PARTY.

Use of the HP-UX Handbook: Hewlett-Packard Company ("HP") authorizes you to view and download the HP-UX Handbook only for internal use by you, a valued HP Contract Customer, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials. You may not modify the HP-UX Handbook in any way or publicly display, perform, or distribute or otherwise use them for any public or purpose outside your own business. The materials comprising the HP-UX Handbook are copyrighted and any unauthorized use of these materials may violate copyright, trademark, and other laws. If you breach any of these Terms, your authorization to use the HP-UX Handbook automatically terminates and you must immediately destroy any downloaded or printed materials.

Links To Other Web Sites: Links to third party Web sites provided by the HP-UX Handbook are provided solely as a convenience to you. If you use these links, you will leave this Site. HP has not reviewed all of these third party sites and does not control and is not responsible for any of these sites or their content. Thus, HP does not endorse or make any representations about them, or any information, software or other products or materials found there, or any results that may be obtained from using them. If you decide to access any of the third party sites linked to this Site, you do this entirely at your own risk.

Disclaimer: THE HP-UX HANDBOOK IS PROVIDED "AS IS" WITHOUT ANY WARRANTIES OF ANY KIND INCLUDING WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF INTELLECTUAL PROPERTY. HP further does not warrant the accuracy and completeness of the materials in the HP-UX Handbook. HP may make changes to the HP-UX Handbook at any time without notice. The HP-UX Handbook may be out of date, and HP makes no commitment to update the HP-UX Handbook. Information in the HP-UX Handbook may refer to products, programs or services that are not available in your country. Consult your local HP business contact for information regarding the products, programs and services that may be available to you.

Limitation of Liability: IN NO EVENT WILL HP, ITS SUPPLIERS, OR OTHER ANY THIRD PARTIES MENTIONED IN THE HP-UX HANDBOOK BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, THOSE RESULTING FROM LOST PROFITS, LOST DATA OR BUSINESS INTERRUPTION) ARISING OUT OF THE USE, INABILITY TO USE, OR THE RESULTS OF USE OF THE HP-UX HANDBOOK, WHETHER BASED ON WARRANTY, CONTRACT, TORT OR ANY OTHER LEGAL THEORY AND WHETHER OR NOT ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS DOES NOT APPLY IN CASE OF INTENT OR IF LIABILITY IS LEGALLY STIPULATED. IF YOUR USE OF THE HP-UX HANDBOOK RESULTS IN THE NEED FOR SERVICING, REPAIR OR CORRECTION OF EQUIPMENT OR DATA, YOU ASSUME ALL COSTS THEREOF.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Applicable Laws: These Terms will be governed by and construed in accordance with the laws of the State of California, without giving effect to any principles of conflicts of laws.

General: HP may revise these Terms at any time by updating this posting. ***Revised Oct 2013***

FEEDBACK or QUESTIONS: please email essam.ackleh@hp.com
(please use subject syntax: *HP-UX Handbook v13.00 Chapter <YY> - <Feedback Title>*

TABLE OF CONTENTS

LVM Versions

As of the March 2010 release of HP-UX 11i Version 3, LVM supports four versions of volume groups. Version 1.0, 2.0, 2.1, and 2.2 enable the configuration of larger volume groups, logical volumes, physical volumes, and other parameters.

Version #		
Supported	Supported	Bootable Volume Group
Version 1.0	HP-UX 11i	yes
Version 2.0	HP-UX 11i V3 March 2008.	no
Version 2.1	HP-UX 11i V3 September 2008	no
Version 2.2	HP-UX 11i V3 March 2010 Update and forward.	yes

Device Files

As of HP-UX 11i Version 3, disk devices can be represented by legacy and persistent device files in the /dev directory.

Legacy device hardware path information such as SCSI bus, target, and LUN encoded in the device file name and minor number. For example, the legacy device file /dev/dsk/c3t2d0 represents the disk at card instance 3, target address 2, and lun address 0.

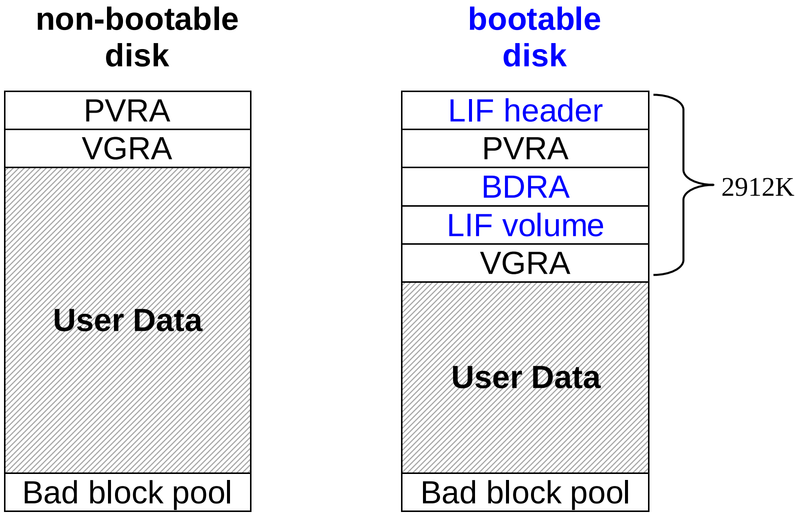
Persistent device files are not tied to the physical hardware path to a disk, but instead map to the disk's unique worldwide identifier (WWID). For example, /dev/disk/disk*n*, where *n* is the instance number assigned to the disk.

Serviceguard A.11.20 uses cluster-wide device files (cDSFs). A cluster device special file provides a consistent set of device special files across a set of cluster nodes. For example, /dev/cdisk/disk*n*, where *n* is the instance number assigned to the disk.

LVM Structural Information

This information applies only to disks belonging to Version 1.0 and Version 2.2 (or higher) volume groups.There are two kinds of LVM disk layouts—for boot disks and all

other LVM disks—and they differ in their data structures. Non-bootable disks have two reserved areas: the physical volume reserved area (PVRA) and the volume group reserved area (VGRA). Bootable disks have a PVRA and VGRA, and additional sectors reserved for the boot data reserved area (BDRA) and boot LIF.
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 have a 100MB EFI partition at the beginning of the disk. Refer to the Itanium Chapter 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 automatically 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 utility lvmmeta displays the complete LVM header:

```
# lvmmeta -d /dev/rdisk/disk3_p2
…
…
0x00002008   uint32_t id1; 0x413b03fc CPU ID = 1094386684
0x0000200c   uint32_t id2; 0x4c0d30d2 Time = Mon Jun  7 11:48:02 2010
```

i.e. pvcreate(1m) was run on CPU with ID 1094386684 at Mon Jun 7 11:48:02 2010

```
…
0x00002010   uint32_t id1; 0x413b03fc CPU ID = 1094386684
0x00002014   uint32_t id2; 0x4c0d30d3 Time = Mon Jun  7 11:48:03 2010
```

i.e. vgcreate(1m) was run on CPU with ID 1094386684 at Mon Jun 7 11:48:03 2010

Since the lvm tool lvmmeta 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/rdisk/disk3_p2
0000000      1094386684      1275932882         1094386684      1275932883
              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) 1094386684
- pvcreate was run at timestamp 1275932882 (seconds after Jan 1st 1970 0:00 UTC)
-
- vgcreate was run at timestamp 1275932883 (1 second later)

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 accidentally 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, /etc/lvmtab_p and vgscan(1M)

The files /etc/lvmtab and /etc/lvmtab_p contain information about all known VGs and their PVs. It is mainly used by vgchange(1M) at VG activation time. lvmtab is used for LVM 1.0 and is a binary file and lvmtab_p is used for LVM 2.x and is a ascii file. To display the lvmtab file 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

To read the lvmtab_p file use the cat command:

```
# cat /etc/lvmtab_p
ÐÐ/dev/vg_lvm2A0000000000000009Thu Jun 23 18:30:55 2011523dfd66-9dfd-11d8-
be52-c83e68f1c426/dev/disk/disk111
```

To read the VGID for any volume group in the /etc/lvmtab file using the command below.

Note: the command below is grepping out volume group vgcat8.

```
# strings -t d /etc/lvmtab | grep /dev | grep -v /dev/dsk | while read
  offset path; do xd -An -j$(($offset+1024)) -N8 -tx /etc/lvmtab | read
  vgid1 vgid2; echo $path $vgid1 $vgid2; done | grep "vgcat8 "
```

The line below is returned containing the VGID:

```
/dev/vgcat8 4ec89e3f 48f7b7ba
```

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.

To disable automatic volume group activation, set AUTO_VG_ACTIVATE to 0.
In /sbin/lvmrc

AUTO_VG_ACTIVATE=0

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.

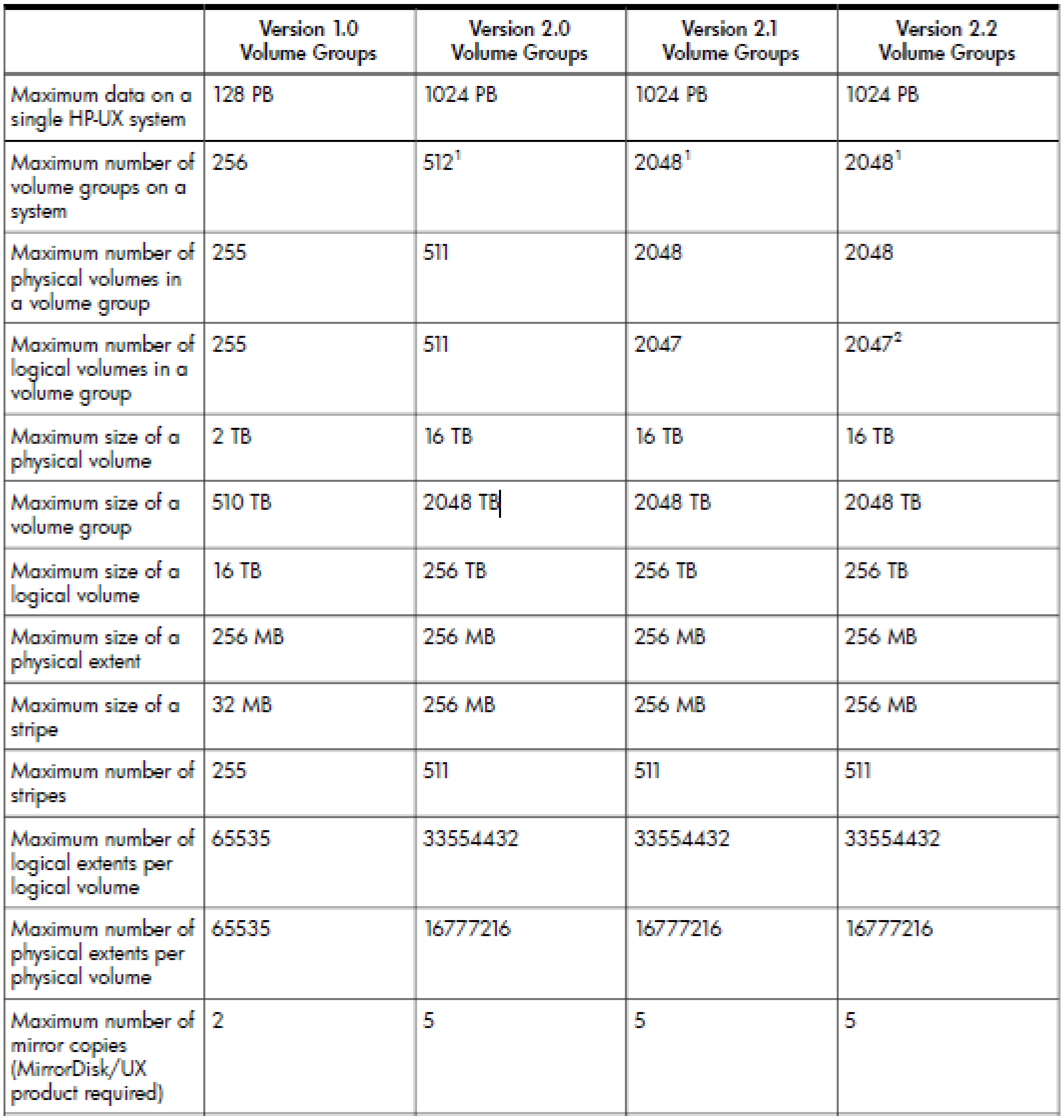
ATTENTION: On systems using data replication products like BusinessCopy/XP, ContinousAccess/XP, EMC SRDF or EMC Timefinder vgscan may accidently add undesired PVs to VGs.

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

The following table summarizes the supported limits in LVM for HP-UX.



	Version 1.0 Volume Groups	Version 2.0 Volume Groups	Version 2.1 Volume Groups	Version 2.2 Volume Groups
Maximum data on a single HP-UX system	128 PB	1024 PB	1024 PB	1024 PB
Maximum number of volume groups on a system	256	512[1]	2048[1]	2048[1]
Maximum number of physical volumes in a volume group	255	511	2048	2048
Maximum number of logical volumes in a volume group	255	511	2047	2047[2]
Maximum size of a physical volume	2 TB	16 TB	16 TB	16 TB
Maximum size of a volume group	510 TB	2048 TB	2048 TB	2048 TB
Maximum size of a logical volume	16 TB	256 TB	256 TB	256 TB
Maximum size of a physical extent	256 MB	256 MB	256 MB	256 MB
Maximum size of a stripe	32 MB	256 MB	256 MB	256 MB
Maximum number of stripes	255	511	511	511
Maximum number of logical extents per logical volume	65535	33554432	33554432	33554432
Maximum number of physical extents per physical volume	65535	16777216	16777216	16777216
Maximum number of mirror copies (MirrorDisk/UX product required)	2	5	5	5

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.

Calculating an optimal extent size for a version 1.0 volume group.

Sometimes when creating a volume group (VG), the *vgcreate(1M)* command may abort with a message that extent size is too small (too big error or with newer patches a more informative error explaining that the VGRA is too big). In this situation the user is expected to increase the extent size and re-issue the *vgcreate(1M)* command. Increasing the extent size increases the data area marked stale when a write to a mirrored logical volume fails and that can increase the time required for re-synchronizing the stale data. Also, more space than intended may be allocated to the logical volume since the space is allocated in units of extent size. Therefore, the optimal extent size is the smallest value that can be used to successfully create the volume group with the given configuration parameters.

The minimum extent size for a volume group is calculated using the maximum number of, logical volumes MAXLVs and physical volumes (MAXPVs) in the volume group and the maximum number of physical extents (MAXPXs) per each physical volume.

For a VG with bootable PVs, the metadata must fit within 768 Kbytes. Therefore, a *vgcreate(1M)* command with a set of values for MAXLVs, MAXPVs and MAXPXs that succeed on a VG without bootable PVs, may fail on a VG with bootable PVs. In this situation, if the user needs to add a bootable PV to a VG, they must recreate the VG by giving lesser values for these arguments. By far the biggest factor in the size of the metadata is the values for MAXPVs and MAXPXs. Alternatively, they can convert the bootable PV to a normal PV by rerunning *pvcreate(1M)* on that PV without '-B' option and then add it to the VG.

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.

		PE size (vgcreate –s *pe_size*) in MB								
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
PVs/VG (vgcreate –p *max_pv*)	**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								
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
PVs/VG (vgcreate –p *max_pv*)	**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

Calculating an optimal extent size for a version 2.x volume group.

Like Version 1.0 Volume Groups, there is a relationship between extent size and maximum volume group size. There is also a limitation on the number of extents an individual volume group can contain. To determine the proper size, the vgcreate command has a new –E option. This option displays the maximum volume group size based on the given physical extent size or minimum physical extent size based on the maximum volume group size.

Example
After you know the VG size you want to provision for, use vgcreate with the –E option to determine the minimum extent size required to achieve it.

What is the minimum extent size to provision a 2.0 VG for 1 PB?

```
# vgcreate -V 2.0 -E -S 1p
Max_VG_size=1p:extent_size=32m
```

What is the maximum 2.0 volume group size with an extent size of 16MB?

```
#vgcreate -V 2.0 -E -s 16
Max_VG_size=512t:extent_size=16m
```

Supported file and file system sizes

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.

The OnlineJFS and HFS supported size could be found in the JFS/OnlieJFS chapter.

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 1.0 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

 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
```

Information on 2.x VGs

```
# vgdisplay  -v /dev/vg_lvm2
--- Volume groups ---
VG Name                    /dev/vg_lvm2
VG Write Access              read/write
VG Status                  available
Max LV                    511
Cur LV                    0
Open LV                    0
Max PV                    511
Cur PV                    1
Act PV                    1
Max PE per PV               2097152
VGDA                      2
PE Size (Mbytes)            4
Total PE                  10229
Alloc PE                  0
Free PE                   10229
Total PVG                  0
Total Spare PVs             0
Total Spare PVs in use      0
VG Version                 2.0
VG Max Size                 8t
VG Max Extents              2097152
```

```
    --- Physical volumes ---
    PV Name                    /dev/disk/disk111
    PV Status                  available
    Total PE                   10229
    Free PE                    10229
    Autoswitch                 On
    Proactive Polling          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 1.0 PVs

The pvdisplay command prints physical volume information, including status of physical extents:

```
  # 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 2.x PVs

```
# pvdisplay  -v /dev/disk/disk111

--- Physical volumes ---
PV Name                    /dev/disk/disk111
VG Name                     /dev/vg_lvm2
PV Status                 available
Allocatable               yes
VGDA                      2
Cur LV                   1
PE Size (Mbytes)           4
Total PE                  10229
Free PE                   10228
Allocated PE               1
Stale PE                  0
IO Timeout (Seconds)       default
Autoswitch                 On
Proactive Polling           On

   --- Distribution of physical volume ---
   LV Name                   LE of LV PE for LV
   /dev/vg_lvm2/lvol1      1        1

   --- Physical extents ---
   PE             Status         LV                              LE
```

```
00000000   current       /dev/vg_lvm2/lvol1     00000000
00000001   free                                 00000000
00000002   free                                 00000000
```

Information on 1.0 LVs

The lvdisplay command prints logical volume information, including mapping and status of logical extents.

```
# 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.

Information on 2.x LVs

```
# lvdisplay -v  /dev/vg_lvm2/lvol1
--- Logical volumes ---
LV Name                     /dev/vg_lvm2/lvol1
VG Name                     /dev/vg_lvm2
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               0
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            4
Current LE                  1
Allocated PE                1
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   NONE
Allocation                  strict
IO Timeout (Seconds)        default

   --- Distribution of logical volume ---
   PV Name            LE on PV  PE on PV
   /dev/disk/disk111  1         1

   --- Logical extents ---
   LE       PV1                PE1       Status 1
   00000000 /dev/disk/disk111  00000000 current
```

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
```

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

1) initialize the disk if not yet done:

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

If you are using a disk on an HP Integrity server, make sure the device file specifies the HP-UX partition number (2).
For example:

```
# pvcreate [-f] /dev/rdisk/disk3_p2
```

2) select a unique minor number for the VG:

```
# ll /dev/*/group
crw-r-----     1 root     sys      64 0x000000 Mar 12  2009   /dev/vg00/group
crw-rw-rw-   1 root     sys      64 0x010000 Jan 19  2010   /dev/vgISO/group
crw-rw-rw-   1 root     sys      64 0x020000 Jun 12  2009   /dev/vg_test/group
crw-r--r--     1 root     sys    128 0x001000 Jun 23 12:30   /dev/vg_local/group
crw-rw-rw-   1 root      sys    128 0x060000 Jan 18  2010  /dev/vgmore/group
```

NOTE: The first 3 device files are LVM 1.0 volume groups and the last two Device files are LVM 2.x volume groups

3) create the /dev directory

```
# mkdir /dev/vgnew
```

4) create a device file named group in the volume group directory with the mknod command.

For example:

```
# mknod /dev/vgname/group c major 0xminor
```

The c following the device file name specifies that group is a character device file.
major is the major number for the group device file.
For a Version 1.0 volume group, it is 64.
For a Version 2.x volume group, it is 128.

minor is the minor number for the group file in hexadecimal.
For a Version 1.0 volume group, *minor* has the form 0x*nn*0000, where *nn* is a unique number across all Version 1.0 volume groups.
For a Version 2.x volume group, *minor* has the form 0x*nnn*000, where *nnn* is a unique number across all Version 2.x volume groups.

5) creating a volume group

Version 1.0 Volume Group

# vgcreate /dev/*vgname* /dev/disk/disk3

Version 2.x Volume Group

# vgcreate -V 2.0 -s *pe_size* -S *vg_size* /dev/*vgname* /dev/disk/disk3

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 1.0 or 2.x 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 modification 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 it is recommend to use the "-f outfile" option with vgexport and vgimport. See section Importing and exporting VGs for details.

```
# 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
```

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.

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
```

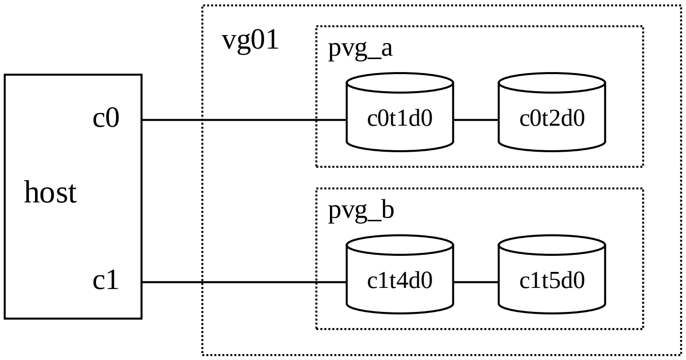
should show "distributed".

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 its 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 (PA-RISC Systems)

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(specify all LVs you need  to reduce)
> do
> lvreduce -m 0 /dev/vg00/$lvol /dev/dsk/c1t6d0
> done
```

Root Mirror (Itanium based Systems)

The following procedure shows how to mirror the root disk. Let c3t2d0 be the existing primary disk and c2t1d0 the new mirror boot disk:

1. Setup the disk partitions
   A. Create a partition description file:

   ```
   # vi /tmp/partitionfile
   2
   EFI 100MB
   HPUX 100%
   ```

   B. Use the idisk(1M) command to partition the disk according to this file:

   ```
   # idisk -wf /tmp/partitionfile /dev/rdsk/c2t1d0
   idisk version: 1.2
   ********************** WARNING ***********************
   If you continue you may destroy all data on this disk.
   Do you wish to continue(yes/no)? yes
   ...
   ```

2. Create the new device files for the new partitions (c2t1d0s1 and c2t1d0s2)

   ```
   # insf -e -C disk
   ```

3. Use mkboot(1M) to format the EFI partition (s1), populate it with the EFI files under /usr/lib/efi/, format the LIF volume (part of s2), and populate it with the LIF files (ISL, AUTO, HPUX, LABEL) under /usr/lib/uxbootlf:

   ```
   # mkboot -e -l /dev/rdsk/c2t1d0
   # efi_ls -d /dev/rdsk/c2t1d0s1              (to check EFI)
     FileName                  Last Modified        Size
     EFI/                    5/19/2003            0
     STARTUP.NSH                     5/19/2003          336
   ```

```
    total space 103215616 bytes, free space 100076032 bytes

# lifls -l /dev/rdsk/c2t1d0s2             (to check LIF)
```

4. Write the contents of the AUTO file to the EFI partition:

```
# mkboot -a "boot vmunix" /dev/dsk/c2t1d0
# efi_cp -d /dev/rdsk/c2t1d0s1 -u /EFI/HPUX/AUTO /tmp/x;

# cat /tmp/x
```

(to check it)

NOTE: Specify -a "boot vmunix -lq" if you want the system to boot up without interruption in case of a disk failure.

5. Initialize the LVM partition (s2) and add it to vg00:

```
# pvcreate [-f] -B /dev/rdsk/c2t1d0s2 (take care to use s2)
# vgextend vg00 /dev/dsk/c2t1d0s2
```

A. Mirror the LVs to the s2 partition:

```
# for i in lvol1 lvol ... lvol8
> do lvextend -m 1 /dev/vg00/$i /dev/dsk/c2t1d0s2
> done
```

B. Write the contents of the LABEL file, i.e. set root, boot, swap and dump device:

```
# lvlnboot -r /dev/vg00/lvol3
# lvlnboot -b /dev/vg00/lvol1
# lvlnboot -s /dev/vg00/lvol2
# lvlnboot -d /dev/vg00/lvol2

# lvlnboot -v                        (to check it)

Boot Definitions for Volume Group /dev/vg00:

Physical Volumes belonging in Root Volume Group:
/dev/dsk/c3t2d0s2 (0/1/1/1.2.0) -- Boot Disk
/dev/dsk/c2t1d0s2 (0/1/1/0.1.0) -- Boot Disk
```

```
Boot: lvol1      on:     /dev/dsk/c3t2d0s2
                         /dev/dsk/c2t1d0s2
Root: lvol3      on:     /dev/dsk/c3t2d0s2
                         /dev/dsk/c2t1d0s2
Swap: lvol2     on:     /dev/dsk/c3t2d0s2
                        /dev/dsk/c2t1d0s2
Dump: lvol2     on:     /dev/dsk/c3t2d0s2, 0
```

6. Specify the mirrored disk as an alternate bootpath

   # *setboot -a <HW path of mirror>*
   # *setboot* (to check it)

7. Create an EFI boot option. This boot option will be stored in NVRAM:

   Boot to EFI and enter the Boot option maintenance menu:

```
------------------------------------------------------------
| EFI Boot Maintenance Manager ver 1.10 [14.61]
|
| Main Menu. Select an Operation
|
|
|        Boot from a File
|   >>>  Add a Boot Option
|        Delete Boot Option(s)
|        Change Boot Order
|
|        Manage BootNext setting
|        Set Auto Boot TimeOut
|
|        Select Active Console Output Devices
|        Select Active Console Input Devices
|        Select Active Standard Error Devices
|
|        Cold Reset
|        Exit
------------------------------------------------------------
```

Select the mirror disk. (Pun1,Lun0) represents SCSI target 1, Lun 0 (c2t1d0) in our case. (Pun2,Lun0) is the original boot disk (c3t2d0):

```
-------------------------------------------------------------------
```

```
    | EFI Boot Maintenance Manager ver 1.10 [14.61]
    |
    | Add a Boot Option.  Select a Volume
    |
    | >>> IA64_EFI [Acpi(HWP0002,100)/Pci(1|0)/
    |                    Scsi(Pun1,Lun0)/HD(Part1,SigB45A0000)
    |    IA64_EFI [Acpi(HWP0002,100)/Pci(1|1)/
    |                    Scsi(Pun2,Lun0)/HD(Part1,Sig958B0000)
    |    EFI DISK [Acpi(HWP0002,600)/Pci(1|0)/Pci(0|0)/Pci(0|0)/
    |         Pci(0|0)/Scsi(Pun0,Lun0)/
    |         HD(Part1,Sig   119E1A60-0B4C-01C3-507B- 9E5F8078F531)
    |    Removable Media Boot [Acpi(HWP0002,0)/Pci(2|0)/
    |             Ata(Primary,Master)]
    |    Load File [EFI Shell [Built-in]]
    |    Load File [Acpi(HWP0002,0)/Pci(3|0)/Mac(00306E3809C6)]
    |    Load File [Acpi(HWP0002,100)/Pci(2|0)/Mac(00306E3889E3)]
    |    Exit
    ---------------------------------------------------------------
```

Now navigate to the HP-UX bootloader, HPUX.EFI, on the disk:

```
-----------------------------------------------------------------
    | EFI Boot Maintenance Manager ver 1.10 [14.61]
    |
    | Select file or change to new directory:
    |
    |  >>>  05/28/03  09:38a <DIR>        512 EFI
    |      [Treat like Removable Media Boot]
    |    Exit

    | Select file or change to new directory:
    |
    |      05/28/03  09:38a <DIR>       512 .
    |      05/28/03  09:38a <DIR>         0 ..
    |  >>>  05/28/03  09:38a <DIR>       512 HPUX
    |      05/28/03  09:38a <DIR>       512 Intel_Firmware
    |      05/28/03  09:38a <DIR>       512 DIAG
    |      05/28/03  09:38a <DIR>       512 HP
    |      05/28/03  09:38a <DIR>       512 TOOLS
    |    Exit

    | Select file or change to new directory:
    |
    |      05/28/03  09:38a <DIR>       512 .
```

```
       |      05/28/03  09:38a <DIR>        512 ..
       |  >>>  05/28/03  11:52a         419,545 HPUX.EFI
       |      05/28/03  11:52a         24,576 NBP.EFI
       |    Exit
       |
       |
       |    Filename: \EFI\HPUX\HPUX.EFI
       |
       | DevicePath:[Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun1,Lun0)/
       |            HD(Part1,SigB45A0000)/\EFI\HPUX\HPUX.EFI]
       |    IA-64 EFI Application 05/28/03  11:52a     419,545 bytes
       |
       |    BootFFFF: Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun1,Lun0)/
       |            HD(Part1,SigB45A0000)/\EFI\HPUX\HPUX.EFI
   ------------------------------------------------------------------
```

Now enter a description for this boot option, e.g. "HP-UX mirror boot disk":

```
   ------------------------------------------------------------------
     |    Enter Description:  HP-UX mirror boot disk
     |
     |    Current BootOption-->Main Menu. Select an Operation
     |    New BootOption Data. ASCII/Unicode strings only, with max of
     |            240 characters
     |    Enter BootOption Data Type [A-Ascii U-Unicode
     |            N-No BootOption] : N
   ------------------------------------------------------------------
```

Finally save the setting to NVRAM:

```
     |    Save changes to NVRAM [Y-Yes N-No]:  Y
```

8. Try to boot from the mirror disk by choosing the appropriate boot option.

PV Links (Alternate Paths)

Physical Volume Links (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 retries 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.

LVM striping

Introduction

The "striping" of Logical Volumes is a well-proven method of balancing I/O across multiple disks and I/O interfaces. It is particularly appropriate for large intensively accessed databases, and is often a more practicable approach to achieving good throughput than any attempt to "manually" position file systems on specific hardware.

For some time now it has been possible to set up "Extent-based" striping quite easily - this involves the distribution of the extents in a Logical Volume in a "round-robin" fashion across two or more PVs. This has always been possible with LVM, but prior to the "-D" option being made available for the "lvcreate" command, it was necessary to use scripting to build LVs with distributed extents.

An alternative "Block" striping method (-i and -I options on lvcreate) is easy to use, but cannot be used with Mirrordisk/UX. It is also impossible to use the "pvmove" command with LVs created in this way, and extension of LVs can be difficult or impossible if PVs are filled in an uneven fashion - extent distribution is much more flexible in this respect.

It is also worth noting that the Extent-based striping approach is thought by many to be more appropriate when working with disk arrays with large cache sizes, such as the HP Storageworks XP and EMC models. This is because there is a large difference between the typical "stripe depth" used with Block striping (often 64Kb or less - though the size can be as large as 32 MB if desired) and Extent-based striping (from 1 MB to 256 MB); where intensive sequential I/O takes place, the cache-management algorithms are likely to work better with the larger stripe depth. With small stripe depths there is a danger that multiple streams of parallel sequential activity may start to be treated by the array management algorithms as if they were random access.

In general, it is better to use the term "Extent distribution" when discussing extend based "striping" - and the objective when setting up LVs in this way is to distribute the I/O load across multiple storage components in a large storage array with significant caching of sequential I/O. This is not the same as the approach to striping which was popular with JBOD disks, where the objective is also to speed up sequential activity, and where a small stripe "depth" may be appropriate.

Striping and Distributed Allocation Policy

LVM supports two types of striping. The first is *block based striping* which is configured using

lvcreate -I <stripe_size> -i <# of stripes>

NOTE: Block based striping is **not** supported in conjunction with LVM mirroring using MirrorDisk/UX.

The second striping option is *extent distribution*. Extent distribution is supported with LVM mirroring. The logical volumes are set up so that the next logical extent added is always placed to a different physical volume from the preceding logical extent. 1 MB is the smallest physical extent size - and it is important to note that the maximum LV size is determined by the extent size, which cannot be changed after a VG has been created - there can be a maximum of 64K extents in a single LV.

It is worth noting that the Distributed attribute may be changed with the lvchange(1M) command, and this includes an option to "force" the attribute onto the LV if its extents are not already distributed. This could be useful if there is a desire for future extension of the LV to use extent distribution, or perhaps if there is a need to create an additional mirror copy with its extents distributed across PVs in another PVG.

Procedure for creation of LVs with extent distribution:

1) Create the VG

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

NOTE: Ensure that the 0x0#0000 minor number is unique for each VG on your system.

```
# vgcreate -s 1 /dev/vgDist /dev/dsk/c0t0d0 /dev/dsk/c0t1d0 \
                      /dev/dsk/c1t2d0 /dev/dsk/c1t3d0
```

NOTE: The -s 1 in the above command specifies a PE size of 1 Mbyte. This is the smallest stripe or extent size available. The default would be 4MB. This is only an example, and the default or larger is appropriate in most cases.

2) Define PVGs

Physical volume groups can be created by using the "-g" option on the vgcreate and vgextend commands, or by manually editing the /etc/lvmpvg file. Refer to lvmpvg**(4)** man page for syntax.

Physical volume groups are used in conjunction with PVG-strict allocation policy. PVG-strict allocation ensures that primary and mirror extents do not reside within the same PVG. Typically each PVG consists of disk devices from the same controller card or set of controllers. Another example would be the case where LVs are mirrored between two separate disk arrays - the PVs in each of the arrays being assigned to separate PVGs.

Example /etc/lvmpvg file:

```
VG /dev/vgDist
PVG PVG1
/dev/dsk/c0t0d0
/dev/dsk/c0t1d0
PVG PVG2
/dev/dsk/c1t2d0
/dev/dsk/c1t3d0
```

NOTE: Notice that both disk devices on the c0 controller card are put in PVG1 and both devices on the c1 card are in PVG2. Extent distributed logical volumes within vgDist will be assured to have one mirror copy spread across the disk devices on the c0 controller and another mirrored copy striped across disk devices on the c1 controller card.

3) Create the distributed LV

```
# lvcreate  -D y -s g -m 1 -L 1000 -n lvdist1  vgDist
```

-D y	specifies distributed allocation policy
-s g	specifies PVG-strict allocation policy
-m 1	specifies 1 mirror copy
-L 1000	specifies size in MBytes = 1000
-n lvdist1	names logical volume dist1

```
# lvdisplay -v /dev/vgDist/lvdist1

--- Logical volumes ---
LV Name                     /dev/vgDist/lvdist1
VG Name                     /dev/vgDist
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               1
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            1000
Current LE                  1000
Allocated PE                2000
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   on
Allocation                  PVG-strict/distributed
IO Timeout (Seconds)        default

   --- Distribution of logical volume ---
   PV Name            LE on PV  PE on PV
   /dev/dsk/c0t0d0    500       500
   /dev/dsk/c0t1d0    500       500
   /dev/dsk/c1t2d0    500       500
   /dev/dsk/c1t2d0    500       500

   --- Logical extents ---
   LE   PV1                PE1  Status 1 PV2                PE2  Status 2
   0000 /dev/dsk/c0t0d0    0000 current  /dev/dsk/c1t2d0    0000 current
   0001 /dev/dsk/c0t1d0    0000 current  /dev/dsk/c1t3d0    0000 current
   0002 /dev/dsk/c0t0d0    0001 current  /dev/dsk/c1t2d0    0001 current
   0003 /dev/dsk/c0t1d0    0001 current  /dev/dsk/c1t3d0    0001 current
   0004 /dev/dsk/c0t0d0    0002 current  /dev/dsk/c1t2d0    0002 current
   ...
   ...
   0995 /dev/dsk/c0t1d0    0497 current  /dev/dsk/c1t3d0    0497 current
   0996 /dev/dsk/c0t0d0    0498 current  /dev/dsk/c1t2d0    0498 current
   0997 /dev/dsk/c0t1d0    0498 current  /dev/dsk/c1t3d0    0498 current
```

```
0998 /dev/dsk/c0t0d0   0499 current  /dev/dsk/c1t2d0   0499 current
0999 /dev/dsk/c0t1d0   0499 current  /dev/dsk/c1t3d0   0499 current
```

Notice how PV1 alternates physical extents between the disk devices in PVG1 and PV2 alternates between the disk devices in PVG2. This is extent based striping.

Comments on Physical Volume Groups (PVGs)

Before the "-D" option of the lvcreate command was introduced, PVGs were generally only set up for VGs with which Mirrordisk/UX was being used.

PVGs are subdivisions of a Volume Group - they are defined in an ASCII text file, /etc/lvmpvg, which has a simple format and can be edited manually. The file can be set up, and amended, by means of the vgcreate, vgextend and vgreduce commands, but it is also possible to set it up manually with "vi". It is important to note that the names of PVGs are only held in this file (so they can be changed easily, for instance). When a VG is exported (using vgexport) its entries are removed from the local /etc/lvmpvg, but they are not recreated when the VG is imported into the same or another system (using vgimport)- so manual maintenance is essential in this case, Note that LVs will retain their "distributed" status, even though no PVG information is imported for the VG by vgimport.

The default characteristic of a Logical Volume is "strict" allocation policy - this means that when the volume is mirrored with Mirrordisk/UX, no extent will have its mirror copy on the SAME PV (this would be irrational, assuming that the mirror copy is being used for data protection, as is usually the case).

If a Logical Volume is configured as "Group Strict" (by means of the "-s g" option of the lvcreate command), then mirrored extents must also be in separate PVGs. The typical approach is to group disks that are connected to the same I/O channel together into PVGs, so that mirrored extents will not be on the same interfaces - thus interface card failure will not completely disable access to a mirrored LV. Note that an existing LV can be made "Group Strict" using the lvchange command, but only BEFORE it is mirrored (even if it is mirrored and complies with the Group Strict requirement already). In contrast, it is possible to change a Logical Volume to make it Distributed, or to cease to be regarded as Distributed.

VGs which are built on disks in arrays, which are independently protected in Mode 1 or Mode 5, do not normally use Mirrordisk facilities, so they used not to be subdivided into PVGs. Mirrordisk is not needed to deal with the possibility of interface (I/O channel) failure with disk arrays, because "Alternate Links" (sometimes known as "PV links") can usually be set up, and these provide a means for LVM to switch from a primary link to a secondary one, to the same single underlying PV - this means that a PV is effectively included twice within the same VG, even though there is only one copy of the data visible to LVM (the disk array "hides" any additional copy or data protection).

However, when LVs are set up using the "-D" option of lvcreate, it is compulsory for them to be configured as "Group strict", in case Mirrordisk will be used as well. This means that it is also compulsory to have at least one PVG (and usually ONE is the correct number of PVGs to have in this case - having more than one may prevent an LV being striped in the manner desired) in the relevant VG. The mechanism by which lvcreate chooses which disks in which PVGs should have extents of a Logical Volume when it is created does not appear to be documented - however it is clear that should a mirror copy of the LV be created subsequently, then it will have its extents allocated on different PVGs, if this is possible; if it is not, then the mirror copy cannot be created.

When lvcreate is used to create an extent-striped LV, it should be assumed that it will place it in the first PVG in the VG with adequate space - it is not possible to specify a PVG when using lvcreate. However the lvextend command does allow a PVG to be specified, so if there is more than one PVG in a VG, it may be worth using lvcreate to create the LV with no space allocated, and then using lvextend to allocate the storage in the desired PVG.

At the time that an LV is created or extended, the PVG that it is in will determine which disks are used to spread its extents across, though there is some flexibility - for instance if one disk in the PVG is full, it will not be used, but as long as two disks have space, then extents can continue to be created.

A common, and effective, method used to distribute extents across PVs in a cached disk array, when there is no need for Mirrordisk replication, is as follows:

- Set up a single PVG for the entire VG. This means that when a striped LV is created, it will use all the PVs in the VG, as will normally be desired.

- The Alternate links may also be in the PVG, though this is not essential.

- Allocate the PVs to the VG and PVG in the sequence that access is desired - for instance, if two I/O channels are to be used, then alternate PVs across them (the Alternate links will of course "alternate in the opposite sequence").

- To get this right, either: Create the VG without specifying PVGs, but making sure to add all the alternate links, then create /etc/lvmpvg manually (perhaps using vgdisplay , then check with vgdisplay, then create LVs.
  Or, create the VG using the "-g pvgname" option on the vgcreate and vgextend commands, taking care to specify all PVs in the desired access sequence.

- Check the /etc/lvmpvg file carefully whenever any PVs are added to, or removed from, VGs. Amend manually if necessary.

Two other areas are worthy of discussion. These are the export/import of VGs, and the question of JFS "hot spots".

When a VG is exported and imported, it is best to avoid the "-s" option if alternate links are being used. This is because, whilst it simplifies the process, it will usually lead to a situation where the imported VG has all its "primary" links to LVs down a single I/O channel (the channel where they were first discovered by the "vgimport -s" process) - and all the alternate links down another. This can be rectified by careful subsequent use of vgreduce/vgextend commands (but NOT by use of pvchange which may only have a temporary effect). However, it is better to specify a complete list of disks, in the correct sequence to set up alternate links as desired, when the vgimport command is run. This has the effect of making the import process much faster too. Whilst it can be tedious to set up the command for import of a large VG in this way, it is helpful to note that a list of disks can be established on the source system when the vgexport is carried out (perhaps in preview mode) by using the "-f" option to capture a list of PVs in a file. With appropriate editing, which will often be necessary to resolve differing channel numbers, the target system can use same the list of disks with its vgimport command. It will be necessary to amend /etc/lvmpvg manually after importing a VG (the entries for an existing VG disappear when the VG is exported.

The JFS file system makes use of an "intent log" to improve performance and integrity. This is held in the first extent of the LV in which a file system resides, and will tend to be accessed more, on average, than the rest of the file system, if a substantial amount of changes occur. It may therefore be worthwhile in some cases (for instance if monitoring of I/O suggests an imbalance) to attempt to locate the first extent of each LV on a different disk, to the degree that this is possible. However, the only easy way to manage this process would be to amend the /etc/lvmpvg file, to change the sequence of the PVs, in between each lvcreate command.

Offline Diagnostic Environment (ODE)

You need the OfflineDiagnostic (ODE) to be able to do HW troubleshooting 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.

ODE is the LIF-LOAD product which is part of the OnlineDiag bundle:

```
# swlist OnlineDiag
# OnlineDiag                B.11.11.15.13  HPUX 11.11 Support Tools Bundle, Dec 2004
  OnlineDiag.Sup-Tool-Mgr      B.11.11.15.13  Support Tools Manager for HPUX
systems
```

```
 OnlineDiag.EMS-KRMonitor     A.11.11.05     EMS Kernel Resource Monitor
 OnlineDiag.EMS-Core         A.04.00.02     EMS Core Product
 OnlineDiag.EMS-Config       A.04.00.02     EMS Config
 OnlineDiag.Contrib-Tools    B.11.11.15.13  Contributed Tools
 OnlineDiag.LIF-LOAD         B.11.11.15.13  HP LIF LOAD Tools
```

The ODE LIF files can be found in the regular files below /usr/sbin/diag/lif/ directory, e.g.:

```
    # 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 using the lifload utility which in turn calls mkboot:

```
   # cd /usr/sbin/diag/lif
   # getconf HW_CPU_SUPP_BITS              (the result is either 32, 32/64 or 64)
   # ./lifload –f updatediaglif                                   (if 32 or 32/64)
   # ./lifload –f updatediaglif2                                          (if 64)
```

NOTE: “lifload –f updatediaglif” performs “mkboot -b updatediaglif -p ISL -p HPUX /dev/rdsk/cXtXdX”, where cXtXdX are all the disks listed in /stand/bootconf. The –p option of mkboot preserves the specified file so that it is not overwritten in LIF.

If you are setting up a mirrored root config you need have the mirror disk listed in /stand/bootconf or use mkboot directly to install the ODE files there.

LVM and 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
- /etc/lvmtab_p
- /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** **/etc/lvmtab_p**	**/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 more information.

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 (c#t#d#s1) contains the EFI (500MB). The former LVM disk is now located at partition 2 of the disk (c#t#d#s2). 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.

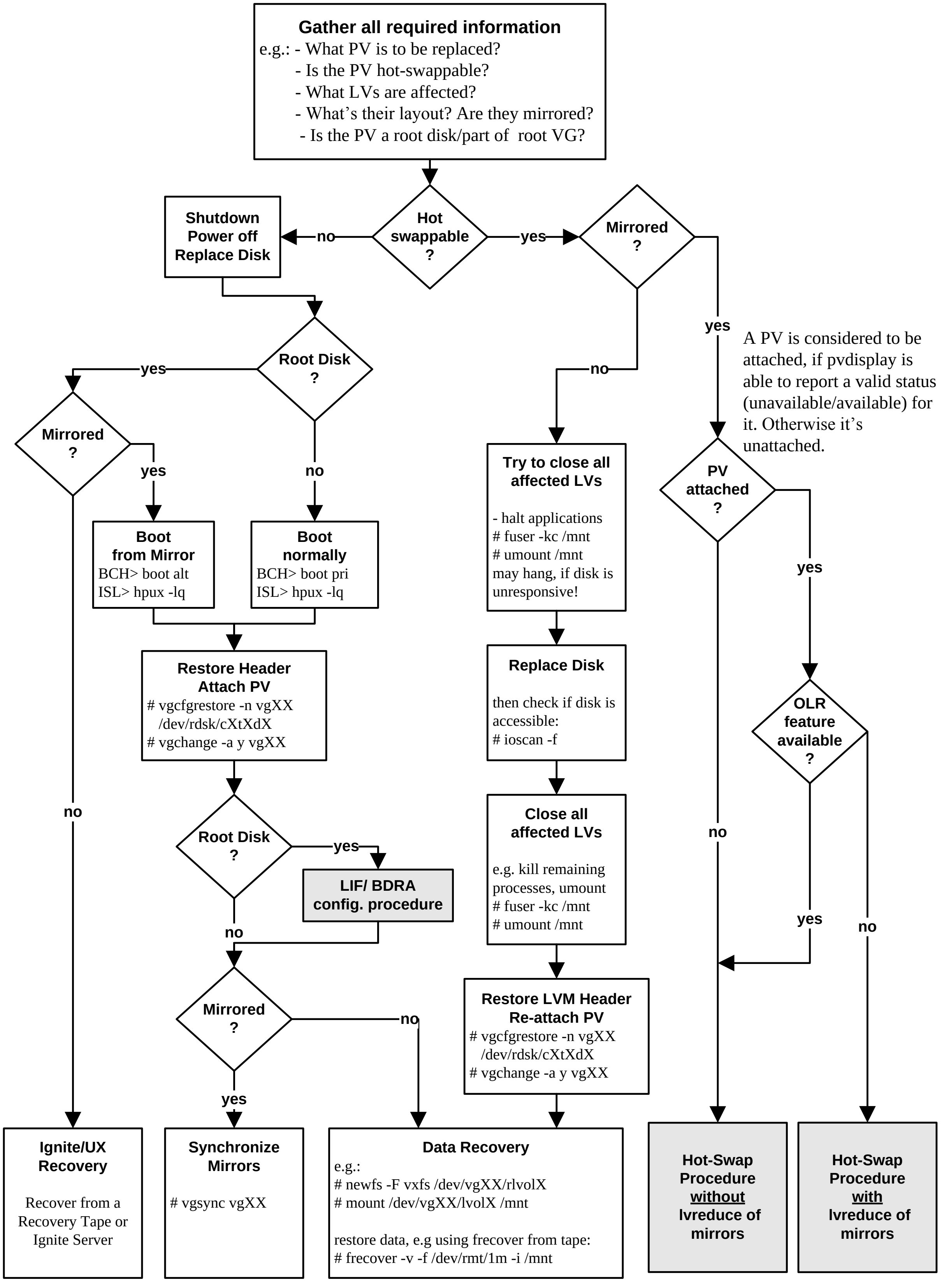
Here are some approaches how to check for typical symptoms of failed disks.

- 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/c#t#d# of=/dev/null bs=256K**) or partially **(dd if=/dev/rdsk/c#t#d# of=/dev/null bs=1024K count=10**) 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.



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 root VG?

Shutdown Power off Replace Disk

A PV is considered to be attached, if pvdisplay is able to report a valid status (unavailable/available) for it. Otherwise it's unattached.

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!

Replace Disk

then check if disk is accessible:

```
# ioscan -f
```

Restore Header Attach PV

```
# vgcfgrestore -n vgXX
   /dev/rdsk/cXtXdX
# vgchange -a y vgXX
```

Close all affected LVs

e.g. kill remaining processes, umount

```
# fuser -kc /mnt
# umount /mnt
```

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
```

Attached vs. Unattached

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*.
It is not allowed to replace an attached disk without having the mirrors reduced before. 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.

New LVM OLR (Online Replacement) feature

There is a new functionality that allows the online replacement of attached disks by adding the new –a option to the pvchange command:

```
# pvchange –a n /dev/dsk/c#t#d#             Detach this path of the PV
# pvchange –a N /dev/dsk/c#t#d#        Detach ALL paths of a PV
# pvchange –a y /dev/dsk/c#t#d#        Attach this path of the PV
```

This eases the replacement procedure for attached PVs since there is no need to reduce the mirror anymore.

You need the following patches (or later) to enable the feature:

UX 11.11   PHKL_31216 & PHCO_30698
UX 11.23   PHKL_32095 & PHCO_32622 (depends on Sep 04 base patch PHKL_31500)

Refer to the pvchange(1M) manpage regarding the new -a option.

Hot-Swap Procedure

Follow these steps to replace a mirrored hot-swap disk module

1. Unattached the PV

   If the PV is still attached and the OLR feature explained above is available then use pvchange to unattach it:

   ```
   # pvchange –a N /dev/dsk/c#t#d#        Detach all paths to the PV
   ```

2. (Optional) Only if PV cannot be unattached reduce the mirrors

   Reduce any LVs 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/c#t#d#). Then you should check for each found LV *how* it is mirrored (use lvdisplay –v /dev/vg##/lvol#). 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/vg##/lvol# <PV list>          for 1 way mirroring, or
   # lvreduce -m 1 -A n /dev/vg##/lvol# <PV list>          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 automatic vgcfgbackup operation, which is likely to get stuck on accessing a defective disk.

3. 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/c#t#d#
   ```

4. (Optional) For fibre channel disks perform the replace_dsk steps

   as described in the section “*How to Replace Disks at Hosts with TachLite HBAs*” in the Fibre Channel chapter.

5. Restore the LVM configuration onto the new disk

   ```
   # vgcfgrestore -n vg## /dev/rdsk/c#t#d#
   ```

   NOTE: Since we did lvreduce **-A n** in step 2 the lvmconf file we are about to restore is outdated. Fortunately, LVM is smart enough to detect this using timestamps. The restored config is "older"... though it's considered out-dated and gets automatically updated during re-activation.

6. Attach the new disk to the active VG

   ```
   # vgchange -a y vg##                    or
   # vgchange -a e vg##                    for exclusively activated Cluster VGs
   ```

   NOTE: This would attach ALL unattached PVs to the VG at once. If you intend to attach only the failed PV then use pvchange instead:

   ```
   # pvchange –a y /dev/rdsk/c#t#d#
   ```

7. (Optional) Configure LIF & BDRA

   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.

8. Sync the mirrors again

   If you did not reduced the mirrors in step 2 then simply sync the mirrors to the new 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 LV(s) are still accessible to users applications during this command.

   ```
   # vgsync vg## &
   ```

   To check the progress of the synchronization you could use:

   ```
   # lvdisplay -v $(find /dev/vg## -type b) | grep –c stale
   ```

9. (Optional) Recreate the mirrors

   If you reduced the mirrors in step 2 then recreate them.

```
# lvextend -m 1 /dev/vg##/lvol# /dev/dsk/c#t#d# &         for 1 way mirroring, or
# lvextend -m 2 /dev/vg##/lvol# /dev/dsk/c#t#d# &         for 2 way mirroring
```

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

```
# for i in lvol1 lvol2 lvol3 ...                    specify any LV you need to
mirror
> do
> lvextend –m 1 /dev/vg##/$i /dev/dsk/c#t#d#
> done
```

There are 2 new white papers available regarding LVM disk replacement:

LVM Online Disk Replacement (LVM OLR)
http://bizsupport.austin.hp.com/bc/docs/support/SupportManual/c01919406/c01919406.pdf

When Good Disks Go Bad: Dealing with Disk Failures Under LVM
http://bizsupport.austin.hp.com/bc/docs/support/SupportManual/c01911837/c01911837.pdf

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:

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
```

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: 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

```
# 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.

How To increase the root LV`s size using Ignite:

1. System Reboot

```
Processor is booting from first available device.
To discontinue, press any key within 10 seconds.
```

2. Pres any key. You will receive a similar message:

```
Boot terminated.
---- Main Menu -------------------------------------------------------------
Command Description
------- -----------
BOot [PRI|ALT|<path>] Boot from specified path
PAth [PRI|ALT] [<path>] Display or modify a path
SEArch [DIsplay|IPL] [<path>] Search for boot devices
COnfiguration menu Displays or sets boot values
INformation menu Displays hardware information
SERvice menu Displays service commands
DIsplay Redisplay the current menu
HElp [<menu>|<command>] Display help for menu or command
RESET Restart the system
----
```

```
Main Menu: Enter command or menu >
```

3. If you know the path to your tape device, enter the following:

```
boot 0/0/1/0.1
```

4. If you don’t know the path to your tape device, you have to search for all bootable paths (with the SEA command).

```
Searching for potential boot devices.
To discontinue search, press any key (termination may not be immediate).
Path# Device Path (dec) Device Path (mnem) Device Type
----- ---------------- ----------------- -----------
P0 0/0/0/0 lan.15.140.10.113 LAN Module
P1 0/0/1/0.6 extscsi.6 Random access media
P2 0/0/1/0.3 extscsi.3 Sequential access media
P3 0/0/1/0.1 extscsi.1 Random access media
P4 0/0/1/1.0 intscsib.0 Random access media
P5 0/0/2/0.0 intscsia.0 Random access media
```

5. Now you have the path and can boot with the command from step 3.

```
Die Frage Interact with IPL (Y, N, or Cancel)?> sollten Sie mit NEIN beantworten.
Booting...
Boot IO Dependent Code (IODC) revision 1
HARD Booted.
ISL Revision A.00.43 Apr 12, 2000
ISL booting hpux (;0):INSTALL
Boot
: disk(0/0/1/0.1.0.0.0.0.0;0):WINSTALL
8941568 + 1642496 + 2596176 start 0x1fef68
```

6. In order to interrupt the non-interactive installation, pres any key

```
WARNING: The configuration information calls for a non-interactive
Installation.
Press <Return> within 10 seconds to cancel batch-mode installation:
Really cancel non-interactive install and start the
User-interface? ([y]/n): y
```

7. After some minutes and some status screens you will see a similar a screen:

```
Welcome to the HP-UX installation/recovery process!
Use the <tab> key to navigate between fields, and the arrow keys
within fields. Use the <return/enter> key to select an item.
```

```
Use the <return> or <space-bar> to pop-up a choices list. If the
menus are not clear, select the "Help" item for more information.
Hardware Summary: System Model: 9000/800/L1000-36
+---------------------+----------------+-------------------+ [ Scan Again ]
| Disks: 17 (271.3GB) | Floppies: 0 | LAN cards: 4 |
| CD/DVDs: 1 | Tapes: 1 | Memory: 256Mb |
| Graphics Ports: 0 | IO Buses: 8 | CPUs: 1 | [ H/W Details ]
+---------------------+----------------+-------------------+
[ Install HP-UX ]
[ Run a Recovery Shell ]
[ Advanced Options ]
[ Reboot ] [ Help ]
```

8. Choose Install HP-UX

```
User Interface and Media Options
This screen lets you pick from options that will determine if an
Ignite-UX server is used, and your user interface preference.
Source Location Options:
[ * ] Media only installation
[ ] Media with Network enabled (allows use of SD depots)
[ ] Ignite-UX server based installation
User Interface Options:
[ ] Guided Installation (recommended for basic installs)
[ * ] Advanced Installation (recommended for disk and filesystem management)
[ ] No user interface - use all the defaults and go
Hint: If you need to make LVM size changes, or want to set the
final networking parameters during the install, you will
need to use the Advanced mode (or remote graphical interface).
[ OK ] [ Cancel ] [ Help ]
```

9. You obtain the following screen:

```
/-------\/----------\/--------\/-------------\/----------\
| Basic || Software || System || File System || Advanced |
| \-----------------------------------------------------------------\
| |
| Configurations: [ HP-UX B.11.11 Default ->] [ Description... ] |
| |
| Environments: [ HP-UX 11i Base OS-64bit ->] (HP-UX B.11.11) |
| |
| [ Root Disk... ] SEAGATE_ST39103LC, 0/0/2/0.0.0, 8683 MB |
| |
| File System: [ Logical Volume Manager (LVM) with VxFS ->] |
| |
| [ Root Swap (MB)... ] 512 Physical Memory (RAM) = 256 MB |
| |
| [ Languages... ] English [ Keyboards... ] [ Additional... ]|
| |
\-------------------------------------------------------------------------/
[ Show Summary... ] [ Reset Configuration ]
--------------------------------------------------------------------------
[ Go! ] [ Cancel ] [ Help ]
```

10. From the "File System" Tab choose "Additional Tasks" -> Logical Volume Paramters -> here you can change the root LV size.

11. Press GO

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 /new_stand/bootconf to c1t1d0:

```
# vi /new_stand/bootconf

l  /dev/dsk/c1t1d0
```

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

```
# setboot –p <HW path of c1t1d0> -b on
# 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            (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/LIF-LOAD files (may be skipped):

   ```
   # cd /usr/sbin/diag/lif
   # getconf HW_CPU_SUPP_BITS        (the result is either 32, 32/64 or 64)
   # ./lifload –f updatediaglif                             (if 32 or 32/64)
   # ./lifload –f updatediaglif2                                    (if 64)
   ```

NOTE: "lifload –f updatediaglif" performs "mkboot -b updatediaglif -p ISL -p HPUX /dev/rdsk/cXtXdX", where cXtXdX are all the disks listed in /stand/bootconf.
The –p option of mkboot preserves 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)
```

What is the new in 11.31 regarding LVM

Native Multipathing in LVM

In HP-UX 11i v3 LVM supports both legacy and persistent DSFs(persistent DSFs are a new type of DSFs. More about them, can be found in the I/O Chapter of this book, under section "New Type of DSFs").
A volume group can be configured, using both types of DSFs. Some volume groups can use legacy device special files, some can use persistent, and some can use a mixture of both types. Suporting a mixed mode configuration allows a volume group that currently uses one DSF naming model to be extended (by vgextend) with the same or another physical volume using a different DSF naming model. Such operations result in a mixed mode volume group and facilitate a phased DSF migration from legacy to agile naming model. The /etc/lvmtab looks in the following way when having a mixed mode supported:

Example:

```
strings /etc/lvmtab
/dev/vg00
/dev/disk/disk6_p2
/dev/disk/disk5
```

```
/dev/disk/disk4
/dev/vg02
/dev/dsk/c3t2d0s1
```

Note: If You have mixed mode activated (if legacy naming model is not deactivated), then you are able to create volume gropus, or physical volumes, or logical volumes using both naming models.

For operating with physical volumes, both types of DSFs can be used. If for example the physical volume was created using the legacy DSF, then You can easily find the persistent equivalent by the command “ioscan –m dsf” and then use the persistent DSF to remove the physical volume:

Example:

```
# ioscan -m dsf
Persistent DSF          Legacy DSF(s)
========================================
/dev/rdisk/disk4        /dev/rdsk/c2t0d0
/dev/rdisk/disk5        /dev/rdsk/c2t1d0
/dev/rdisk/disk6        /dev/rdsk/c3t2d0
/dev/rdisk/disk6_p1       /dev/rdsk/c3t2d0s1
/dev/rdisk/disk6_p2       /dev/rdsk/c3t2d0s2
/dev/rdisk/disk6_p3       /dev/rdsk/c3t2d0s3
/dev/rdisk/disk7        /dev/rdsk/c0t0d0
```

```
# vgremove /dev/vg02
Volume group "/dev/vg02" has been successfully removed.
```

```
# pvremove /dev/rdisk/disk6_p1
The physical volume associated with "/dev/rdisk/disk6_p1" has been removed.
```

Nevertheless, HP recommends using persistent DSFs for LVM configurations, and especially for new volume groups. Existing volume groups configured with legacy DSFs should be migrated to use corresponding persistent DSFs.

New options have been added to vgimport (option –N) and vgscan (options –N –B), to specify the persistent naming model.

- vgimport –N - Configure the volume group using persistent DSFs. In the absence of the –N option, legacy DSFs are used. This option can only be ised together with the scan option –s.
- vgscan –N – Recover /etc/lvmtab file using persistent DSFs, with the following exception: Active volume groups configured with legacy DSFs. In this case, vgscan populates the /etc/lvmtab file using the legacy DSFs.

- vgscan –B – Recover /etc/lvmtab file using both persistent and legacy DSFs. This option can be used to migrate a volume group configured with legacy DSFs to use corresponding persistent DSFs.

Example:

```
#strings /etc/lvmtab
/dev/vg00
/dev/disk/disk6_p2
/dev/disk/disk5
/dev/disk/disk4
/dev/dsk/c2t0d0
/dev/dsk/c2t1d0
/dev/dsk/c3t2d0s2
# strings /etc/lvmtab.old
/dev/vg00
/dev/disk/disk6_p2
/dev/disk/disk5
/dev/disk/disk4
```

Note: The example above shows lvmtab and lvmtab.old, after renaming the lvmtab in lvmtab.old, and recreating the lvmtab with the option of vgscan "-B". The new lvmtab includes both types of DSFs, legacy and persistent.

Alternate Links in HP-UX 11.31

LVM supports alternate links to a device to allow continued access to the device if the primary link fails. In previous HP-UX releases LVM's multipathing solution increases the data availability but does not allow multiple paths to be used simultaneously. LVM still supports this functionality in HP-UX 11i v3, but the behaviour could be different, based on how LVM and the mass storage stacks are configured.
With the introduction of mass storage stack's native multipathing functionality, it is no longer required nor recommended to configure LVM alternate links.

Listed below are the alternate link behavioral differencies that could be seen, when compared with previous releases:

- Regardless of whether LVM alternate links are configured or not, the massstorage stack processes the I/O operations to the device, using all available paths. That is, LVM selects the LUN and the massstorage stack select the lunpath to be used.
- If LVM alternate links are configured, no link switching ever happens unless the mass storage stack's native multipathing feature is disabled using the scsimgr command and only legacy DSFs are used in the volume group configuration.

- Path switch options of the pvchange (-s and –S options) command do not switch links or stop I/O operations as they did in earlier releases. In order to disable I/O operations on a given path, use the scsimgr command (scsimgr [-f] disable Lunpath).
- While LVM alternate link functionality supports up to 8 paths to a physical volume, mass storage stack's native multipathing feature can support up to 32 lunpaths.

If legacy LVM alternate link functionality is preferred over the mass storage stack's native multi pathing functionality, the native multipathing feature can be disabled using the scsimgr command.

The following example shows how to enable legacy LVM alternate link functionality in a backward compatible manner. In this example, the vg01 is configured with a single persistent DSF:

Example:

1. Get a list of DSFs configured in the volume group:

```
#vgdisplay –v –F vg01 | grep pv_name
```

2. Find legacy DSFs corresponding to the persistent DSF:

```
# ioscan –m dsf /dev/disk/disk5
Persistent DSF           Legacy DSF(s)
========================================================
/dev/disk/disk5        /dev/dsk/c2t1d0
                         /dev/dsk/c3t1d0
```

3. Add legacy DSFs as alternate links to the persistent DSF:

```
# vgextend vg01 /dev/dsk/c2t1d0 /dev/dsk/c3t1d0
```

4. Remove the persistent DSF from the volume group:

```
# vgreduce vg01 /dev/disk/disk5
```

5. Disable native multipathing through legacy DSFs:

scsimgr command can be run in any of the two ways listed below:

- Disable native multipathing for a specific LUN in a non-persistent way:

```
#scsimgr set_attr –D /dev/rdisk/disk5 –a leg_mpath_enable=false
    Value of attribute leg_mpath_enable set successfully
```

- Disable native multipathing globally for all LUNs in the system, in a non-persistent way:

```
#scsimgr set_attr –a leg_mpath_enable=false
    Value of attribute leg_mpath_enable set successfully
```

6. Verify the resaulting configuration:

The scsimgr get_attr command should show the current setting of leg_mpath_enable attribute value set to false, for all assosiated persistent DSFs in the volume group.:

```
scsimgr get_attr -D /dev/rdisk/disk5 -a leg_mpath_enable

        SCSI ATTRIBUTES FOR LUN : /dev/rdisk/disk5

name = leg_mpath_enable
current = false
default = false
saved =
```

LVM Migration

It is recommended that in HP-UX 11i v3 persistent DSFs are used over legacy DSFs. There are different alternatives to migrate lvm configurations from legacy to persistent DSFs. All of them update the /etc/lvmtab automatically, and no explicit user action is required. However some alternatives require, that for Physical Volume Groups /etc/lvmpvg is manually edited.

Note: Each alternative updates one volume group at a time. The steps must be repeated in the alternative chosen for all volume groups to be migrated.

1. Backup the /etc/lvmpvg file, if it exists. Deactivate and export the volume group. Re-import the same volume group using persistent DSFs in place of legacy DSFs. Persistent DSFs, corresponding to legacy DSFs, can be found using ioscan –m dsf <legacy_dsf> command. Restore the backed-up /etc/lvmpvg file and replace legacy DSFs listed under the volume group being migrated with corresponding persistent DSFs. (in case of mulitipath devices, a single persistent DSF can correspond to multiple legacy DSFs). The disadvantage of this method is the necessity of volume group deactivation.
2. Run vgscan-B command followed by volume group activation.

   For each of the physical volumes configured in the volume group, vgscan –B command populates the /etc/lvmtab file with both persistent and legacy DSFs. However, /etc/lvmtab file supports a maximum of 8 paths per physical volume. Therefor, if a physical volume has more than 8 paths configured, only 7 are retained in order to allow space for addition of persistent DSF.

   Note: The vgscan command will not update existing volume group entries in

/etc/lvmtab file unless the –f option is used. The –f option restricts the operation to the specified volume group (overwrite existing volume group entries in /etc/lvmtab). If the /etc/lvmtab is moved to a new location before running the vgscan –B comand, full effect of the command is taken on all configured volume groups and /etc/lvmtab file gets recreated.
Following the invocation of vgscan –B command, use vgchange to reactivate the corresponding volume group. This reconfigures the volume group with the DSFs(both legacy and persistent DSFs corresponding to each of the physical volume) in the /etc/lvmtab file. Legacy DSFs can later be removed using vgreduce, leaving behind corresponding persistent DSFs. This facilitates a phased migration from legacy to agile naming model, while the volume group continues to remain active.

Backup the /etc/lvmpvg file, if it exists. After the vgreduce operations, restore the backed-up /etc/lvmpvg file and replace legacy DSFs listed under the volume group being migrated with corresponding persistent DSFs.

3. There is a shell script in /usr/contrib/bin/vgdsf, that can perform the migration while the volume group remains active. The vgdsf performs following tasks:
   - Extend the volume group with corresponding persistent DSFs:
     1. Identify the legacy DSFs configured in the volume group, using vgdisplay –v. Ignore alternate links.
     2. For each legacy DSF found, find its corresponding persistent DSF, using ioscan –m command.
     3. If a physical volume is configured with 8 paths, remove one using vgreduce. This will make room for additional persistent DSF.
     4. Extend the volume group to add the persistent DSF, keeping it in some physical volume group configuration, if any (/etc/lvmpvg file gets automatically updated).

   - Reduce the volume group of all legacy DSFs :
     1. Identify all legacy DSFs configured in the volume group, using vgdisplay –v
     2. For each of the legacy DSF that also has a corresponding persistent DSF configured, reduce the legacy DSF from the volume group. If the corresponding persistent DSF is not configured in the volume group, a message is displayed on screen and the command continues with other DSFs.

   - Back up the resault configuration, using the vgcfgbackup command.
   - Activate all confugred volume groups in the system and identify the ones requiring a migration, using vgchange –a.

- For reference, make a note of the volume group configuration. Save the outputs from LVM commands: vgdisplay, lvdisplay, pvdisplay in verbose mode.
- Ensure all physical volumes configured in the volume group are online using ioscan –P health.
- Run /usr/contrib.bin.vgdsf –c <vg_name> to migrate each volume group. Ensure that no failures are reported.
- Verify the resaulting volume group configuration using LVM display commands. Use ioscan –m dsf to validate DSF mappings.

Below the vgdsf migration script:

```
#!/usr/bin/ksh
  DEV=$1
  LAST=""
  if [[ -s /etc/lvmpvg ]]; then
    if /usr/bin/grep -q "$DEV" /etc/lvmpvg; then
      TOKS=`/usr/bin/grep -e PVG -e "$DEV" /etc/lvmpvg|sed "s/PVG *//"| \
           /usr/bin/awk '{ print $1 }'`
      for TOK in $TOKS
      do
        if [[ $TOK = $DEV ]]; then
          if [[ -n $LAST ]]; then
            PVG="-g $LAST"
          fi
          return
        fi
        LAST=$TOK
      done
    fi
  fi
}
```

Common Issues

There are different common issues, that appear during execution of LVM commands. Some of them are fixed with patches, Lab requests or in later versions of HP-UX. Nevertheless a description of some of these is not useless in problem situations:

vgdisplay

Among the most common error messages is the following:
vgdisplay "Could not query physical volume"
Causes for this error message could be different. Here are some of them, as well as suggested workarounds:

1. PROBLEM - vgdisplay error: can't query physical volume

RESOLUTION

vgdisplay error: can't query physical volume

The error message will include the device in question.
Use the information with the output of the ioscan command.

```
/sbin/ioscan -fnC disk
```

Is the device in the output of the command, reporting both the hardware path and device file?
If not, there may be a possible hardware issue.

If the output of the ioscan reports the disk correctly, compare to the output of the strings command to check /etc/lvmtab.

```
strings /etc/lvmtab
```

Does this reflect the device in the correct volume group ?
If not, check the device with the pvdisplay command. It will display the status and characteristics of the physical volume.

```
pvdisplay /dev/dsk/c#t#d#
 --- Physical volumes ---
PV Name                   /dev/dsk/c#t#d#
VG Name                   /dev/vg##
```

If the disk reports incorrectly, restore the information to the disk with the vgcfgrestore command. Check the file /etc/lvmconf/vg##.conf for accuracy, before restoring.

```
strings /etc/lvmconf/vg##.conf
```

If this reports accurately, restore the LVM configuration information to the physical volume.

By default the volume group will need to be de-activated.
(see man page vgcfgrestore 1(m) for additional options)

```
vgchange /dev/vg## -a n
vgcfgrestore -n /dev/vg00 /dev/rdsk/c#t#d#
strings /etc/lvmtab
```

If /etc/lvmtab does not reflect the change, use the vgscan command. This allows the re-creation of the /etc/lvmtab and possibly the associated volume group device files.

```
-p   Preview the actions that would be taken but do not update
     file /etc/lvmtab.  This option is best used in conjunction
     with the -v option.

-v   Print verbose messages
```

To protect /etc/lvmtab , move the file.
If the vgscan command fails it can be moved back.

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

This will allow the preview of the results from vgscan, without making the changes. If the information is correct continue without the -p option.

```
vgscan -v
strings /etc/lvmtab
vgchange /dev/vg## -a y
vgdisplay -v /dev/vg##
```

2. PROBLEM – vgdisplay shows error after disk replacement

After replacing the system mirror root disk, the vgcfgrestore command worked successfully, but now the vgdisplay command displays this error message:

```
Warning: Could not query physical volume
```

What is causing this problem?

RESOLUTION

The problem is that vgdisplay or pvdisplay reads the information that is loaded into memory when the volume group gets activated.

Follow these steps:

1.1. Restore the LVM config/headers onto the replaced disk with vgfgrestore.

1.2. Reactivate the volume group.

1.3. Notify the volume group about the the replaced drive by running vgchange:

```
vgchange -a y /dev/vg00
```

Note: This step will cause LVM to become aware that the disk is available again, and will do a resync.

1.4. Synchronize the PEs that are most likely marked as stale:

```
vgsync /dev/vg00
```

Note: This process might take a while depending on the vg size and data size.

3. PROBLEM

When running the command vgdisplay on a volume group(vgdisplay /dev/vg01) the following error occurs:

```
vgdisplay: Warning: couldn't query physical volume "/dev/dsk/c0t0d0":
The specified path does not correspond to physical volume attached to
this volume group
vgdisplay: Warning: couldn't query all of the physical volumes.
```

RESOLUTION

Simply try to activate the volume group with the vgchange command:

```
vgchange –a y /dev/vgXX
```

It is possible, that the system was rebooted or the volume group was activated with a disk or path missing. The disk or path is later restored, but the volume group is unaware of the change, until it is re-activated.

4. PROBLEM – Hardware issue

Please also bear in mind, that this error, could be due to a hardware problem as well. It is recommended, that hardware on the system is also checked, to exclude hardware failures, before further troubleshooting is performed.

Hardware problems on a disk, can be checked with the **dd** command (for exact options, check the dd manpage). Please note, that the **dd** command, alhough completed with success, is not a guarantee for no hardware issues. It only reads from the disk, and does not write, so further examinations should be made, in order to make sure, that hardware problems are not present.

5. PROBLEM – the "couldn't query" message Cur and Act PV number disagree

Typically vgreduce -f is used when the volume group's vgdisplay(1m) output shows Cur PV and Act PV disagree. Often vgdisplay(1m) will give a message like: couldn't query physical volume /dev/dsk/cXtXdX when Cur and Act PV number disagree. The "couldn't query" message usually indicates one of these scenarios:

1. The disk device(s) specified is not responding due to a hardware error or incorrect device file. Here are some commands that may help isolate if there is a disk device problem:

   - ioscan -fn
     Hardware path should show as claimed and with the correct device file.

   - insf -H HW_PATH_FROM_IOSCAN -e
     Recreate device file.

   - diskinfo /dev/rdsk/cXtXd0
     Verify product number and size.

   - dd if=/dev/dsk/cXtXdX of=/dev/null bs=10k count=30
     Verify the disk can read from without error and the blocks in and out match. You should get 30+0 records in 30+0 records out.

2. The disk is responding but does not contain correct lvm information to attach it to the volume group. Every LVM physical volume contains a volume group reserve area (VGRA) on the front of that disk device. The VGRA contains information, shared by all disks attached to the same volume group, to know how disk devices belonging to that volume group are partitioned. If this area becomes corrupt or overwritten it will result in the "couldn't query physical volume message" from vgdisplay(1m).

vgcfgrestore(1m) can be used to restore the VGRA.

Example: vgcfgrestore -n /dev/vg01 /dev/rdsk/c1t5d2

3. The disk device's alternate path was added to another volume group. The method used to verify alternate paths will be unique to the type of disk device being used. The operating system will usually not allow an alternate path to be added to a different volume group than the primary path unless pvcreate -f is first run.

NOTE: While Cur PV and Act PV do not agree the volume group should not be modified. Since vgcfgbackup(1m) will fail when Cur PV and Act PV do not agree, modifing the volume group can make it so the disk data structures (VGRA) no longer match the information in the last vgcfgbackup(1m) file residing in the /etc/lvmconf directory. Modifing the volume group when Cur and Act PV disagree can make recovery more difficult!

RESOLUTION

vgreduce -f should be used as a last resort. If possible use vgcfgrestore(1m) to restore the lvm information. See step 2 above. If vgcfgrestore(1m) cannot be used to make the Cur PV and Act PV agree again, then vgreduce -f may be required. Here are the steps to successfully use vgreduce -f:

1. Get a list of logical volumes belonging to the volume group.

   Use: vgdisplay -v /dev/vg_name to get a list of logical volumes for the volume group.

2. Find out which logical volume reside on the disk device(s) to be forcibly reduced.

   Use: lvdisplay -v /dev/vg_name/lv_name | more to see if any of the logical volumes extents show ??? in the PV section. Page through every logical extent for each logical volume in the volume group. ??? indicate that the extents shown reside on a physical volume that the system is unable to query. Any logical volume with ??? will have to be removed using lvremove(1m) in order for vgreduce -f to complete successfully.

3. Remove logical volumes with ??? in their lvdisplay(1m) output.

   Since logical volumes that show ??? have missing or unavailable data they will have to removed. In order for vgreduce -f to succeed all logical volumes with extents on the physical volume to be reduced must first be removed. Once the volume group is in the correct state, Cur PV = Act PV, the logical volumes can be recreated and any lost data restored.

Use: lvremove /dev/vg_name/lvol_name

4. Forcibly reduce out the physical volume.

   Use: vgreduce -f /dev/vg_name

   NOTE: The above command does not require a physical volume argument. It must be run on a active volume group.

5. If the vgreduce -f command does not work or does not give any error and vgdisplay(1m) still shows that Cur PV and Act PV disagree then use the following steps to vgexport and vgimport the volume group prior to trying Step 4 again.

   This procedure can be used when vgreduce fails to reduce a physical volume that can no longer be queried by the system. If executing the following procedure on the root volume group, usually vg00, you must first boot into LVM maintenance mode (** For steps see below).

   a. Get the /dev/vg_name/group minor number and physical volumes belonging to the volume group.

      Use: ll /dev/vg00/group to get 0x###### minor number.
      vgdisplay -v /dev/vg_name to get physical volumes.

   b. vgchange -a n /dev/vg_name
      Skip this step if booting maintanence mode for root volume group.

   c. vgexport -m /mapfile /dev/vg_name

   d. mkdir /dev/vg_name

   e. mknod /dev/vg_name/group c 64 0x0#0000
      Re-use minor number obtained from step a.

   f. vgimport -m /mapfile /dev/vg_name pv_name [pv_name ...]

      NOTE: Specify all the physical volumes obtained from step a. Do not include the physical volume that you are trying to remove or that couldn't be queried.

** Steps to boot into maintenance mode and active. :
1. shutdown -hy now
2. interrupt boot sequence
3. boot from primary boot path and interact with ISL

NOTE: Procedure used for steps 2 and 3 may very slightly depending on machine model.

4. enter the following at the IPL> prompt:

```
IPL> hpux -lm (;0)/stand/vmunix
        or

IPL> hpux -lm
```

6. Retry the vgreduce -f command specified in step 4.

This time the vgreduce should succeed and give you a message similar to: "PV with key # sucessfully deleted from vg /dev/vg_name". It should also display:

Repair done, please do the following steps:
1. save /etc/lvmtab to another file
2. remove /etc/lvmtab
3. use vgscan(1m) -v to recreate /etc/lvmtab
4. NOW use vgcfgbackup(1m) to save the LVM setup

Follow the above steps.

vgdisplay /dev/vg_name should now show Cur PV and Act PV agree.

vgscan

1. PROBLEM – vgscan returns error:

```
# vgscan -v
Creating "/etc/lvmtab".
vgscan: Couldn't access the list of physical volumes for volume
group "/dev/vgXX".
```

```
# vgscan -v
Creating "/etc/lvmtab".
vgscan: Couldn't access the list of physical volumes for volume
group "/dev/vg00".
vgscan: Couldn't access the list of physical volumes for volume
group "/dev/vg01".
vgscan: Couldn't access the list of physical volumes for volume
group "/dev/vg02".

vgscan: Physical volume "/dev/dsk/c14t5d0" is not a block special file.
vgscan: Physical volume "/dev/dsk/c14t6d0" is not a block special file.
vgscan: Physical volume "/dev/dsk/c15t5d0" is not a block special file.
vgscan: Physical volume "/dev/dsk/c15t6d0" is not a block special file.

Couldn't stat physical volume "/dev/dsk/c14t5d0":
Invalid argument
Couldn't stat physical volume "/dev/dsk/c14t6d0":
Invalid argument
Couldn't stat physical volume "/dev/dsk/c15t5d0":
Invalid argument
Couldn't stat physical volume "/dev/dsk/c15t6d0":
Invalid argument

/dev/vg00
/dev/dsk/c1t2d0
/dev/dsk/c2t2d0

The Volume Group /dev/vg01/group was not matched with any Physical Volumes.
The Volume Group /dev/vg02/group was not matched with any Physical Volumes.
Scan of Physical Volumes Complete.
```

```
# strings /etc/lvmtab
/dev/vg00
/dev/dsk/c1t2d0
/dev/dsk/c2t2d0
```

--> vg01 and vg02 are missing because vgscan had errors!

The devicefiles that vgscan complains about are not existing on the system:

```
# ll /dev/dsk/c14*
/dev/dsk/c14* not found
```

```
    # ll /dev/dsk/c15*
    /dev/dsk/c15* not found
```

RESOLUTION

The reason for that is that the instance numbers have changed.
They have changed because the system was booted from the boot disk of another system which was supposed to be identical in HW (this is a common "HA" solution bypassing the need of professional cluster software like MC/ServiceGuard, usualy used with boot disk on fibre channel devices).
As a result the devicefiles for the disks changed.
The system is still accessing the PVs through their old devicefiles,
VGs could be activated, FS coud be mounted, applications could run.
But HW scanning commands like vgscsan fail under such circumstances.
The solution is to use insf(1M) to rename the devicefiles according the given the ext_bus instance numbers and to recreate lvmtab using vgscan to include the new devicefiles.

```
# insf -e
...

# vgscan -v
...


# strings /etc/lvmtab
/dev/vg00
/dev/dsk/c1t2d0
/dev/dsk/c2t2d0
/dev/vg02
/dev/dsk/c14t5d0  <<< new
/dev/dsk/c15t5d0  <<< new
/dev/vg01
/dev/dsk/c14t6d0  <<< new
/dev/dsk/c15t6d0  <<< new
```

```
what changed? c4t*d* --> c14t*d*  and
              c6t*d* --> c15t*d*
```

If there is any application that has the names of disk devicefiles hard coded in it's configuration you need to change it accordingly.

2.PROBLEM - vgscan: unable to match physical volume to a volume group

The vgimport of a volume group fails. vgscan -p -v -a gives the following error:

```
   The following Physical Volumes belong to one Volume Group.
   Unable to match these Physical Volumes to a Volume Group.
   Use the vgimport command to complete the process.
```

RESOLUTION
1. Find the disks that appear to belong to a volume group in the lvm configuration but cannot be matched with one:

```
    # vgscan -p -v -a
```

(There is no need to move the /etc/lvmtab because vgscan
will run in the preview mode and not actaully build the /etc/lvmtab)

In the output of the vgscan you will see information that looks something like this:

```
    The following Physical Volumes belong to one Volume Group.
    Unable to match these Physical Volumes to a Volume Group.
    Use the vgimport command to complete the process.
```

A list of the device files for each of the disks's will be listed beneath the message.

These are most likely the disks that you are working with, but this does not mean that all the disks belong to the same volume group.

To match the group of disks to a single volume group do the following:
eg..

FYI: 0x2008?4 is always the offset of the lvm data structure.

```
# echo 0x2008?4D|adb -o  /dev/disk/disk3_p2

0x2008:     418374281     915654113     418374281     909512229
```

The volume group name is not important at this point, but it is important to find the disks that have the same vgid or volume group id.
Notice the first field to the left: "2008" this represents the address
on the disk for the lvm data. It will always be the same.
next field: "418374281" this is the system or cpu id. You should see the same number returned from uname -i.

next field: "915654113" is the pvid, not important at this time.
next field: "418374281" is the cpu id again
Now the last field is the one we want to pay attention to: "909512229"
This is the vgid field and represents the unique volume group identifier.

This will need to be done for each disk that is not matched to a volume group. Once you have matched the vgid's of the disks then use the vgimport command to import the disks into a volume group.

for example..

```
    # mkdir /dev/vgtest
    # mknod /dev/vgtest/group c 64 0x140000
```

NOTE: The minor number used must be a minor number that is not already being used. It it is a required not to use a number higher than the value of the maxvg kernel parameter. If there are gaps in the existing minor numbers one can be selected to fill in the gaps. If 0x00, 0x01 0x03, 0x04 are in use then 0x02 would be a good choice for the next group file, but the numbers do not have to be used in any specific order.

To determine the value of maxvg:

```
    # sysdef | grep maxvg
```

Tip: to convert hex to decimal and decimal to hex you can use adb.

To convert decimal 127 to hex:

```
    #  echo '0d127=X' | adb -o
                7F
```

To convert hex 7F to decimal:

```
    #  echo '0x7f=D' | adb -o
                127
```

To find the existing minor numbers of existing volume groups do the following:

```
    # ll /dev/*/group
```

Then next step is to vgimport the volume groups:

```
    # vgimport /dev/vgtest /dev/dsk/cxtxdx /dev/dsk/cxtxdx(for each disk in
```

vg)

Once the volume group has been imported, the next step is to determine the previous name of the volume group by mounting the filesystems, and viewing their contents.

Once the actual name of the volume group has been determined, it can be renamed back to it's original name.

To rename the volume group do the following:

```
    # cd /dev
    # mv vgtest
    # mv /etc/lvmtab /etc/lvmtab.orig
    # vgscan -v
    # strings /etc/lvmtab
```

Now corrected volume group name should appear in the lvmtab file.

Note: After the vgimport and standard logical volume names wil need to be renamed from the standard lvol1,lvol2, etc.. --DO NOT VGCFGRESTORE AT THIS POINT--

To rename the lvols back to the original names do the following:

```
    #cd /dev/vgxx
```

For each lvol there will be a block and character device file (lvolx and rlvolx).

Both the character device file and the block device file will need to be renamed to their original names.

```
    # mv /dev/vgxx/lvol1 /dev/vgxx/
    # mv /dev/vgxx/rlvol1 /dev/vgxx/r
```

simply add a r to the name of the lvol name for the character device.

After renaming the device files there is no need to recreate the lvmtab, because lvm will read the volume group directory each time it accesses the logical volume.

Finally make sure that the lvm structures are backed up:

```
  # vgcfgbackup /dev/vgxx
```

vgremove

1.PROBLEM – VGREMOVE: COULDN’T REMOVE VOLUME GROUP /dev/vgXX

When vgremove is used on a volume group that conatins an alternate link without first doing a vgreduce, the volume group is left in an unusable state. The vgremove gives an error:

```
vgremove: Couldn't remove the entry "/dev/vg*" from "/etc/lvmtab".
vgremove: Couldn't remove volume group "/dev/vg*"
```

RESOLUTION

When vgremove is used on a volume group that conatins an alternate link without first doing a vgreduce, the volume group is left in an unusable state. The vgremove gives an error:

```
vgremove: Couldn't remove the entry "/dev/vg*" from "/etc/lvmtab".
vgremove: Couldn't remove volume group "/dev/vg*"
```

It appears the vgremove didn't remove the volume group.

vgdisplay on the volume group returns the following:

```
  /usr/sbin/vgdisplay [-v] [vg_name ...]
```

```
vgdisplay: Volume group not activated.
vgdisplay: Cannot display volume group "/dev/vg*"
```

vgchange on the volume group returns the following:

```
  /usr/sbin/vgchange -a y /dev/vg*
```

```
vgchange: Warning: Couldn't attach to the volume group physical volume
"/dev/dsk/c*":Cross-device link
vgchange: Warning: couldn't query physical volume "/dev/dsk/c*":
The specified path does not correspond to physical volume attached to
this volume group.
vgchange: Warning: couldn't query all of the physical volumes.
vgchange: Couldn't activate volume group "/dev/vg*":
Quorum not present, or some physical volume(s) are missing.
```

Recreate the lvmtab with the vgscan command.

The vgscan command allows the re-creation of the /etc/lvmtab file and possibly the associated volume group device files.

```
mv /etc/lvmtab /etc/lvmtab.old
```

```
/usr/sbin/vgscan -p -v
```

-p Preview the actions that would be taken but do not update file /etc/lvmtab. This option is best used in conjunction

-v Print verbose messages

Since vgscan searches each disk on the system in the order of where they are configured, when vgscan reconstructs /etc/lvmtab file, the order of disks in the file could be different than it was before. The following will happen:

The designated primary and alternate link might not be the same as it was configured before.

Alternate links will be added to the /etc/lvmtab file even if they might not be configured in the volume group initially.

The boot information might be incorrect due to different order of disks in the new /etc/lvmtab file.

In order to correct the above problems, do the following:

Use vgchange with -a option to activate all volume groups.

Use lvlnboot with -R option to correct boot information on disk.

Use vgreduce to reduce any alternate links that were added to the /etc/lvmtab file by vgscan, but they were not needed.

If the original primary path of a disks become an alternate path after /etc/lvmtab file is reconstructed, the order can be easily reverted by using vgreduce to remove the primary path and use vgextend to add the path back again.

If /etc/lvmtab is destroyed, do not use vgscan to re-construct /etc/lvmtab if the system is heavily loaded by an application. Otherwise, vgscan will create an incomplete /etc/lvmtab due to a known

NIKE/LVM limitation issue. It's important to quiesce the logical volume's I/O before re-constructing the /etc/lvmtab.

If for some reason, there is a need to re-construct /etc/lvmtab when the system is running production application, vgscan will create a partial /etc/lvmtab. In this case, most of the primary paths should be included in the /etc/lvmtab. Use vgextend to include any missing alternate paths in the VG.

If the preview mode has no errors, continue with

```
/usr/sbin/vgscan
```

Check /etc/lvmtab with

```
strings /etc/lvmtab
```

vgchange

1. PROBLEM – vgchange returns “cross device link”

After replacing a bad disk in a mirrored pair in a VG (vg02) that belonged to Serviceguard pkg, activation of vg02 fails even after vgcfgrestore to both pri and mirror disks.

vgchange -a y /dev/vg02 gives error "cross device link"

RESOLUTION

First one must test activation on the other node or nodes that share this VG in Serviceguard. Success of activation on those nodes means the problem is not with the disks and their LVM headers but with local OS information on the node where the problem exists, either lvmtab, /dev/vgname, or corruption with LVM in memory.

Second, one needs to try and go through all the standard methods to fix that error "cross device link". Here are the possible solutions:

If that does not work, then we can try using vgscan. So one can try to move /etc/lvmtab to /etc/lvmtab.keep and run vgscan.

```
# mv /etc/lvmtab /etc/lvmtab.keep
# vgscan -va
# vgchange -a y /dev/vgname
```

If that fails, one can try to reimport using map file from any other node where activation works. On one of those nodes, one can get a good copy of such map file for that VG using:

```
# vgexport -pvs -m mapfile /dev/vgname
```

Then one must rcp or ftp that mapfile to the node that is having the problem. There one can try to re-import that VG to see if problem is resolved:

```
# cp /etc/lvmtab.keep /etc/lvmtab   copies back original lvmtab
# vgexport -v /dev/vgname
# mkdir /dev/vgname
# mknod /dev/vgname/group c 64 0xNM0000
# vgimport -vs -m mapfile /dev/vgname
# vgchange -a y /dev/vgname
```

Here the minor number 0xNM0000 where NM=unique two digit hex integer, such as 01, or 02 or 5C (limited in value to the kernel parameter maxvgs). If that fails, one can try to reimport the VG using the dev files of the disks that should be in that VG. You can do strings on /etc/lvmtab to see those dev files. But one needs to edit the mapfile and remove the first line that has VGID in it:

```
# cp /etc/lvmtab.keep /etc/lvmtab
# strings /etc/lvmtab     <<<< see which disks are under that VG
# vgexport -v /dev/vgname
# mkdir /dev/vgname
# mknod /dev/vgname/group c 64 0xNM0000
# vi mapfile        <<<< remove VGID line and save it
# export D=/dev/dsk
# vgimport -v -m mapfile /dev/vgname $D/disk1 $D/disk2 .. $D/diskN
# vgchange -a y /dev/vgname
```

Where diskM=cXtYdZ for a given disk in the VG, so that $D/c0t6d0 is really /dev/dsk/c0t6d0 which simplifies typing, and saves space on command line in case there are lots of disks. If there are 1 or 2 or 3 disks, you can simply skip the export D command and type in the full dev files instead of $D/diskX syntax.

If all the above fails, the problem could be related to corruption regarding LVM in memory so a reboot is necessary. This will become apparent when the VG activates fine on node2, but fails on this node.

lvextend

1. PROBLEM lvextend: Error detected when reading from file "/etc/lvmpvg"

When trying to extend a logical volume through SAM, a lvextend error is reported:

```
lvextend: Error detected when reading from file "/etc/lvmpvg".
```

The system does not have an /etc/lvmpvg file. Is it necessary to have this file?

RESOLUTION

Not every system needs to have an /etc/lvmpvg file. It is only necessary if you want to have PVG's, or physical volume groups. They basically define subgroups of physical volumes within a volume group.

In this case the allocation policy for the logical volume was set to PVG-strict and distributed:

```
# lvdisplay /dev/vg03/lvol3
...
 Allocation     PVG-strict/distributed
```

With the allocation policy set to PVG-strict, lvextend will look for the file /etc/lvmpvg and report an error if the file is not found.
The /etc/lvmpvg file was probably removed after the logical volume was created.
In this case the logical volume resided on a VA array which already striped the data. A decision was made to turn off the allocation policies PVG-strict/distributed:

```
# lvchange -D n -s n /dev/vg03/lvol3
```

The lvextend command now succeeded.

For further information consult the man pages for lvcreate, lvchange and lvmpvg.

2. PROBLEM LVM: Extending a Striped And Mirrored Logical Volume

When using lvextend (1M) to increase the size of logical volumes (lvols) which are striped and mirrored, you must account for the allocation policy which by default is assigned to those lvols. The allocation policy is that of distributed allocation (lvextend -D) where only one free extent is allocated from the first available physical volume. The next free extent is allocated from the next available physical volume. Allocation of free extents proceeds in round-robin order on the list of available physical volumes.

The distributed allocation policy REQUIRES the PVG-strict allocation policy (-s g) to ensure that mirrors of distributed extents do not overlap (for maximum availability).

lvcreate(1M) will obtain the list of available physical volumes
from /etc/lvmpvg.

When a logical volume with distributed extents is mirrored, the resulting layout is commonly referred to as EXTENT-BASED MIRRORED STRIPES.

Note that EXTENT-BASED MIRRORED STRIPES can be created without the distributed
allocation policy by adding one extent at a time to the desired physical volumes through lvextend(1M).

Examples

Assume /dev/vg01/lvol1 is 4gb and striped across c1t5d0 and c1t6d0 with with mirrors on c0t3d0 and c0t6d0 respectively, each of these disks being 2gb PVs.

Prior to beginning the file /etc/lvmpvg would look like this:

```
# more /etc/lvmpvg

VG /dev/vg01
PVG PVG0
/dev/dsk/c1t5d0
/dev/dsk/c1t6d0
PVG PVG1
/dev/dsk/c0t3d0
/dev/dsk/c0t6d0
```

Assume you want to increase the lvol by another 1gb. Conventional thought would indicate that you should add another disk but actually you must add 4 more disks to satisfy the requirements of striping and mirroring with distributed allocation and PVG strict policies.

```
# vgextend /dev/vg01 /dev/dsk/c1t3d0 /dev/dsk/c1t4d0 /dev/dsk/c0t5d0 \
/dev/dsk/c0t4d0
```

After you added 4 disks using vgextend, i.e. c1t3d0l, c1t4d0, c0t5d0 and c0t4d0, the /etc/lvmpvg file would need to be edited so it looked like this:

```
# more /etc/lvmpvg

VG /dev/vg01
PVG PVG0
/dev/dsk/c1t5d0
/dev/dsk/c1t6d0
/dev/dsk/c1t3d0
/dev/dsk/c1t4d0
PVG PVG1
/dev/dsk/c0t3d0
/dev/dsk/c0t6d0
/dev/dsk/c0t5d0
/dev/dsk/c0t4d0
```

Now to increase the lvol, you may use the lvextend command as normal. Note that you may substitute the names of the two PVGs for the name(s) of the physical volumes when directing the added physical extents.

```
    # lvextend -L 5000 /dev/vg01/lvol1 PVG0 PVG1
```

Note also that you may alternately specify the 4 physical volumes which were added, in lieu of specifying the volumes by PVG name.

```
    # lvextend -L 5000 /dev/vg01/lvol1 /dev/dsk/c1t3d0 \
    /dev/dsk/c1t4d0 /dev/dsk/c0t5d0 /dev/dsk/c0t4d0
```

You must however use one of these methods to specify all 4 disks simultaneously. A single disk would complain about the strict allocation policy. You cannot turnoff this policy when striping. This prevents you from striping to the same disk.

The following commands will help to determine your configuration:
1) lvdisplay /dev/vgxx/lvolxx shows the information over how many disks the lvol is striped and what the stripe size it.

```
    # lvdisplay -v /dev/vgxx/lvolxx

    --- Logical volumes ---
    LV Name                    /dev/vgxx/lvolxx
    VG Name                    /dev/vgxx
    LV Permission              read/write
    LV Status                  available/syncd
    Mirror copies              0
    Consistency Recovery       MWC
    Schedule                   parallel
    LV Size (Mbytes)           1600
    Current LE                 400
```

3. PROBLEM lvextend: "LogicalExtentsNumber" is bigger than the maximum value allowed

While trying to make a file system larger using the command:

```
lvextend -L 300000 /dev/vgxx/lvol1
```

it fails with the error:

```
lvextend: "LogicalExtentsNumber" is bigger than the maximum value allowed.
```

The total physical volume size would allow a file system of 323616 Mbytes (4 Mbytes * 80904):

```
# vgdisplay /dev/vgxx
...
PE size (Mbytes) 4
total PE 80904
free PE 19404
```

RESOLUTION

The lvextend man page states for the -L option:

"lv_size is rounded up to the nearest multiple of the logical extent size, equivalent to the physical extent size defined for the volume group by the vgcreate command (see vgcreate(1M))."

The physical extent (PE) size for the volume group and such the logical extent (LE) size is 4 Mbytes. The maximum number of logical extents is 65535. That allows for a maximum logical volume size of 262140 Mbytes (4 Mbytes * 65535).

It is not possible to increase the logical volume size beyond 262140 Mbytes for the existing volume group.
To solution is to recreate the volume group with a larger PE size. A PE size of 8 Mbytes for example allows for a logical volume and file system of up to 524280 Mbytes.

Example:

```
# vgcreate -s 8 /dev/vgxx /dev/dsk/c1t0d0
```

Note: The file system data will be lost when recreating the volume group.
Backup and restore any data you already have on the volume group's
file systems.

4. PROBLEM - lvextend -l 20 /dev/vg_test/lvol2 lvextend: Not enough free physical extents availabe, at mirrored logical volumes, parts of PVGs

When Physical Volume Groups are available and there is a mirror created, make sure, Physical Volume Groups of both parts of the mirror, are given:

```
# lvextend -l 20 /dev/vg_test/lvol2 ABC    ← ABC – PVG Name
lvextend: Not enough free physical extents available. ← Only one PVG is given in the
command line, and the error appears
Logical volume "/dev/vg_test/lvol2" could not be extended.
Failure possibly caused by PVG-Strict allocation policy.
# lvextend -l 20 /dev/vg_test/lvol2 ABC XYZ ← ABC XYZ – PVG Names
Logical volume "/dev/vg_test/lvol2" has been successfully extended.
Volume Group configuration for /dev/vg_test has been saved in
/etc/lvmconf/vg_test.conf
```

The same issue is resaulted after adding the physical volumes paths.

5. PROBLEM – When extending logical volume on a mirrored physical volume groups with lvextend, a cross mirror could appear

In order to escape cross mirroring when extending a logical volume on a mirrored physical volume group, make sure following steps are executed:

1. Allocation policy is set to PVG-Strict

   Note: If Allocation policy is set to non-strict, cross-mirroring appears:

   Example:

```
# lvdisplay -v /dev/vg01/lvol2
--- Logical volumes ---
LV Name                     /dev/vg01/lvol2
VG Name                     /dev/vg01
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               1
```

```
Consistency Recovery       MWC
Schedule                   parallel
LV Size (Mbytes)           40
Current LE                 10
Allocated PE               20
Stripes                    0
Stripe Size (Kbytes)       0
Bad block                  on
Allocation                 non-strict
IO Timeout (Seconds)       default

   --- Distribution of logical volume ---
   PV Name            LE on PV  PE on PV
   /dev/dsk/c0t9d0    10        10
   /dev/dsk/c0t11d0   10        10

   --- Logical extents ---
   LE    PV1                PE1   Status 1 PV2               PE2   Status 2
   00000 /dev/dsk/c0t9d0    00010 current  /dev/dsk/c0t11d0  00010 current
   00001 /dev/dsk/c0t9d0    00011 current  /dev/dsk/c0t11d0  00011 current
   00002 /dev/dsk/c0t9d0    00012 current  /dev/dsk/c0t11d0  00012 current
   00003 /dev/dsk/c0t9d0    00013 current  /dev/dsk/c0t11d0  00013 current
   00004 /dev/dsk/c0t9d0    00014 current  /dev/dsk/c0t11d0  00014 current
   00005 /dev/dsk/c0t9d0    00020 current  /dev/dsk/c0t11d0  00020 current
   00006 /dev/dsk/c0t9d0    00021 current  /dev/dsk/c0t11d0  00021 current
   00007 /dev/dsk/c0t9d0    00022 current  /dev/dsk/c0t11d0  00022 current
   00008 /dev/dsk/c0t9d0    00023 current  /dev/dsk/c0t11d0  00023 current
   00009 /dev/dsk/c0t9d0    00024 current  /dev/dsk/c0t11d0  00024 current

# lvextend -l 15 /dev/vg01/lvol2 /dev/dsk/c0t9d0
Logical volume "/dev/vg01/lvol2" has been successfully extended.
Volume Group configuration for /dev/vg01 has been saved in /etc/lvmconf/vg01.conf
# lvdisplay -v /dev/vg01/lvol2
--- Logical volumes ---
LV Name                    /dev/vg01/lvol2
VG Name                    /dev/vg01
LV Permission              read/write
LV Status                  available/syncd
Mirror copies              1
Consistency Recovery       MWC
Schedule                   parallel
LV Size (Mbytes)           60
Current LE                 15
Allocated PE               30
Stripes                    0
```

```
Stripe Size (Kbytes)        0
Bad block                   on
Allocation                  non-strict
IO Timeout (Seconds)        default

   --- Distribution of logical volume ---
   PV Name            LE on PV  PE on PV
   /dev/dsk/c0t9d0    15        20
   /dev/dsk/c0t11d0   10        10

   --- Logical extents ---
   LE    PV1                PE1   Status 1 PV2               PE2   Status 2
   00000 /dev/dsk/c0t9d0    00010 current  /dev/dsk/c0t11d0   00010 current
   00001 /dev/dsk/c0t9d0    00011 current  /dev/dsk/c0t11d0   00011 current
   00002 /dev/dsk/c0t9d0    00012 current  /dev/dsk/c0t11d0   00012 current
   00003 /dev/dsk/c0t9d0    00013 current  /dev/dsk/c0t11d0   00013 current
   00004 /dev/dsk/c0t9d0    00014 current  /dev/dsk/c0t11d0   00014 current
   00005 /dev/dsk/c0t9d0    00020 current  /dev/dsk/c0t11d0   00020 current
   00006 /dev/dsk/c0t9d0    00021 current  /dev/dsk/c0t11d0   00021 current
   00007 /dev/dsk/c0t9d0    00022 current  /dev/dsk/c0t11d0   00022 current
   00008 /dev/dsk/c0t9d0    00023 current  /dev/dsk/c0t11d0   00023 current
   00009 /dev/dsk/c0t9d0    00024 current  /dev/dsk/c0t11d0   00024 current
   00010 /dev/dsk/c0t9d0    00025 current  /dev/dsk/c0t9d0    00026 current ←
   00011 /dev/dsk/c0t9d0    00027 current  /dev/dsk/c0t9d0    00028 current ←
   00012 /dev/dsk/c0t9d0    00029 current  /dev/dsk/c0t9d0    00030 current ←
   00013 /dev/dsk/c0t9d0    00031 current  /dev/dsk/c0t9d0    00032 current ←
   00014 /dev/dsk/c0t9d0    00033 current  /dev/dsk/c0t9d0    00034 current ←
```

Note : After the execution of lvextend with non-strict allocation policy set, cross mirror appears.

2.The order of the physical volumes in /etc/lvmpvg is also important. The first device which stays in /etc/lvmpvg is possible to be used as first device to write to after execution of lvextend.

Note: This is only tested behaviour, but it could also depend on the particular configuration.

Booting:

When LVM Data Structures Are Lost

When critical LVM data structures have been lost, you will need to use the recovery portion of the Support Media included in the HP-UX product kit to restore the corrupted disk image from your backup tape. For more information, see the *Support Media User's Manual*.

After you have made the LVM disk minimally bootable, the system can be booted in maintenance mode using the -lm option of the hpux command at the ISL> prompt. This causes the system to boot to single-user state without primary swap, dump, or LVM to access the root file system. See "Booting in Maintenance Mode" in Chapter 2 for more information. (The information is identical for Series 700 or 800 systems.)

CAUTION: The system *must not* be brought to multi-user mode (that is, run-level 2 or greater) when in LVM maintenance mode. Corruption of the root file system might result.

To exit LVM maintenance mode, use reboot -n.

When a Volume Group Will Not Activate

Normally, volume groups are automatically activated during system startup. Unless you intentionally deactivate a volume group using vgchange, you will probably not need to reactivate a volume group.

However, LVM does require that a "quorum" of disks in a volume group be available. During normal system operation, LVM needs a quorum of more than half of the disks in a volume group for activation. If, during run time, a disk fails and causes quorum to be lost, LVM alerts you with a message to the console, but keeps the volume group active.

When booting the system, LVM also will require a quorum of one more than half of the disks in the root volume group. This means, for example, that LVM cannot activate a two-disk root volume group with one of the disks offline. As a result, if this happens, you will have a problem booting.

Another possible problem pertaining to activation of a volume group is a missing or corrupted /etc/lvmtab file. You can use the *vgscan*(1M) command to re-create the /etc/lvmtab file.

Quorum Problems

Quorum Problems with a Non-Root Volume Group

If you attempt to activate a non-root volume group when not enough disks are present to establish a quorum, you will see error messages similar to those in this example:

```
# vgchange –a y /dev/vg01

vgchange: Warning: Couldn't attach to the volume group
                physical volume "/dev/dsk/c1t0d2":
The path of the physical volume refers to a device that does not
exist, or is not configured into the kernel.

vgchange: Couldn't activate volume group "/dev/vg01":
Either no physical volumes are attached or no valid VGDAs were found
on the physical volumes.
```

If a non-root volume group does not get activated because of a failure to meet quorum, try the following:

- Check the power and data connections of all the disks that are part of the volume group that you cannot activate. Return all disks (or, at least enough to make a quorum) to service. Then, use the vgchange command to try to activate the volume group again.
- If there is no other way to make a quorum available, the -q option to the vgchange command will override the quorum check.

  EXAMPLE:

  ```
  # vgchange –a y –q n /dev/vg01
  ```

CAUTION: This will activate the volume group but a quorum will not be present. You might get messages about not being able to access logical volumes. This is because some or all of a logical volume might be located on one of the disks that is not present. Whenever you override a quorum requirement, you run the risk of using data that is not current. Be sure to check the data on the logical volumes in the activated volume group as well as the size and locations of the logical volumes to ensure that they are up-to-date.

You should attempt to return the disabled disks to the volume group as soon as possible. When you return a disk to service that was not online when you originally activated the volume group, use the activation command again to attach the now accessible disks to the volume group.

EXAMPLE:

```
# vgchange –a y /dev/vg01
```

Quorum Problems with Your Root Volume Group

Your root volume group might also have a quorum problem. This might occur if you have physically removed a disk from your system because you no longer intended to use it with the system, but did not remove the physical volume from the volume group using vgreduce. Although you should *never* remove an LVM disk from a system without first removing it from its volume group, you can probably recover from this situation by booting your system with the quorum override option, hpux -lq.

Problems after Reducing the Size of a Logical Volume

If you extend the logical volume *without* extending its file system, you can subsequently safely reduce the logical volume's size as long as it remains as big as its file system. (Use *bdf*(1) to determine the size of your file system.) Once you use the extendfs command to expand the file system, you can no longer safely reduce the size of the associated logical volume.

If you reduce the size of a logical volume containing a file system to a size smaller than that of a file system within it using the lvreduce command without subsequently creating a new smaller file system, you may crash your system. If this occurs:

1. Reboot your system in single-user state.
2. If you have already have a good current backup of the data in the now corrupt file system, skip this step.

   Only if you do not have such backup data *and* if that data is critical, you may want to *try* to recover whatever part of the data that may remain intact by attempting to back up the files on that file system in your usual way.
3. Immediately unmount the corrupted file system, if it is mounted.
4. You can now use the logical volume for swap space or raw data storage, or use SAM or the newfs command to create a *new* file system in the logical volume. This new file system will now match the current reduced size of the logical volume.
5. If you have created a new file system on the logical volume, you can now do *one* of the following:

- If you have a good prior backup, restore its contents. Since the new file system in the smaller logical volume will be smaller than the original file system, you may not have enough space to restore all your original files.
- Use the new file system for creating and storing a new set of files (not for trying to restore the original files).
- If you do not have a good prior backup, *attempt* to restore as many files as possible from any backup. Again, there is no guarantee that complete data will be recoverable from this backup.

No Response or Program Output from an Inaccessible Disk

You might occasionally see long periods of apparent inactivity by programs that are accessing disks. Such programs may be "hung", waiting for access to a currently inaccessible disk. Messages indicating the disk is offline will also appear on your system console.

If the logical volume is mirrored onto another disk, the hang will only last until the disk driver returns (about a minute or perhaps a little longer). LVM then marks the disk as offline and continues the operation on any remaining mirror disk. LVM will only write to or access the mirror copy until the offline disk is returned to service.

If the logical volume is not mirrored, or if the mirror copies of the logical volume are also not available, the program will remain hung until a disk becomes accessible. Therefore, if your program hangs, you should check for problems with your disk drives and, if necessary, restore them to service as soon as possible.

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: 11.31 LVM commands in `/usr/sbin`

```
# ls -alil -iF vg* lv* pv* nomwc*
  4865 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvchange@ ->
./lvm_wrapper
  4866 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvcreate@ ->
./lvm_wrapper
  4867 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvdisplay@ -
> ./lvm_wrapper
  4868 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvextend@ ->
./lvm_wrapper
  4869 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvlnboot@ ->
./lvm_wrapper
  4870 -r-sr-xr-x   1 root       sys       247648 Dec  8  2009 lvm_wrapper*
  4871 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvmadm@ ->
./lvm_wrapper
  4872 -r-xr-xr-x   1 bin        bin         1627 Jan 11  2010 lvmchk*
  4873 -r-sr-xr-x   1 root       sys      1892028 Jan 11  2010 lvmcmd*
  4874 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvmerge@ ->
./lvm_wrapper
  4875 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvmove@ ->
./lvm_wrapper
  4876 -r-sr-xr-x   1 root       sys      2563004 Jan 11  2010 lvmpcmd*
  4877 -r-sr-xr-x   1 root       sys        81920 Dec  8  2009 lvmpud*
  4878 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvreduce@ ->
./lvm_wrapper
  4879 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvremove@ ->
./lvm_wrapper
  4880 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvrmboot@ ->
./lvm_wrapper
  4881 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvsplit@ ->
./lvm_wrapper
  4882 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 lvsync@ ->
./lvm_wrapper
  4883 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 nomwcsyncd@
-> ./lvm_wrapper
  4884 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 pvchange@ ->
./lvm_wrapper
  4885 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 pvck@ ->
./lvm_wrapper
  4886 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 pvcreate@ ->
./lvm_wrapper
  4887 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 pvdisplay@ -
> ./lvm_wrapper
  4888 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 pvmove@ ->
./lvm_wrapper
  4889 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 pvremove@ ->
./lvm_wrapper
  4890 lrwxrwxrwx   1 root       sys            13 Jun  7  2010 vgcfgbackup@
-> ./lvm_wrapper
  4891 lrwxrwxrwx   1 root       sys            13 Jun  7  2010
vgcfgrestore@ -> ./lvm_wrapper
```

```
  4892 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgchange@ ->
./lvm_wrapper
  4893 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgchgid@ ->
./lvm_wrapper
  4894 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgcreate@ ->
./lvm_wrapper
  4895 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgdisplay@ -
> ./lvm_wrapper
  4896 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgexport@ ->
./lvm_wrapper
  4897 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgextend@ ->
./lvm_wrapper
  4898 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgimport@ ->
./lvm_wrapper
  4899 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgmodify@ ->
./lvm_wrapper
  4900 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgmove@ ->
./lvm_wrapper
  4901 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgreduce@ ->
./lvm_wrapper
  4902 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgremove@ ->
./lvm_wrapper
  4903 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgscan@ ->
./lvm_wrapper
  4904 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgsync@ ->
./lvm_wrapper
  4905 lrwxrwxrwx   1 root       sys             13 Jun  7  2010 vgversion@ -
> ./lvm_wrapper
```