Paper 1010
HP 9000/800 Bootable Backup

Todd Loeppke, Southwestern Bell Telephone Company
Chris Freeman, Hewlett-Packard Company

Backup and recovery policies are like buying insurance for your system. It can take a lot of time, planning and resources to make it useful which cost money. The trick is, (like buying insurance), to answer the question how much do I need to spend to make the loss tolerable. How much data can I afford to lose and how long can it take to restore the system to the original state. As you will find in this paper you can spend an infinite amount of time preparing for hardware data loss but this time is spent knowing that there may never be a return on the investment. On the other hand it could save your job too.

A.) WHAT IS A BOOTABLE BACKUP TAPE:

A bootable backup tape (BBT) is a solution that can be used to restore a system in the event of a hardware failure with little or no human intervention. This tape is created by running a command on the system (maketape_sh) you are protecting. The output creates a "snap shot" of the logical volume manager and filesystem configuration of the system. The tape can also contains your backup/recovery software or an entire backup of your system depending on the amount of the used disk space.

For a system running HP-UX 9.x the BBT can be created 2 ways. The first is with a normal hp-ux install file. This solution is only partially automated. Some system information must always be entered during the recovery. The second is to use an unsupported hp-ux install tape special from Hewlett-Packard. This tape allows programmatic input of critical system information that is normally entered manually during a system install.

HP took into consideration the necessity of creating a BBT when designing HP-UX 10.0. As a result all of the features available through the HP-UX 9.x unsupported special are available as HP-UX 10.x standard features. Some of these features will be discussed in this paper. For additional information on this topic please reference the "HP-UX 10.0 System Installation Tools" white paper by David Arko and Mark Egeland of Hewlett-Packard.

Attached documents in this paper represent a BBT for an HP 9000/800 9.x system unless noted otherwise. Notes and comments appropriate to equivalent functionality will be added as it pertains to creating a BBT for 10.x. No script examples are provided in this paper for a 10.x implementation.

B.) WHEN WOULD YOU USE IT:

Although you hope to never have to use this solution, if needed it can save in down time and may make the difference in whether you can restore your system to its original state.

In the area of disk hardware failures there are four basic categories of disk data and they vary in their range of importance for the system to be able to operate correctly. The BBT in this paper addresses the first and most difficult category but could be modified to address the second category also.

1. The first type of disk/data loss would be to loose all disk drives on the system. Although this is probably not likely in a system with multiple disk drives or disk racks it is possible in a smaller system with only a few drives and could be more likely depending upon the hardware layout. This type of failure would also be possible if the system is effected by a natural or man made disaster. When this type of failure occurs the options for recovery are limited. Assuming you have hardware to recover you have two basic options:

   a.) Reload from your Install CD and then restore from your backups

   b.) Use a BBT and then restore from your backups if needed

   Method one would require several steps. First would be to reload the operating system. To do this you would need to know the configuration of the original root volume group. You would want to then check to see if your lu's are assigned the way they were on the original system and if not correct your device files. Once all the device files are set-up correctly you would need to either reconfigure all your LVM disk or recover the /etc/lvmconf files, use vgcfgrestore and then vgimport. Then you would need to run newfs on the appropriate volumes, create the original mount points and mount the filesystems. Now you are ready to start your system restore from your backups with the overwrite newer files option enabled.

   The second method or recovery would consist of putting the BBT in the tape drive and booting from it at the "ISL>" prompt. From there the system would boot from the tape with no intervention, restore the LIF file to the root disk, restore a minimal unix system, create the correct disk device files,
restore the lvm information to the disk drives, run newfs recreate mount points, mount all filesystems and then recover whatever else you put on the BBT (could be the entire system if it fits on the tape) and then the system would reboot. All of this happen automatically with no intervention needed. At that point you would recover whatever else you might want from your regular backups. This second method does not require the "restorer" to know anything about the original Logical Volume Manager (LVM) or filesystem and takes much less time.

2. The second type of disk/data loss would be one that effected the "root drive" or one with operating system type files like "/usr" on it. In that case you have four options for restoring the information after the disk(s) are replaced.

   a.) Reload from your Install CD and then restore from your backups

   b.) Use the support CD to help with recovery

   c.) Use COPYUTIL on the support CD

   d.) Could use a BBT and then restore from your backups if needed

   Solution one (described above) is very involved and might mean doing unnecessary work and additional downtime.

Solution two requires extensive knowledge of how to use the support media and might require input from the HP Support Line. With this method you will again need to know the LVM and filesystem layout and will require the use of the vgcfgrestore command. Solution three uses the COPYUTIL which is a utility on the support media that allows you to copy an entire disk including the boot area to tape. The downside of this is you must shutdown your system to run COPYUTIL and it takes and hour or more for each disk you run it on.

The fourth solution could be to use a BBT but would take some modifications to the "rebuild" script.

3. The third type of disk/data loss would be to loose the disk drive which contains your backup/recovery software. In this case you would need to reinstall the software or have a separate backup of the software on tape using a unix tool/command like tar or cpio.

4. The last type of disk/data lose would be to lose a disk containing data or application software on it. With this type of failure you would replace the disk and use vgcfgrestore, newfs and your normal backup/recovery software.

C.) TECHNICAL DISCUSSION

1. HP-UX SYSTEM INSTALLATION THEORY

To create a BBT you need to understand at least three basic things. They are as follows:

a.) Format of the install tape

This tape is the starting point in creating a BBT. Without it's contents it would be impossible to create. The tape as it is shipped from HP consist of 2 basic components. They are the Logical Interchange Format (LIF) file and a basic operating system in tar format hereto referred to as the "hp-ux tarball".

The installation process between an HP 9000/800 system running HP-UX 9.x and one running HP-UX 10.x are basically the same. Note, however, that the mechanics of how this is accomplished are somewhat different. But for the purposes of creating a BBT the structure of the install tape is the same. We will discuss that mechanics of how an HP 9000/800 system running HP-UX 9.x and one running 10.x install later in the paper.

Before you can customize the contents of the BBT you need to copy them from the install media. A DAT tape provides the most accessible media. Execute the dd command with the following syntax to accomplish the task of copying the install tape to a work area:

For the LIF file
dd if=/dev/rmt/0m of=/dir/of/choice/file bs=2k

For the hp-ux tarball
dd if=/dev/rmt/0m of=/dir/of/choice/file bs=10k

b.) Contents of the LIF file and hp-ux tarball

The LIF file is in reality a container for several files. Normally you never access the contents the LIF file. Please note that modification of any of these files is not support by Hewlett-Packard. However, for the purposes of creating an automated BBT for a 9.x system some modification of it's contents is absolutely necessary. For future reference the contents of the LIF file will be notated as such: LIF:xxx where xxx is the file contained in the LIF file.

Access to the contents of a LIF file is done through the use of the lifcp, lifrm and lifls commands. Fortunately these are include with the operating system.

The lifls allows you to view the contents of the LIF file. The "-l" or long listing option provides useful information about the LIF file. Use this command on the newly dd'd file. See the hp-ux lifls man page for additional information.

Lifrm does as you would expect. It removes the file from the LIF file. Use this with caution. Additional information on this command can be obtained via hp-ux man pages.

Lifcp gives you the capability of copying the LIF file files into and out of the actual LIF file. This is critical to a 9.x BBT solution. The lifcp options necessary for the implementation of a BBT solution are shown in code exapomles in Appendix A. Additional information on this command can be obtained via hp-ux man pages.

Two files contained in the LIF file need to be modified. They are the LIF:AUTO file and the LIF:INSTALL file. The LIF:AUTO file contains the physical address of the "install to" disk. The LIF:INSTALL file contains the logic responsible for requesting and allocating the root and swap file systems sizes. With a standard LIF:INSTALL file the only way to provide this information to the install process is to either accept defaults presented during the installation or to override them.

The hp-ux tarball is a standard tar file. Therefore, it can be extracted using the "tar xvf /hpux.tarball" command. The best way to examine the contents of this file is to extract it and snoop around. This will uncover that you have just extracted a complete operating system. Certain parts of this operating system will need to be modified when creating a 9.x BBT. These will be discussed later in this paper.

Because HP took into consideration the need for a BBT when designing 10.x modifying files in the hp-ux tarball is no longer necessary. This greatly simplifies the process of creating a BBT in a 10.x environment.

c.) The third thing that is necessary to have is a basic understanding of what occurs during a system installation.

Installation of a 9.x system follows these basic steps:

1) Power on
2) Select boot device. System will determine a new install if the device is a CD-ROM or DAT tape.
3) If using a CD-ROM or DAT tape the LIF:INSTALL file is read and executed. It will install a root file system on the designated root disk This is when the hp-ux tarball is installed.
4) Install will also ask for additional information such as:

   Do you wish to install LVM?
   Do you wish to change Root or swap sizes?
   Etc.

5) After system has rebooted it will prompt you to enter the install media.
6) System will reboot and execute the inittab file. This is a special inittab file containing commands to install LVM
7) The system will install all appropriate files and enable networking upon recovery.

Installation of a 10.x system follows these basic steps:

1) Steps 1 through 3 of the previous section are performed.
2) Install asks for information concerning LVM configuration, filesystem types and networking through a menu driven user interface.
3) After making appropriate selections the system will reboot.
4) Modifications to the standard configuration are stored in a config.local file. This file is used to extend the contents of the LIF file from which the system is now booting.
4) Depending on whether you selected an installation from media or network will determine from where the core operating system is installed.
   5) The system will install all appropriate files and enable networking upon recovery.

2. CREATING THE TAPE

a.) CREATE TAPE WALK THROUGH

Regardless of what OS level you are using some essential system information must be gathered. They are as follows:

current boot device addressing information ("install to" device)
current root, swap information
current volume group, logical volume and filesystem information

When creating a fully automated BBT it is necessary to use the unsupported hp-ux installation tape special. A tool is provided with this special that modifies the LIF:INSTALL file called mod_lif_file. This tool reads the contents of a configuration file containing root and swap information and loads it into the LIF:INSTALL file. See Appendix A "sizes.exampl" for an example config file. The script cannot be used on a

standard LIF file because modifications to the LIF:INSTALL file where required to accommodate this information. In addition the special LIF:INSTALL file will perform the installation without requiring any user input by accepting default answers to installation questions. These two features combined are what allow for a fully automated recovery..

In 10.x all tools are included with the operating system. All modifications to installation process can be made by modifying the /usr/lib/sw/hpux.install/config.local file. This file, once modified, will be placed into a reserved area on the 10.x install LIF:INSTALLFS file by the use of the instl_adm command. Please reference the hp-ux man pages for instl_adm (1) and instl_adm (4) for more information.

b.) CHECKING THE SYSTEM

Before you can create the BBT you must interrogate your system either manually or programetically to determine it's current configuration. This configuration information will be stored in various places which include the lvminst file, LIF:AUTO and configuration file containing the root and swap information. See Appendix A "newcauto.sh" for example code. Note that similar information is will be required for a 10.x implementation of a BBT. However this information will be stored in the config.local file. See Appendix A "config.local" for an example config.local file.

c.) THE lvminst FILE

It is important to mention what the lvminst file is and what impact it has on the installation of an HP-UX 9000/800 9.x system. As mentioned in section C.1.c.6 a special inittab file is executed during the installation process. Among other things this file executes a binary called lvminst which is responsible for installing LVM. By replacing this file with a script which performs equivalent activities the installation process can be cleanly interrupted. The new lvminst should contain lvm commands to recreate the VGs, lvols and filesytems of the system being recovered at the time the BBT was created. It also needs commands to recover the system and then reboot. The lvminst file is located the /xxx/local directory where xxx is the directory of the extracted hp-ux tarball.

c.) CREATING THE LIF

Creating the LIF involves the following steps:

1) Get a working copy of the LIF file
2) Extract the LIF:AUTO file
   ex: lifcp uxbootlf.tape:AUTO AUTO
3) Modify the LIF:AUTO file with the address of the boot disk
4) Replace the LIF:AUTO file
   ex: lifrm uxbootlf.tape:AUTO
   lifcp -r -T -12289 AUTO uxbootlf.tape:AUTO
5) Modify the LIF:INSTALL file via the mod_lif_file program using system parameters

ex: mod_lif_file -i instl_adm -l uxbootlf.tape
note: the readme file for the hp-ux install tape special defines the usage of the mod_lif_file tool.

6) dd the new LIF file to the 1st position on your new BBT
ex: dd if=uxbootlf.tape of=/dev/rmt/0mn bs=2k

Please reference Appendix A "maketape.sh and newauto.sh" for example code.

d.) CREATING THE REVISED HP-UX TARBALL

The hp-ux tarball is complete with the exception of the lvminst file and a few extra operating system commands necessary to recover the system. The additional commands required depend on how you implement you BBT. To create the hp-ux tarball requires the following steps:

1) Put a copy of the hp-ux tarball in working directory
ex: dd if=/dev/rmt/0m
2) Extract the contents of the tarball
ex: tar xvf /working_dir/hp-ux_tarball
3) Replace the lvminst file with the modified lvminst file
4) Put any additional commands in the appropriate locations:
ex: mt, ksh, vi, etc.
5) tar the extracted tarball contents onto the BBT as the second image. It may be necessary to rewind the tape and position past the first image to insure this is done successfully.
ex: tar cvf /workingdir/hp-ux_tarball

Please reference Appendix A "maketape.sh" for example code.

Below is an example of output generated when executing the maketape.sh script.

e.) CREATING THE lvminst FILE:

Thehp-ux install tape special supplied by HP for the BBT executes a shell in /local called "lvminst" when the system eventually boots off the target root disk drive. On a normal Install Media this is usually a binary file but in the case of the special bits it is just a shell script. This is the file that is customized to produce the desired results (in our case a "hands-off" full system recovery).

During the recovery the lvminst script performs the following functions:

1.) run rmsf to remove potentially incorrect /dev/dsk files

This must be done because the way the BBT initially installs the disk device files is unpredictable as far as what lu's are mapped to what disk drives.

2.) recreate the /dev/dsk files with insf

The "rebuild" program takes the lu mapping from the original system and creates insf commands to recreate the disk device files.

3.) run ioinit -f

This forces the information from the original ioconfig to overwrite any logical unit information in the kernel

4.) run vgcfgrestore on all LVM disk drives

This recreate the volume group areas on each disk and is easier than rebuilding all the volume groups from scratch.

5.) run rm, mkdir and mknod on the LVM device directories

This prepares the system for running vgimport

6.) run vgimport and the vgchange to activate the original volume groups

This seems to be the "cleanest" method of rebuilding the volume group device files and a correct /etc/lvmtab file.

7.) save root volume group information with lvlnboot

This updates the physical volumes in the root volume group so the system can boot properly.

8.) run "mount -u"

This insures a correct /etc/mnttab file.

9.) run newfs on all LVM filesystem

Rebuild unix filesystems on the disk drives.

10.) run "mkdir -p"

This creates mount points for all mountable filesystems.

11.) run "mount -a"

Mount all filesystem listed in the original /etc/checklist.

12.) run frecover to recovery the third image off the BBT

This recovers what ever is in the third image on the BBT.

13.) reboot the system with "reboot -s"

This step sync's the filesystems and reboots the system.

For an example of the rebuild shell script please reference Appendix A "rebuild". Rebuild is executed on the system where you are building the BBT. It generates the "lvminst" script. Reference Appendix A "lvminst" for example output of the rebuild shell script.

g.) DETERMINING THE BACKUP TOOL FOR THE THIRD BBT IMAGE

The third image on the tape allows for a lot of flexibility. The "tar" command was used to for the first two images on the tape. With the third image we wanted to use a command that would allow us to pick and choose exactly what went on the tape. We decided to use fbackup because if would allow
us to use a configuration file (graph file) where we could list exactly what we wanted to backup. With "tar" and "cpio" this is not easily done.

h.) CREATING THE THIRD IMAGE

The next problem we encountered was that fbackup/frecover wants to rewind the tape and read the read/write header information when invoked. We needed the backup image to be the third image on the tape and besides that the LIF information was the first thing on the tape not the fbackup/frecover header. The way we where able to get around not rewinding the tape and not reading/writing a tape header was to us the "dd" command. An example of the command is shown below:

To create the third image we used:

/etc/fbackup -v -y -g ${bt_dir}/fbackup.graph -f - | dd of=/dev/rmt/0mn bs=10k

To recover the data we used:

/bin/dd if=/dev/rmt/0mn bs=10k | /etc/frecover -vr -o -f -

3. RECOVERY WITH THE BBT

a.) 9.04 BOOT SPECIAL

Recovery with the 9.x BBT create with the unsupported hp-ux install tape special initially follows the same process as any other hp-ux install. Places where the special tape and modified files will be clearly noted.

1) Boot tape is loaded.
2) System is powered on and tape device is selected as boot device.
3) No additional user interaction is required.
4) System loads the modified LIF:INSTALL file.
5) Root file system is installed on disk address in modified LIF:AUTO file.
6) System reboots and executes inittab which in turn runs modified lvminst file.
7) Modified lvminst creates all necessary VGs, lvols, and filesystems.
8) Modified lvminst executes the restore command (assume whole system restore).
9) After restore completes system reboots again.
10) User logs on to a restored system.

b.) 10.X BOOT SPECIAL

1) Boot tape is loaded.
2) System is powered on and tape device is selected as boot device.
3) No additional user interaction is required.
4) System loads the modified LIF:INSTALLFS file.
5) LIF:CONFIG and LIF:INSTALLFS parse configuration information. (Note: LIF:INSTALLFS was modified with config.local information)
6) Root file system is installed on disk address designated in config.local
7) VG's, lvols and filesystems are restored as defined in config.local
8) After restore completes system reboots again
9) User logs on to a restored system

c.) EXAMPLE RECOVERY OUTPUT

Please reference Appendix B for output from a 9.x BBT system recovery using the unsupported hp-ux install special.

d.) RECOVERING DATA

Once your operating system is recovered and all the filesystems are mounted you are then ready to recover the rest of your system. What you need to restore depends on how much you include in the 3rd tape image. If an entire backup fixs on the tape then you

are done except for any files that have changed since you created your BBT. If the entire system will not fit on the tape you will want to make sure your third image contains whatever backup and recovery software you use. Make sure that any backup/recovery patches needed are part of the 3rd tape image and also use the "over write newer file" option so you get the most current files. If your backup/recovery software is network based you will need to run your normal restore after the system reboots.

D.) OTHER ISSUES:

As mentioned in the "WHEN WOULD YOU USE IT" section of the paper we only addressed the loss of an entire system in this paper. With second type of disk/data loss where you lose a part of your our operating system like /usr or the root drive the lvminst script could be generated so the person restoring the system is asked before the
different drives and filesystems are restored. In this case most of the disk drives are uneffected you would not want to run the newfs on all the drives. Recovering only what needs to be recovered will also save time.

If you have logical volumes that do not use the default volume names (lvol1, lvol2, ...) you will need to modify your "rebuild" script include a "map file" for vgimport to use. If this is the case you will probably need to boot the system in "LVM maintenance mode" and run the "rebuild" script at that time so you can get a map file from vgexport command. Volume groups must not be "active" for vgexport to work. Make sure you use the "-p" option for preview so you do not remove the volume group. If your root volume group uses default volume name then you will not need to go in to "LVM maintenance mode" to create the "map file". You will just need to umount the file systems, deactivate the volume group, run vgexport with the "-p and -m" options and then activate and mount the filesystems again.

This BBT only works on systems where all disk space ("cooked" or "raw") uses LVM and any replacement hardware must have its hardware address settings set to the original values.

E.) SUMMARY

This paper has covered a lot of territory. And to be honest many hours of work went into creating a flexible 9.x BBT solution. All this effort has culminated in a reasonably effective and reliable method of creating an automated disaster recovery solution. Remember that all of the shell scripts and config files you can create it to suit your own recovery needs.

Hopefully what has been communicated are the what's, why's and to some degree the how's involved in creating a bootable tape solution. The bottom line is it's an important venture to embark upon and it a doable solution. Enjoy the challange and good luck.

APPENDIX A

maketape.sh code example

```
==========================================================================
#
# Execute command to create the first two images on the boottape
#
# Define variables for this script
#
bt_dir=/usr/local/boottape
bt_lf=${bt_dir}/uxbootlf.tape
bt_lfinstall=${bt_dir}/uxbootlf.install
bt_tar=${bt_dir}/tarball
bt_tarfile=${bt_dir}/tarchive.tar
bt_newauto=${bt_dir}/newauto.sh
bt_lf_orig=${bt_dir}/orig/uxbootlf.tape
tape=/dev/rmt/0m
tapen=/dev/rmt/0mn
#
# Create lvminst
#
rm ${bt_tar}/local/lvminst
${bt_dir}/rebuild
cp ${bt_dir}/rebuild.out ${bt_tar}/local/lvminst
chmod 755 ${bt_tar}/local/lvminst
#
# Copy a good clean uxbootlf.tape for the orig directory
#
echo Copy a good clean uxbootlf.tape for the orig directory
cp ${bt_lf_orig} ${bt_lf}

#
# Execute the mod_lif_file using the system parameters
#
echo "Modify the lif file"
echo "${bt_dir}/mod_lif_file -i ${bt_dir}/instl_adm -l ${bt_lf}"
${bt_dir}/mod_lif_file -i ${bt_dir}/instl_adm -l ${bt_lf}
#
# Execute the newauto.sh
#
${bt_newauto}

# Make sure the tape is in the drive
echo "Put the tape in the drive and then hit Enter"
read reply
echo "Rewinding boottape (Just in case)"
/bin/mt -t ${tape} rew
#
# dd the lf image onto the tape
#
```

```
echo "/bin/dd if=${bt_lf} of=${tapen} bs=2k"
/bin/dd if=${bt_lf} of=${tapen} bs=2k

#
# tar the tarball stuff onto the tape
#
cd ${bt_tar}
echo "Creating tarfile ${bt_tarfile}"
/bin/tar cvf ${bt_tarfile} .
echo "Positioning boottape"
/bin/mt -t ${tape} rew
/bin/mt -t ${tapen} fsf 1
echo "Putting tar image on tape"
echo "/bin/dd if=${bt_tarfile} of=${tapen} bs=10k"
/bin/dd if=${bt_tarfile} of=${tapen} bs=10k
#
# Do the fbackup ?
#
echo "Positioning boottape"
/bin/mt -t ${tape} rew
/bin/mt -t ${tapen} fsf 2
echo "Beginning fbackup"
echo "/etc/fbackup -v -y -g ${bt_dir}/fbackup.graph -f - | dd of=${tapen} bs=10k"
/etc/fbackup -v -y -g ${bt_dir}/fbackup.graph -f - | dd of=${tapen} bs=10k
#
# Rewind and eject the tape
#
echo "Finished creating boottape"
echo "Rewinding and ejecting tape"
/bin/mt -t ${tape} offl
```

APPENDIX A

newauto.sh code example

```
#!/bin/ksh
# ----------------------------
# --- Description
# The script will determine which disk is be
# be the system boot disk in the event of a
# system recovery using the recovery boot tape.
# It modifies the AUTO file in the boot lif to
# point to the correct install destination.
```

```
# ----------------------------

# ----------------------------
# --- Define variables
LVFILE=/tmp/lvfile.dat
BTDIR=/usr/local/boottape
AUTOORIG=${BTDIR}/AUTO.orig
AUTO=${BTDIR}/AUTO
SEDSHELL=/tmp/sedshell.sh
SEDTMP=/tmp/sedtmp.dat
# ----------------------------

# ----------------------------
# --- Determine the disk the system currently booted from
# Assume that vg00 is the root vg
# determine the logical volumes in the root vg
LVOL=`vgdisplay -v vg00 | grep "LV Name" | awk '{print $3}'`
# ----------------------------

# ----------------------------
# --- Get the physical volume information on the first entry in the
# --- LVOL
PVOL=`lvdisplay -v ${LVOL} | sed -n '17p' | awk '{print $1}'`
# ----------------------------

# ----------------------------
# --- Get the hw address of the boot disk
# HWADDR=`ioscan -f ${PVOL} | tail -1 | awk '{print $3}'`
HWADDR="56/40.1.0"
# Put in escape character in HWADDR (\) if necessary
if [ `echo ${HWADDR} | grep \/` ]; then
  echo ${HWADDR} | sed 's/\/\\\//' > ${SEDTMP}
  HWADDR=`cat ${SEDTMP}`
  # cleanup /tmp
  rm ${SEDTMP}
fi
# ----------------------------

# ----------------------------
# --- Modify the AUTO file
# cp ${AUTOORIG} ${AUTO}
echo "sed 's/TOADDR/${HWADDR}/' ${AUTOORIG} > ${AUTO}" > ${SEDSHELL}
chmod +x ${SEDSHELL}
${SEDSHELL}
# cleanup /tmp
rm ${SEDSHELL}
# Determine if you want to continue with the AUTO modifications
echo "Do you wish to modify the AUTO file with the following:\n"
```

```
echo "=================================================="
cat ${AUTO}
echo "=================================================="
echo "\nPlease enter choice [y,n]: \c"
read answer
if [ "${answer}" = "n" ]; then
    exit
fi
# ----------------------------

# ----------------------------
# --- Remove AUTO from boot lif
echo "\nContinuing with AUTO modifications"
lifrm uxbootlf.tape:AUTO
# ----------------------------

# ----------------------------
# --- Copy the modified AUTO into the boot lif
lifcp -r -T -12289 AUTO uxbootlf.tape:AUTO
# ----------------------------

# ----------------------------
# --- List the lif files for verification
lifls -l uxbootlf.tape
ls -l uxbootlf.tape*
# ---------------------------- End of newauto.sh


#############

LIF:AUTO file as copied from install tape LIF file
======================================================================
hpux -a (TOADDR;0) (;0xa0000):INSTALL

#############

LIF:AUTO as modified by newauto.sh
======================================================================
hpux -a (56/40.1.0;0) (;0xa0000):INSTALL
```

APPENDIX A

config.local example

```
==========================================================================

INIT "Configuration from \"a2610cxf\"" = FALSE
"Configuration from \"a2610cxf\"" help_text
"Selecting this option will give you a configuration similar to that
used when the system \"a2610cxf\" was installed.  This will include
the disk layout as well as software selection."
LLA_ADDR=="080009350943"
{
        system_name="a2610cxf"
        ip_addr="15.31.72.90"
        netmask="255.255.248.0"
        gateway="15.31.72.1"
        RUN_UI=FALSE
        RUN_SD=FALSE
        POST_CFG_CMD+="
                restore commands
                ...
                ...
                ..."
}
best=disk[2/0/1.6.0]
"Configuration from \"a2610cxf\""
{
        #
        # START DISK VOLUME CONFIGURATION
        #
        volume_group "vg00" {
                physical_volume disk[best] {
                } # end pv_options
                physical_volume disk[2/0/1.5.0] {
                } # end pv_options
                max_physical_extents=2000
                physical_extent_size=4
                logical_volume {
                        mount_point="/"
                        usage=HFS
                        size=307200K
                        FILE_LENGTH=LONG
                        bad_block_relocate=FALSE
                        contiguous_allocation=TRUE
                } # end logical_volume
                logical_volume {
                        mount_point="primary"
                        usage=SWAP
                        size=49152K
                        bad_block_relocate=FALSE
```

```
                        contiguous_allocation=TRUE
                } # end logical_volume
                logical_volume {
                        mount_point="secondary"
                        usage=SWAP
                        size=102400K
                        bad_block_relocate=FALSE
                        contiguous_allocation=TRUE
                } # end logical_volume
                logical_volume {
                        mount_point="/usr"
                        usage=VxFS
                        size=204800K
                } # end logical_volume
                logical_volume {
                        mount_point="/home"
                        usage=VxFS
                        size=49152K
                } # end logical_volume
        } # end volume_group "vg00"
} # end "Configuration from \"a2610cxf\""
```

APPENDIX A

rebuild example

```
================================================================
##############################################################################
#
# PROGRAM NAME: rebuild
#
# DISCRIPTION: This program create the /local/lvminst shell which is
#              executed during the boot on the target root disk.
#
# AUTHOR: J. Todd Loeppke         COMPANY: Southwestern Bell Telephone
#
# DATE: 5/13/96
#
##############################################################################
#
# Location of the boottape source
#
bt_dir=/usr/local/boottape
#
# Copy original /etc/ioconfig /etc/checklist to tarball
#
cp /etc/ioconfig ${bt_dir}/tarball/etc
cp /etc/checklist ${bt_dir}/tarball/etc
#
# remove old disk rebuild profile
#
rm ${bt_dir}/rebuild.out >/dev/null 2>&1

#
# CREATION OF lvminst STARTS
# "rebuild.out is moved to /usr/local/boottape/tarball/local/lvminst by
# the "maketape.sh" script
#

#
# Remove the disk device files
#
echo "/bin/echo \"/etc/rmsf -a /dev/dsk/*\"" >> ${bt_dir}/rebuild.out
echo "/etc/rmsf -a /dev/dsk/*" >> ${bt_dir}/rebuild.out
#
# cd to /dev to created disk device files
#
echo "/bin/echo \"cd /dev\"" >> ${bt_dir}/rebuild.out
echo "cd /dev" >> ${bt_dir}/rebuild.out
#
# Remove  and rebuild device files to insure correct lu's
#
# List all disk drives.  for loop output looks like: i=/dev/dsk/c0d0s2
```

```
for i in `ls -1 /dev/dsk/*s2`
do
        correct_lu=`lssf ${i} | cut -d " " -f 3`
        correct_hw=`ioscan   -C disk -l ${correct_lu} |grep disk|cut -d " " -f 1`
        echo "/bin/echo \"/etc/insf -H ${correct_hw} -l ${correct_lu}\"" >> ${bt_dir}/rebuild.out
        echo "/etc/insf -H ${correct_hw} -l ${correct_lu}" >> ${bt_dir}/rebuild.out
done

#
# Force and ioinit update to kernel
#
echo "/bin/echo \"/etc/ioinit -f\"" >> ${bt_dir}/rebuild.out
echo "/etc/ioinit -f" >> ${bt_dir}/rebuild.out
#
#
# Determine Root Disk  output looks like: c0d0s2
# Determine Root Volume  output looks like: /dev/vg00/lvol1
# Determine Primary Swap Volume output looks like: /dev/vg00/lvol2
#
boot_dsk=`lvlnboot -v | grep "Root:" | cut -d / -f 4`
root_vol=`bdf | grep /$ | cut -d " " -f 1`
ps_swap=`swapinfo | grep dev | head -n 1 | cut -c 58-78`
#
# null out variables
#
i=;j=;q=;disk=
#
# save volume group information to ${bt_dir}/tarball/etc/lvmconf
#
# List all volume groups.  for loop output looks like: i=vg00
for i in `vgdisplay -v | grep "VG Name" | cut -c 29-39`
do
        vgcfgbackup -f ${bt_dir}/tarball/etc/lvmconf/${i}.conf /dev/${i}
#
# Restore volume group information on all LVM disks during rebuild
# List all disks that have LVM on them.  For loop output looks like: j=/c0d0s2
#
    for j in `vgdisplay -v /dev/${i} | grep dsk | cut -c 35-42`
        do
                echo "/bin/echo \"/etc/vgcfgrestore -n  /dev/${i} /dev/rdsk${j}\"" >>
${bt_dir}/rebuild.out
                echo "/etc/vgcfgrestore -n  /dev/${i} /dev/rdsk${j}" >> ${bt_dir}/rebuild.out
        done
        echo "/bin/echo \"/bin/rm -rf /dev/${i} 2>&1\"" >> ${bt_dir}/rebuild.out
    echo "/bin/rm -rf /dev/${i} 2>&1" >> ${bt_dir}/rebuild.out
    echo "/bin/echo \"/bin/mkdir /dev/${i} 2>&1\"" >> ${bt_dir}/rebuild.out
    echo "/bin/mkdir /dev/${i} 2>&1" >> ${bt_dir}/rebuild.out
```

```
      echo "/bin/echo \"/etc/mknod /dev/${i}/group c `/bin/ll /dev/${i}/group | cut -c 35-45` 2>&1\""
>> ${bt_dir}/rebuild.out
          echo "/etc/mknod /dev/${i}/group c `/bin/ll /dev/${i}/group | cut -c 35-45` 2>&1" >>
${bt_dir}/rebuild.out
#
# import volume groups to create /etc/lvmtab and device volume groups
# string together the disk drives in one variable to the vgimport
# For loop output looks like: q=/dev/dsk/c8d0s2 /dev/dsk/c9d0s2
#
          for q in `vgdisplay -v /dev/${i} | grep dsk |cut -c 27-43`
          do
                    disk="${disk} ${q}"
          done
          echo "/bin/echo \"/etc/vgimport -v /dev/$i ${disk}\"" >> ${bt_dir}/rebuild.out
          echo "/etc/vgimport -v /dev/$i ${disk}" >> ${bt_dir}/rebuild.out

          echo "/bin/echo \"/etc/vgchange -a y /dev/$i\"" >> ${bt_dir}/rebuild.out
          echo "/etc/vgchange -a y /dev/$i" >> ${bt_dir}/rebuild.out
          disk=
done

#
# Install boot program on root disk
#
echo "/bin/echo \"/etc/mkboot /dev/rdsk/${boot_dsk} 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/etc/mkboot /dev/rdsk/${boot_dsk} 2>&1" >> ${bt_dir}/rebuild.out

#
# Update the root LVM area
#
echo "/bin/echo \"/etc/lvlnboot -r  ${root_vol} 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/etc/lvlnboot -r ${root_vol} 2>&1" >> ${bt_dir}/rebuild.out

echo "/bin/echo \"/etc/lvlnboot -s ${ps_swap} 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/etc/lvlnboot -s ${ps_swap} 2>&1" >> ${bt_dir}/rebuild.out

#
# Force update to /etc/mnttab
#
echo "/bin/echo \"/etc/mount -u 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/etc/mount -u 2>&1" >> ${bt_dir}/rebuild.out
#
# Rebuild Filesystem With newfs
# List of all logical volumes.  For loop output looks like: vol_name=/dev/vg00/lvol1
#
for vol_name in `bdf | grep \% | cut -d " " -f 1`
do
#
```

```
# List all disks that have LVM on them.  For loop output looks like: disk_name=c0d0s2
#
        for disk_name in `vgdisplay -v | grep "PV Name" | cut -c 36-45`
        do
                pvdisplay -v /dev/dsk/${disk_name} | grep "current  ${vol_name} " >/dev/null
2>&1
                if [ "$?" -eq 0 ]
                then
                        target_vol=`bdf | grep "${vol_name} " | cut -c 6-19 | sed 's/\//\/r/'`
                        disk_type=`diskinfo /dev/rdsk/${disk_name} | grep id | cut -c 22-34`
                        if [ "${vol_name}" != "${root_vol}" ]
                        then
                                echo "/bin/echo \"/etc/newfs -L /dev/${target_vol} ${disk_type}
2>&1\"" >> ${bt_dir}/rebuild.out
                                echo "/etc/newfs -L /dev/${target_vol} ${disk_type} 2>&1" >>
${bt_dir}/rebuild.out
                        fi
                        break
                fi
        done
done

#
# Make mount points
# List all mounted filesystems on the system.  For loop output looks like: q=/usr
#
for q in `df | cut -d " " -f 1`
do
        if [ "${q}" != "/" ]
        then
                echo "/bin/echo \"/bin/mkdir -p ${q} 2>&1\"" >> ${bt_dir}/rebuild.out
                echo "/bin/mkdir -p ${q} 2>&1" >> ${bt_dir}/rebuild.out
        fi
done

#
# Clean up /usr before mounting
#
echo "/bin/echo \"/bin/rm -rf /usr/bin 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/bin/rm -rf /usr/bin 2>&1" >> ${bt_dir}/rebuild.out
echo "/bin/echo \"/bin/rm -rf /usr/lib 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/bin/rm -rf /usr/lib 2>&1" >> ${bt_dir}/rebuild.out

#
# Mount all filesystem
#
echo "/bin/echo \"/etc/mount -a 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/etc/mount -a 2>&1" >> ${bt_dir}/rebuild.out
```

```
#
# Begin restore of data in the third image on this tape.
#
# Rewind the tape
echo "/bin/echo \"/bin/mt -t /dev/rmt/0m rew 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/bin/mt -t /dev/rmt/0m rew 2>&1" >> ${bt_dir}/rebuild.out

#
# Skip the first two images
#
echo "/bin/echo \"/bin/mt -t /dev/rmt/0mn fsf 2 2>&1\"" >>  ${bt_dir}/rebuild.out
echo "/bin/mt -t /dev/rmt/0mn fsf 2 2>&1" >> ${bt_dir}/rebuild.out

#
# User frecover to recover the data
#
echo "/bin/echo \"/bin/dd if=/dev/rmt/0mn bs=10k | /etc/frecover -vr -o -f - 2>&1\"" >>
${bt_dir}/rebuild.out
echo "/bin/dd if=/dev/rmt/0mn bs=10k | /etc/frecover -vr -o -f - 2>&1" >> ${bt_dir}/rebuild.out
#
# Recovery should be done.  Reboot the system
#
echo "/bin/echo \"/etc/reboot -s 2>&1\"" >> ${bt_dir}/rebuild.out
echo "/etc/reboot -s 2>&1" >> ${bt_dir}/rebuild.out
lvminst example
==========================================================================
/bin/echo "/etc/rmsf -a /dev/dsk/*"
/etc/rmsf -a /dev/dsk/*
/bin/echo "cd /dev"
cd /dev
/bin/echo "/etc/insf -H 56/40.1.0 -l 0"
/etc/insf -H 56/40.1.0 -l 0
/bin/echo "/etc/insf -H 56/52.2.0 -l 1"
/etc/insf -H 56/52.2.0 -l 1
/bin/echo "/etc/insf -H 56/40.2.0 -l 2"
/etc/insf -H 56/40.2.0 -l 2
/bin/echo "/etc/insf -H 56/40.3.0 -l 3"
/etc/insf -H 56/40.3.0 -l 3
/bin/echo "/etc/insf -H 56/40.4.0 -l 4"
/etc/insf -H 56/40.4.0 -l 4
/bin/echo "/etc/insf -H 56/40.5.0 -l 5"
/etc/insf -H 56/40.5.0 -l 5
/bin/echo "/etc/insf -H 56/40.14.0 -l 6"
/etc/insf -H 56/40.14.0 -l 6
/bin/echo "/etc/insf -H 56/40.15.0 -l 7"
/etc/insf -H 56/40.15.0 -l 7
/bin/echo "/etc/insf -H 56/52.5.0 -l 8"
```

```
/etc/insf -H 56/52.5.0 -l 8
/bin/echo "/etc/insf -H 56/52.6.0 -l 9"
/etc/insf -H 56/52.6.0 -l 9
/bin/echo "/etc/ioinit -f"
/etc/ioinit -f
/bin/echo "/etc/vgcfgrestore -n  /dev/vg00 /dev/rdsk/c0d0s2"
/etc/vgcfgrestore -n  /dev/vg00 /dev/rdsk/c0d0s2
/bin/echo "/bin/rm -rf /dev/vg00 2>&1"
/bin/rm -rf /dev/vg00 2>&1
/bin/echo "/bin/mkdir /dev/vg00 2>&1"
/bin/mkdir /dev/vg00 2>&1
/bin/echo "/etc/mknod /dev/vg00/group c 64 0x000000 2>&1"
/etc/mknod /dev/vg00/group c 64 0x000000 2>&1
/bin/echo "/etc/vgimport -v /dev/vg00  /dev/dsk/c0d0s2"
/etc/vgimport -v /dev/vg00  /dev/dsk/c0d0s2
/bin/echo "/etc/vgchange -a y /dev/vg00"
/etc/vgchange -a y /dev/vg00
/bin/echo "/etc/vgcfgrestore -n  /dev/vg01 /dev/rdsk/c2d0s2"
/etc/vgcfgrestore -n  /dev/vg01 /dev/rdsk/c2d0s2
/bin/echo "/bin/rm -rf /dev/vg01 2>&1"
/bin/rm -rf /dev/vg01 2>&1
/bin/echo "/bin/mkdir /dev/vg01 2>&1"
/bin/mkdir /dev/vg01 2>&1
/bin/echo "/etc/mknod /dev/vg01/group c 64 0x010000 2>&1"
/etc/mknod /dev/vg01/group c 64 0x010000 2>&1
/bin/echo "/etc/vgimport -v /dev/vg01  /dev/dsk/c2d0s2"
/etc/vgimport -v /dev/vg01  /dev/dsk/c2d0s2
/bin/echo "/etc/vgchange -a y /dev/vg01"
/etc/vgchange -a y /dev/vg01
/bin/echo "/etc/vgcfgrestore -n  /dev/vg02 /dev/rdsk/c3d0s2"
/etc/vgcfgrestore -n  /dev/vg02 /dev/rdsk/c3d0s2
/bin/echo "/bin/rm -rf /dev/vg02 2>&1"
/bin/rm -rf /dev/vg02 2>&1
/bin/echo "/bin/mkdir /dev/vg02 2>&1"
/bin/mkdir /dev/vg02 2>&1
/bin/echo "/etc/mknod /dev/vg02/group c 64 0x020000 2>&1"
/etc/mknod /dev/vg02/group c 64 0x020000 2>&1
/bin/echo "/etc/vgimport -v /dev/vg02  /dev/dsk/c3d0s2"
/etc/vgimport -v /dev/vg02  /dev/dsk/c3d0s2
/bin/echo "/etc/vgchange -a y /dev/vg02"
/etc/vgchange -a y /dev/vg02
/bin/echo "/etc/vgcfgrestore -n  /dev/vdog /dev/rdsk/c8d0s2"
/etc/vgcfgrestore -n  /dev/vdog /dev/rdsk/c8d0s2
/bin/echo "/etc/vgcfgrestore -n  /dev/vdog /dev/rdsk/c9d0s2"
/etc/vgcfgrestore -n  /dev/vdog /dev/rdsk/c9d0s2
/bin/echo "/bin/rm -rf /dev/vdog 2>&1"
/bin/rm -rf /dev/vdog 2>&1
/bin/echo "/bin/mkdir /dev/vdog 2>&1"
```

```
/bin/mkdir /dev/vdog 2>&1
/bin/echo "/etc/mknod /dev/vdog/group c 64 0x030000 2>&1"
/etc/mknod /dev/vdog/group c 64 0x030000 2>&1
/bin/echo "/etc/vgimport -v /dev/vdog  /dev/dsk/c8d0s2 /dev/dsk/c9d0s2"
/etc/vgimport -v /dev/vdog  /dev/dsk/c8d0s2 /dev/dsk/c9d0s2
/bin/echo "/etc/vgchange -a y /dev/vdog"
/etc/vgchange -a y /dev/vdog
/bin/echo "/etc/mkboot /dev/rdsk/c0d0s2 2>&1"
/etc/mkboot /dev/rdsk/c0d0s2 2>&1
/bin/echo "/etc/lvlnboot -r  /dev/vg00/lvol1 2>&1"
/etc/lvlnboot -r /dev/vg00/lvol1 2>&1
/bin/echo "/etc/lvlnboot -s /dev/vg00/lvol2 2>&1"
/etc/lvlnboot -s /dev/vg00/lvol2 2>&1
/bin/echo "/etc/mount -u 2>&1"
/etc/mount -u 2>&1
/bin/echo "/etc/newfs -L /dev/vg02/rlvol2     DSP3107LSW    2>&1"
/etc/newfs -L /dev/vg02/rlvol2     DSP3107LSW    2>&1
/bin/echo "/etc/newfs -L /dev/vg02/rlvol1     DSP3107LSW    2>&1"
/etc/newfs -L /dev/vg02/rlvol1     DSP3107LSW    2>&1
/bin/echo "/etc/newfs -L /dev/vdog/rlvol4     C2247M1       2>&1"
/etc/newfs -L /dev/vdog/rlvol4     C2247M1       2>&1
/bin/echo "/etc/newfs -L /dev/vdog/rlvol3     C2247M1       2>&1"
/etc/newfs -L /dev/vdog/rlvol3     C2247M1       2>&1
/bin/echo "/etc/newfs -L /dev/vdog/rlvol2     C2247M1       2>&1"
/etc/newfs -L /dev/vdog/rlvol2     C2247M1       2>&1
/bin/echo "/etc/newfs -L /dev/vdog/rlvol1     C2247M1       2>&1"
/etc/newfs -L /dev/vdog/rlvol1     C2247M1       2>&1
/bin/echo "/etc/newfs -L /dev/vg01/rlvol3     DSP3107LSW    2>&1"
/etc/newfs -L /dev/vg01/rlvol3     DSP3107LSW    2>&1
/bin/echo "/etc/newfs -L /dev/vg01/rlvol2     DSP3107LSW    2>&1"
/etc/newfs -L /dev/vg01/rlvol2     DSP3107LSW    2>&1
/bin/echo "/etc/newfs -L /dev/vg01/rlvol1     DSP3107LSW    2>&1"
/etc/newfs -L /dev/vg01/rlvol1     DSP3107LSW    2>&1
/bin/echo "/etc/newfs -L /dev/vg00/rlvol3     DSP3107LSW    2>&1"
/etc/newfs -L /dev/vg00/rlvol3     DSP3107LSW    2>&1
/bin/echo "/bin/mkdir -p /usr 2>&1"
/bin/mkdir -p /usr 2>&1
/bin/echo "/bin/mkdir -p /fs11 2>&1"
/bin/mkdir -p /fs11 2>&1
/bin/echo "/bin/mkdir -p /fs12 2>&1"
/bin/mkdir -p /fs12 2>&1
/bin/echo "/bin/mkdir -p /fs13 2>&1"
/bin/mkdir -p /fs13 2>&1
/bin/echo "/bin/mkdir -p /fsvdog1 2>&1"
/bin/mkdir -p /fsvdog1 2>&1
/bin/echo "/bin/mkdir -p /fsvdog2 2>&1"
/bin/mkdir -p /fsvdog2 2>&1
/bin/echo "/bin/mkdir -p /fsvdog3 2>&1"
```

```
/bin/mkdir -p /fsvdog3 2>&1
/bin/echo "/bin/mkdir -p /fsvdog4 2>&1"
/bin/mkdir -p /fsvdog4 2>&1
/bin/echo "/bin/mkdir -p /fs21 2>&1"
/bin/mkdir -p /fs21 2>&1
/bin/echo "/bin/mkdir -p /fs22 2>&1"
/bin/mkdir -p /fs22 2>&1
/bin/echo "/bin/rm -rf /usr/bin 2>&1"
/bin/rm -rf /usr/bin 2>&1
/bin/echo "/bin/rm -rf /usr/lib 2>&1"
/bin/rm -rf /usr/lib 2>&1
/bin/echo "/etc/mount -a 2>&1"
/etc/mount -a 2>&1
/bin/echo "/bin/mt -t /dev/rmt/0m rew 2>&1"
/bin/mt -t /dev/rmt/0m rew 2>&1
/bin/echo "/bin/mt -t /dev/rmt/0mn fsf 2 2>&1"
/bin/mt -t /dev/rmt/0mn fsf 2 2>&1
/bin/echo "/bin/dd if=/dev/rmt/0mn bs=10k | /etc/frecover -vr -o -f - 2>&1"
/bin/dd if=/dev/rmt/0mn bs=10k | /etc/frecover -vr -o -f - 2>&1
/bin/echo "shelling out 2>&1"
/bin/sh 2>&1
/bin/echo "/etc/reboot -s 2>&1"
/etc/reboot -s 2>&1
```

APPENDIX A

sizes.example

```
================================================================
ROOT_SIZE=120
SWAP_SIZE=136
```

APPENDIX B
Console output from 9.x BBT recovery