STORAGETEK®

APPLICATION NOTE

Using Shared Virtual Array® (SVA™) disk systems and SnapShot software with NetWorker on HP-UX

APRIL 2004

HARDWARE AND SOFTWARE APPLICATIONS USED

	Name	Vendor	Comments
Operation system	HP-UX 11.0	HP	
SVA software	SVAA 3.1.0 for HP-UX	StorageTek	
SVA hardware	SVA V960	StorageTek	
Hardware	HP L1000 HBA A5158A	HP	HP PCI Tachyon TL Fibre

1 ABSTRACT

The V-Series Shared Virtual Array® (SVA™) disk system has a feature, called SnapShot software, which allows you to make instantaneous copies of any single virtual disk defined within the disk subsystem. In order to take advantage of the benefits of this feature, it is necessary to integrate the SnapShot software operational steps with the common Relational Database Management System (RDBMS) used on the host systems. An RDBMS is a program that lets you create, update and administer a relational database.

Today the Open Systems market is leading the disk storage market and all the integration efforts remain to be done in this arena. This paper intends to present a simple integration of SnapShot software with LEGATO NetWorker software. The tests performed here are based on an HP environment.

We assume the reader has a good knowledge of UNIX disk administration, LEGATO NetWorker software, SVA principles and SVA administration.

The last part of this document describes the different tests of backup with SVA and NetWorker.

All the scripts, prerequisites, constraints are also detailed in each test.

2 UNDERSTANDING SNAPSHOT SOFTWARE

SnapShot software is a powerful data duplication product that harnesses the unique virtual disk architecture of the V960 Shared Virtual Array® (SVA™). SnapShot duplicates entire Open Systems Logical Units (LUNs) in seconds, using zero additional physical capacity.

SnapShot software's virtually instantaneous copying speed is accomplished by manipulating pointers to data within the virtual disk subsystem, instead of physically moving it. With no physical data movement required, SnapShot software creates copies of LUNs in approximately two to five seconds, no matter how much data is copied. SnapShot software also has the unique ability to create any number of copies needed or desired by the user. By eliminating the use of resources such as Central Processing Unit (CPU) cycles, channel utilization, I/O operations and, most importantly, time. SnapShot software creates a new paradigm in data duplication. Traditional data duplication products require physical data movement, which is expensive in terms of resources used and precious time.

How does it work?

SnapShot software simply updates pointers within the virtual disk subsystem mapping tables for the data views being duplicated at electronic memory speeds. These updated pointers reference to the same disk array locations as the original source data — one physical copy of the data with multiple host "views" **(Figure 1).**

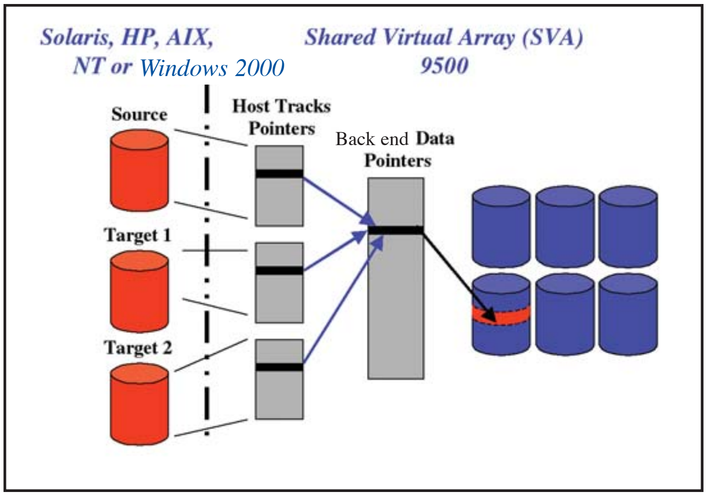

Figure 1. SnapShot mechanism.

SnapShot software copies use no disk storage initially since only one physical copy of the data exists at the point in time when the snap is created. All copies and the original copy are totally independent, and can be updated changed or read without affecting one another. Only changes to either the source data or copies use additional physical storage, and any parts of the data that remain common are shared. The result of a SnapShot copy is the same as that of a traditional copy, the main difference is that the time required to create the copy with SnapShot software is measured in mere seconds versus traditional physical duplication products measured in minutes or hours.

3 ARCHITECTURE EXAMPLE

The aim of this section is to present the test environment for the SVA using the NetWorker software.

The figure below illustrates the architecture used as an example during this study. This basic architecture intends to reproduce a typical schema with data production server and backup server:

> The production server (HP C200 AGADIR) uses two LUNs of the SVA to store the data files and the SnapShot copy. It has a Fibre Channel connection (Emulex LP8000) to the subsystem on the domain 2. It is configured as NetWorker client.

> The backup server (HP L1000 LHERS) uses one LUN (the SnapShot copy) to back up the data files. It has connections both to the disk system and to the tape drive. It is configured as NetWorker server.

> A library is shared across ASCSL software installed on the server LHERS.

> Both servers are on the same LAN.

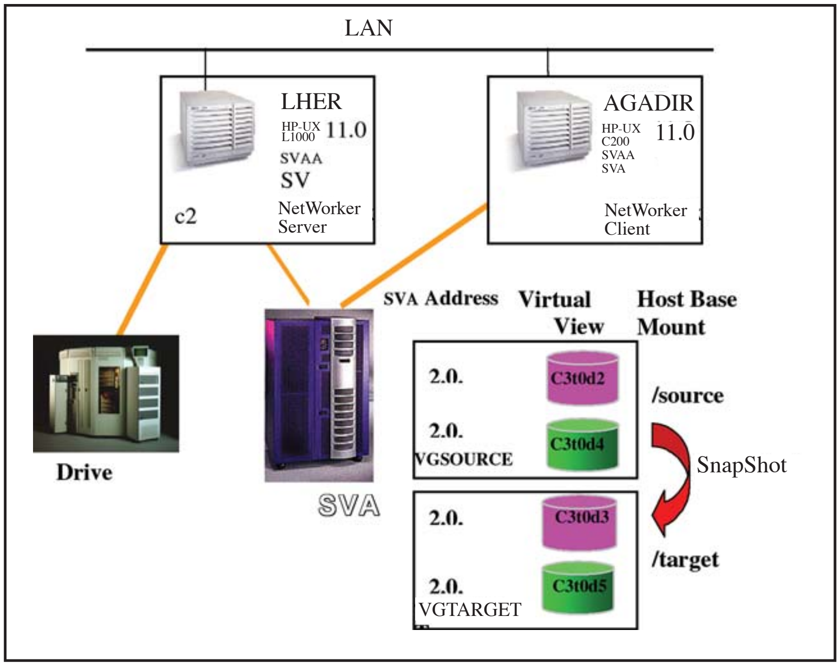

Figure 2. General architecture.

4 CONFIGURATION STEPS

4.1 SVA CONFIGURATION

> Configure the SVA on the NetWorker client.

> Define a functional device using SVAA.

> Configure two devices for two volumes groups on SVA: one volume group for the customer data and one for SnapShot both on the NetWorker client and NetWorker server.

> Configure the NetWorker software.

> Configure the NetWorker server.

> Configure the NetWorker client.

Note Here the target is always defined on the SVA, but this device does not use physical capacity of the SVA. After the backup, a release command is performed on the target volume to remove the capacity of potential data updates after the SnapShot. The SCSI partition release facility informs the SVA subsystem that a given SCSI partition no longer contains any useful data. So the subsystem can release the space used by the partition. Thus, partition release enables you to take better advantage of an SVA's extended capacity option to lower overall storage costs and, at the same time, to securely erase unwanted data.

4.1.1 Define functional SCSI LUNs devices with SVAA

In this section we define single LUNs, but you can use larger LUNs. The following commands are mentioned here to show all the configuration steps.

> Declaration of SCSI LUNs

```
#sibadmin
SIB> qsubsys
SVAA Server: agadir           Date/Time: 07-01-2002 10:56:13 GMT

ECLIPSE
SIB> defdev -subsys ECLIPSE -fdid 0D0 -devtyp SCSIB -scsiaddr 2.0.2 -scsiblks 4096 -
scsirw yes -name HP_src1
SIB9830I: Device (FDID 0D0) successfully defined with an exact size of 7.4 GB.

SIB> defdev -subsys ECLIPSE -fdid 0D1 -devtyp SCSIB -scsiaddr 2.0.3 -scsiblks 4096 -
scsirw yes -name HP_trg1
SIB9830I: Device (FDID 0D1) successfully defined with an exact size of 7.4 GB.
SIB> defdev -subsys ECLIPSE -fdid 0D2 -devtyp SCSIB -scsiaddr 2.0.4 -scsiblks 4096 -
scsirw yes -name HP_src2
SIB9830I: Device (FDID 0D2) successfully defined with an exact size of 7.4 GB.

SIB> defdev -subsys ECLIPSE -fdid 0D3 -devtyp SCSIB -scsiaddr 2.0.5 -scsiblks 4096 -
scsirw yes -name HP_trg2
SIB9830I: Device (FDID 0D3) successfully defined with an exact size of 7.4 GB.
```

4.1.2 LUNs configuration on AGADIR (NetWorker client)

> Create the directory for the new device.

```
#insf
#ioscan –fnC disk
```

> Configure the SVA path.

Scan the new device for SVA Path:

```
#sppath –v
…..
SPD=3 c3t0d2 dev=188,0x030200 type=2 SANID="STK 9500 00000000003000D0"
SPD=4 c3t0d3 dev=188,0x030300 type=2 SANID="STK 9500 00000000003000D1"
SPD=5 c3t0d4 dev=188,0x030400 type=2 SANID="STK 9500 00000000003000D2"
SPD=6 c3t0d5 dev=188,0x030500 type=2 SANID="STK 9500 00000000003000D3"
```

Generate kernel configuration files:

```
#setsp -g
#setsp -S
#setsp
==============================================================================
 spd   Path/disk     Status Primary Exclude Buf Balance RtrCnt  RtrDly FailBack
==============================================================================
==============================================================================
  3    c3t0d2         Good     X       X     1000    1      20     3000      1
 spd3 = c3t0d2                                    ID = "STK 9500
00000000003000D0"
==============================================================================
  4   c3t0d3          Good     X       X     1000    1      20     3000      1
 spd4 = c3t0d3                                    ID = "STK 9500
00000000003000D1"
==============================================================================
  5    c3t0d4           Good      X        X      1000     1       20      3000       1
 spd5 = c3t0d4                                            ID = "STK 9500
0000000000003000D2"
===================================================================================
  6   c3t0d5            Good      X        X      1000     1       20      3000       1
 spd6 = c3t0d5                                            ID = "STK 9500
0000000000003000D3"
===================================================================================
```

The four new devices are:

> c3t0d2 and c3t0d4 for the customer data in the **vgsource** volume group.

> c3t0d3 and c3t0d5 for the snapshot devices in the **vgtarget** volume group.

Due to SVA path, new LUNs are excluded by default, then we must include those:

```
#setsp —e0 —l3
#setsp —e0 —l4
#setsp —e0 —l5
#setsp —e0 —l6
```

> Generate the volume group source.

> vgsource building.

```
# mkdir -p /dev/vgsource
# mknod /dev/vgsource/group c 64 0x060000
# pvcreate /dev/rdsk/c3t0d2
# pvcreate /dev/rdsk/c3t0d4
# vgcreate /dev/vgsource /dev/dsk/c3t0d2 /dev/dsk/c3t0d4
# lvcreate -L7000 -n source1  /dev/vgsource
# lvcreate -L4000 -n source2 /dev/vgsource
```

> Construct new files systems.

```
# newfs -F vxfs /dev/vgsource/rsource1
# newfs -F vxfs /dev/vgsource/rsource2
```

> Mount the source logical volumes.

```
# mkdir -p /source/source1
# mount -F vxfs /dev/vgsource/source1  /source/source1
# mkdir -p /source/source2
# mount -F vxfs /dev/vgsource/source2  /source/source2
```

> Generate the volume group target.

> vgtarget building.

```
# mkdir -p /dev/vgtarget
# mknod /dev/vgtarget/group c 64 0x070000
# pvcreate /dev/rdsk/c3t0d3
# pvcreate /dev/rdsk/c3t0d5
# vgcreate /dev/vgtarget /dev/dsk/c3t0d3 /dev/dsk/c3t0d5
```

4.1.3 LUNs configuration on LHERS (NetWorker server)

```
#insf
#ioscan –fnC disk
#sppath –v
#setsp -g
#setsp -S
#setsp
```

4.2 NETWORKER CONFIGURATION

The objective of this section is to explain the configuration required using the SVA SnapShot with NetWorker.

The first most important concept of NetWorker is the "Group" concept. A group represents a group of clients that needs to be backed up. A list of groups can be defined using the graphical interface. Then those groups are used to perform the backups. The NetWorker server will initiate groups that will be saved in the order that they appear on the *NetWorker Groups* configuration windows.

The client uses only one Networker group: **dummy_agadir-group —** the client (AGADIR) belongs to this group. It is important that the name of the client server was present in the composition of the group name. A group name must always be composed with the following elements:*dummy_nameofclient-group*

Launch the graphic interface on the NetWorker server:

```
# nwadmin &
```

> Create a client NetWorker group.
> Click on **Customize > Groups**… and finally on the **Create** button.
> Fill in the **Name** zone with the name of the client NetWorker group. In our example, we choose dummy_agadir-grp.
> Click now on the Apply button to validate.

> Create a NetWorker client .
> Click on **Clients > Client Setup**… then, on the **Create** button.
> Fill in the Name field with the client name. Here we take ***agadir***.
> Choose the NetWorker group by clicking on the NetWorker group name ***(dummy_agadir-grp)***.
> Complete the **Save Set** field with ***/dev/null*** and the **Backup command** field with ***savepnpc***.
> Click on the **Apply** button to validate.

4.2.1 NetWorker client (AGADIR)

The NetWorker client started with the **dummy_agadir-group** that uses a pre-processing shell ***dummy.sh***.

> Pre-processing steps:

1) Take a snapshot of the target device.
2) Exclude the target LUNs from the client view.
3) Launch the remote script shell *backup_agadir.sh* on the Networker server LHERS.

Note Do not forget to configure the /.rhosts and /etc/hosts.equiv files on LHERS.

4) Include each target LUN on the client.
5) Release the snapshot.

The NetWorker software uses the *dummy_agadir-group.res* file, located in the directory /nsr/res on AGADIR.

The save set to backup is required, so we can initialize the field with /dev/null.

Note All shell scripts must be present in the directory of LEGATO NetWorker binary files and their rights must be set to 700.

4.2.2 NetWorker server (LHERS)

The NetWorker client initiates the backup on the NetWorker server by remotely executing the *backup_agadir.sh* script. This shell script performs the following:

> Shell script's steps :

1) Get configuration files through ftp with *recover.sh* script shell.
2) Include each target LUN from the server view.
3) Generate the target volume group and mount the file system for the backup.
4) Save the mounted target file systems.
5) Dismount the file systems and remove the target volume group.
6) Exclude the target LUNs from the server view.

Note All shell scripts must be present in the directory of LEGATO NetWorker binary files and their rights must be set to 700.

5 EXECUTION PROCESS OVERVIEW

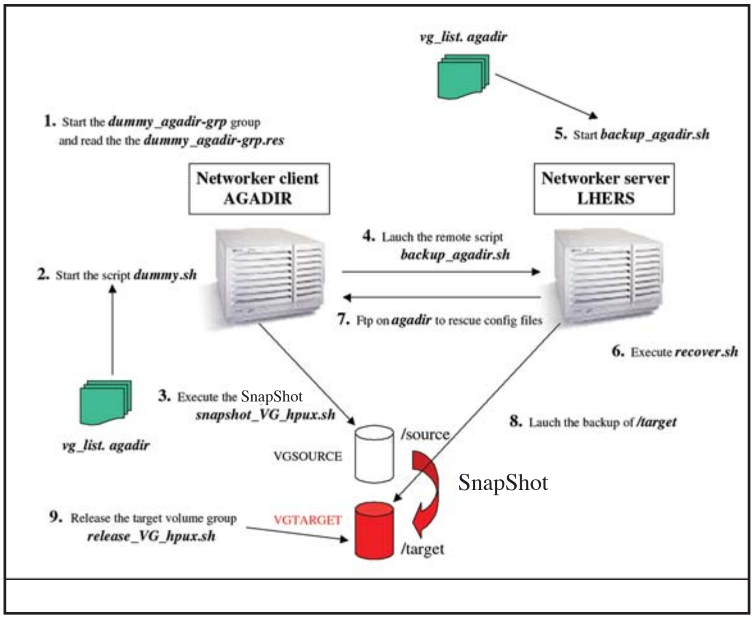
vg_list. agadir
1. Start the dummy_agadir-grp group
and read the the dummy_agadir-grp.res
5. Start backup_agadir.sh
Networker client
AGADIR
Networker server
LHERS
4. Lauch the remote script
backup_agadir.sh
2. Start the script dummy.sh
7. Ftp on agadir to rescue config files
6. Execute recover.sh
3. Execute the SnapShot
snapshot_VG_hpux.sh
8. Lauch the backup of /target
vg_list. agadir
/source
VGSOURCE
SnapShot
9. Release the target volume group
release_VG_hpux.sh
VGTARGET
/target

Figure 3. Execution overview.

Content of file **dummy_agadir-group.res**:

```
type: savepnpc;
precmd: dummy.sh;
timeout: "12:00:00";
```

6 CRITICAL ANALYSIS

Advantages	Drawbacks
Simple: there is only one SVA command per file system.	
The data can be backed up to tape using another server, which will mount the SVA snapshot.	
The target SVA devices do not need to be mounted on the server.	
All the other advantages of a cold backup remain.	

SnapShot software works by creating different "views" of data rather than copying the data itself. By eliminating the need to physically move data for duplication, SnapShot software creates copies, almost instantly. With SnapShot software's ultra-fast duplication capability, the backup operation is not on the critical path any more and could run in parallel with other processes.

7 APPENDICES

This section contains a copy of each script used in this project.

APPENDIX A: THE LUNS CONFIGURATION FILE

vg_list.agadir

```
#==========================================================
#
#       Volume group              Volume group                 Mounting point
#       source                    snapshot target              snapshot target
#       on Networker
#       client
#
#==========================================================
        vgsource                  vgtarget                     /target
```

APPENDIX B: THE BACKUP SCRIPTS

dummy.sh

```
#!/bin/sh
#
# This script is called by NetWorker across the savepncp script
# during the group backup

# Variables to initialise
SERVER=lhers
CLIENT=`hostname`
NETWORKER=/opt/networker/bin

# Initialise the contain of this variable with the pathname of the volumes group list file
# The contain of this file must be completed.
VG_LIST_FILE=/opt/networker/bin/vg_list.$CLIENT

# Existence of the volumes group list verification
if [ ! -f $VG_LIST_FILE ]
then
        print "VG_LIST_FILE not present"
        print "Check if file exists: $VG_LIST_FILE"
        exit 1
fi

# Config files building
mkdir -p /tmp/map
touch /tmp/devices_info
touch /tmp/mounts_info
touch /tmp/minors_info

egrep -v '^#|^$' $VG_LIST_FILE |
while read  VG_SOURCE VG_TARGET MOUNT_TARGET
do
        touch /tmp/map/$VG_SOURCE.map
        IND=1
        for LV in `vgdisplay -v $VG_SOURCE | grep "LV Name" | cut -d'/' -f4`
        do
                print $IND $LV >> /tmp/map/$VG_SOURCE.map
                IND=`expr $IND + 1`
        done
        for MPOINT in `df | grep /dev/$VG_SOURCE/$LV | awk '{print $1}'`
        do
                print /dev/$VG_SOURCE $LV $MPOINT `fstyp /dev/$VG_SOURCE/$LV` >>
/tmp/mounts_info
        done

        # Snapshot the target device
        snapshot_VG_hpux.sh /dev/$VG_SOURCE /dev/$VG_TARGET

        # Volume group minor number rescue
        ls -l /dev/$VG_TARGET | awk '{print $6}' | tail -n1 | cut -c3,4 | read MINOR
        print $VG_TARGET $MINOR >> /tmp/minors_info

        # Device id config file building
        for DEVICE in `vgdisplay -v $VG_TARGET | grep "PV Name" | cut -d'/' -f4`
        do
                setsp | grep "= "$DEVICE | awk '{print $8}' | cut -d'"' -f1 | read DEVICE_ID
                print $VG_TARGET $DEVICE_ID $DEVICE >> /tmp/devices_info
        done

        # Volume group desactivation
        vgchange -a n /dev/$VG_TARGET

        # Volume group target removing
        vgexport $VG_TARGET

        # All devices of the volume group exclusion
        egrep $VG_TARGET /tmp/devices_info |
        while read VG DEVICE_ID DEVICE
        do
                setsp -e1 -l`setsp | grep $DEVICE | sed 's/spd//g' | awk '{print
$1}' | tail -n1`
        done
done

# Start the backup_OSSAU.sh script on the Networker server
# savegrp -G backup_$CLIENT-grp
if [ $CLIENT = $SERVER ]
then
        $NETWORKER/backup_$CLIENT.sh
else
        remsh $SERVER $NETWORKER/backup_$CLIENT.sh
fi


egrep -v '^#|^$' $VG_LIST_FILE |
while read VG_SOURCE VG_TARGET MOUNT_TARGET
do
        # All devices of the volume group inclusion
        egrep $VG_TARGET /tmp/devices_info |
        while read VG DEVICE_ID DEVICE
        do
                print "inclusion ds dummy"
                setsp -e0 -l`setsp | grep $DEVICE | sed 's/spd//g' | awk '{print
$1}' | tail -n1`
        done
        # VG import
        mkdir -p /dev/$VG_TARGET
        egrep $VG_TARGET /tmp/minors_info |
        while read VG MINOR
        do
                mknod /dev/$VG_TARGET/group c 64 "0x"$MINOR"0000"
        done

        DEVICE_LIST=""
        egrep $VG_TARGET /tmp/devices_info |
        while read VG DEVICE_ID DEVICE
        do
                DEVICE_LIST=$DEVICE_LIST" /dev/dsk/"$DEVICE
        done
        vgimport -m /tmp/map/$VG_SOURCE.map /dev/$VG_TARGET $DEVICE_LIST

        # Volume group reactivation
        vgchange -a y /dev/$VG_TARGET

        # Volume group release
        release_VG_hpux.sh /dev/$VG_TARGET
done

# Files cleaning
rm /tmp/devices_info
rm /tmp/mounts_info
rm /tmp/minors_info
rm -r /tmp/map

print "End of the script."

exit 0
```

backup_agadir.sh

```
#!/bin/sh

CL=`print $0 | cut -d'_' -f2 | cut -d'.' -f1`

# Variables to initialise
VG_LIST_FILE=/opt/networker/bin/vg_list.$CL
ROOT=root_password_on_agadir
NETWORKER_BIN=/opt/networker/bin

if [ ! -f $VG_LIST_FILE ]
then
        print "VG_LIST_FILE not present"
        print "Check if file exists: $VG_LIST_FILE"
        exit 1
fi

HOSTNAME=`hostname`
DIRECT=`pwd`


if [ $CL = $HOSTNAME ]
then
        egrep -v '^#|^$' $VG_LIST_FILE |
        while read VG_SOURCE VG_TARGET MOUNT_TARGET
        do
                # Devices inclusion and devices list building
                DEVICE_LIST=""
                egrep $VG_TARGET /tmp/devices_info |
                while read VG DEVICE_ID DEVICE
                do
                        /sbin/setsp -e0 -l`/sbin/setsp | grep $DEVICE | sed
's/spd//g' | awk '{print $1}' | tail -n1`
                        DEVICE_LIST=$DEVICE_LIST" /dev/dsk/"$DEVICE
                done

                # VG import
                mkdir -p /dev/$VG_TARGET
                egrep $VG_TARGET /tmp/minors_info |
                while read VG MINOR
                do
                        mknod /dev/$VG_TARGET/group c 64 "0x"$MINOR"0000"
                done

                vgimport -m /tmp/map/$VG_SOURCE.map /dev/$VG_TARGET $DEVICE_LIST

                # VG activation
                vgchange -a y $VG_TARGET

                # Mounting points building
                egrep /dev/$VG_SOURCE /tmp/mounts_info |
                while read VG LV MPOINT FSYS
                do
                        # Checking
                        fsck -F $FSYS /dev/$VG_TARGET/$LV
                        # Tree generation
                        mkdir -p $MOUNT_TARGET$MPOINT
                        # Mounting
                        mount -F $FSYS /dev/$VG_TARGET/$LV $MOUNT_TARGET$MPOINT
                done
                save -c $CL -s $HOSTNAME -i -v $MOUNT_TARGET/*
                # Umounting
                egrep /dev/$VG_SOURCE /tmp/mounts_info |
                while read VG LV MPOINT FSYS
                do
                        umount $MOUNT_TARGET$MPOINT
                done
                rm -r $MOUNT_TARGET
                mkdir -p $MOUNT_TARGET
                done
                # VG desactivation
                vgchange -a n $VG_TARGET

                # VG removing
                vgexport $VG_TARGET

                # exclusion
                egrep $VG_TARGET /tmp/devices_info |
                while read VG DEVICE_ID DEVICE
                do
                        /sbin/setsp -e1 -l`/sbin/setsp | grep $DEVICE | sed
's/spd//g' | awk '{print $1}' | tail -n1`
                done
        done
else
        mkdir -p /tmp/map
        cd /tmp/map
        # Getting the mounting points info file
        $NETWORKER_BIN/recover.sh $CL $ROOT
        cd $DIRECT
        # Getting the mounting points info file
        egrep -v '^#|^$' $VG_LIST_FILE |
        while read VG_SOURCE VG_TARGET MOUNT_TARGET
        do
                # Devices inclusion
                DEVICE_LIST=""
                touch /tmp/device.tmp
                egrep $VG_TARGET /tmp/devices_info |
                while read VG DEVICE_ID DEVICE
                do
                        /sbin/setsp | grep $DEVICE_ID | awk '{print $3}' | read DEV
                        print $VG_TARGET $DEV >> /tmp/device.tmp
                        DEVICE_LIST=$DEVICE_LIST" /dev/dsk/"$DEV
                        /sbin/setsp -e0 -l`/sbin/setsp | grep $DEV | sed 's/spd//g'
| awk '{print $1}' | tail -n1`
                done
                        DEVICE_LIST=$DEVICE_LIST" /dev/dsk/"$DEV
                # VG import
                mkdir -p /dev/$VG_TARGET

                # VG Minor max extraction
                MAX=0
                for VG in `/sbin/vgdisplay | grep "VG Name" | cut -d'/' -f3`
                do
                        ls -l /dev/$VG | awk '{print $6}' | tail -n1 | cut -c3,4 | read
MINOR
                        if [ $MAX -lt $MINOR ]
                        then
                                MAX=$MINOR
                        fi
                done
                MINOR=`expr $MAX + 1`MINOR
                if [ $MINOR -le 9 ]
                then
                        MINOR=0$MINOR
                fi
                /sbin/mknod /dev/$VG_TARGET/group c 64 "0x"$MINOR"0000"

                /sbin/vgimport -m /tmp/map/$VG_SOURCE.map /dev/$VG_TARGET $DEVICE_LIST

                # VG activation
                /sbin/vgchange -a y $VG_TARGET

                # Mounting points building
                egrep /dev/$VG_SOURCE /tmp/mounts_info |
                while read VG LV MPOINT FSYS
                do
                        # Checking
                        /sbin/fsck -F $FSYS /dev/$VG_TARGET/$LV
                        # Tree generation
                        mkdir -p $MOUNT_TARGET$MPOINT
                        # Mounting
                        /sbin/mount -F $FSYS /dev/$VG_TARGET/$LV $MOUNT_TARGET$MPOINT
                done
                $NETWORKER_BIN/save -c $CL -s $HOSTNAME -i -v $MOUNT_TARGET/*
                # Umounting
                egrep /dev/$VG_SOURCE /tmp/mounts_info |
                while read VG LV MPOINT FSYS
                do
                        /sbin/umount $MOUNT_TARGET$MPOINT
                done
                rm -r $MOUNT_TARGET
                mkdir -p $MOUNT_TARGET

                # VG desactivation
                /sbin/vgchange -a n $VG_TARGET

                # VG removing
                /sbin/vgexport $VG_TARGET

                # exclusion
                egrep $VG_TARGET /tmp/device.tmp |
                while read VG DEV
                do
                        /sbin/setsp -e1 -l`/sbin/setsp | grep $DEV | sed 's/spd//g'
| awk '{print $1}' | tail -n1`
                done
                rm /tmp/device.tmp
        done
        # Config files cleaning
        rm -r /tmp/map
        rm /tmp/devices_info
        rm /tmp/mounts_info
fi


exit 0
```

recover.sh

```
#!/bin/sh

ftp -in $1 << -debut
        user root $2
        cd /tmp/map
        mget *.map
        cd ..
        get devices_info /tmp/devices_info
        get mounts_info /tmp/mounts_info
        bye
-debut
```

APPENDIX C: THE SNAPSHOT SCRIPT

snapshot_VG_hpux.sh

```
#!/bin/sh

# --- Important Note ---
#
```

```
# The volume group target must be existant and its
# physical structure must be identical than the volume group source !
#

# Synopsis of the snapshot_VG_hpux.sh script
synopsis()
{
        print "Synopsis"
        print "########"
        print ""
        print "snapshot_VG_hpux.sh Volume_Group_Name_Source Volume_Group_Name_Target
[mounting_point_target]"
        print ""
        print "Example : snapshot_VG_hpux.sh /dev/vg01 /dev/vg03
/mounting_point_target"
        print ""
        print "Important note : The both volumes group must have identical physical
structure !"
}

# Parameters verification
if [ $# -ne 3 ]
then
        if [ $# -ne  2 ]
        then
                synopsis
                exit 1
        else
                if [ `print $1 | cut -d'/' -f2` != "dev" ]
                then
                        synopsis
                        exit 1
                else
                        if [ `print $2 | cut -d'/' -f2` != "dev" ]
                        then
                                synopsis
                                exit 1
                        else
                                VG_SOURCE=$1
                                VG_TARGET=$2
                                VG_SOURCE_NAME=`print $VG_SOURCE | cut -d'/' -f3`
                                VG_TARGET_NAME=`print $VG_TARGET | cut -d'/' -f3`
                                # Mounting label
                                MOUNTING=1
                        fi
                fi
        fi
else
        if [ `print $1 | cut -d'/' -f2` != "dev" ]
        then
                synopsis
                exit 1
        else
                if [ `print $2 | cut -d'/' -f2` != "dev" ]
                then
                        synopsis
                        exit 1
                else
                        VG_SOURCE=$1
                        VG_TARGET=$2
                        VG_SOURCE_NAME=`print $VG_SOURCE | cut -d'/' -f3`
                        VG_TARGET_NAME=`print $VG_TARGET | cut -d'/' -f3`
                        MOUNTING=0
                        # Mounting point base for the snapshot target
                        MOUNTING_BASE=$3
                fi
        fi
fi

# The complete path where you can find the binary sibadmin
SIBADMIN_PATH="/opt/storagetek/SVAA3.1.0/bin"

# Don't initialize those variables
DEV_TARGET_LIST=""
DEV_SOURCE_LIST=""
LV_SOURCE_LIST=""

# Volumes Group existence verification
vgdisplay $VG_SOURCE > /dev/null 2> /dev/null
if [ $? -ne 0 ]
then
        print $VG_SOURCE" doesn't exist or is not activated !"
        exit 2
fi
vgdisplay $VG_TARGET > /dev/null 2> /dev/null
if [ $? -ne 0 ]
then
        print $VG_TARGET" doesn't exist or is not activated !"
        exit 2
#else
#       print $VG_TARGET" is actived, all the data on this volume group will be destroy !"
#       print "Do you want to continue the snapshot operation ? (default = y)[y/n]"
#       read REP
#       while [ $REP != y ] && [ $REP != Y ] && [ $REP != n ] && [ $REP != N ]
#       do
#               print "Do you want to continue the snapshot operation ? (default =
y)[y/n]"
#               read REP
#       done
#       if [ $REP = n ] || [ $REP = N ]
#       then
#               print "Snapshot operation aborted !"
#               exit 3
#       fi
fi

# Logical volumes of the volume group source extraction and map file generation
IND=1
touch /tmp/$VG_SOURCE_NAME.map
for LV in `vgdisplay -v $VG_SOURCE | grep "LV Name" | cut -d'/' -f4`
do
        LV_SOURCE_LIST=$LV_SOURCE_LIST$LV" "
        print $IND $LV >> /tmp/$VG_SOURCE_NAME.map
        IND=`expr $IND + 1`
done

# Devices lists extraction
for PV in `vgdisplay -v $VG_SOURCE | grep "PV Name" | cut -d'/' -f4`
```

```
do
        DEV_SOURCE_LIST=$DEV_SOURCE_LIST$PV" "
done

for PV in `vgdisplay -v $VG_TARGET | grep "PV Name" | cut -d'/' -f4`
do
        DEV_TARGET_LIST=$DEV_TARGET_LIST$PV" "
done

# Physical structure verification for both volumes group : the structures must be identical
!
# Number of physical devices
print $DEV_SOURCE_LIST | awk '{print NF}' | read NS
print $DEV_TARGET_LIST | awk '{print NF}' | read NT
if [ $NS -ne $NT ]
then
        print $VG_SOURCE" and "$VG_TARGET" haven't got identical physical structures !"
        exit 4
fi
# Total number of PE for each volume group
IND=1
for PE_S in `vgdisplay -v $VG_SOURCE | grep "Total PE" | awk '{print $3}'`
do
        vgdisplay -v $VG_TARGET | grep "Total PE" | awk '{print $3}' | head -n $IND
| tail -n1 | read PE_T
        IND=`expr $IND + 1`
        if [ $PE_S -ne $PE_T ]
        then
                print $VG_SOURCE" and "$VG_TARGET" haven't got identical physical
structures !"
                exit 4
        fi
done

# Volume group target minor number rescue
ls -l $VG_TARGET | awk '{print $6}' | tail -n1 | cut -c3,4 | read MINOR

# All logical volumes target desactivation
# All mounting points umounting
for MPOINT in `df | grep $VG_TARGET | awk '{print $1}'`
do
        umount $MPOINT
done
# Removing
for LV in `vgdisplay -v $VG_TARGET | grep "LV Name" | cut -d'/' -f4`
do
        lvremove -f $VG_TARGET/$LV
done

# Volume group target desactivation
vgchange -a n $VG_TARGET

# Volume group target removing
vgexport $VG_TARGET

# Snapshot
print "Snapshot ..."
IND=1
while [ $IND -le $NS ]
```

```
do
        SOURCE=`print $DEV_SOURCE_LIST | awk '{print $'$IND'}'`
        TARGET=`print $DEV_TARGET_LIST | awk '{print $'$IND'}'`
        print "$SIBADMIN_PATH/sibadmin snap -source /dev/rdsk/$SOURCE -target
/dev/rdsk/$TARGET"
        $SIBADMIN_PATH/sibadmin snap -source /dev/rdsk/$SOURCE -target
/dev/rdsk/$TARGET
        IND=`expr $IND + 1`
done

# Volume group target rebuilding
mkdir $VG_TARGET
mknod $VG_TARGET/group c 64 "0x"$MINOR"0000"

BDEV_TARGET_LIST=""
IND=1
while [ $IND -le $NS ]
do
        BDEV_TARGET_LIST=$BDEV_TARGET_LIST"/dev/dsk/"`print $DEV_TARGET_LIST | awk
'{print $'$IND'}'`" "
        IND=`expr $IND + 1`
done
vgimport -m /tmp/$VG_SOURCE_NAME.map $VG_TARGET $BDEV_TARGET_LIST
rm /tmp/$VG_SOURCE_NAME.map

# Reactivation of the volume group target
vgchange -a y $VG_TARGET

# Configuration of the volume group target backup
vgcfgbackup $VG_TARGET

if [ $MOUNTING -ne 1 ]
then
        # Directories tree target building and mounting
        mkdir -p $MOUNTING_BASE
        print > $MOUNTING_BASE/mount.log

        # Directories tree target building
        for LV in `vgdisplay -v $VG_TARGET | grep "LV Name" | cut -d'/' -f4`
        do
                # Checking
                fsck -F `fstyp $VG_TARGET/$LV` $VG_TARGET/$LV
                # Mounting point building
                for MPOINT in `df | grep $VG_SOURCE/$LV | awk '{print $1}'`
                do
                        mkdir -p $MOUNTING_BASE$MPOINT
                        mount -F `fstyp $VG_TARGET/$LV` $VG_TARGET/$LV
$MOUNTING_BASE$MPOINT
                        print "mount -F `fstyp $VG_TARGET/$LV` $VG_TARGET/$LV
$MOUNTING_BASE$MPOINT" >> $MOUNTING_BASE/mount.log
                done
        done
fi

exit 0
```

APPENDIX D: THE RELEASE SCRIPT

release_VG_hpux.sh

```
#!/bin/sh

# Synopsis of the release_VG_hpux.sh script
synopsis()
{
        print "Synopsis"
        print "########"
        print ""
        print "release_VG_hpux.sh Volume_Group_Name"
        print ""
        print "Examples : release_VG_hpux.sh /dev/vg01"
}

# The complete path where you can find the binary sibadmin
SIBADMIN_PATH="/opt/storagetek/SVAA3.1.0/bin"

# Don't initialize those variables
PV_LIST=""

# Parameter number verification
if [ $# -ne  1 ]
then
        synopsis
        exit 1
else
        if [ `print $1 | cut -d'/' -f2` != "dev" ]
        then
                synopsis
                exit 1
        else
                VOLUME_GROUP=$1
                VOLUME_GROUP_NAME=`print $VOLUME_GROUP | cut -d'/' -f3`
        fi
fi
# Volume Group existence verification
vgdisplay $VOLUME_GROUP > /dev/null 2> /dev/null
if [ $? -ne 0 ]
then
        print $VOLUME_GROUP" doesn't exist or is not activated !"
        exit 2
#else
#       print $VOLUME_GROUP " is actived, all the data on this volume group will be
destroy !"
#       print "Do you want to continue the release operation ? (default = y)[y/n]"
#       read REP
#       while [ $REP != y ] && [ $REP != Y ] && [ $REP != n ] && [ $REP != N ]
#       do
#               print "Do you want to continue the release operation ? (default =
y)[y/n]"
#               read REP
#       done
#       if [ $REP = n ] || [ $REP = N ]
#       then
#               print "Release operation aborted !"
#               exit 3
#       fi
```

```
fi

# Physical volumes of the volume group extraction
NPV=0
for PV in `vgdisplay -v $VOLUME_GROUP | grep "PV Name" | cut -d'/' -f4`
do
        PV_LIST=$PV_LIST$PV" "
        NPV=`expr $NPV + 1`
done

# Volume group minor number rescue
ls -l $VOLUME_GROUP | awk '{print $6}' | tail -n1 | cut -c3,4 | read MINOR

# All logical volumes desactivation
# All mounting points umounting
for MPOINT in `df | grep $VOLUME_GROUP | awk '{print $1}'`
do
        umount $MPOINT
done
# Removing
for LV in `vgdisplay -v $VOLUME_GROUP | grep "LV Name" | cut -d'/' -f4`
do
        lvremove -f $VOLUME_GROUP/$LV
done

# Volume group desactivation
vgchange -a n $VOLUME_GROUP

# Volume group removing
vgexport $VOLUME_GROUP

# Release
IND=1
while [ $IND -le $NPV ]
do
        DEV=`print $PV_LIST | awk '{print $'$IND'}'`
        $SIBADMIN_PATH/sibadmin release -force -path /dev/rdsk/$DEV
        IND=`expr $IND + 1`
done


# Volume group rebuilding
mkdir $VOLUME_GROUP
mknod $VOLUME_GROUP/group c 64 "0x"$MINOR"0000"

IND=1
while [ $IND -le $NPV ]
do
        DEV=`print $PV_LIST | awk '{print $'$IND'}'`
        pvcreate /dev/rdsk/$DEV
        IND=`expr $IND + 1`
done

BDEVICE_LIST=""
IND=1
while [ $IND -le $NPV ]
do
        DEV=`print $PV_LIST | awk '{print $'$IND'}'`
        BDEVICE_LIST=$BDEVICE_LIST"/dev/dsk/"$DEV" "
        IND=`expr $IND + 1`
done
vgcreate $VOLUME_GROUP $BDEVICE_LIST

# Reactivation of the volume group
vgchange -a y $VOLUME_GROUP

exit 0
```

STORAGETEK®

ABOUT STORAGETEK®

Storage Technology Corporation (NYSE: STK), a $2 billion worldwide company with headquarters in Louisville, CO, has been delivering a broad range of storage management solutions designed for IT professionals for over 30 years. StorageTek offers solutions that are easy to manage, integrate well with existing infrastructures and allow universal access to data across servers, media types and storage networks. StorageTek's practical and safe storage solutions for tape automation, disk storage systems and storage integration, coupled with a global services network, provide IT professionals with confidence and know-how to manage their entire storage management ecosystem today and in the future.

StorageTek products are available through a worldwide network. For more information, visit www.storagetek.com, or call 1.800.275.4785 or 01.303.673.2800.

WORLD HEADQUARTERS

Storage Technology Corporation
One StorageTek Drive
Louisville, Colorado 80028 USA
1.800.877.9220 or 01.303.673.5151