APPLICATION NOTE

Using Shared Virtual Array® (SVA™) disk systems and SnapShot software with Time Navigator for HP-UX

APRIL 2004

1 ABSTRACT 6

2 UNDERSTANDING SNAPSHOT SOFTWARE 6

2.1 HOW DOES SNAPSHOT WORK? 6

3 ARCHITECTURE EXAMPLE 7

3.1 SINGLE-SERVER BACKUP 7

3.2 PRODUCTION SERVERLESS BACKUP 8

4 CONFIGURATION STEPS 9

4.1 PRODUCTION SERVER CONFIGURATION 9

4.1.1 Single-server architecture 9

4.1.1.1 SVA configuration 9

4.1.1.2 Volume group configuration 13

4.1.2 Dedicated backup server architecture 14

4.1.2.1 SVA configuration 14

4.1.2.2 Volume group configuration 16

4.2 BACKUP SERVER CONFIGURATION 16

4.2.1 Tape configuration 16

4.2.2 Time navigator configuration 16

4.2.2.1 Creating hosts 17

4.2.2.2 Creating drives 17

4.2.2.3 Creating cartridge pools 18

4.2.2.4 Creating backup strategies 18

4.2.2.5 Creating backup classes 18

5 THE MAIN PROCESSES 19

5.1 SINGLE-SERVER ARCHITECTURE 19

5.1.1 Backup execution overview 19

5.1.2 Pre-/Post-processing scripts for Strategy A 19

5.1.3 Development overview 20

5.2 DEDICATED BACKUP SERVER ARCHITECTURE 22

5.2.1 Backup execution overview 22

5.2.2 Pre-/Post-processing scripts on the backup server 23

5.2.3 Pre-/Post-processing scripts on the production server 23

5.2.4 Development overview 23

6 HOW TO RESTORE 26

7 CRITICAL ANALYSIS 27

8 APPENDICES 28
APPENDIX A: LUN CONFIGURATION FILE (1 SERVER) 28
APPENDIX B: PRE-/POST-PROCESSING SCRIPTS (1 SERVER) 29
APPENDIX C: SNAP, MOUNT, UNMOUNT, RELEASE SCRIPTS (1 SERVER) 31
APPENDIX D: EXAMPLE OF SUCCESSFUL BACKUP (1 SERVER) 37
APPENDIX E: LUNS CONFIGURATION FILE (2 SERVERS) 38
APPENDIX F: PRE-/POST-PROCESSING SCRIPTS OF STRATEGY A ON TAILLON (2 SERVERS) 38
APPENDIX G: PRE-PROCESSING SCRIPT OF STRATEGY A ON LHERS (2 SERVERS) 41
APPENDIX H: PRE-PROCESSING SCRIPT OF STRATEGY B ON LHERS (2 SERVERS) 42
APPENDIX I: SNAP, MOUNT, UNMOUNT, RELEASE SCRIPTS (2 SERVERS) 43
APPENDIX J: EXAMPLE OF SUCCESSFUL BACKUP (2 SERVERS) 55

HARDWARE AND SOFTWARE APPLICATIONS USED

Name	Vendor	Comments	
ISV	Time Navigator (Quadratec)	Atempo	
Operation system	HP-UX 11.0	HP	
SVA software	SVAA 3.1.0 patch 12 SVAC 3.1.0 patch 12 (Windows) SVA PATH 3.3	StorageTek	 Optional* Optional**
SVA hardware	SVA 9200, 9500 and V960	StorageTek	Tested on SVA 9200
Backup software	TINA 3.5.0.1	Atempo	
Backup hardware	Quantum DLT drive DLT tape	Quantum	

* SVAC is the graphical interface to SVAA functions. The use of the SVAC is optional; the devices can be created using the CLI.

** SVA Path is the failover product for SVA. The scripts included in this document contain SVA Path command lines.

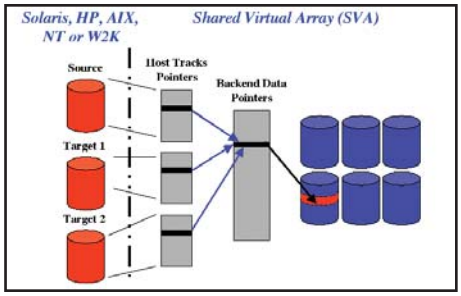


Figure 1. Snapshot mechanism.

1 ABSTRACT

The V-Series Shared Virtual Array® (SVA™) disk system features a great functionality, called SnapShot software. This SnapShot software allows the making of instantaneous copies of any single virtual disk defined within the disk subsystem. In order to take best advantage of all SnapShot software benefits, SnapShot operational steps must be integrated with the common Relational Database Management System (RDBMS) used on the host systems. RDBMS is a program that helps you create, update and administer a relational database.

Today the Open Systems market leads the disk storage market, and integration efforts remain to be made in this area. This paper intends to present a simple integration of SnapShot software with Atempo (Quadratec) Time Navigator software. The tests performed here are based on an HP-UX environment. For more information about the same scenario based on a Solaris platform, refer to the White Paper *Using SVA SnapShot with Time Navigator on Solaris* (November 2001).

This paper addresses those of you who have a good knowledge of UNIX disk administration, Atempo Time Navigator (TINA) software, SVA principles and SVA administration.

All scripts, prerequisites, and constraints are detailed for every test.

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

2 UNDERSTANDING SNAPSHOT SOFTWARE

SnapShot is a powerful data duplication software product that harnesses the unique virtual disk architecture of StorageTek's SVA. SnapShot software duplicates entire Open Systems logical units (LUN) in seconds, using almost no 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 LUN in approximately 2–5 seconds, no matter how much data is copied. SnapShot software also has the unique ability to create any number of copies that are 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 by reducing data duplication 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 and time.

2.1 HOW DOES SNAPSHOT WORK?

SnapShot software simply updates pointers within the virtual disk subsystem. Mapping tables for the data views are duplicated at electronic memory speed. Every updated pointer references the same disk array locations as the original source data—one physical copy of the data with multiple host "views" (**Figure 1**).

Initially, snapshot copies use no disk storage 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 each other. 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 being that the time required creating the copy with SnapShot software can be measured in seconds, whereas traditional physical duplication products take minutes or hours.

3 ARCHITECTURE EXAMPLE

The aim of this section is to present the test environment for the StorageTek SVA product and its use of the snapshot feature with the Atempo Time Navigator software. Two main types of architecture are considered:

> A simple SAN architecture with a single server serving as the production and backup server
> An architecture using an additional backup server to perform a "production server-free backup".

3.1 SINGLE-SERVER BACKUP

The figure below illustrates a simple backup configuration. In this configuration, the production server and the backup server are configured on the same machine. This section presents a backup scenario with no additional backup server. This section presents a second test environment for the StorageTek SVA product and the Atempo Time Navigator software.

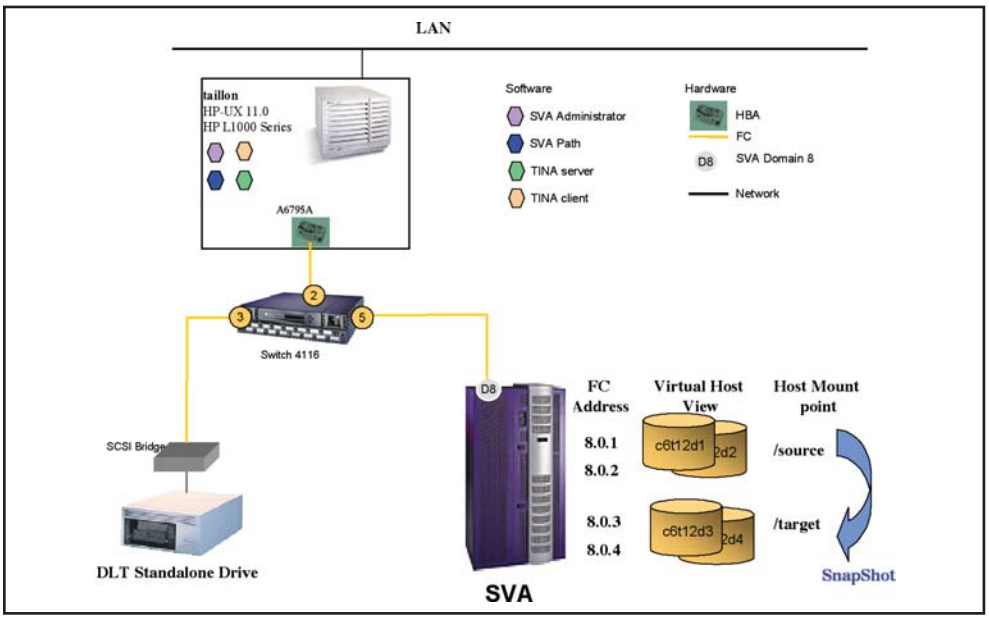


Figure 2. Single production / Backup server configuration.

> The production server named taillon (HP-UX) uses one volume group with two of the SVA LUN to store data files. Another volume group with the same characteristics is used to get a snapshot copy of the original data files. The server uses a Fibre Channel connection (HBA type: A6795A) to communicate with the SVA subsystem on domain 8. It is configured as a TINA local client.
> The same server acts as the backup server as well. It is connected to the tape drive via the Fibre Channel connection (HBA type: A6795A) and as such, it is also configured as the TINA server.
> The stand-alone tape drive uses a DLT tape. A SCSI bridge 3100 connects to the drive.

3.2 PRODUCTION SERVERLESS BACKUP

This section presents a second test environment for the StorageTek SVA product and the Atempo Time Navigator software.

The figure below illustrates the basic architecture of a typical setup with a data production server and a backup server:

> The production server named lhers (HP-UX) uses one volume group with two SVA LUN to store the data and another volume group to receive the snapshot copy. A Fibre Channel connection links the HP host to the SVA subsystem on domain 8 (HBA type A5158A).
> The backup server named taillon (HP-UX) has a fibre connection to the SVA on the same domain for access to the same LUN. The HBA is used to connect to the stand-alone tape drive (via a SCSI bridge 3100). The backup server is configured as a TINA Server.
> Both servers are on the same LAN.

Note **All software applications required to manage the StorageTek SVA product are installed on both servers.**

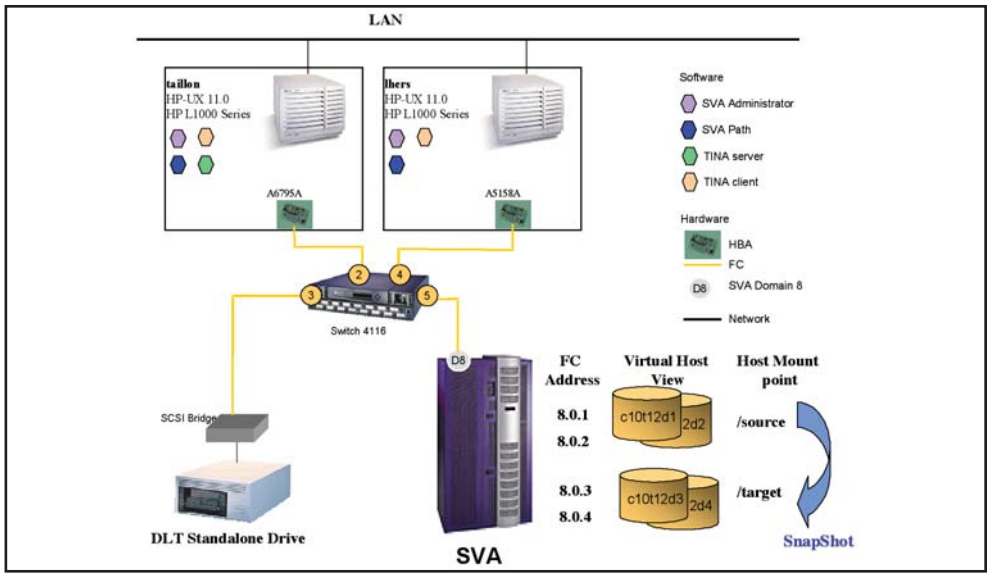


Figure 3. Dedicated backup server configuration.

4 CONFIGURATION STEPS

4.1 PRODUCTION SERVER CONFIGURATION

Two volume groups must be configured on the production server: one for customer data and one for snapshot. Each of these volume groups is composed of two SVA LUN. Here, we suppose that the snapshot target is physically present on the SVA yet it uses much less physical SVA capacity than that necessary for source data. After the backup, a release command is executed to clean the device and remove any data updates on the snapshot target.

The SCSI partition release facility informs an SVA subsystem that a given SCSI partition no longer contains useful data so the subsystem can release the related space. Thus, with partition release capabilities, you can take better advantage of SVA's extended capacity, reducing overall storage costs, while securely getting rid of unwanted data.

In this section, details are given on:

> SVA configuration
> Volume group configuration.

The production server configuration operations will be described considering two types of architecture:

> Single-server
> Dedicated backup server.

4.1.1 Single-server architecture

4.1.1.1 SVA configuration

In order to create two volume groups (source and target), four LUN are created. Another LUN will be created (the ECAM device) to connect to the SVA subsystem. This section details the operations necessary to:

> Create the ECAM device
> Define SVA LUN for volume groups.

4.1.1.1.1 Defining one functional SCSI LUN device with SVAA

> Declaration of SCSI LUN.

```
SIB> defdev -subsys ECLIPSE -fdid 18c -scsiaddr 8.0.0 -name ECAM -
fcapa 1g -devtyp SCSI
SIB9830I: Device (FDID 18c) successfully defined with an exact size of 2.5 GB.
```

You must specify the first LUN as an ECAM device to be able to connect to the subsystem from the server used.

> Make this LUN available for the system.

```
# insf
# ioscan –fnC disk
Class      I  H/W Path                    Driver S/W State   H/W Type     Description
==========================================================================================
...
disk       3  0/3/0/0.2.21.255.12.12.0  sdisk  CLAIMED     DEVICE       STK      9200
                                          /dev/dsk/c6t12d0   /dev/rdsk/c6t12d0
# sppath –v
…
SPD=0  c6t12d0 dev=188,0x06c000 type=2 SANID="STK 9200 000000000030018C"
# setsp -g
# setsp -S
# setsp
==========================================================================================
 spd   Path/disk           Status   Pri Exc Buf Balance RtrCnt  RtrDly FailBack
==========================================================================================
   0   c6t12d0            Excluded   X        1000     1       20      3000       1
 spd0 = c6t12d0                                    ID = "STK 9200 000000000030018C"
==========================================================================================
# setsp –l0 –e0
```

> Use the sibadmin CLI for Server and subsystem information.

```
# sibadmin
Starting CLI...
SIB> dserver
SVAA Server: TINADEMO           Date/Time: 11-22-2002 13:30:30 GMT

Version: 3.1
Description:
Trace Status: Disabled
Installed Features:
   SnapShot
Maintenance Level:
   0 PPFinfo PTF=L2P008T Patch=12 FIX=0 Issue=768241
Host:
   Name: taillon
   TCP/IP Port: 41248
Host OS Level:
     B.11.00
     U
```

If there is no subsystem attached to the server (or ECAM device), use the addsubsyspath command in the CLI. For more information on the sibadmin commands, type *help* or *help command*.

```
SIB> addsubsyspath -devpath /dev/rdsk/c6t12d0
SIB> qsubsys
SVAA Server: TINADEMO           Date/Time: 11-29-2002 13:10:34 GMT

ECLIPSE
```

4.1.1.1.2 LUNs configuration on the server named taillon (TINA client and server)

> Create devices for the source and target volume groups.

```
# sibadmin
Starting CLI...
SIB> defdev -subsys ECLIPSE -fdid 18d -scsiaddr 8.0.1 -name Source1 -fcapa 4g -devtyp
SCSI
SIB9830I: Device (FDID 18d) successfully defined with an exact size of 4.9 GB.
SIB> defdev -subsys ECLIPSE -fdid 18f -scsiaddr 8.0.2 -name Source2 -fcapa 4g -devtyp
SCSI
SIB9830I: Device (FDID 18f) successfully defined with an exact size of 4.9 GB.
SIB> defdev -subsys ECLIPSE -fdid 191 -scsiaddr 8.0.3 -name Target1 -fcapa 4g -devtyp
SCSI
SIB9830I: Device (FDID 191) successfully defined with an exact size of 4.9 GB.
SIB> defdev -subsys ECLIPSE -fdid 193 -scsiaddr 8.0.4 -name Target2 -fcapa 4g -devtyp
SCSI
SIB9830I: Device (FDID 193) successfully defined with an exact size of 4.9 GB.
```

You can either use the CLI or the SVAC to create the source and target LUN.

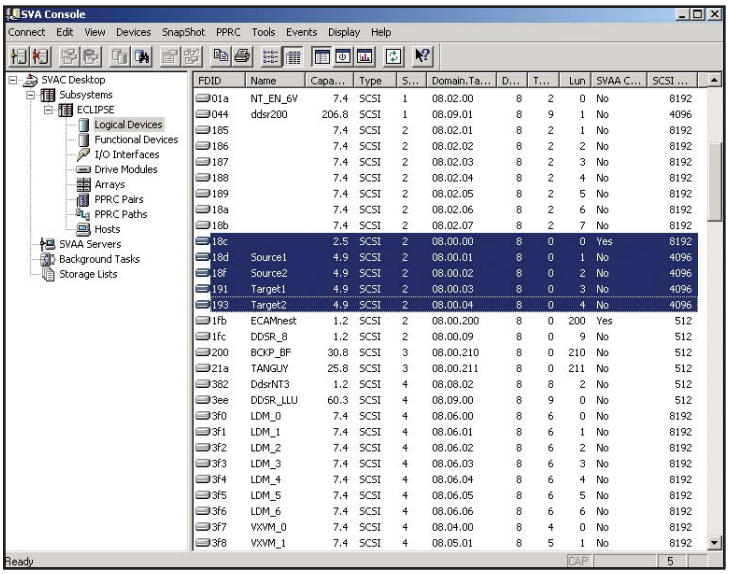


Figure 4. SVA console interface (connection with ECLIPSE).

To create a device, go to the menu Devices > Define New ... and complete the fields with the corresponding information.

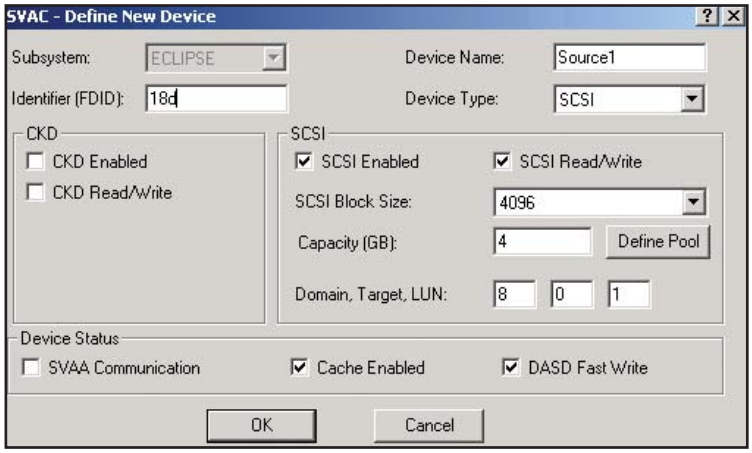


Figure 5. New device definition (SVAC interface).

Note **This section provides details on source and target volume group creation. The source volume group may be composed of several LUN. This volume will store the data that is used by the production server. The target volume group will obtain the snapshot copy of the source data. The SnapShot software operation can only be achieved if the target volume group is identical to the source volume group. This means that all device parameters in the source and target volume groups must match (size, block size, device type, etc.).**

> Create the device files for the new devices.

```
# insf
# ioscan -fnC disk
Class      I  H/W Path                    Driver S/W State   H/W Type     Description
===========================================================================
disk       3  0/3/0/0.2.21.255.12.12.0  sdisk   CLAIMED     DEVICE       STK      9200
                                          /dev/dsk/c6t12d0   /dev/rdsk/c6t12d0
disk       4  0/3/0/0.2.21.255.12.12.1  sdisk   CLAIMED     DEVICE       STK      9200
                                          /dev/dsk/c6t12d1   /dev/rdsk/c6t12d1
disk       5  0/3/0/0.2.21.255.12.12.2  sdisk   CLAIMED     DEVICE       STK      9200
                                          /dev/dsk/c6t12d2   /dev/rdsk/c6t12d2
disk       6  0/3/0/0.2.21.255.12.12.3  sdisk   CLAIMED     DEVICE       STK      9200
                                          /dev/dsk/c6t12d3   /dev/rdsk/c6t12d3
disk       7  0/3/0/0.2.21.255.12.12.4  sdisk   CLAIMED     DEVICE       STK      9200
                                         /dev/dsk/c6t12d4   /dev/rdsk/c6t12d4
```

> Configure the SVA path.

Scan the new device for SVA Path:

```
# sppath -v
...
SPD=0  c6t12d0 dev=188,0x06c000 type=2 SANID="STK 9200 0000000000030018C"
SPD=1  c6t12d1 dev=188,0x06c100 type=2 SANID="STK 9200 0000000000030018D"
SPD=2  c6t12d2 dev=188,0x06c200 type=2 SANID="STK 9200 0000000000030018F"
SPD=3  c6t12d3 dev=188,0x06c300 type=2 SANID="STK 9200 0000000000300191"
SPD=4  c6t12d4 dev=188,0x06c400 type=2 SANID="STK 9200 0000000000300193"
```

Generate kernel configuration files:

```
# setsp -g
# setsp -S
# setsp
================================================================================
 spd   Path/disk           Status   Pri Exc Buf Balance RtrCnt  RtrDly FailBack
================================================================================
   0   c6t12d0              Good      X        1000     1       20      3000       1
 spd0 = c6t12d0                                  ID = "STK 9200 0000000000030018C"
================================================================================
   1   c6t12d1             Excluded  X    X   1000     1       20      3000       1
 spd1 = c6t12d1                                  ID = "STK 9200 0000000000030018D"
================================================================================
   2   c6t12d2             Excluded  X    X   1000     1       20      3000       1
 spd2 = c6t12d2                                  ID = "STK 9200 0000000000030018F"
================================================================================
   3   c6t12d3             Excluded  X    X   1000     1       20      3000       1
 spd3 = c6t12d3                                  ID = "STK 9200 0000000000300191"
================================================================================
   4   c6t12d4             Excluded  X    X   1000     1       20      3000       1
 spd4 = c6t12d4                                  ID = "STK 9200 0000000000300193"
================================================================================
```

4.1.1.2 Volume group configuration

The 4 new devices are:

> c6t12d1 and c6t12d2 for the customer data in the vgsource volume group

> c6t12d3 and c6t12d4 for the snapshot devices in the vgtarget volume group.

Due to SVA path, every new LUN is excluded by default, which means that we must include them:

```
#setsp -e0 -l1
#setsp -e0 -l2
#setsp -e0 -l3
#setsp -e0 -l4
```

4.1.1.2.1 Volume group source building

> Generate the volume group source named vgsource.

```
# mkdir -p /dev/vgsource
# mknod /dev/vgsource/group c 64 0x060000
# pvcreate /dev/rdsk/c6t12d1
# pvcreate /dev/rdsk/c6t12d2
# vgcreate /dev/vgsource /dev/dsk/c6t12d1 /dev/dsk/c6t12d2
# lvcreate -i 2 -I 64 -L 600 /dev/vgsource
```

> Construct the new files systems.

```
# newfs -F vxfs /dev/vgsource/rlvol1
# vgchange -a y /dev/vgsource
```

> Mount the source logical volumes.

```
# mkdir -p /source
# mount -F vxfs /dev/vgsource/lvol1  /source
```

4.1.1.2.2 Volume group target building

> Generate the volume group target (vgtarget building).

```
# mkdir -p /dev/vgtarget
# mknod /dev/vgtarget/group c 64 0x070000
# pvcreate /dev/rdsk/c6t12d3
# pvcreate /dev/rdsk/c6t12d4
# vgcreate /dev/vgtarget /dev/dsk/c6t12d3 /dev/dsk/c6t12d4
# lvcreate -i 2 -I 64 -L 600 /dev/vgtarget
```

4.1.2 Dedicated backup server architecture

Production server configuration is the same whether it is for single server architecture or dedicated backup server. This means that the above operations be performed on the production server for this dedicated backup server configuration (Ihers).

The following gives another solution to move the volume groups created on taillon to the new production host named Ihers. The steps described are:

> Exporting the volume groups from taillon to Ihers

> Configuring the volume groups on Ihers.

4.1.2.1 SVA configuration

4.1.2.1.1 Exporting the volume groups created from taillon

Execute the following steps on taillon for the vgsource volume group and the vgtarget volume group:

> Dismount volume group vgsource (if mounted).

```
# unmount /source
```

> Deactivate the volume group.

```
# vgchange -a n /dev/vgsource
```

> Delete volume group vgsource on taillon.

```
# vgexport /dev/vgsource
```

4.1.2.1.2 Devices configuration on LHERS

> Connect the host to the SVA on the same domain (D8)

> Make the previously created LUNs available for the system.

```
# insf
# ioscan -fnC disk
disk      33  0/4/0/0.2.21.255.12.12.0  sdisk  CLAIMED     DEVICE        STK      9200
                                         /dev/dsk/c10t12d0   /dev/rdsk/c10t12d0
disk      34  0/4/0/0.2.21.255.12.12.1  sdisk  CLAIMED     DEVICE        STK      9200
                                         /dev/dsk/c10t12d1   /dev/rdsk/c10t12d1
disk      35  0/4/0/0.2.21.255.12.12.2  sdisk  CLAIMED     DEVICE        STK      9200
                                         /dev/dsk/c10t12d2   /dev/rdsk/c10t12d2
disk      36  0/4/0/0.2.21.255.12.12.3  sdisk  CLAIMED     DEVICE        STK      9200
                                         /dev/dsk/c10t12d3   /dev/rdsk/c10t12d3
disk      37  0/4/0/0.2.21.255.12.12.4  sdisk  CLAIMED     DEVICE        STK      9200
                                         /dev/dsk/c10t12d4   /dev/rdsk/c10t12d4
# sppath -v
.....
SPD=5  c10t12d0 dev=188,0x0ac000 type=2 SANID="STK 9200 0000000000030018C"
SPD=6  c10t12d1 dev=188,0x0ac100 type=2 SANID="STK 9200 0000000000030018D"
SPD=7  c10t12d2 dev=188,0x0ac200 type=2 SANID="STK 9200 0000000000030018F"
SPD=8  c10t12d3 dev=188,0x0ac300 type=2 SANID="STK 9200 0000000000300191"
SPD=9  c10t12d4 dev=188,0x0ac400 type=2 SANID="STK 9200 0000000000300193"
# setsp
================================================================================
 spd   Path/disk          Status   Pri Exc Buf Balance RtrCnt  RtrDly FailBack
================================================================================
   5   c10t12d0          Excluded    X       1000    1       20     3000       1
 spd5 = c10t12d0                                  ID = "STK 9200 0000000000030018C"
================================================================================
   6   c10t12d1          Excluded    X       1000    1       20     3000       1
 spd6 = c10t12d1                                  ID = "STK 9200 0000000000030018D"
================================================================================
   7   c10t12d2          Excluded    X       1000    1       20     3000       1
 spd7 = c10t12d2                                  ID = "STK 9200 0000000000030018F"
================================================================================
   8   c10t12d3          Excluded    X       1000    1       20     3000       1
 spd8 = c10t12d3                                  ID = "STK 9200 0000000000300191"
================================================================================
   9   c10t12d4          Excluded    X       1000    1       20     3000       1
 spd9 = c10t12d4                                  ID = "STK 9200 0000000000300193"
================================================================================
# setsp -l5 -e0
# setsp -l6 -e0
# setsp -l7 -e0
# setsp -l8 -e0
# setsp -l9 -e0
```

4.1.2.2 Volume group configuration

Perform the following steps on lhers for the vgsource and the vgtarget volume groups:

> Import volume group vgsource.

```
# mkdir -p /dev/vgsource
# mknod /dev/vgsource/group c 64 0x060000
# vgimport /dev/vgsource/group /dev/dsk/c10t12d1 /dev/dsk/c10t12d2
```

> Make the volume group active.

```
# vgchange -a y /dev/vgsource
```

> Mount volume group vgsource.

```
# mkdir -p /source
# mount -F vxfs /dev/vgsource/lvol1  /source
```

4.2 BACKUP SERVER CONFIGURATION

4.2.1 Tape configuration

> Connect the tape drive or library to the backup host

> Discover the tape device.

```
# ioscan -fnC disk
Class      I  H/W Path                    Driver S/W State   H/W Type     Description
================================================================================
...
tape       1  0/3/0/0.2.19.255.0.0.0  stape  CLAIMED     DEVICE       Quantum DLT4000
                           /dev/rmt/1m       /dev/rmt/1mnb       /dev/rmt/c5t0d0BESTn
                           /dev/rmt/1mb      /dev/rmt/c5t0d0BEST /dev/rmt/c5t0d0BESTnb
                           /dev/rmt/1mn      /dev/rmt/c5t0d0BESTb
```

This DLT® 4000 tape drive will be used when performing backup operations from the disk devices to the tapes.

4.2.2 Time navigator configuration

This section gives the operations that have to be performed on the backup server. Please refer to the *Time Navigator Administration Guide* for more details.

To use the backup utilities, launch the Time Navigator Administrator Interface.

Go to the TINA binaries directories. In this case, the installation directory is /opt/tina.

```
# /opt/tina/Bin/tina_adm &
```

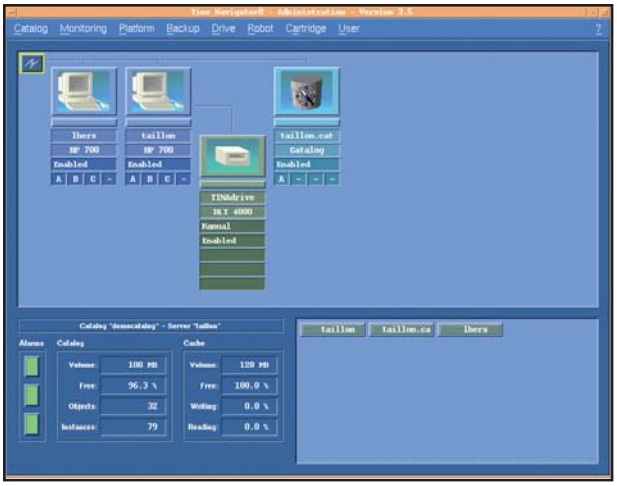


Figure 6. The TINA Administration Interface.

The TINA Server is installed on the backup server. The TINA client application must be installed on the production server to backup local devices. Snapshots of these devices will be taken from the production server and the snapshots will be mounted on the backup server for the backup operation.

4.2.2.1 Creating hosts

All the client machines to be backed up from the backup server must be created as hosts from the TINA administration interface (**Figure 6**). You can create a host at any time.

> Go to the Platform Menu > Host > Create
> Complete the fields with the correct values (Type of OS, Name ...).

The new host is displayed in the *NetWork* area and its icon in the *Icons* area. Now that the host is successfully created, you have to enable it:

> Select the host to enable
> Go to Platform Menu > Use > Enabled or right click on the host icon and select Enabled.

4.2.2.2 Creating drives

Create and configure the drive(s) used to perform the backup operations on the backup host.

> Select the host to which the drive is connected
> Go to the Drives Menu > Create
> Select the type of Drive you want to add
> Complete the fields with the correct values (Drive Name, Device Descriptor ...).

Note In the Device Descriptor Field, enter the full path of the tape drive device, for instance /dev/rmt/1mn (choose the no rewind option to be able to perform restore operations).

Now that the drive is successfully created, you have to enable it:

> Select the drive to enable
> Go to Drive Menu > Use > Enabled or right click on the drive icon and select Enabled.

It is recommended to get a drive diagnosis to check whether the drive is in working conditions and has been correctly configured.

> Select the drive to diagnose
> Go to Drive Menu > Use > Diagnosis or right click on the drive icon and select Diagnosis.

4.2.2.3 Creating cartridge pools

A name, a label, drives and a management policy define a cartridge pool. To create a pool:

> Go to Cartridge Menu > Pool > Create
> Enter the name of the cartridge pool in the Pool Name textbox
> Enter a label to define a label for the pool (all the cartridges in the pool will bear that label)
> Define a Management policy
> Add the associated drives.

4.2.2.4 Creating backup strategies

To schedule automatic backups, you must define backup strategies. You can define up to four strategies (A, B, C, D) for each host.

> Select a host
> Go to Backup Menu > Strategy A > Edit
> Give the main cartridge pool name in the Full and Incremental tabs
> Complete the Advanced tab with the Pre-processing and Post-processing command paths.

Please refer to the following sections for further details on backup strategies to be created on each host:

> In the Single-server architecture
- **5.1.1 Backup execution overview**
- **5.1.2 Pre-/Post-processing scripts for Strategy A**

> In the Dedicated backup server architecture
- **5.2.1 Backup execution overview**
- **5.2.2 Pre-/Post-processing scripts on the backup server**
- **5.2.3 Pre-/Post-processing scripts on the production server**

4.2.2.5 Creating backup classes

Files and directories to be backed up must be part of a backup class.

> Select a host
> Go to Backup Menu > Class to edit the list of classes defined
> Choose the directory to be backed up and define the backup class settings.

Please refer to the sections mentioned below for more information on the backup classes to be created:

5.1.3 Development overview in the Single-Server Architecture

5.2.4 Development overview in the Dedicated Backup Server Architecture.

5 THE MAIN PROCESSES

5.1 SINGLE-SERVER ARCHITECTURE

This section describes the different processes generated by the backup operation using the TINA backup software application with the SVA snapshot in a single server configuration.

All the scripts mentioned here are detailed in **8 Appendices**.

Note **The scripts used here do not include the use of SVA Path. But you can add SVA Path command lines (to include and exclude the target devices).**

5.1.1 Backup execution overview

Only one backup strategy is needed in this configuration:

> Strategy A calls the pre-processing and the post-processing scripts

> The pre-processing script launches the snapshot and mount operations

> The backup classes specified for this strategy contain the files and directories to be backed up

> The post-processing script launches the dismount and release operations after backup is done.

The figure below (**Figure 7**) details the backup operation processes.

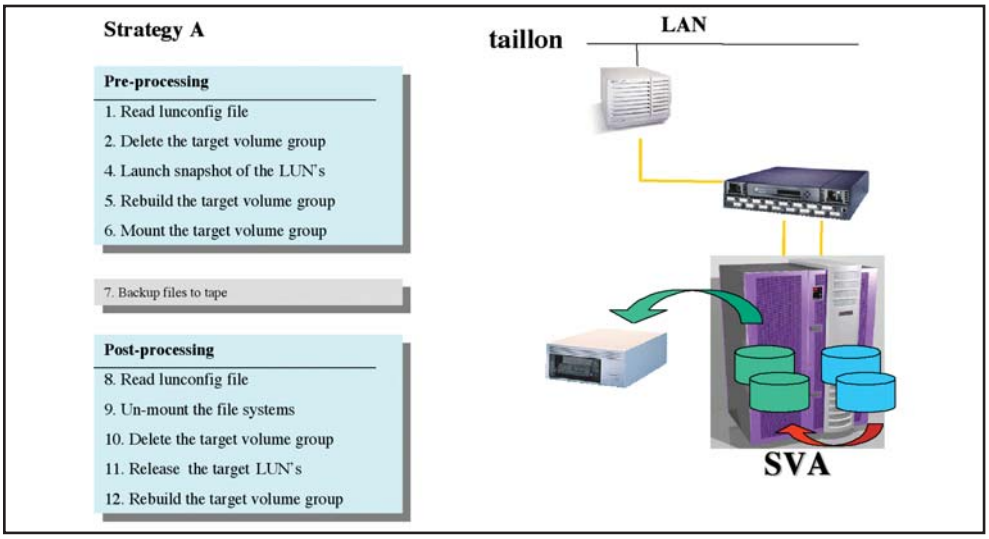
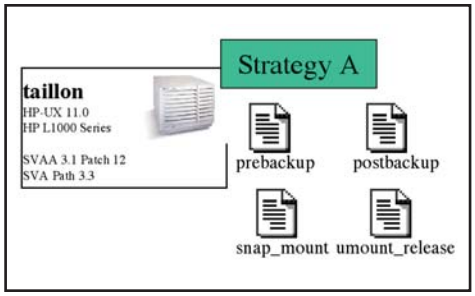


Figure 7. Backup operations (no additional backup server).

5.1.2 Pre-/Post-processing scripts for Strategy A

Please refer to **Appendix B: Pre-/Post-processing scripts (1 Server)** for the detailed pre-backup and post-backup scripts.



> Pre-processing script (pre-backup)
- Read the lunconfig file
- Delete vgtarget
- Launch snapshot of each LUNs
- Rebuild vgtarget
- Mount the volume group to the file system.

> Post-processing script (post-backup)
- Read the lunconfig file
- Unmount the file systems
- Delete vgtarget
- Release the target LUNs
- Rebuild vgtarget.

5.1.3 Development overview

In this single server configuration, after all configuration steps are completed, perform the following steps:

> Choose a custom script directory and initialize the STK_SCRIPT_DIR environment variable.

```
STK_SCRIPT_DIR=/export/scripts
```

> Copy all the configuration scripts to the directory:
- lunconfig (**Appendix A**)
- pre-backup (**Appendix B**)
- post-backup (**Appendix B**)
- snap_mount **(Appendix C**)
- unmount_release (**Appendix C**)

Note Make changes in each script to reflect your own organization.

> Define the backup class (**Figure 8**)
- Select the host to which files and directories will be backed up
- Go to the Backup Menu > Class
- Type the full path of the directory you're interested in (/target).

Note The directory mentioned here must be the one you specify in the lunconfig file as target mount point. If more than one target is mentioned, create as many backup classes as needed. All these classes will use backup strategy A.

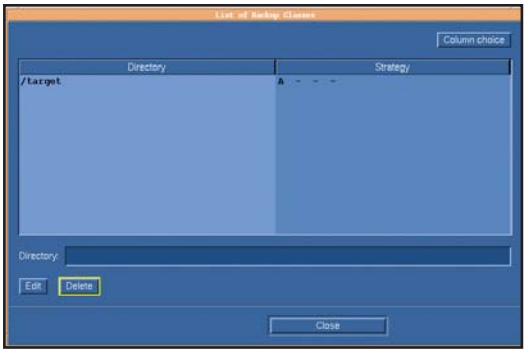


Figure 8. List of backup classes (no additional backup server).

> Fill the Pre- and Post-processing fields of the backup strategy A

- Select the host
- Go to Backup Menu > Strategy A > Edit
- Complete the fields with the script's full paths.

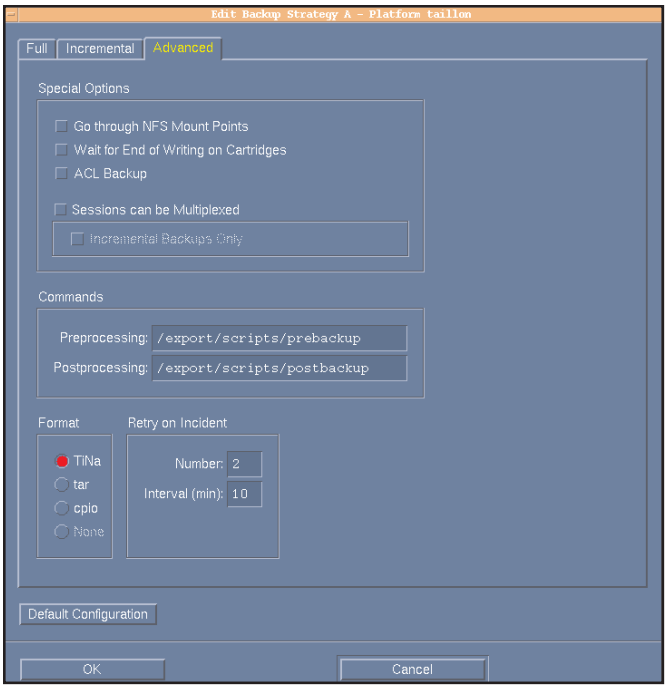


Figure 9. Backup Strategy A (No additional backup server).

> Launch the backup operation

- Select the host
- Go to Backup Menu > Strategy A > Full Session Now

> Go to Monitoring Menu > Jobs to visualize the backup operation.

Please refer to **Appendix D: Example of successful backup (1 server)** for a detailed log of a successful backup.

5.2 DEDICATED BACKUP SERVER ARCHITECTURE

This section describes the different processes generated by the backup operation using the TINA backup software application with the SVA snapshot in the dedicated backup server architecture.

All scripts mentioned here are detailed in **8 Appendices.**

Note **The scripts used here include the use of SVA Path. This configuration requires the use of SVA Path.**

5.2.1 Backup execution overview

One backup strategy is defined on the backup server (taillon)

Strategy A is the main strategy, calling for all other strategies:

> Pre-processing:
- Call Strategy A on lhers to perform the snapshot operation
- Mount the target devices to perform the backup operation.

> Post-processing:
- Unmount the target devices from the backup server view
- Call Strategy B on lhers to perform the release operation.

Two backup strategies are defined on the production server (lhers)

> Strategy A launches the snapshot process and performs a dummy backup
> Strategy B launches the release process and performs a dummy backup

The figure below (**Figure 10**) details backup operation processes.

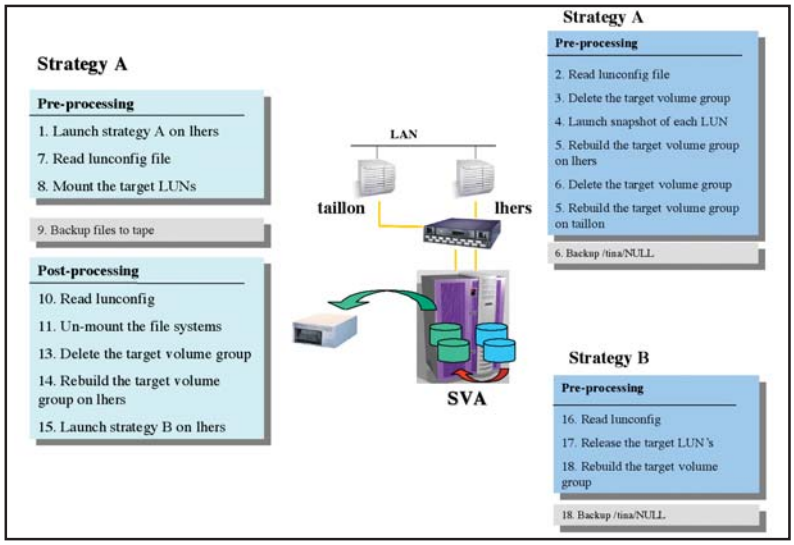
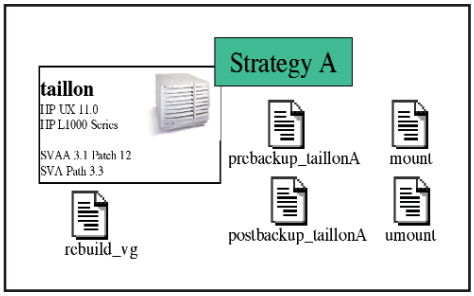


Figure 10. Backup operations (dedicated backup server).

5.2.2 Pre-/Post-processing scripts on the backup server

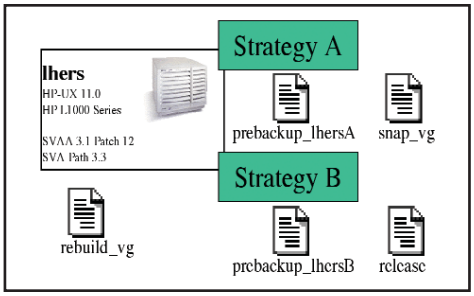


Please refer to **Appendix F: Pre-/Post-processing scripts of Strategy A on taillon (2 Servers)** for the detailed pre-backup_taillonA and post-backup_taillonA scripts.

> Pre-processing script (pre-backup_taillonA)
- Call backup Strategy A on lhers
- Read the lunconfig file
- Mount the volume group to the file system.

> Post-processing script (post-backup_taillonA)
- Read the lunconfig file
- Unmount the file systems
- Delete vgtarget
- Rebuild vgtarget
- Call backup Strategy B on lhers
- Rebuild vgtarget.

5.2.3 Pre-/Post-processing scripts on the production server



Please refer to
Appendix G: Pre-processing script of Strategy A on LHERS (2 Servers)
Appendix H: Pre-processing script of Strategy B on LHERS (2 Servers)
for the detailed scripts named pre-backup_lhersA and pre-backup_lhersB.

> Pre-processing script of Strategy A (pre-backup_lhersA)
- Read the lunconfig file
- Delete vgtarget
- Launch snapshot of each LUN
- Rebuild vgtarget on lhers
- Delete vgtarget
- Rebuild vgtarget on taillon.

> Post-processing script of Strategy B (pre-backup_lhersB)
- Read the lunconfig file
- Release the target LUN
- Rebuild vgtarget.

5.2.4 Development overview

In this dedicated backup server configuration, after all configuration steps are completed on the production server and on the backup server, perform the following steps:

> Choose a custom script directory and initialize the STK_SCRIPT_DIR environment variable in the scripts used either on the production or the backup hosts.

```
STK_SCRIPT_DIR=/export/scripts
```

> Copy all the configuration scripts to the directory:

- On the Backup Server (taillon)
  - lunconfig (**Appendix E**)
  - pre-backup_taillonA (**Appendix F**)
  - post-backup_taillonA (**Appendix F**)
  - mount (**Appendix I**)
  - unmount (**Appendix I**)
- On the Production Server (lhers)
  - lunconfig (**Appendix E**)
  - pre-backup_lhersA (**Appendix G**)
  - pre-backup_lhersB (**Appendix H**)
  - snap_vg (**Appendix I**)
  - release (**Appendix I**)

Note **Make changes in each script to reflect your own organization.**

> Define backup classes (**Figure 11** and **Figure 12**)

- Select the host to which files and directories will be backed up
- Go to the Backup Menu > Class
- Type the full path of the directory you're interested in (/target)

Note **The directory mentioned here must be the one you specify in the lunconfig file as target mount point. If more than one target is mentioned, create as many backup classes as needed. All these classes will use backup Strategy A.**

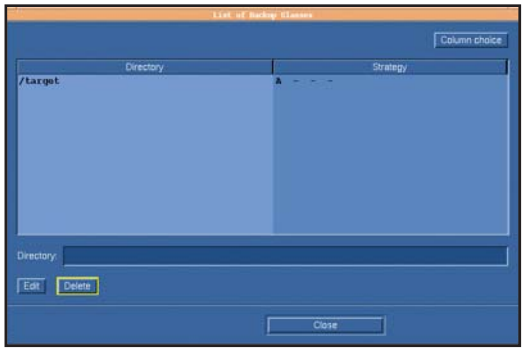


Figure 11. List of backup classes on taillon (additional backup server).

Note **The directory mentioned here (Figure 12) serves as a dummy backup. This directory must be created on the production server. As Strategy A and Strategy B perform the dummy backups, specify Strategy A and B for the backup class /tina/NULL.**

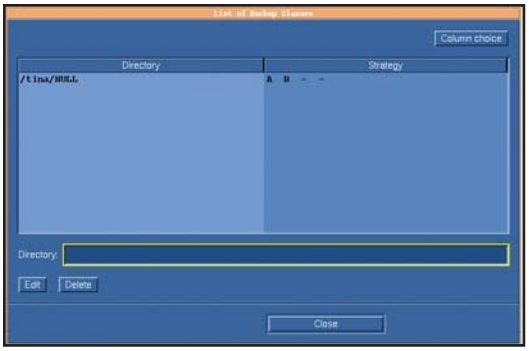


Figure 12. List of backup classes on LHERS (production host).

> Fill the Pre- and Post-processing fields of each backup strategy
- Select the host taillon
- Go to Backup Menu > Strategy A > Edit
- Complete the fields with the full script pathnames:
  – pre-backup_taillonA
  – post-backup_taillonA
- Select the host lhers
- Go to backup Menu > Strategy A > Edit.

> Complete the fields with the full script pathnames:
  – pre-backup_lhersA
- Select the host lhers
- Go to backup Menu > Strategy B > Edit
- Complete the fields with the full script pathnames:
  – pre-backup_lhersB.

> Launch the backup operation
- Select the host lhers
- Go to Backup Menu > Strategy A > Full Session Now

> Go to Monitoring Menu > Jobs to visualize the backup operation.

Please refer to **Appendix J: Example of successful backup (2 servers)** for a detailed log of a successful backup.

6 HOW TO RESTORE

Use the TINA interface to perform restore operations:

> Select the host where are located the files or directories to restore

> Go to Platform Menu > TINA (the TINA client interface is launched — **Figure 13**)

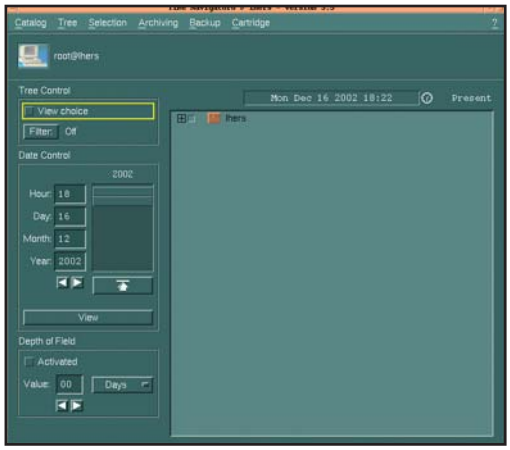


Figure 13. Time Navigator Client Interface.

> Go to Tree Menu > Search

> Complete the search fields to find the directory or file to be restored

> Select the object found that you want to restored

> Synchronize the contents

> Select the files or/and directories to be restored

> Go to Backup Menu > Restore > Choice

> Complete the fields with the restore parameters

> Launch the restore operation

> View the restore operation progress (go to Monitoring Menu > Jobs).

7 CRITICAL ANALYSIS

ADVANTAGES	DRAWBACKS
Simple: there is only one SVA command per file system.	Dummy data is sent from the client to the backup server during the execution of pre-processing and post-processing scripts of Strategy A on the backup server.
The data can be backed up to tape using another server that will mount the SVA snapshot.	
All the other advantages of a cold backup remain.	

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

8 APPENDICES

This section contains a copy of each script mentioned in this document.

APPENDIX A: LUN CONFIGURATION FILE (1 SERVER)

lunconfig file

```
#
#     Database          Database          Mounting
#     Volume Group      Volume Group      point base
#     source            target            target
#
#================================================
      /dev/vgsource     /dev/vgtarget     /target
```

APPENDIX B: PRE-/POST-PROCESSING SCRIPTS (1 SERVER)

pre-backup

```
#! /bin/sh

#
#
#  This is the preprocessing script
#
#

STK_SCRIPT_DIR=`dirname $0`
RESULT_FILE=$STK_SCRIPT_DIR/result_file
SVA_LUNCONFIG_FILE=$STK_SCRIPT_DIR/lunconfig

# Get the production server name and the TINA catalog name
PROD_NAME=taillon
CATALOG_NAME=democatalog
STARTEGY=C

echo “BEGIN OF STK SVA BACKUP - `date`” > $RESULT_FILE
echo >> $RESULT_FILE
echo “============== Pre-processing Script starting - `date`” >> $RESULT_FILE
echo >> $RESULT_FILE

echo “Initializing variables” >> $RESULT_FILE
SOURCE=””
TARGET=””
MOUNT_TARGET=””

# LUNCONFIG file existence
if [ ! -f $SVA_LUNCONFIG_FILE ]
then
   echo “SVA_LUNCONFIG_FILE not present” >> $RESULT_FILE
   echo “Check if file exists: $SVA_LUNCONFIG_FILE” >> $RESULT_FILE
   exit 1
fi

# Call the snapshot and mount script
echo “************* Calling the snap_mount script - `date` *********************” >>
$RESULT_FILE
/bin/egrep -v ‘^#|^$’ $SVA_LUNCONFIG_FILE |
while read SOURCE TARGET MOUNT_TARGET
do
   $STK_SCRIPT_DIR/snap_mount $SOURCE $TARGET $MOUNT_TARGET >> $RESULT_FILE
done
echo “********* End of execution of the snap_mount script - `date` *************” >>
$RESULT_FILE



echo >> $RESULT_FILE
echo “============ End of the Preprocessing script - `date`”  >> $RESULT_FILE
echo >> $RESULT_FILE
exit 0
```

post-backup

```
#!/bin/sh

#
#  This is the postprocessing script
#

STK_SCRIPT_DIR=`dirname $0`
RESULT_FILE=$STK_SCRIPT_DIR/result_file
SVA_LUNCONFIG_FILE=$STK_SCRIPT_DIR/lunconfig

echo >> $RESULT_FILE
echo "================================ Post-processing Script starting - `date`" >>
$RESULT_FILE
echo >> $RESULT_FILE

echo "Initializing variables" >> $RESULT_FILE
SOURCE=""
TARGET=""
MOUNT_TARGET=""

# LUNCONFIG file existence
echo "$SVA_LUNCONFIG_FILE is the LUN configuration file" >> $RESULT_FILE
if [ ! -f $SVA_LUNCONFIG_FILE ]
then
   echo "SVA_LUNCONFIG_FILE not present">> $RESULT_FILE
   echo "Check if file exists: $SVA_LUNCONFIG_FILE">> $RESULT_FILE
   exit 1
fi

# Unmount and Release the target LUNs on the SVA
echo "**************** Calling the unmount_release script - `date`
**********************">> $RESULT_FILE
/bin/egrep -v '^#|^$' $SVA_LUNCONFIG_FILE |
while read SOURCE TARGET MOUNT_TARGET
do
   $STK_SCRIPT_DIR/unmount_release $TARGET >> $RESULT_FILE
done
echo "************ End of execution of the unmount_release script - `date`
***************">> $RESULT_FILE

echo >> $RESULT_FILE
echo "============================== End of the Postprocessing script - `date`" >>
$RESULT_FILE
echo >> $RESULT_FILE
echo "END OF STK SVA BACKUP - SUCCESSFUL - `date`" >> $RESULT_FILE
exit 0
```

Note **The scripts detailed here do not contain the absolute path of each system executables. If the PATH environment variable does not contain the needed path, you have to modify the PATH value or modify the script with the full path of each command.**

APPENDIX C: SNAP, MOUNT, UNMOUNT, RELEASE SCRIPTS (1 SERVER)

snap_mount script

```
#!/bin/sh

# -- Important Note --
#
# The volume group target must be existent and its
# physical structure must be identical to the volume group source !
#

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

# Synopsis of the snap_cmd script
synopsis()
{
      print "Synopsis"
      print "########"
      print ""
      print "snap_cmd Volume_Group_Name_Source Volum_Group_Name_Target
[mounting_point_target]"
      print ""
      print "Examples : snap_cmd /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


# 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
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 deactivation
# All mounting points umounting
for MPOINT in `df | grep $VG_TARGET | awk '{print $1}'`
do
      unmount $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 deactivation
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'}'`
      echo snap -source /dev/rdsk/$SOURCE -target /dev/rdsk/$TARGET
      IND=`expr $IND + 1`
done | $SIBADMIN 1>/dev/null 2>&1

# 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 $MOUNT-
ING_BASE$MPOINT" >> $MOUNTING_BASE/mount.log
               done
      done
fi

exit 0
```

unmount_release script

```
#!/bin/sh

# Synopsis of the unmount_release script
synopsis()
{
      print “Synopsis”
      print “########”
      print “”
      print “unmount_release Volume_Group_Name”
      print “”
      print “Examples : unmount_release /dev/vg01”
}

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

# 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
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 deactivation
# All mounting points umounting
for MPOINT in `df | grep $VOLUME_GROUP | awk ‘{print $1}’`
do
      unmount $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 deactivation
vgchange -a n $VOLUME_GROUP

# Volume group removing
vgexport $VOLUME_GROUP

# Release
echo “Release ...”
IND=1
while [ $IND -le $NPV ]
do
      DEV=`print $PV_LIST | awk ‘{print $’$IND’}’`
      echo release -force -path /dev/rdsk/$DEV
      IND=`expr $IND + 1`
done | $SIBADMIN 1>/dev/null 2>&1

# 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
```

APPENDIX D: EXAMPLE OF SUCCESSFUL BACKUP (1 SERVER)

```
BEGIN OF STK SVA BACKUP - Thu Nov 28 11:23:42 WET 2002

============= Pre-processing Script starting - Thu Nov 28 11:23:42 WET 2002

Initializing variables
********** Calling the snap_mount script - Thu Nov 28 11:23:42 WET 2002 *************
Volume group "/dev/vgtarget" has been successfully changed.
Snapshot ...
Warning: A backup of this volume group may not exist on this machine.
Please remember to take a backup using the vgcfgbackup command after activating the
volume group.
Activated volume group
Volume group "/dev/vgtarget" has been successfully changed.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
log replay in progress
replay complete - marking super-block as CLEAN
***** End of execution of the snap_mount script - Thu Nov 28 11:23:58 WET 2002 *******

============ End of the Preprocessing script - Thu Nov 28 11:23:58 WET 2002


============== Post-processing Script starting - Thu Nov 28 11:24:28 WET 2002

Initializing variables
/export/scripts/lunconfig is the LUN configuration file
*************** Calling the unmount_release script - Thu Nov 28 11:24:28 WET 2002
**********************
Logical volume "/dev/vgtarget/lvol1" has been successfully removed.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
Volume group "/dev/vgtarget" has been successfully changed.
Release ...
Physical volume "/dev/rdsk/c6t12d3" has been successfully created.
Physical volume "/dev/rdsk/c6t12d4" has been successfully created.
Increased the number of physical extents per physical volume to 1173.
Volume group "/dev/vgtarget" has been successfully created.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
Volume group "/dev/vgtarget" has been successfully changed.
************ End of execution of the unmount_release script - Thu Nov 28 11:24:47 WET
2002 ***************

=============== End of the Postprocessing script - Thu Nov 28 11:24:47 WET 2002

END OF STK SVA BACKUP - SUCCESSFUL - Thu Nov 28 11:24:47 WET 2002
```

APPENDIX E: LUNS CONFIGURATION FILE (2 SERVERS)

lunconfig

```
#
#     Database          Database          Mounting
#     Volume Group      Volume Group      point base
#     source            target            target
#
#=====================================================
      /dev/vgsource     /dev/vgtarget     /target
```

This file is the same on the production host as it is on the backup server.

APPENDIX F: PRE-/POST-PROCESSING SCRIPTS OF STRATEGY A ON TAILLON (2 SERVERS)

pre-backup_taillonA

```
#! /bin/sh

#
#
#  This is the preprocessing script for the Strategy A on the backup server
#  The aim of this script is to initiate the SVA backup
#
#

STK_SCRIPT_DIR=/export/scripts
RESULT_FILE=$STK_SCRIPT_DIR/result_file
SVA_LUNCONFIG_FILE=$STK_SCRIPT_DIR/lunconfig

# Get the production server name and the TINA catalog name
PROD_SERVER=lhers
CATALOG_NAME=democatalog
STRATEGY=A

/bin/echo "BEGIN OF STK SVA BACKUP - `date`" > $RESULT_FILE
/bin/echo >> $RESULT_FILE
/bin/echo "============= Pre-processing Script starting on backup server (Strategy A) -
`date`" >> $RESULT_FILE
/bin/echo >> $RESULT_FILE

/bin/echo "Initializing variables" >> $RESULT_FILE
SOURCE=""
TARGET=""
MOUNT_TARGET=""

# LUNCONFIG file existence
if [ ! -f $SVA_LUNCONFIG_FILE ]
then
   /bin/echo "SVA_LUNCONFIG_FILE not present" >> $RESULT_FILE
   /bin/echo "Check if file exists: $SVA_LUNCONFIG_FILE" >> $RESULT_FILE
   exit 1
fi

# Call the snapshot
/bin/echo "********** Calling the Strategy A on production server $PROD_SERVER (snap) -
`date` **************" >> $RESULT_FILE
/opt/tina/Bin/tina_archive -backup -full -strat $STRATEGY -host $PROD_SERVER -catalog
$CATALOG_NAME

# Mount the target devices on the backup server
/bin/echo "************ Calling the mount script - `date` ********************" >>
$RESULT_FILE
/bin/egrep -v '^#|^$' $SVA_LUNCONFIG_FILE |
while read SOURCE TARGET MOUNT_TARGET
do
   /bin/echo SOURCE=$SOURCE TARGET=$TARGET MOUNT_TARGET=$MOUNT_TARGET >> $RESULT_FILE
   /bin/echo $STK_SCRIPT_DIR/mount $SOURCE $TARGET $MOUNT_TARGET >> $RESULT_FILE
   $STK_SCRIPT_DIR/mount $SOURCE $TARGET $MOUNT_TARGET >> $RESULT_FILE
done
/bin/echo "********* End of execution of the mount script - `date` *************" >>
$RESULT_FILE

/bin/echo >> $RESULT_FILE
/bin/echo "============ End of the Preprocessing script on the backup server (Strategy
A) - `date`"  >> $RESULT_FILE
/bin/echo >> $RESULT_FILE
exit 0
```

post-backup_taillonA

```
#!/bin/sh

#
#  This is the postprocessing script
#

STK_SCRIPT_DIR=`dirname $0`
RESULT_FILE=$STK_SCRIPT_DIR/result_file
SVA_LUNCONFIG_FILE=$STK_SCRIPT_DIR/lunconfig

# Get the production server name and the TINA catalog name
PROD_SERVER=lhers
CATALOG_NAME=democatalog
STRATEGY=B

/bin/echo >> $RESULT_FILE
/bin/echo "======================== Post-processing Script starting on the backup
server (Strategy A) - `date`" >> $RESULT_FILE
/bin/echo >> $RESULT_FILE

/bin/echo "Initializing variables" >> $RESULT_FILE
SOURCE=""
TARGET=""
MOUNT_TARGET=""

# LUNCONFIG file existence
/bin/echo "$SVA_LUNCONFIG_FILE is the LUN configuration file" >> $RESULT_FILE
if [ ! -f $SVA_LUNCONFIG_FILE ]
then
   /bin/echo "SVA_LUNCONFIG_FILE not present">> $RESULT_FILE
   /bin/echo "Check if file exists: $SVA_LUNCONFIG_FILE">> $RESULT_FILE
   exit 1
fi

# Unmount the target LUNs on the SVA from the backup server
/bin/echo "**************** Calling the unmount script - `date`
**********************">> $RESULT_FILE
/bin/egrep -v '^#|^$' $SVA_LUNCONFIG_FILE |
while read SOURCE TARGET MOUNT_TARGET
do
   $STK_SCRIPT_DIR/unmount $TARGET >> $RESULT_FILE
done
/bin/echo "************ End of execution of the unmount script - `date`
***************">> $RESULT_FILE

# Release the target devices on the production server $PROD_SERVER
/bin/echo "********** Calling the Strategy B on production server $PROD_SERVER
(release) - `date` **************" >> $RESULT_FILE
/opt/tina/Bin/tina_archive -backup -full -strat $STRATEGY -host $PROD_SERVER -catalog
$CATALOG_NAME >> $RESULT_FILE

/bin/echo >> $RESULT_FILE
/bin/echo "============================== End of the Postprocessing script - `date`" >>
$RESULT_FILE
/bin/echo >> $RESULT_FILE
/bin/echo "END OF STK SVA BACKUP - SUCCESSFUL - `date`" >> $RESULT_FILE
exit 0
```

APPENDIX G: PREPROCESSING SCRIPT OF STRATEGY A ON LHERS (2 SERVERS)

pre-backup_lhersA

```
#! /bin/sh

#
#
#  This is the preprocessing script of the Strategy A on the production server
#  The aim of this script is to perform the SVA Snapshot
#
#

STK_SCRIPT_DIR=`dirname $0`
RESULT_FILE=$STK_SCRIPT_DIR/result_file
SVA_LUNCONFIG_FILE=$STK_SCRIPT_DIR/lunconfig

echo > $RESULT_FILE
echo "=========== Pre-processing Script on lhers starting (Startegy A) - `date`" >>
$RESULT_FILE
echo >> $RESULT_FILE

echo "Initializing variables" >> $RESULT_FILE
SOURCE=""
TARGET=""
MOUNT_TARGET=""

# LUNCONFIG file existence
if [ ! -f $SVA_LUNCONFIG_FILE ]
then
   echo "SVA_LUNCONFIG_FILE not present" >> $RESULT_FILE
   echo "Check if file exists: $SVA_LUNCONFIG_FILE" >> $RESULT_FILE
   exit 1
fi

# Call the snapshot
echo "************* Calling the snap script - `date` **********************" >>
$RESULT_FILE
/bin/egrep -v '^#|^$' $SVA_LUNCONFIG_FILE |
while read SOURCE TARGET MOUNT_TARGET
do
   $STK_SCRIPT_DIR/snap_vg $SOURCE $TARGET >> $RESULT_FILE
done
echo "********* End of execution of the snap script - `date` *************" >>
$RESULT_FILE
echo >> $RESULT_FILE

echo >> $RESULT_FILE
echo "========== End of the Preprocessing on lhers script (Strategy A) - `date`"  >>
$RESULT_FILE
echo >> $RESULT_FILE
exit 0
```

APPENDIX H: PREPROCESSING SCRIPT OF STRATEGY B ON LHERS (2 SERVERS)

pre-backup_lhersA

```
#! /bin/sh

#
#
#  This is the preprocessing script of the Strategy A on the production server
#  The aim of this script is to perform the release of the target devices of the SVA
#
#

STK_SCRIPT_DIR=`dirname $0`
RESULT_FILE=$STK_SCRIPT_DIR/result_file
SVA_LUNCONFIG_FILE=$STK_SCRIPT_DIR/lunconfig

echo >> $RESULT_FILE
echo "=========== Pre-processing Script on lhers starting (Startegy B) - `date`" >>
$RESULT_FILE
echo >> $RESULT_FILE

echo "Initializing variables" >> $RESULT_FILE
SOURCE=""
TARGET=""
MOUNT_TARGET=""

# LUNCONFIG file existence
if [ ! -f $SVA_LUNCONFIG_FILE ]
then
   echo "SVA_LUNCONFIG_FILE not present" >> $RESULT_FILE
   echo "Check if file exists: $SVA_LUNCONFIG_FILE" >> $RESULT_FILE
   exit 1
fi

# Do the release of the target devices on the SVA
echo "************* Calling the release script - `date` **********************" >>
$RESULT_FILE
/bin/egrep -v '^#|^$' $SVA_LUNCONFIG_FILE |
while read SOURCE TARGET MOUNT_TARGET
do
   $STK_SCRIPT_DIR/release $TARGET >> $RESULT_FILE
done
echo "********* End of execution of the release script - `date` *************" >>
$RESULT_FILE
echo >> $RESULT_FILE

echo >> $RESULT_FILE
echo "========== End of the Preprocessing on lhers script (Strategy B) - `date`"  >>
$RESULT_FILE
echo >> $RESULT_FILE
exit 0
```

APPENDIX I: SNAP, MOUNT, UNMOUNT, RELEASE SCRIPTS (2 SERVERS)

snap_vg on lhers

```
#!/bin/sh

# -- Important Note --
#
# The volume group target must be existent and its
# physical structure must be identical to the volume group source !
#

# The complete path where you can find the binary sibadmin
SIBADMIN="/opt/storagetek/SVAA3.1.0/bin/sibadmin"
BACKUP_SERVER=taillon
SCRIPTS_DIR=/export/scripts

# Synopsis of the snap_vg script
synopsis()
{
      /bin/echo "Synopsis"
      /bin/echo "########"
      /bin/echo ""
      /bin/echo "snap_vg Volume_Group_Name_Source Volum_Group_Name_Target"
      /bin/echo ""
      /bin/echo "Examples : snap_vg /dev/vg01 /dev/vg03"
      /bin/echo ""
      /bin/echo "Important note : The both volumes group must have identical physical
structure !"
}

# Parameters verification
if [ $# -ne 2 ]
then
      synopsis
      exit 1
else
      if [ `/bin/echo $1 | /bin/cut -d'/' -f2` != "dev" ]
      then
               synopsis
               exit 1
      else
               if [ `/bin/echo $2 | /bin/cut -d'/' -f2` != "dev" ]
               then
                        synopsis
                        exit 1
               else
                        VG_SOURCE=$1
                        VG_TARGET=$2
                        VG_SOURCE_NAME=`/bin/echo $VG_SOURCE | /bin/cut -d'/' -f3`
                        VG_TARGET_NAME=`/bin/echo $VG_TARGET | /bin/cut -d'/' -f3`
               fi
      fi
fi


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

```
SANID_TARGET_LIST=""
# Volumes Group existence verification
/usr/sbin/vgdisplay $VG_SOURCE
/usr/sbin/vgdisplay $VG_SOURCE > /dev/null 2> /dev/null
if [ $? -ne 0 ]
then
      /bin/echo $VG_SOURCE" doesn't exist or is not activated !"
      exit 2
fi
/usr/sbin/vgdisplay $VG_TARGET > /dev/null 2> /dev/null
if [ $? -ne 0 ]
then
      /bin/echo $VG_TARGET" doesn't exist or is not activated !"
      exit 2
fi

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

# Devices lists extraction
for PV in `/usr/sbin/vgdisplay -v $VG_SOURCE | /bin/grep "PV Name" | /bin/cut -d'/' -
f4`
do
      DEV_SOURCE_LIST=$DEV_SOURCE_LIST$PV" "
done

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

for PV in `/usr/sbin/vgdisplay -v $VG_TARGET | /bin/grep "PV Name" | /bin/cut -d'/' -
f4`
do
        SANID_TARGET_LIST=$SANID_TARGET_LIST`/sbin/setsp | /bin/grep $PV | /bin/grep
"ID" | /bin/awk '{print $8}'| /bin/cut -d\" -f1`" "
done

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

# Volume group target minor number rescue
MINOR=`/bin/ls -l $VG_TARGET | /bin/awk ‘{print $6}’ | /bin/tail -n1 | /bin/cut -c3,4`

# All logical volumes target deactivation
# All mounting points umounting
for MPOINT in `df | /bin/grep $VG_TARGET | /bin/awk ‘{print $1}’`
do
      /usr/sbin/unmount $MPOINT
done
# Removing
for LV in `/usr/sbin/vgdisplay -v $VG_TARGET | /bin/grep “LV Name” | /bin/cut -d’/’ -
f4`
do
      /usr/sbin/lvremove -f $VG_TARGET/$LV
done

# Volume group target deactivation
/usr/sbin/vgchange -a n $VG_TARGET

# Volume group target removing
/usr/sbin/vgexport $VG_TARGET

# Snapshot
/bin/echo “Snapshot ...”
IND=1
while [ $IND -le $NS ]
do
      SOURCE=`/bin/echo $DEV_SOURCE_LIST | /bin/awk ‘{print $’$IND’}’`
      TARGET=`/bin/echo $DEV_TARGET_LIST | /bin/awk ‘{print $’$IND’}’`
      echo snap -source /dev/rdsk/$SOURCE -target /dev/rdsk/$TARGET
      IND=`expr $IND + 1`
done | $SIBADMIN 1>/dev/null 2>&1

# Volume group target rebuilding
/bin/mkdir $VG_TARGET
/usr/sbin/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/”`/bin/echo $DEV_TARGET_LIST |
/bin/awk ‘{print $’$IND’}’`” “
      IND=`expr $IND + 1`
done
/usr/sbin/vgimport -m /tmp/$VG_SOURCE_NAME.map $VG_TARGET $BDEV_TARGET_LIST
```

```
# Reactivation of the volume group target
/usr/sbin/vgchange -a y $VG_TARGET

# Configuration of the volume group target backup
/usr/sbin/vgcfgbackup $VG_TARGET

# Volume group target deactivation
/usr/sbin/vgchange -a n $VG_TARGET

# Volume group target removing
/usr/sbin/vgexport $VG_TARGET

# Copy the map file to the backup server
/bin/cp /tmp/$VG_SOURCE_NAME.map /tmp/$VG_TARGET_NAME.map
/bin/rcp /tmp/$VG_TARGET_NAME.map $BACKUP_SERVER:/tmp/$VG_TARGET_NAME.map
/bin/touch /tmp/$VG_TARGET_NAME.SANID_list
/bin/echo $SANID_TARGET_LIST > /tmp/$VG_TARGET_NAME.SANID_list
/bin/rcp /tmp/$VG_TARGET_NAME.SANID_list $BACKUP_SERVER:/tmp/$VG_TARGET_NAME.SANID_list

# Exclude the target devices from the production server view
/bin/echo "Excluding the target devices from the production server view"
IND=1
while [ $IND -le $NT ]
do
        SANID=`/bin/echo $SANID_TARGET_LIST | /bin/awk '{print $'$IND'}'`
        SPD=`/sbin/setsp | /bin/grep $SANID | /bin/awk '{print $1}' | /bin/cut -dd -
f2`
        /sbin/setsp -e1 -l$SPD
        IND=`expr $IND + 1`
done

# Volume group target rebuilding on the backup server
/bin/echo "Rebuilding the target volume group on the backup server $BACKUP_SERVER"
/bin/remsh $BACKUP_SERVER $SCRIPTS_DIR/rebuild_vg $VG_TARGET

exit 0
```

rebuild_vg on taillon

```
#!/bin/sh

# Synopsis of the rebuild_vg script
synopsis()
{
      print "Synopsis"
      print "########"
      print ""
      print "rebuild_vg Volum_Group_Name_Target"
      print ""
      print "Examples : rebuild_vg /dev/vgtarget"
}

# Parameters verification
if [ $# -ne 1 ]
then
      synopsis
      exit 1
else
        if [ `print $1 | cut -d'/' -f2` != "dev" ]
        then
```

```
                synopsis
                exit 1
        else
                VG_TARGET=$1
                VG_TARGET_NAME=`print $VG_TARGET | cut -d'/' -f3`
        fi
fi

# Initialization
MAP_FILE="/tmp/$VG_TARGET_NAME.map"
SANID_list_file=/tmp/$VG_TARGET_NAME.SANID_list
/bin/egrep -v '^#|^$' $SANID_list_file |
while read SANID
do
      SANID_TARGET_LIST=$SANID_TARGET_LIST$SANID" "
done
print $SANID_TARGET_LIST

print $SANID_TARGET_LIST | /bin/awk '{print NF}' | read NT

# include the target devices on the backup server
echo "Including the target devices on the backup server"
IND=1
while [ $IND -le $NT ]
do
      SANID=`print $SANID_TARGET_LIST | /bin/awk '{print $'$IND'}'`
        SPD_backup=`/sbin/setsp | /bin/grep $SANID | /bin/awk '{print $1}' | /bin/cut
-dd -f2`
        /sbin/setsp -e0 -l$SPD_backup
        IND=`expr $IND + 1`
done

# Determine the MINOR number
ls -l /dev/vg* | /bin/grep root | /bin/awk '{print $6}' | cut -c3,4 | sort -u | tail -
1 | read MINOR
MINOR=`expr $MINOR + 1 `
echo MINOR=$MINOR

# Rebuild the target volume group
mkdir $VG_TARGET
/usr/sbin/mknod $VG_TARGET/group c 64 "0x0"$MINOR"0000"

BDEV_TARGET_LIST=""
IND=1
while [ $IND -le $NT ]
do
        SANID=`print $SANID_TARGET_LIST | /bin/awk '{print $'$IND'}'`
        BDEV_TARGET_LIST=$BDEV_TARGET_LIST"/dev/dsk/"`/sbin/setsp | /bin/grep $SANID |
/bin/awk '{print $3}'`" "
        IND=`expr $IND + 1`
done
#/usr/sbin/vgimport -m $MAP_FILE $VG_TARGET $BDEV_TARGET_LIST
echo /usr/sbin/vgimport $VG_TARGET $BDEV_TARGET_LIST
/usr/sbin/vgimport $VG_TARGET $BDEV_TARGET_LIST

# Reactivation of the volume group target on the backup server
/usr/sbin/vgchange -a y $VG_TARGET

# Configuration of the volume group target backup
/usr/sbin/vgcfgbackup $VG_TARGET

exit 0
```

mount on taillon

```
#!/bin/sh

# Synopsis of the mount script
synopsis()
{
       /bin/echo “Synopsis”
       /bin/echo “########”
       /bin/echo “”
       /bin/echo “mount Volum_Group_Name_Source Volum_Group_Name_Target
mounting_point_target”
       /bin/echo “”
       /bin/echo “Examples : mount /dev/vg01 /dev/vg03 /mounting_point_target”
}

PROD_SERVER=lhers

# Parameters verification
if [ $# -ne 3 ]
then
       synopsis
       exit 1
else
         if [ `/bin/echo $1 | /bin/cut -d’/’ -f2` != “dev” ]
         then
                  synopsis
                  exit 1
         else
                  if [ `/bin/echo $2 | /bin/cut -d’/’ -f2` != “dev” ]
                  then
                           synopsis
                           exit 1
                  else
                           VG_SOURCE=$1
                           VG_TARGET=$2
                           VG_SOURCE_NAME=`remsh $PROD_SERVER “/bin/echo $VG_SOURCE |
/bin/cut -d’/’ -f3”`
                           VG_TARGET_NAME=`/bin/echo $VG_TARGET | /bin/cut -d’/’ -f3`
                           MOUNTING_BASE=$3
                  fi
         fi
fi

# Don’t initialize those variables
DEV_TARGET_LIST=””
LV_SOURCE_LIST=””

# Volumes Group existence verification
/usr/sbin/vgdisplay $VG_TARGET > /dev/null 2> /dev/null
if [ $? -ne 0 ]
then
       /bin/echo $VG_TARGET” doesn’t exist or is not activated !”
       exit 2
fi

# Directories tree target building and mounting
/bin/echo “Mouting $VG_TARGET_NAME on $MOUNTING_BASE”
/usr/bin/mkdir -p $MOUNTING_BASE
/bin/echo > $MOUNTING_BASE/mount.log
```

```
# Directories tree target building
for LV in `/usr/sbin/vgdisplay -v $VG_TARGET | /bin/grep “LV Name” | /bin/cut -d’/’ -
f4`
do
      # Checking
      /bin/echo “Checking the file system ...”
      /usr/sbin/fsck -F `fstyp $VG_TARGET/$LV` $VG_TARGET/$LV
      # Mounting point building
      /bin/echo “Building mounting point”
      MPOINT_LIST=`/usr/bin/remsh $PROD_SERVER df | /bin/grep $VG_SOURCE/$LV |
/bin/awk ‘{print $1}’`
      for MPOINT in $MPOINT_LIST
      do
               /usr/bin/mkdir -p $MOUNTING_BASE$MPOINT
               /usr/sbin/mount -F `fstyp $VG_TARGET/$LV` $VG_TARGET/$LV
$MOUNTING_BASE$MPOINT
               /bin/echo “mount -F `fstyp $VG_TARGET/$LV` $VG_TARGET/$LV
$MOUNTING_BASE$MPOINT” >> $MOUNTING_BASE/mount.log
      done
done

exit 0
```

unmount on taillon

```
#!/bin/sh

# Synopsis of the unmount script
synopsis()
{
      /bin/echo “Synopsis”
      /bin/echo “########”
      /bin/echo “”
      /bin/echo “unmount Volume_Group_Name”
      /bin/echo “”
      /bin/echo “Examples : unmount /dev/vg01”
}

PROD_SERVER=lhers
SCRIPTS_DIR=/export/scripts
# Don’t initialize those variables
PV_LIST=””
SANID_LIST=””

# Parameter number verification
if [ $# -ne  1 ]
then
      synopsis
      exit 1
else
      if [ `/bin/echo $1 | /bin/cut -d’/’ -f2` != “dev” ]
      then
               synopsis
               exit 1
      else
               VOLUME_GROUP=$1
               VOLUME_GROUP_NAME=`/bin/echo $VOLUME_GROUP | /bin/cut -d’/’ -f3`
      fi
```

```
fi
# Volume Group existence verification
/usr/sbin/vgdisplay $VOLUME_GROUP > /dev/null 2> /dev/null
if [ $? -ne 0 ]
then
      /bin/echo $VOLUME_GROUP" doesn't exist or is not activated !"
      exit 2
fi

# SANID of the source and target devices extraction
for PV in `/usr/sbin/vgdisplay -v $VOLUME_GROUP | /bin/grep "PV Name" | /bin/cut -d'/'
-f4`
do
        SANID_LIST=$SANID_LIST`/sbin/setsp | /bin/grep $PV | /bin/grep "ID" | /bin/awk
'{print $8}'| /bin/cut -d\" -f1`" "
done

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

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

# Volume group deactivation
/usr/sbin/vgchange -a n $VOLUME_GROUP

# Volume group removing
/usr/sbin/vgexport $VOLUME_GROUP

NP=`/bin/echo $SANID_LIST | /bin/awk '{print NF}'`

# Exclude the target devices from the backup server view
/bin/echo "Excluding the target devices from the backup server view"
IND=1
while [ $IND -le $NP ]
do
        SANID=`/bin/echo $SANID_LIST | /bin/awk '{print $'$IND'}'`
        SPD=`/sbin/setsp | /bin/grep $SANID | /bin/awk '{print $1}' | /bin/cut -dd -
f2`
        /sbin/setsp -e1 -l$SPD
        IND=`expr $IND + 1`
done

# Volume group target rebuilding on the production server
/bin/echo "Rebuilding the target volume group on the production server $PROD_SERVER"
/bin/remsh $PROD_SERVER $SCRIPTS_DIR/rebuild_vg $VOLUME_GROUP

exit 0
```

rebuild_vg on lhers

```
#!/bin/sh

# Synopsis of the rebuild_vg script
synopsis()
{
       print “Synopsis”
       print “########”
       print “”
       print “rebuild_vg Volum_Group_Name_Target”
       print “”
       print “Examples : rebuild_vg /dev/vgtarget”
}

# Parameters verification
if [ $# -ne 1 ]
then
       synopsis
       exit 1
else
         if [ `print $1 | cut -d’/’ -f2` != “dev” ]
         then
                  synopsis
                  exit 1
         else
                  VG_TARGET=$1
                  VG_TARGET_NAME=`print $VG_TARGET | cut -d’/’ -f3`
         fi
fi

# Initialization
MAP_FILE=”/tmp/$VG_TARGET_NAME.map”
SANID_list_file=/tmp/$VG_TARGET_NAME.SANID_list
/bin/egrep -v ‘^#|^$’ $SANID_list_file |
while read SANID
do
       SANID_TARGET_LIST=$SANID_TARGET_LIST$SANID” “
done
print $SANID_TARGET_LIST

print $SANID_TARGET_LIST | /bin/awk ‘{print NF}’ | read NT

# include the target devices on the backup server
echo “Including the target devices on the backup server”
IND=1
while [ $IND -le $NT ]
do
       SANID=`print $SANID_TARGET_LIST | /bin/awk ‘{print $’$IND’}’`
         SPD_backup=`/sbin/setsp | /bin/grep $SANID | /bin/awk ‘{print $1}’ | /bin/cut
-dd -f2`
         /sbin/setsp -e0 -l$SPD_backup
         IND=`expr $IND + 1`
done

# Determine the MINOR number
ls -l /dev/vg* | /bin/grep root | /bin/awk ‘{print $6}’ | cut -c3,4 | sort -u | tail -
1 | read MINOR
MINOR=`expr $MINOR + 1 `
echo MINOR=$MINOR
```

```
# Rebuild the target volume group
mkdir $VG_TARGET
/usr/sbin/mknod $VG_TARGET/group c 64 "0x0"$MINOR"0000"

BDEV_TARGET_LIST=""
IND=1
while [ $IND -le $NT ]
do
        SANID=`print $SANID_TARGET_LIST | /bin/awk '{print $'$IND'}'`
        BDEV_TARGET_LIST=$BDEV_TARGET_LIST"/dev/dsk/"`/sbin/setsp | /bin/grep $SANID |
/bin/awk '{print $3}'`" "
        IND=`expr $IND + 1`
done
#/usr/sbin/vgimport -m $MAP_FILE $VG_TARGET $BDEV_TARGET_LIST
echo /usr/sbin/vgimport $VG_TARGET $BDEV_TARGET_LIST
/usr/sbin/vgimport $VG_TARGET $BDEV_TARGET_LIST

# Reactivation of the volume group target on the backup server
/usr/sbin/vgchange -a y $VG_TARGET

# Configuration of the volume group target backup
/usr/sbin/vgcfgbackup $VG_TARGET

exit 0
```

release on lhers

```
#!/bin/sh

# Synopsis of the release script
synopsis()
{
      /bin/echo "Synopsis"
      /bin/echo "########"
      /bin/echo ""
      /bin/echo "release Volume_Group_Name_Target"
      /bin/echo ""
      /bin/echo "Examples : release /dev/vg01"
}

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

# Don't initialime those variables
PV_LIST=""

# Parameter number verification
if [ $# -ne  1 ]
then
      synopsis
      exit 1
else
      if [ `/bin/echo $1 | /bin/cut -d'/' -f2` != "dev" ]
      then
               synopsis
               exit 1
      else
               VOLUME_GROUP=$1
```

```
                VOLUME_GROUP_NAME=`/bin/echo $VOLUME_GROUP | /bin/cut -d'/' -f3`
      fi
fi

# Volume Group existence verification
/usr/sbin/vgdisplay $VOLUME_GROUP > /dev/null 2> /dev/null
if [ $? -ne 0 ]
then
        /bin/echo $VOLUME_GROUP" doesn't exist or is not activated !"
        exit 2
fi

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

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

# All logical volumes deactivation
# All mounting points umounting
for MPOINT in `df | /bin/grep $VOLUME_GROUP | /bin/awk '{print $1}'`
do
        /usr/sbin/unmount $MPOINT
done

# Volume group deactivation
/usr/sbin/vgchange -a n $VOLUME_GROUP

# Volume group removing
/usr/sbin/vgexport $VOLUME_GROUP

# Release
/bin/echo "Release ..."
IND=1
while [ $IND -le $NPV ]
do
      DEV=`/bin/echo $PV_LIST | /bin/awk '{print $'$IND'}'`
      /bin/echo release -force -path /dev/rdsk/$DEV
      IND=`expr $IND + 1`
done | $SIBADMIN 1>/dev/null 2>&1

# Volume group rebuilding
/bin/mkdir $VOLUME_GROUP
/usr/sbin/mknod $VOLUME_GROUP/group c 64 "0x0"$MINOR"0000"

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

BDEVICE_LIST=""
```

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

/usr/sbin/vgcreate $VOLUME_GROUP $BDEVICE_LIST

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

exit 0
```

APPENDIX J: EXAMPLE OF SUCCESSFUL BACKUP (2 SERVERS)

On the backup server

```
BEGIN OF STK SVA BACKUP - Wed Dec  4 14:33:44 WET 2002

== Pre-processing Script starting on taillon (Strategy A) - Wed Dec 4 14:33:44 WET
2002

Initializing variables
Calling the Strategy A on production server lhers (snap) - Wed Dec 4 14:33:44 WET 2002
************ Calling the mount script - Wed Dec  4 14:34:22 WET 2002 *************
Mouting vgtarget on /target
Checking the file system ...
log replay in progress
replay complete - marking super-block as CLEAN
Building mounting point
******* End of execution of the mount script - Wed Dec  4 14:34:24 WET 2002 *********

== End of the Preprocessing script on the backup server (Strategy A) - Wed Dec  4
14:34:24 WET 2002


======================== Post-processing Script starting on the backup server (Strategy
A) - Wed Dec  4 14:35:03 WET 2002

Initializing variables
./lunconfig is the LUN configuration file
******** Calling the unmount script - Wed Dec  4 14:35:03 WET 2002 ****************
Logical volume "/dev/vgtarget/lvol1" has been successfully removed.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
Volume group "/dev/vgtarget" has been successfully changed.
Excluding the target devices from the backup server view
Rebuilding the target volume group on the production server lhers
Including the target devices on the production server
Warning: A backup of this volume group may not exist on this machine.
Please remember to take a backup using the vgcfgbackup command after activating the
volume group.
Activated volume group
Volume group "/dev/vgtarget" has been successfully changed.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
****** End of execution of the unmount script - Wed Dec  4 14:35:06 WET 2002
**********

****** Calling the Strategy B on production server lhers (release) - Wed Dec  4
14:35:06 WET 2002 **************

========== End of the Postprocessing script - Wed Dec  4 14:35:06 WET 2002

END OF STK SVA BACKUP - SUCCESSFUL - Wed Dec  4 14:35:06 WET 2002
```

On the production server

```
== Pre-processing Script on lhers starting (Strategy A) - Wed Dec  4 14:32:28 WET 2002

Initializing variables
********** Calling the snap script - Wed Dec  4 14:32:28 WET 2002 *****************
Logical volume "/dev/vgtarget/lvol1" has been successfully removed.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
Volume group "/dev/vgtarget" has been successfully changed.
Snapshot ...
Warning: A backup of this volume group may not exist on this machine.
Please remember to take a backup using the vgcfgbackup command after activating the
volume group.
Activated volume group
Volume group "/dev/vgtarget" has been successfully changed.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
Volume group "/dev/vgtarget" has been successfully changed.
Excluding the target devices from the production server view
Rebuilding the target volume group on the backup server taillon
Including the target devices on the backup server
Warning: A backup of this volume group may not exist on this machine.
Please remember to take a backup using the vgcfgbackup command after activating the
volume group.
Activated volume group
Volume group "/dev/vgtarget" has been successfully changed.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
****** End of execution of the snap script - Wed Dec  4 14:32:42 WET 2002 ***********


== End of the Preprocessing on lhers script (Strategy A) - Wed Dec 4 14:32:43 WET 2002


=== Pre-processing Script on lhers starting (Startegy B) - Wed Dec 4 14:36:12 WET 2002

Initializing variables
******* Calling the release script - Wed Dec  4 14:36:12 WET 2002 ***************
Volume group "/dev/vgtarget" has been successfully changed.
Release ...
Physical volume "/dev/rdsk/c10t12d3" has been successfully created.
Physical volume "/dev/rdsk/c10t12d4" has been successfully created.
Increased the number of physical extents per physical volume to 1173.
Volume group "/dev/vgtarget" has been successfully created.
Volume Group configuration for /dev/vgtarget has been saved in
/etc/lvmconf/vgtarget.conf
Volume group "/dev/vgtarget" has been successfully changed.
***** End of execution of the release script - Wed Dec  4 14:36:27 WET 2002 **********


== End of the Preprocessing script on lhers (Strategy B) - Wed Dec 4 14:36:27 WET 2002
```

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

MS 9150 B 04/04