SUN BLUEPRINTS [illegible] APPROVED

JumpStart™ Mechanics: Using JumpStart Application for Hands-Free Installation of Unbundled Software - Part 2

Automatic Encapsulation of the Root Disk

By John S. Howard - Enterprise Engineering

Sun BluePrints™ OnLine - June 2000

Sun microsystems
ENTERPRISE ENGINEERING

http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-5684-10
Revision 01, June 2000

Please
Recycle

Adobe PostScript™

JumpStart™ Mechanics: Using JumpStart Application for Hands-Free Installation of Unbundled Software - Part 2

Introduction

Automating and standardizing the installation of the Solaris™ Operating Environment along with the associated unbundled software products and datacenter management tools is one of the largest challenges facing System Administrators managing large environments or datacenters. In this article, a procedure for automating the initial configuration of Sun Enterprise Volume Manager™ (SEVM) software and encapsulation of the root disk will be presented.

As encapsulation is a complicated process, a brief overview of encapsulation will also be presented. This article will only address basic configuration and encapsulation of the root disk, and is meant to provide an examination of advanced techniques for installation of the Solaris Operating Environment and JumpStart™ Server configuration.The techniques and procedures presented here are meant to be a starting point for your storage configuration. A reference configuration for SEVM controlled boot disks will be presented in a future article.

This article uses the terms boot disk and root disk interchangeably to refer to the physical disk that the Solaris Operating Environment has been installed on and is the default boot device.

Assumptions

This article assumes that the reader has a moderate level of experience with both JumpStart technology and the SEVM software. Additionally, it is assumed that you have a JumpStart server configured to install the Solaris Operating Environment and SEVM 2.6, as explained in Part 1 of this series. For clarity and simplicity, throughout this article we will create the `rootdg` disk group containing only an encapsulated boot disk. Extending and augmenting the following scripts and procedures is left as an exercise for the reader: mirroring the boot disk, adding hotspares, configuring recovery disks to `rootdg`, and creating and populating additional disk groups.

This article presents several scripts and procedures, it is important to keep in mind that these scripts are not "production ready". These scripts are presented to demonstrate a technique, and as such, the scripts will contain limited or no error checking.

A Brief Overview of Encapsulation

Encapsulation is the method by which the SEVM software takes over management of a disk which has data that needs to be preserved. The most common use of encapsulation is to put the Solaris Operating Environment boot disk under SEVM control. Placing the boot disk under SEVM software control is desired to mirror the boot disk, to provide a higher level of reliability, availability and serviceability (RAS) of the system.

All disks under SEVM software control (whether encapsulated or initialized) are divided into 2 regions:

1. The public region, used for SEVM's volumes, subdisks, and other virtual devices.
2. The private region, used by volume manager for internal storage of its configuration information.

When encapsulating a disk, the SEVM software will require a small amount of space (typically several cylinders) for its private region. The challenge that the SEVM software faces is that all space on the boot disk may have already been allocated. In this case, the SEVM software will steal several cylinders from the swap partition and appropriately mask off the stolen area by creating a `rootdiskPriv` subdisk. Encapsulation is usually accomplished by running `vxinstall` or `vxdiskadm` and responding to the prompts.

A Look Under the Hood of Encapsulation

Due to being extremely interactive and difficult to script, the usual methods of encapsulation are not viable for execution from a JumpStart server finish script. In order to provide a hands-free method of invoking encapsulation we will examine two techniques: hand-crafting SEVM control files directly and using `/usr/lib/vxvm/bin/vxencap` (the SEVM encapsulation preparation script) to build the control files.

The actual work of encapsulation is performed by `/etc/init.d/vxvm-reconfig` at system boot. `vxvm-reconfig` is driven by the contents or presence of several control files in the `/etc/vx/reconfig.d` directory.

Licensing

It is important to remember that the SEVM software requires a license, that license can be in the form of hardware such as a Sun StorEdge™ A5200, or a software license key. Regardless of which licensing method is in use at your site, the SEVM software must be licensed before attempting to encapsulate your boot disk.

If the software license method is in use at your site, the scripts and procedures presented here should be modified to license the SEVM product before any SEVM commands are executed or configuration is done. To create a software license, the finish script should be modified to either run `vxserial`, or manually create the license key in the `/etc/vx/elm` directory.

`vxvm-reconfig`, The Hardest Working Script in SEVM

All of the actual work of encapsulation, such as creating the private region and creating any needed subdisks is done by `vxvm-reconfig`. In brief, `vxvm-reconfig` checks for the presence of several flag files and control files in `/etc/vx/reconfig.d`. Based upon the presence and contents of these files `vxvm-reconfig` will bring disks under SEVM software control and make other

SEVM software configuration changes. An in-depth examination of vxvm-reconfig is outside the scope of this article, however a careful reading of the script will provide insight into the details of SEVM software start-up.

vxencap Created Control Files

Creation of the commands and the encapsulated disk layout are done by the vxencap script. By simply executing vxencap and specifying the desired disk media name (rootdisk), the boot disk will take care of the set-up for encapsulation:

```
/usr/lib/vxvm/bin/vxencap rootdisk=c0t0d0
```

vxencap is the preferred method for scripting encapsulation, it is a standard SEVM utility complete with a man page. However, vxencap does not function as expected from a JumpStart server finish script. The reason that vxencap should not be used from the finish script is due to vxparms, one of the utilities used by vxencap to query the environment and kernel. vxparms functions correctly from the finish script, however, it is returning information relative to the currently booted installation mini-root environment. The mini-root environment is different in several key areas such as device major/minor numbers and device drivers. For example, the SEVM device drivers are not loaded in the JumpStart server mini-root.

It is possible to build a mini-root with the SEVM drivers installed, such as the MR system, that is served by the JumpStart server. However, working around the device numbering issues becomes very problematic and inefficient in terms of time. The method that is easiest to maintain, is to have the JumpStart finish script add a start-up script to the client system, such as /etc/rc3.d/S99encap-root, to be executed whenever the client system enters run-level 3. The function of S99encap-root is to test for the presence of a flag file. If this flag file exists, then the S99encap-root script will execute vxencap and initiate a re-boot. At that re-boot, vxvm-reconfig will complete the actual work of encapsulating the boot disk.

When using vxinstall to encapsulate the boot disk and configure SEVM, vxinstall gathers information on disks and disk controllers currently attached to the system, and decides specifically what disks should be encapsulated and what disks need to be initialized. Our boot disk is c0t0d0, but there is one other disk on the c0 controller - c0t1d0. We will need to hand-craft the files instructing SEVM to ignore c0t1d0.

Having `vxvm-reconfig` ignore disks, is accomplished by listing the disks to be ignored in a file that is named for the controller, and then listing the controllers containing disks to be ignored in the `/etc/vx/reconfig.d/cntrls` file. For example, the following two commands will create and populate the necessary files to ignore `c0t1d0`:

```
echo c0t1d0 >/etc/vx/reconfig.d/c0
echo c0 >/etc/vx/reconfig.d/cntrls
```

Finally, we need to signal vxvm-reconfig to reconfigure on the next reboot:

```
touch /etc/vx/reconfig.d/state.d/reconfig
```

Automated Encapsulation After Installation

The JumpStart finish script used to create S99encap-root and the flag file is:

```
#!/bin/sh

BASEDIR=/a
SLASHAMNT=${BASEDIR}/mnt
VXPRODUCT=${SLASHAMNT}/Product
VXPATCH=${SLASHAMNT}/Patches
OK=0
NOTOK=1

mount -F nfs blackmesa:/JumpStart/sun_sevm_2_6 ${SLASHAMNT}
if [ $? != 0 ]; then
        exit ${NOTOK}
fi

echo "Installing SUNWvxvm..."
pkgadd -a ${SI_CONFIG_DIR}/noask -d ${VXPRODUCT} \
  -R ${BASEDIR} SUNWvxvm
echo "Installing SUNWvxva..."
pkgadd -a ${SI_CONFIG_DIR}/noask -d ${VXPRODUCT} \
  -R ${BASEDIR} SUNWvxva
echo "Installing SUNWasevm..."
pkgadd -a ${SI_CONFIG_DIR}/noask \
  -r ${SI_CONFIG_DIR}/SUNWasevm.response \
  -d ${VXPRODUCT} -R ${BASEDIR} SUNWasevm
echo "Installing SUNWvmsa..."
pkgadd -a ${SI_CONFIG_DIR}/noask \
  -r ${SI_CONFIG_DIR}/SUNWvmsa.response \
  -d ${VXPRODUCT} -R ${BASEDIR} SUNWvmsa
echo "Installing SUNWvmman..."
pkgadd -a ${SI_CONFIG_DIR}/noask -d ${VXPRODUCT} \
  -R ${BASEDIR} SUNWvmman
echo "Installing SUNWvmdev..."
pkgadd -a ${SI_CONFIG_DIR}/noask -d ${VXPRODUCT} \
  -R ${BASEDIR} SUNWvmdev
echo "Preparing for encapsulation of c0t0d0 ..."
#
# If needed, setup software license would go here
#
```

```
# Create any disk exclusion and inclusion files

#
cat > ${BASEDIR}/etc/rc3.d/S99encap-root <<EOENCAP-ROOT
#!/bin/sh

if [ ! -f /.S99encap-root-flag ]; then
        exit 0
fi

if [ ! -f /etc/vx/reconfig.d/state.d/install-db ]; then
        gettext "ERROR: S99encap-root: VxVM appears to be \
          already configured.\n"
        gettext "TO FIX: this setup program will not attempt\
          to reinitialize the system.\n"
        exit 1
fi

# Verify the disk we are operating on is c0t0d0s0.
gettext "Verifying root device ... "
A=“df -k / | awk ’/^/dev// {print \$1}’“
if [ "\$A" != "/dev/dsk/c0t0d0s0" ]; then
        echo
        gettext "ERROR: S99encap-root: current boot device \
          must be /dev/dsk/c0t0d0s0.\n"
        gettext "TO FIX: ensure system is not yet configured \
          with VxVM.\n"
        exit 1
fi
gettext "ok.\n"

# Setup root disk for encapsulation, the actual encapsulation
# occurs at the next reboot.
/usr/lib/vxvm/bin/vxencap rootdisk=c0t0d0
echo c0t0d0 >/etc/vx/reconfig.d/disks-cap-part
echo c0t1d0 >/etc/vx/reconfig.d/c0
echo c0 >/etc/vx/reconfig.d/cntrls
touch /etc/vx/reconfig.d/state.d/init-cap-part
touch /etc/vx/reconfig.d/state.d/reconfig
#
rm -f /.S99encap-root-flag
/sbin/init 6
EOENCAP-ROOT
chmod 755 ${BASEDIR}/etc/rc3.d/S99encap-root
touch ${BASEDIR}/.S99encap-root-flag
exit ${OK}
```

This method adds one reboot to the installation process as we: boot off the JumpStart server to install the Solaris Operating Environment, reboot off of the newly installed disk, then reboot a third time after the `vxencap` completes. However this is certainly an equitable trade-off for having an automated, documented, and easily maintainable method for installation and encapsulation.

Hand-Crafting the Control Files

By hand-crafting the control files in `/etc/vx/reconfig.d` from the JumpStart server's finish script, we can cause encapsulation to occur on the next reboot after the Solaris Operating Environment installation completes. This technique is only presented to give understanding to the procedure of encapsulation, for reasons that shall soon become painfully apparent, this technique is not recommended.

The primary purpose of `vxencap` is to map out the specified disk to be encapsulated. This mapping process involves specifying the subdisk layout information for the soon-to-be encapsulated disk and saving a copy of the disk's current volume table of contents (VTOC). This saved VTOC is required if the disk is ever to be unencapsulated and reverted to the underlying partitions. Additionally, the disk media name that SEVM should use for the device will need to be specified. For every device to be encapsulated (controlled by the disk names listed in `/etc/vx/reconfig.d/disks-cap-part`), these items are stored in three files in a subdirectory named for the device, under the `/etc/vx/reconfig.d/disk.d` directory. For example, we will be encapsulating `/dev/dsk/c0t0d0`, the encapsulation control files that need to be created are:

File Name	Contents
`/etc/vx/reconfig.d/disks-cap-part`	The disk(s) device names to be encapsulated
`/etc/vx/reconfig.d/disk.d/c0t0d0`	Directory
`/etc/vx/reconfig.d/disk.d/c0t0d0/dmname`	The media name for this disk
`/etc/vx/reconfig.d/disk.d/c0t0d0/newpart`	Subdisk layout
`/etc/vx/reconfig.d/disk.d/c0t0d0/vtoc`	VTOC

The contents of both the `newpart` and `vtoc` file are dependant on the underlying disk geometry. This implies that a unique file will need to be hand crafted for each disk geometry to be encapsulated. The SEVM utility `vxprtvtoc` can be used to generate the `vtoc` file, however the subdisk layout information and offsets will still need to be manually calculated and created for each disk.

Obviously, this procedure is a manual, labor intensive process that is extremely susceptible to human error. Additionally, this procedure requires the System Administrator to be intimately aware of the disk geometry of every disk to be encapsulated. As this procedure has so many opportunities for error and very few checkpoints for verification, this technique can not be recommended.

Automated Encapsulation From the Finish Script

Following is the JumpStart server finish script from Part 1 of this series, augmented to automatically encapsulate the root disk (`c0t0d0`) after installation of Solaris 2.6 Operating Environment and the SEVM 2.6 software. The remainder of this article assumes that the JumpStart server has been configured as specified in Part 1. Also, as in Part 1, the JumpStart server hostname is `blackmesa` and the install client hostname is `ravenswood`. Licensing is taken care of by the Sun StorEdge A5200's attached to `ravenswood`.

As with any installation procedure or script, this should be thoroughly tested and understood before using. Initialization of disks by SEVM re-partitions the disks, irrevocably erasing all pre-existing data on the disks.

The finish script we are using for this example is:

```
#!/bin/sh

BASEDIR=/a
SLASHAMNT=${BASEDIR}/mnt
VXPRODUCT=${SLASHAMNT}/Product
VXPATCH=${SLASHAMNT}/Patches
RECON=${BASEDIR}/etc/vx/reconfig.d
DISKD=${RECON}/disk.d
STATED=${RECON}/state.d
ROOTDEV=c0t0d0
OK=0
NOTOK=1
```

```
mount -F nfs blackmesa:/JumpStart/sun_sevm_2_6 ${SLASHAMNT}
if [ $? != 0 ]; then
        exit ${NOTOK}
fi

echo "Installing SUNWvxvm..."
pkgadd -a ${SI_CONFIG_DIR}/noask -d ${VXPRODUCT} \
  -R ${BASEDIR} SUNWvxvm
echo "Installing SUNWvxva..."
pkgadd -a ${SI_CONFIG_DIR}/noask -d ${VXPRODUCT} \
  -R ${BASEDIR} SUNWvxva
echo "Installing SUNWasevm..."
pkgadd -a ${SI_CONFIG_DIR}/noask \
  -r ${SI_CONFIG_DIR}/SUNWasevm.response \
  -d ${VXPRODUCT} -R ${BASEDIR} SUNWasevm
echo "Installing SUNWvmsa..."
pkgadd -a ${SI_CONFIG_DIR}/noask \
  -r ${SI_CONFIG_DIR}/SUNWvmsa.response \
  -d ${VXPRODUCT} -R ${BASEDIR} SUNWvmsa
echo "Installing SUNWvmman..."
pkgadd -a ${SI_CONFIG_DIR}/noask -d ${VXPRODUCT} \
  -R ${BASEDIR} SUNWvmman
echo "Installing SUNWvmdev..."
pkgadd -a ${SI_CONFIG_DIR}/noask -d ${VXPRODUCT} \
  -R ${BASEDIR} SUNWvmdev
echo "Preparing for encapsulation of c0t0d0 ..."
#
# If needed, setup software license would go here
#
# Create any disk exclusion and inclusion files
#
echo ${ROOTDEV} > ${RECON}/disks-cap-part
echo c0t1d0 > ${RECON}/c0
echo c0 > ${RECON}/cntrls
#
touch ${STATED}/init-cap-part
touch ${STATED}/reconfig
#
mkdir -p ${DISKD}/${ROOTDEV}
cat > ${DISKD}/${ROOTDEV}/newpart <<EOPART
```

```
# volume manager partitioning for drive c0t0d0
 0 0x2 0x200        0  3105360
 1 0x3 0x201  3106880  1047280
 2 0x5 0x200        0  4154160
 3 0xe 0x201        0  4154160
 4 0xf 0x201  3105360     1520
 5 0x0 0x000        0        0
 6 0x0 0x000        0        0
 7 0x0 0x000        0        0
#vxmake sd rootdiskPriv disk=rootdisk offset=3105360 len=1519
putil0=PRIVATE
comment="Private region area
#vxmake vol rootvol plex=rootvol-%%00 usetype=root logtype=none
#vxmake plex rootvol-%%00 sd=rootdisk-B0,rootdisk-%%00
#vxmake sd rootdisk-%%00 disk=rootdisk offset=0 len=3105359
#vxmake sd rootdisk-B0 disk=rootdisk offset=3105359 len=1
putil0=Block0
comment="Remap of block 0
#vxvol start rootvol
#rename c0t0d0s0 rootvol
#vxmake vol swapvol plex=swapvol-%%01 usetype=swap
#vxmake plex swapvol-%%01 sd=rootdisk-%%01
#vxmake sd rootdisk-%%01 disk=rootdisk offset=3106879 len=1047280
#vxvol start swapvol
#rename c0t0d0s1 swapvol
EOPART
chroot ${BASEDIR} /usr/lib/vxvm/bin/vxprtvtoc \
  -f /etc/vx/reconfig.d/disk.d/c0t0d0/vtoc /dev/dsk/c0t0d0
echo rootdisk > ${DISKD}/${ROOTDEV}/dmname
exit ${OK}
```

Note that the contents of the `newpart` file created with the in-line input redirection (or “here is file”) to the `cat` command contains several commented commands. These commands are commented out in the `newpart` file and are referenced later by the SEVM software when the SEVM plexes and subdisks are created.

Conclusion

This article has provided the procedures to fully automate the initial configuration of SEVM and automate encapsulation of the boot disk. JumpStart application is a powerful and flexible tool that enables the system administrator to have a fully automated and customizable method to standardize installation and configuration of the Solaris Operating Environment and unbundled products. Additionally, JumpStart application provides a framework to help ensure uniformity and adherence to site standards or conventions for all Solaris Operating Environment installations. However, this power and flexibility requires careful planning and testing by the System Administrator.

This article covered basic configuration and encapsulation of the boot disk, and is meant to provide an examination of advanced JumpStart configuration techniques. The techniques and procedures presented here are a starting point for your storage configuration. These procedures are the foundation for building an automated implementation of reference configurations to be presented in future articles.

Acknowledgements

Richard Elling provided scripting assistance and his questions served as the impetus for writing this article.

Author's Bio: John S. Howard

John S. Howard is currently a Staff Engineer in the Enterprise Engineering group at Sun Microsystems in San Diego, California. He has worked as a software engineer and systems administrator for the past 19 years. Prior to Enterprise Engineering, John worked in Enterprise Services as an Area System Support Engineer for five years. As an ASSE, he was responsible for developing and performing Reliability, Accessibility, and Serviceability (RAS) studies of customer datacenters and the development of proactive Enterprise RAS Services. Prior to Sun, John held engineering positions at: The Chicago Board of Trade Clearing Corporation, Datalogics Inc, and Rand McNally. Throughout his career he has developed: pagination and publishing software, loose-leaf publishing systems, extensive SGML systems development, database publishing systems, text editors and WYSIWIG systems, and device drivers.