**Alternative : Building a SPOT from a mksysb resource**

**A** NIM SPOT is generally created from an lpp\_source. You can also create a SPOT resource from a mksysb image. This was designed to allow for NIM masters that do not have a lot of disk space available to house a whole series of different leveled SPOT resources. Using this method you can store all of your mksysb resources and just build SPOT resources as needed, then remove them once the install is complete. Let’s say we have a mksysb resource named clientA\_mksysb and will use that to create a NIM SPOT resource called clientA\_spot.

# nim -o define -t spot -a server=master -a source=clientA\_mksysb -a location=/export/nim/spot -a auto\_expand=yes -a comments='5300-05 Created from clientA\_mksysb’ clientA\_spot  
-or-  
# smitty nim\_mkres  
Next you select “SPOT” as the resource type.  
  
\* Resource Name [clientA\_spot]   
\* Resource Type spot  
\* Server of Resource [master]  
\* Source of Install Images [clientA\_mksysb]  
\* Location of Resource [/export/nim/spot/]  
Expand file systems if space needed? yes  
Comments [Created from clientA\_mksysb]

This SPOT is only good for using in conjunction with this mksysb. Do not attempt to use this SPOT resource with any other lpp\_source or mksysb image, as it can have unpredictable and even failure results. You also can not update this SPOT with any fixes, service packs, or TL updates.

NIM uses mksysb images that are taken to file. A NIM mksysb resource can either be defined from an existing mksysb file, or the NIM master can create a new mksysb image of one of his own clients. We’ll look at both processes below. Before doing that we’ll want to create a separate filesystem to hold these images. Following suit with our lpp\_source and SPOT filesystems, we’ll call this one /export/nim/mksysb, and create it the same way we created our lpp\_source and SPOT filesystems.

# crfs -v jfs2 -g nimvg -m /export/nim/mksysb -a size=3G  
-or-  
# smitty crfs

Mount the filesystem up and we’re ready to use it.

**\*\*NOTE\*\***

If any of your mksysb files are going to be greater than 2gig in size, you will want to make sure that the filesystem you create on your NIM master is a “Large File Enabled” filesystem. JFS2 filesystems are by default set with this parameter, however if you are creating jfs filesystems, you will want to take note of the appropriate choices when creating the filesystem using smitty or add the “-a bf=true” attribute in your command line. You will also want to make sure your master’s root user and the client’s root user both have the authority to create large files.

To check this you can run the following command :

# ulimit -a  
Time(seconds) unlimited  
File(blocks) 2097151  
Data(kbytes) 131072  
Stack(kbytes) 32768  
Memory(kbytes) 32768  
Coredump(blocks) 2097151  
Nofiles(descriptors) 2000

The “File(blocks)” entry is what you’re looking for. By default this is set to 1G. To change this you’ll want to run the following command :

# chuser fsize=-1 root

Changing the “fsize = 2097151” to “fsize = -1” will allow for unlimited file sizes. You will need to log out and log back in for this to take effect.

**Creating a mksysb resource from an existing mksysb file :**

Depending on how you prefer to make mksysb images, you might wish to create mksysb images locally and ftp them to your NIM master. Using a NIM client “ClientA” as an example I will log into that NIM client and run the following command into a filesystem that can hold the file.

# mksysb -i /myfs/ClientA\_mksysb

You’ll then ftp this file over to your NIM master under our generated filesystem /export/nim/mksysb. Once there we can execute our NIM commands to define it as a resource.

**From command line :**

# nim -o define -t mksysb -a server=master -a location=/export/nim/mksysb/ClientA\_mksysb -a comments="Mksysb of ClientA" ClientA\_mksysb

Notice that the file name and the resource name are the same. That is fine, and helpful because we know exactly what it is just by the name alone.

**From SMIT :**

# smitty nim\_mkres  
-or-  
# smitty nim   
=>Perform NIM Administration Tasks => Manage Resources => Define a Resource

Next, select “mksysb” as the resource type. (There are more options listed that we’ll cover later)

\* Resource Name [ClientA\_mksysb]  
\* Resource Type mksysb  
\* Server of Resource [master]  
\* Location of Resource [/export/nim/mksysb/ClientA\_mksysb]  
 Comments [Mksysb of ClientA]  
  
# lsnim -l ClientA\_mksysb  
ClientA\_mksysb:  
Class = resources  
Type = mksysb  
Arch = power  
Rstate = ready for use  
Prev\_state = ready for use  
Location = /export/mksysb/ClientA\_mksysb  
Version = 5  
Release = 3  
Mod = 0  
Oslevel\_r = 5300-05  
Alloc\_count = 0  
Server = master

**Using your nim master to create a mksysb resource of an existing client :**

Your NIM master can also create mksysb images of his clients. In order to do this the client machine must have the bos.sysmgt.nim.client fileset installed, and must have an existing /etc/niminfo file. (This is covered in the previous “Defining Nim Clients” section).

For our purposes, we are using rsh as our communication protocol. You also have the option of using ‘nimsh’ as a communication protocol, however for this guide, the NIM master has rsh permission to the client.

**From command line :**

# nim -o define -t mksysb -a mk\_image=yes -a mksysb\_flags="-i" -a source=ClientA -a location=/export/nim/mksysb/ClientA\_mksysb -a server=master ClientA\_mksysb

Since the mksysb image hasn’t already been created we need to specify more “-a <attribute>” flags than we did in the previous example.

**From SMIT :**

# smitty nim\_mkres  
-or-  
# smitty nim   
=>Perform NIM Administration Tasks => Manage Resources => Define a Resource

Next, select “mksysb” as the resource type.

\* Resource Name [ClientA\_mksysb]  
\* Resource Type mksysb  
\* Server of Resource [master]  
\* Location of Resource [/export/nim/mksysb/ClientA\_mksysb]  
 Comments [Mksysb of ClientA]  
  
 Source for Replication []  
 -OR-  
 System Backup Image Creation Options:  
 CREATE system backup image? yes  
 NIM CLIENT to backup [ClientA]  
 PREVIEW only? no  
 IGNORE space requirements? no  
 EXPAND /tmp if needed? no  
 Create MAP files? no  
 Backup extended attributes? yes  
 Number of BLOCKS to write in a single output []  
 (leave blank to use system default)  
 Use local EXCLUDE file? no  
 (specify no to include all files in backup)  
 -OR-  
 EXCLUDE\_FILES resource []

As you can see again since we are not only defining the resource, but creating the mksysb image as well, there are more options to consider. Once it is complete you will have a mksysb resource that you can view using the ‘lsnim -l’ command :

# lsnim -l ClientA\_mksysb  
ClientA\_mksysb:  
Class = resources  
Type = mksysb  
Arch = power  
Rstate = ready for use  
Prev\_state = ready for use  
Location = /export/mksysb/ClientA\_mksysb  
Version = 5  
Release = 3  
Mod = 0  
Oslevel\_r = 5300-05  
Alloc\_count = 0  
Server = master

Outside of NIM you can find a bosinst.data file in the root (/) directory of most systems. If there is not one in that location you can find one in /var/adm/ras. The purpose of the bosinst.data file is generally to run “non-prompted” installations. When you boot from the AIX Installation media you are presented with a series of choices. Every choice that you make affects the outcome and configuration of your installation (i.e. Disks in rootvg, language environments, desktop, additional packages installed...). There might be a case where you have to install a system across the hall, in a different building, or across the country. If no one is on the other side making these choices, then your NIM client will patiently sit and wait at the “Please Define the System Console” prompt for all eternity. Using a bosinst\_data resource will preset these choices for you.

Excluding the comments, a bosinst.data file will look something like this :

control\_flow:  
 CONSOLE = Default  
 INSTALL\_METHOD = overwrite  
 PROMPT = yes  
 EXISTING\_SYSTEM\_OVERWRITE = yes  
 INSTALL\_X\_IF\_ADAPTER = yes  
 RUN\_STARTUP = yes  
 RM\_INST\_ROOTS = no  
 ERROR\_EXIT =  
 CUSTOMIZATION\_FILE =  
 TCB = no  
 INSTALL\_TYPE =  
 BUNDLES =  
 SWITCH\_TO\_PRODUCT\_TAPE =  
 RECOVER\_DEVICES = Default  
 BOSINST\_DEBUG = no  
 ACCEPT\_LICENSES =  
 DESKTOP = CDE  
 INSTALL\_DEVICES\_AND\_UPDATES = yes  
 IMPORT\_USER\_VGS =  
 ENABLE\_64BIT\_KERNEL = yes  
 CREATE\_JFS2\_FS = yes  
 ALL\_DEVICES\_KERNELS = yes  
 GRAPHICS\_BUNDLE = yes  
 MOZILLA\_BUNDLE = no  
 KERBEROS\_5\_BUNDLE = no  
 SERVER\_BUNDLE = no  
 ALT\_DISK\_INSTALL\_BUNDLE = no  
 REMOVE\_JAVA\_118 = no  
 HARDWARE\_DUMP = yes  
 ADD\_CDE = no  
 ADD\_GNOME = no  
 ADD\_KDE = no  
 ERASE\_ITERATIONS = 0  
 ERASE\_PATTERNS =  
  
locale:  
 BOSINST\_LANG = en\_US  
 CULTURAL\_CONVENTION = en\_US  
 MESSAGES = en\_US  
 KEYBOARD = en\_US  
  
target\_disk\_data:  
 PVID = 000048eda0243fa5  
 PHYSICAL\_LOCATION = U0.1-P2/Z1-A8  
 CONNECTION = scsi0//8,0  
 LOCATION = 1S-08-00-8,0  
 SIZE\_MB = 34715  
 HDISKNAME = hdisk0

You will have 1 “target\_disk\_data” stanza for every disk that will be part of the rootvg. Non root volume groups will not have “target\_disk\_data” stanzas.

For further detailed information on the rest of the options you can visit this link : <http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.install/doc/insgdrf/bosinst.data_file_stanza_desc.htm>

Notable information concerning a few bosinst\_data resource options:

1. Minimally, if you wanted to do a non-prompted installation you would need to change the “**PROMPT =** yes” to “**PROMPT =** no”.
2. **Recover\_Devices** can have 3 different values that are important when installing a mksysb image. The ‘rte’ and ‘spot copy’ installations should always have this set to “Default”.
3. yes - This will attempt to bring back certain system attributes - mainly tcpip configuration and aio settings.
4. no - This will not import in from the mksysb any of the hostname/tcpip/aio settings.
5. Default - This will only bring back the settings if being restored back to the same system it was taken from, otherwise, it will not.
6. **Install\_devices\_and\_updates** run an update\_all operation after the install is complete. For example if you have a 5300-05 lpp\_source and SPOT, but are installing a 5300-04 mksysb image, after the restore the system will run an update\_all operation against the lpp\_source, in this case updating the system to 5300-05.
7. **Create\_JFS2\_fs** - At the time of writing this document there is no supported (or unsupported) way to change a jfs filesystem to jfs2 without recreating it. Simply switching this to “yes” will not perform this conversion for you. This applies to a new installation or migration of existing jfs2 filesystems.
8. The **target\_disk\_data stanza** does not have to have all of its entries filled in. Only 1 valid entry is required to properly identify a disk. For example you could only have “HDISKNAME = hdisk0” if you know that every time you want to do a non-prompted install with this resource, that you’ll want to install to hdisk0.

Now that you are familiar with what a bosinst.data file does, we will create one. If you are looking to have a bosinst.data file to use for general purpose non-prompted overwrite installations, you can simply use the one from your NIM master. Following our previous naming we’ll create a location to hold all of our bosinst.data files that we define as resources. We will not however be creating a new filesystem to hold these as they are simply very small text files, so my / (root) filesystem should be able to handle the space.

On the NIM master from / :

# mkdir /export/nim/bosinst\_data  
# cp /bosinst.data /export/nim/bosinst\_data/bosinst.noprompt\_ow

I’ve made a copy of my bosinst.data file and labeled it so that I can recognize it as a nonprompted overwrite bosinst.data file. Now we’ll edit it to reflect what a general purpose nonprompted overwrite would use.

control\_flow:  
 CONSOLE = Default  
 INSTALL\_METHOD = overwrite  
 PROMPT = no  
 EXISTING\_SYSTEM\_OVERWRITE = yes  
 INSTALL\_X\_IF\_ADAPTER = yes  
 RUN\_STARTUP = yes  
 RM\_INST\_ROOTS = no  
 ERROR\_EXIT =  
 CUSTOMIZATION\_FILE =  
 TCB = no  
 INSTALL\_TYPE =  
 BUNDLES =  
 SWITCH\_TO\_PRODUCT\_TAPE =  
 RECOVER\_DEVICES = Default  
 BOSINST\_DEBUG = no  
 ACCEPT\_LICENSES =  
 DESKTOP = CDE  
 INSTALL\_DEVICES\_AND\_UPDATES = yes  
 IMPORT\_USER\_VGS =  
 ENABLE\_64BIT\_KERNEL = yes  
 CREATE\_JFS2\_FS = yes  
 ALL\_DEVICES\_KERNELS = yes  
 GRAPHICS\_BUNDLE = yes  
 MOZILLA\_BUNDLE = no  
 KERBEROS\_5\_BUNDLE = no  
 SERVER\_BUNDLE = no  
 ALT\_DISK\_INSTALL\_BUNDLE = no  
 REMOVE\_JAVA\_118 = no  
 HARDWARE\_DUMP = yes  
 ADD\_CDE = no  
 ADD\_GNOME = no  
 ADD\_KDE = no  
 ERASE\_ITERATIONS = 0  
 ERASE\_PATTERNS =  
  
locale:  
 BOSINST\_LANG = en\_US  
 CULTURAL\_CONVENTION = en\_US  
 MESSAGES = en\_US  
 KEYBOARD = en\_US  
  
target\_disk\_data:  
 PVID =   
 PHYSICAL\_LOCATION =   
 CONNECTION =   
 LOCATION =   
 SIZE\_MB =   
 HDISKNAME = hdisk0

Basically all I did in this case was switch the “prompt” value and blanked out the identifying “target\_disk\_data” stanzas to remove any specific reference except for the disk I want to install to. If I wanted to have the install run to 2 disks, I would simply make a second “target\_disk\_data” stanza and add “hdisk1” under that hdiskname field.

With an overwrite, migration, or preservation install you can not “pre-setup” mirroring. Having 2 target\_disk\_data stanzas in this resource will spread the rootvg over two disks.

Another scenario you might encounter, is a case where you have a mksysb image that you would like to either install back to the client it was taken from, or clone that image over to new clients. In a case where you are using a mksysb image, there is a bosinst.data file already built into that image. Allocating a bosinst\_data resource during an install will trump the bosinst.data file that is built into the mksysb image.

Also, what if you have a mksysb image that was created with the “prompt=no” already set and you don’t want the install to be non-prompted. You can extract the bosinst.data file from the mksysb image, edit it, and create a new NIM bosinst\_data resource from that extracted bosinst.data file. Though we haven’t covered mksysb resources yet, that really won’t be important for what we’re looking at here. To extract the bosinst.data file we first need to find out where the mksysb file is located. This should already be defined as a NIM resource. For this example we’ll call the mksysb resource : mksysb1.

To find the location we use the ‘lsnim -l’ command, which you should be familiar with.

# lsnim -l mksysb1  
mksysb1:  
Class = resources  
Type = mksysb  
Arch = power  
Rstate = ready for use  
Prev\_state = ready for use  
Location = /export/mksysb/mksysb1  
Version = 5  
Release = 3  
Mod = 0  
Oslevel\_r = 5300-05  
Alloc\_count = 0  
Server = master

The mksysb file is export/mksysb/mksysb1. We just need to run a restore command to extract it.

# cd /export/mksysb  
# restore -xqvf mksysb1 ./bosinst.data  
  
New volume on mksysb1:  
Cluster size is 51200 bytes (100 blocks).  
The volume number is 1.  
The backup date is: Thu Aug 18 09:13:54 CDT 2006  
Files are backed up by name.  
The user is root.  
x 1139 ./bosinst.data  
The total size is 1139 bytes.  
The number of restored files is 1.

Now, we have /export/mksysb/bosinst.data. We treat this as any other by moving it into our /export/nim/bosinst\_data directory, edit it, and define it as a resource.

Take note that the file name itself does not have to be “bosinst.data”. If that were the case then you could only have 1 bosinst.data file in any one given directory location. The system doesn’t care what the filename is so feel free to give it a descriptive name.

**Defining the bosinst\_data resource**

**From command line :**

Using the file /export/nim/bosinst\_data/bosinst.noprompt\_ow :

# nim -o define -t bosinst\_data -a location=/export/nim/bosinst\_data/bosinst.noprompt\_ow -a server=master -a comments="Non-prompted overwrite bosinst.data" bi\_noprompt\_ow

**From SMIT :**

# smitty nim\_mkres  
-or-  
# smitty nim   
=> Perform NIM Administration Tasks => Manage Resources => Define a Resource

Next, we’ll select “bosinst\_data” as the resource type.

\* Resource Name [bi\_noprompt\_ow]  
\* Resource Type bosinst\_data  
\* Server of Resource [master]  
\* Location of Resource [/export/nim/bosinst\_data/bosinst.noprompt\_ow]  
 Comments [Non-prompted overwrite bosinst.data]

Nim resources can not use certain special characters in their names. A period is one of them, so most people use underscore characters instead.

# lsnim -l bi\_noprompt\_ow  
bi\_noprompt\_ow:  
Class = resources  
Type = bosinst\_data  
Comments = Non-prompted overwrite bosinst.data  
Rstate = ready for use  
Prev\_state = unavailable for use  
Location = /export/nim/bosinst\_data/bosinst.noprompt\_ow  
Alloc\_count = 0  
Server = master

It is now ready for use with any of your installs.

Outside of NIM you can find an image.data file in the root (/) directory of most systems. If there is not one in that location you can make one using the ‘mkszfile’ command. This file contains information concerning the structure of your rootvg, the sizes of logical volumes, their corresponding mountpoint names (filesystem names), and other important information. This information can be edited to change the configuration of a system. At this time we will only be using this for altering mksysb installations. In this guide we will not be going into more advanced functions like pre-setting your own filesystem sizes, or presetting a system to be mirrored. Once you become familiar with editing this file and familiarize yourself with the bosinst.data file - you can pretty much figure out how to do that on your own. Specifics might be given in a later more advanced guide.....let’s just get through the basics first.

For simplicity, we’ll break the image.data file down into 4 sections. (I’ve removed any commented sections).

Rootvg Info :  
  
image\_data:  
 IMAGE\_TYPE= bff  
 DATE\_TIME= Tue Apr 24 14:05:05 CDT 2007  
 UNAME\_INFO= AIX shadoebso 3 5 000048ED4C00  
 PRODUCT\_TAPE= no  
 USERVG\_LIST= nimvg  
 PLATFORM= chrp  
 OSLEVEL= 5.3.0.50  
 OSLEVEL\_R= 5300-05  
 CPU\_ID= 000048ED4C00  
 LPAR\_ID=  
  
logical\_volume\_policy:  
 SHRINK= no  
 EXACT\_FIT= no  
  
ils\_data:  
 LANG= en\_US  
  
vg\_data:  
 VGNAME= rootvg  
 PPSIZE= 64  
 VARYON= no  
 VG\_SOURCE\_DISK\_LIST= hdisk1 hdisk0  
 QUORUM= 2  
 ENH\_CONC\_CAPABLE= no  
 CONC\_AUTO= no  
 BIGVG= no  
 TFACTOR= 1

As you can see there’s all sort of system information and rootvg information listed here. We can see the hostname, user volume groups, os and ml info, and disk information - all can be useful for later editing of the file, and for future possible troubleshooting

Source disk data :   
  
source\_disk\_data:  
 PVID= 000048ed24f90748  
 PHYSICAL\_LOCATION= U0.1-P2/Z1-A9  
 CONNECTION= scsi0//9,0  
 LOCATION= 1S-08-00-9,0  
 SIZE\_MB= 34715  
 HDISKNAME= hdisk1  
  
source\_disk\_data:  
 PVID= 000048eda0243fa5  
 PHYSICAL\_LOCATION= U0.1-P2/Z1-A8  
 CONNECTION= scsi0//8,0  
 LOCATION= 1S-08-00-8,0  
 SIZE\_MB= 34715  
 HDISKNAME= hdisk0

Similar to a /bosinst.data file, the /image.data file has information on what disks the rootvg currently owns. Remember, the /bosinst.data file contains information on how the rootvg was originally setup, the /image.data is the current setup. It is very important that this file be kept up to date when creating a mksysb.

**Mkszfile :**

I want to break here for just a second because this command is very important. If you create your mksysb backups in smit, the default option is already set to run this command so you don’t need to change anything. When running a mksysb from command line you use the ‘-i’ flag to make sure the /image.data file gets updated. Alternately you can execute it manually :

# mkszfile

The reason this is so important is this....

Lets say you have your original system build with your standard filesystems. You create a set of your own filesystems and throw some data in there. You then take a mksysb and do not have the /image.data file updated. The image.data file, being responsible for rebuilding the system structure during mksysb restore, has no knowledge of the added filesystems. What will happen is all of that extra data will be put into your / (root) filesystem. It will likely fill up to 100%, and your mksysb restore will fail.

There can be some cases where you do not have the image.data file updated intentionally, usually because you’ve edited it manually yourself for a specific reason, and having the ‘mkszfile’ executed will remove your changes. Be very careful about editing this file, as any simple character mistake can cause the restore to fail.

Lv\_data :   
  
lv\_data:  
 VOLUME\_GROUP= rootvg  
 LV\_SOURCE\_DISK\_LIST= hdisk0 hdisk1  
 LV\_IDENTIFIER= 000048ed00004c000000011224adbe0a.1  
 LOGICAL\_VOLUME= hd5  
 VG\_STAT= active/complete  
 TYPE= boot  
 MAX\_LPS= 512  
 COPIES= 2  
 LPs= 1  
 STALE\_PPs= 0  
 INTER\_POLICY= minimum  
 INTRA\_POLICY= edge  
 MOUNT\_POINT=  
 MIRROR\_WRITE\_CONSISTENCY= on/ACTIVE  
 LV\_SEPARATE\_PV= yes  
 PERMISSION= read/write  
 LV\_STATE= closed/syncd  
 WRITE\_VERIFY= off  
 PP\_SIZE= 64  
 SCHED\_POLICY= parallel  
 PP= 2  
 BB\_POLICY= relocatable  
 RELOCATABLE= no  
 UPPER\_BOUND= 32  
 LABEL= primary\_bootlv  
 MAPFILE=  
 LV\_MIN\_LPS= 1  
 STRIPE\_WIDTH=  
 STRIPE\_SIZE=  
 SERIALIZE\_IO= no  
 FS\_TAG=  
 DEV\_SUBTYP=

Every logical volume on the system will have an lv\_data stanza, the first always being the stanza for your hd5 - the boot logical volume. This is where, in general, most of the editing of the file is done. The main reason (and only reason we’ll cover here) to edit this stanza is to manually break mirroring. The situation would be where you have a mksysb of a mirrored rootvg and need to restore it to only 1 disk. The mksysb BOS menus do not have an option to have you restore a mirrored mksysb in this manner. To accomplish this you need to pull the image.data file from your mksysb file (similar to how we pulled the bosinst.data file from the mksysb) and edit it to manually break the mirrors. If this is a case where the system you took the mksysb from is up and running and you’re using the image to clone to another system, do NOT just go over to that system and cp the image.data file over. Always use the image.data file from the mksysb.

Fs\_data :  
  
fs\_data:  
 FS\_NAME= /  
 FS\_SIZE= 393216  
 FS\_MIN\_SIZE= 63156  
 FS\_LV= /dev/hd4  
 FS\_JFS2\_BS= 4096  
 FS\_JFS2\_SPARSE= yes  
 FS\_JFS2\_INLINELOG= no  
 FS\_JFS2\_SIZEINLINELOG= 0  
 FS\_JFS2\_EAFORMAT= v1  
 FS\_JFS2\_QUOTA= no  
 FS\_JFS2\_DMAPI= no  
 FS\_JFS2\_VIX= no

This section contains information about the filesystems. When editing the image.data file, ignore this section. For our purposes, it’s just there to look pretty.

In a case where you need to edit the image.data file in an existing NIM mksysb resource, we first pull the ./image.data from the mksysb resource. Using our example from the bosinst.data :

# lsnim -l mksysb1  
mksysb1:  
Class = resources  
Type = mksysb  
Arch = power  
Rstate = ready for use  
Prev\_state = ready for use  
Location = /export/mksysb/mksysb1  
Version = 5  
Release = 3  
Mod = 0  
Oslevel\_r = 5300-05  
Alloc\_count = 0  
Server = master

The mksysb file is export/mksysb/mksysb1. We just need to run a restore command to extract it.

# cd /export/mksysb  
# restore -xqvf mksysb1 ./image.data  
New volume on mksysb1:  
Cluster size is 51200 bytes (100 blocks).  
The volume number is 1.  
The backup date is: Thu Aug 18 09:13:54 CDT 2006  
Files are backed up by name.  
The user is root.  
x 9430 ./image.data  
The total size is 9430 bytes.  
The number of restored files is 1.  
Now, we have /export/mksysb/image.data. We treat this as any other by moving it into our /export/nim/image\_data directory, edit it, and define it as a resource.

**Defining the image\_data resource**

**From command line :**

# nim -o define -t image\_data -a server=master -a location=/export/nim/image\_data/image.data -a comments=”Edited image.data file for mksysb1” image\_data\_mksysb1

**From SMIT :**

# smitty nim\_mkres  
-or-  
# smitty nim   
=> Perform NIM Administration Tasks => Manage Resources => Define a Resource

Next, we’ll select “image\_data” as the resource type.

\* Resource Name [image\_data\_mksysb1]  
\* Resource Type image\_data  
\* Server of Resource [master]  
\* Location of Resource [/export/nim/image\_data/image.data]  
 Comments [Edited image.data file for mksysb1]  
  
# lsnim -l image\_data\_mksysb1  
image\_data\_mksysb1:  
Class = resources  
Type = image\_data  
Comments = Edited image.data file for mksysb1  
Rstate = ready for use  
Prev\_state = unavailable for use  
Location = /export/nim/image\_data/image.data  
Alloc\_count = 0  
Server = master

You can choose to edit your file either before or after you define it as a resource. NIM does not care that you change the contents of the file, as long as the file is valid. Since image.data files change according to the system they are taken from, it is a good idea to remove any image.data files you are not actively using. You only want to use your edited image.data file in conjunction with the mksysb it was intended for.

**Breaking mirroring in an existing mksysb’s image.data file :**

Looking at our lv\_data stanza :

lv\_data:  
 VOLUME\_GROUP= rootvg  
 LV\_SOURCE\_DISK\_LIST= hdisk0 hdisk1  
 LV\_IDENTIFIER= 000048ed00004c000000011224adbe0a.1  
 LOGICAL\_VOLUME= hd5  
 VG\_STAT= active/complete  
 TYPE= boot  
 MAX\_LPS= 512  
 COPIES= 2  
 LPs= 1  
 STALE\_PPs= 0  
 INTER\_POLICY= minimum  
 INTRA\_POLICY= edge  
 MOUNT\_POINT=  
 MIRROR\_WRITE\_CONSISTENCY= on/ACTIVE  
 LV\_SEPARATE\_PV= yes  
 PERMISSION= read/write  
 LV\_STATE= closed/syncd  
 WRITE\_VERIFY= off  
 PP\_SIZE= 64  
 SCHED\_POLICY= parallel  
 PP= 2  
 BB\_POLICY= relocatable  
 RELOCATABLE= no  
 UPPER\_BOUND= 32  
 LABEL= primary\_bootlv  
 MAPFILE=  
 LV\_MIN\_LPS= 1  
 STRIPE\_WIDTH=  
 STRIPE\_SIZE=  
 SERIALIZE\_IO= no  
 FS\_TAG=  
 DEV\_SUBTYP=

Note the two entries in bold. This is the lv\_data stanza for our boot logical volume hd5. In order to break mirroring we need to change those two bold entries.

**COPIES= 2 to COPIES= 1**

**PP= 2 to PP= 1**

For every lv\_data stanza in this file you will switch the number of copies to 1. You’ll then change the “PP=##” in every stanza to half of the current value. If you don’t feel like doing the math, you can also just look under the “COPIES=#” and see the “Lps=##”. This is what the “PP=##” should be after cutting it in half.

ALL lv\_data stanzas need to be edited in this manner. Once this is done, saving the file will give you a non-mirrored rootvg after your NIM restore completes.