IBM

NIM Setup Guide

Technote (FAQ)

Question
This is a guide intended for those who are new to NIM and would like an easy to follow start to finish setup guide.

Answer

NIM SETUP / HOW TO / TROUBLESHOOTING

The Starter Guide

-
Storm M Harris

Document information

Software version:
5.3

Operating system(s):
AIX

Software edition:
Standard

Reference #:
T1010383

Modified date:
2009-06-30

NIM Setup and How-To Outline

Introduction

This guide is intended for those who are new to NIM, haven't worked with NIM in a while and need a refresher, or need some support with some common NIM tasks. This is also a guide intended for those who hate walking through setup guides. I've tried my best to balance out the *yawn* aspect of every other guide and make this something you'll actually recommend to others.

This also is in no way an official publication and is intended as a tool to help with those who are interested in having a starting point in learning and becoming proficient with NIM. As with my other guides I try as much as I can to not use a lot of "techtalk" and keep it as reader friendly as possible. As always, please feel free to make suggestions on how to improve this document, and if there are any procedures you would like to see added in the future, let me know and I'll do my best to do so. Finally, if I get anything wrong don't hesitate to call me out on it and I'll have it fixed ASAP. Though this is intended to be a great reference and training guide, I'm just trying to have some fun with teaching people how to setup NIM - so don't expect a boring read.....hopefully you'll be impressed with how easy NIM can actually be.

For a more advanced, "official," or higher level guide you can check out the recently released redbook :

NIM From A-Z in AIX 5L

Just go to http://www.redbooks.ibm.com/ and type in SG24-7296-00 in the search field on the upper right hand side of the page.

In ALL references from here on out through the rest of the guide, I am going to presume our "NIM master" machine is a dedicated "NIM master" and nothing else. I would not recommend having your #1 super-important business critical no I can't reboot this system for 93 more days system as your NIM master. If you are in a position where

you don't have the luxury of having a dedicated NIM master, follow your normal business practices in cases where I might say things such as "its ok if we corrupt this file, we can just reboot the system and fix it." You know your environment better than I do.

Finally, there always will be someone to tell you how you should appropriately type out "commands to be executed." Some people prefer that commands are put in single quotes, others prefer that they're in italics.....whenever there is an actual command to execute, it will look just like your command line interface on your AIX system.

For example if I want you to do a listing of files in a directory I would have :

```
# ls -al
```

I prefer to do it this way because this is exactly what you'll see when you type it in....(and ok, it's also easier to cut and paste the commands from my test system into this document once I've verified they work correctly). System SMIT screens and expected output will be displayed in the same manner.

What this guide does and does not cover :

Does :

- SMIT and Command line processes for most of the operations I cover
- Setup of the NIM master environment
- Setup of various resource types
- Setup and installation of NIM clients
- Backing up and installation of maintenance to NIM clients

Does not (and my reasons why) :

- Any websm interface procedures on how to use NIM. Websm is not used enough to spend any time going into separate sections on how to use it....and it hangs my test system when I bring it up so....no websm for you!

- Diskless and Dataless clients. These were pretty much phased out for all intents and purposes, then they *phoenix-like* rose from the ashes and have all sorts of new "mk*" commands associated with them. Even as such, since they tried to revive the use of them I still haven't had anyone ask about using them, so I've opted not to devote any time to them.

- Alternate_Disk features. I'd like to cover this here, but I've opted to add this to my next NIM guide which covers more intermediate level NIM tasks.

- NIM communication information (including nimsh and firewall issues). Again, I just don't have time to get into this with this guide and have it finished in a reasonable timeframe. It will be included in the next one.

- nimol (NIM over Linux). Since the introduction of nimol I think I've only had one or two calls on it. It's just not used and besides, once you're an expert in NIM, nimol shouldn't be too difficult to pick up.

- nimdef. The 'nimdef' command is an "easy way" to setup a NIM master and environment by using a template file. This goes against this whole guide as far as "learning" the setup process and about the NIM environment. I've also had no calls on it, so it falls under the "unused" category in my book, and therefore will get no recognition.

- Blades (JS20/21....etc). Again, the differences are there, but not enough to go into a whole section on it. Also, much like the others, it falls in the "not used enough" category.

- CSM. The "csm nim" stuff (much like pssp) is irrelevant to this guide. The SP/REGATTA team handles this piece of it so if you need or would like information on that, they can provide it to you.

Key words and definitions :

If you are unfamiliar with NIM I highly recommend reading through this section and also use it as a reference while reading through this guide. If you're figuring, "eh he'll explain all this later so I'll just skip this part," you're going to be flipping back to this part all mad later thinking, "ok fine so I should have read through this before....".

So this is sort of like a "reverse-glossary". I have it before any of the "How to" portions of the guide because it is important to know what it is we are talking about. I'll give the best and easiest to understand description of these terms so that you'll hopefully have a much easier time understanding any new concepts you're unfamiliar with. In all cases - actually using the files/keywords are handled in greater detail in their corresponding "How To..." sections.

Important Files and Directories:

- /etc/bootptab :
This file will exist on the NIM master. In a quiet NIM environment with no operations that require a client to boot, this file will be empty (except for the pre-existing commented section). This file gets updated automatically by the NIM master when a NIM operation is executed that requires the client machine to boot from a NIM SPOT. If this file contains incorrect information about either the master or the client, the boot operation will fail. While this file "can" be edited manually to fix a bootp issue - it should not be, as you are only applying a "band-aid" fix to an existing issue in your NIM environment....but, sometimes it's 5pm on a Friday and you're ready to go home, right ? (Also note related entry 'bootp')

- /etc/exports :
This is not a "NIM specific" file, it is a NIM critical file. Any sort of installation, boot, mksysb, savevg....etc operation requires the use of NFS. This file will be updated with which locations are NFS exported from the master to the client and the permissions associated with those exports. If these entries are incorrect or incomplete you will run into boot failures, permission problems, and other errors commonly associated with NFS. This is a text file and also "can" be edited manually to sometimes "band-aid" a problem, but should only be done so with care in knowing exactly what you're doing. The good thing is, if we mess up this file we can remove it and recycle NFS. The file can be recreated.

- /etc/hosts :
While not a "NIM specific" file, it is also a NIM critical file. This file is sort of like a phone book. It gives a relationship between a system's hostname and an ip address. Much like a telephone, if you dial the wrong number you get the wrong person. In NIM, if your ip address does not match up to the correct hostname, your install fails. This is a text file and can be edited manually. There should also only be 1 entry per ip/hostname. I personally prefer to make sure my NIM master has all entries in the /etc/hosts file and are of the following format :
<ipadress> <shortname> <longname>
If the client machine is up and running, it should also have a good entry in there for the NIM master as well.

- /etc/niminfo:
This file should always exist on the NIM master and sometimes will exist on a NIM client.
On the Master : This file is built when you first initialize the NIM environment. This is simply a text file so feel free to 'cat' or 'more' the file and look at the entries included in there. You do not want to manually edit this file if there is a mistake in the definition of the master. In this case you will want to redefine the master, or use the feature in NIM to change the master's attributes (hostname, gateway....etc).
On the Client : This file is "optional" depending on what sort of operations you are performing on the client. If the NIM client is up and running, and you intend to perform operations on the client (like take backups, or install maintenance) you will want to make sure this file exists. This file contains not only hostname information for the client, but tells the client who its master is.
This also should not be edited manually. If there is incorrect information in the file, it should be removed and recreated.

- /tftpboot :
This directory should always exist on the NIM master. The main purpose of this directory is to hold the boot images that are created by NIM when a boot or installation is initiated. This directory also holds informational files about the clients that are having a boot or installation operation performed. The file names that are generated in both cases are very descriptive file names For example :
The boot image created might be named : 53_spot.chrp.mp.ent.
The format of the file name is <spotname>.<system_architecture>.<processor>.<adapter_type>
The client info files are aptly named : <clientname>.info
The NIM master will create the <client_hostnamename> file and link it to the boot image. This boot image is what is sent over to the NIM client during a boot/installation operation.

Important NIM Commands :

nim :
This is the command line interface that performs all NIM operations. The 'nim' commands that do various operations are easy to figure out, but you typically don't see people use them that often because they can be really long commands. For most operations we will discuss in this guide I will try and give you the command line as well as 'smitty' ways to perform the function.

The easiest thing to do is understand the format of how to use the command and you're good to go. Take a look at it below :

nim { -o Operation} [-F] [-t Type] [-a Attribute=Value . . .] {ObjectName}

Easy enough right. There are about 25 or so different operations you can perform, and you'll only regularly use maybe 10 of them. I'll go over the more common in the chart below, the rest can be found in the manpage if you're interested. Again, in this guide, we will only be reviewing flags and operations that are the most commonly used. Once you master these you should have no problems understanding the rest.

Common NIM -o {operation} values

allocate	Allocates a specific resource to a client.
alt_disk_install	Sets up for an alternate disk installation.
bos_inst	Sets up or kicks off a BOS installation. (i.e. rte/mksysb, pull/push)
check	Allows you to check the various states of a NIM object.
cust	Allows you to perform software installation to an object.
deallocate	Deallocates resources from a NIM client.
define	Defines a new NIM object. (resource, client, network….etc)
fix_query	Allows you limited 'instfix' functionality against a NIM SPOT.
lppmgr	Allows you to run a cleanup against an lpp_source in NIM.
lslpp	Allows you to use the 'lslpp' command against the SPOT to determine filesets installed into that resource.
maint	Allows software update/remove/reject/commits to be performed.
remove	Removes a resource definition from the NIM environment
reset	Attempts to reset a NIM resource state to "ready for NIM operation".
unconfig	Will unconfigure NIM from the current master machine.

Continuing with the command you see the "-F" flag. This is commonly known as a "force." Typically you should only need to use the "-F" flag with the "reset" operation. I won't make a chart for the [-t Type] simply because, as you'll see later, they are pretty easy to figure out.
The [-a Attribute=….] and {Object Name} are also easily remembered once we get to the section on the nim command and see some examples.

lsnim :
This is going to be the command that provides you information about the various NIM objects in your environment. There is a huge combination of flags and attributes, and if I started listing them all out here with all possible permutations there would be this giant wall of text and your eyes would start bleeding from the stress - we don't want that.
With this command we'll simply be listing out the most useful commands and how to use them. They'll be covered throughout the entire guide so take note of them as you go along.

Important Keywords :

Allocate/Allocation :
This process is what allows your NIM client to access resources in NIM. The master uses NFS to perform the allocation process. Resources can be allocated to one or more NIM clients at the same time. You can see which resource types are allocated to clients by using the following command :

```
# lsnim -a spot
clientA:
   spot = 53H_spot
ClientB:
   spot = 52Q_spot
ClientC:
   spot = 53E_spot
```

So we can see, Client A has a SPOT called 53H_spot allocated to it. ClientB has a SPOT called 52Q_spot allocated to it, and ClientC has a SPOT, 53E_spot, allocated to it.

Base Operating System Installation :
Also commonly called (and referred to from here on out) as a **bos_inst** operation. This simply refers to the fact that you are initiating a boot and installation to a client machine. There are other installation types that do NOT require a boot. A **bos_inst** operation always means a boot of the client machine is involved.

Bootp :
This is the initial communication made between the NIM master and client during a boot or bosinst operation. In order for this to be successful several factors must be met :

1. - bootpd must be running on the NIM master
2. - the NIM client and master must have correct ip information about each other
3. - the /etc/bootptab must be populated correctly
4. - If the master and client systems are on separate networks, the router must be set to forward bootp packets.

There are other causes of failure, but checking/verifying those 4 will solve most bootp issues.

Tftp (Trivial File Transfer Protocol):
When the NIM client has been rebooted for a boot or bos_inst operation you don't have access to normal TCP communication. Once bootp connection has successfully been achieved, the NIM master uses tftp to transfer over the <clientname>.info file and the boot image to the client.

if1= : Or interface 1
This is known as the 'nim network'. Every machine, even the master, is placed on a defined NIM network. A machine who has multiple adapters defined to NIM will have "if2=" and "if3="….etc attributes. Not all adapters on a client need to be defined in NIM, only the ones that you wish to use with NIM. When your NIM master is generated, you will create a network name for the master and every client on the same subnet as the master. If you name this network "master_net" for example, then all clients on the same subnet as the master will have their "if1=" line set to "master_net". If you add additional clients that are on separate subnets, then you will need to create new network names. You can see the "if1=" information from an :

```
# lsnim -l master |more
-or-
# lsnim -l
```

You can get further information about the network name by running an 'lsnim -l <network_name>'.
Having incorrect networking information is probably the leading cause of NIM installation failures. This attribute and the information used when creating networks is extremely important to make sure you have correct.

Client (nim client) :
Any standalone machine or lpar in a NIM environment other than the NIM master. Clients use resources that reside on the NIM master to perform various software maintenance, backup, or other utility functions.
*Note that NIM resources do not always have to reside on the NIM master, but for our purposes they all will.

Groups (machine groups) :
In the spirit of convenience you can create a machine group which consists of a number of NIM clients. All NIM operations initiated from the master to that machine group subsequently are performed to all machines that are part of that group. For example, you can define a machine group and call it "Group1". Group1 has ClientA, ClientB, ClientC, and ClientD in it. You can initiate a bos_inst operation to each individual client, or if all clients are being installed with the same image, you can initiate the bosinst operation to Group1. All client systems will be installed at the same time. The downside to this however, is that you sacrifice performance for convenience. It is best if you decide to use machine groups to test out what sort of load your network and NIM master can hold before seeing diminishing returns.

Master (nim master) :
The one and only one machine in a NIM environment that has permission to run commands remotely on NIM clients. A client can only have one master, and a master can not be a client of any other master. The NIM master must also be at an equal or higher OS/TL/SP level than any client in the NIM environment. The NIM master also can not create SPOT resources at a higher level than it is currently installed at. Finally, the NIM master can not install any clients with an OS/TL/SP higher than his own. Long story short, for all intents and purposes, for any NIM operation, make sure you master is at an equal or higher level.
The NIM master also will hold all of our NIM resources. Due to this we'll want to make sure the NIM master has plenty of space available to it. Ideally, having a separate volume group (nimvg) is beneficial, so the rootvg does not get out of control in size.

Resource (nim resources) :
This can be a single file or up to a whole filesystem that is used to provide some sort of information to, or perform an operation on a NIM client. Resources are allocated to NIM clients using NFS and can be allocated to multiple clients at the same time. Various resource types will be explained below. I've decided to order them in a logical order of description rather than alphabetical order. It should make more sense to read through them in this manner.

Resource (nim resources) lpp_source:
When running an installation of a system outside of NIM, you use an installation CD. NIM uses resources. Two of the most important resources are made using the installation CD. First of all let's understand what exactly is on an installation CD that allows us to install a system. There are 4 parts :
- The filesets that get installed.
- The .toc file so the system knows what filesets are on the media.

- The boot images so the CD can boot the system initially
- A /usr filesystem to run the commands needed to install the system.

The lpp_source is created from an AIX installation CD and is responsible for holding :
- The filesets that get installed.
- The .toc file so NIM knows what is available in the lpp_source to be installed to the client.

In short, the lpp_source is simply a depot. It's just a directory that holds all of the filesets and the .toc file.

Resource (nim resources) SPOT:
The SPOT resource (stands for Shared Product Object Tree in case you were wondering) is responsible for the following :
- Creating a boot image to send to the client machine over the network.
- Running the commands needed to install the NIM client.

Essentially the SPOT is a /usr filesystem just like the one on your NIM master. You can think of it as having multiple "mini-systems" on your NIM master, because each SPOT is its own /usr filesystem. You can upgrade it, add fixes to it, use it to boot a client system....etc. Just like your NIM master's /usr filesystem, going in there manually and messing around with files can easily corrupt it. The good thing about a SPOT however, is that it is easily rebuilt.
You can also create a SPOT from a NIM mksysb resource. This SPOT however is not as versatile as one created from an lpp_source and can not be upgraded with any fixes and can only be used with the mksysb resource it was created from.

Resource (nim resources) mksysb:
This is simply a mksysb image of a machine. The mksysb image can be of the NIM master, a NIM client, or a machine outside of the NIM environment. This resource can be defined in one of two ways.
- From an existing mksysb taken to file that resides on the NIM master.
- Creating a new mksysb image of a currently existing NIM client.

At this time there is no supported way to use a mksysb tape or mksysb on CD/DVD, as an input device to define a mksysb resource in NIM.

Resource (nim resources) bosinst_data:
When booting from installation media to install or upgrade a system you boot to what are known as the "BOS Menus" or Base Operating System Installation Menus. Here you select your console, what language to use, what disks to install to.....and many other options. In NIM we can create a "bosinst_data" resource that will answer these questions for us. By doing this we can perform a "non-prompted" installation. So if you have a NIM client in another building, down the road, or half way across the country, you can create this type of NIM resource which will provide the answers to those questions, so once you kick off the install from the NIM master no further interaction is required. The system should (ideally) install and reboot itself afterward.

A mksysb (as discussed above) has a "built in" bosinst.data file. If the option in that file
(PROMPT =) is set to yes, this file really does nothing as the choices you make in the BOS menus will override the options in the file. However, if the mksysb was created to have that option set to no, then we can create a new bosinst_data resource which will trump the one that is part of the mksysb.

Resource (nim resources) image_data:
Outside of NIM this file is responsible for knowing how your rootvg is built. It contains information like the partition size of rootvg, the disks belonging to rootvg, all of the filesystems (and their sizes) that belong to rootvg, whether the rootvg is mirrored, and other information. As with the bosinst_data file, a mksysb also has one of these "built in". If this built in file needs to be altered in any way, we can accomplish this by creating and allocating an image_data resource.

Initial NIM Master Configuration:

We're going to bypass any further discussion other than at the introduction of this guide concerning the who/what/why/when/where of choosing a suitable NIM master. That's totally up to you and I've already mentioned that it ideally shouldn't be a system you're not able to play with and reboot at will. The only thing we need is space. Make sure you have space directly related to the size of the NIM environment you want to have.

Some choose to create a different volume group (i.e. nimvg) and house everything NIM related in that volume group. This is usually a good idea if you're looking at a very large environment as you typically want to keep your rootvg as small as possible. If you do not have that luxury - no problem, feel free to house your NIM environment in rootvg.

There are a 3 options you have for the initial setup which I will go through in this section.
1. Totally manual setup for the whole environment. (My personal recommendation because you end up learning more).
2. Setup using EZ-NIM
3. Setup using a basic startup method.

For future reference, all further mentions using the word "media" will refer to Base AIX Installation CDs, unless otherwise specified.

For simplicity, all references for any device (cdrom, ethernet, tape...etc) will always be cd0, ent0, rmt0...unless otherwise noted. You may, depending on your environment, need to use other devices...substitute as needed.

Full Manual Setup of NIM

As I mentioned earlier, going through this at least once is definitely recommended. Manually creating the master and defining resources will get you comfortable with the environment and most importantly, you learn much more as you go this way.

Installing the required filesets

You will need Volume 1 of your Base AIX Installation media CD. If you have a directory where you've used the bffcreate utility to copy down the contents of the media to disk, that is fine as well. What we're looking for basically is the base level versions of the NIM filesets.
There are 3 filesets we will need to deliver the NIM software to our future NIM master.

1. bos.sysmgt.nim.master
2. bos.sysmgt.nim.client
3. bos.sysmgt.nim.spot

Put Volume 1 of your media in the drive and from and command line you can run the following command to automatically grab all 3 required filesets in the nim package :

```
# installp -acgXd /dev/cd0 bos.sysmgt.nim
```

Using SMIT:

```
# smitty install_all
* INPUT device / directory for software              /dev/cd0
* SOFTWARE to install                               [bos.sysmgt.nim]
  PREVIEW only? (install operation will NOT occur)    no
  COMMIT software updates?                            yes
  SAVE replaced files?                                no
  AUTOMATICALLY install requisite software?           yes
  EXTEND file systems if space needed?                yes
  OVERWRITE same or newer versions?                   no
  VERIFY install and check file sizes?                no
  DETAILED output?                                    no
  Process multiple volumes?                           yes
  ACCEPT new license agreements?                      no
  Preview new LICENSE agreements?                     no
```

Initializing the nim master :

This is basically going to tell the system, "Hey...you're a NIM master". Easy right.

Command line :

```
# nimconfig -a pif_name=en0 -a master_port=1058 -a netname=master_net -a cable_type=bnc
```

pif_name = This is your primary interface for your NIM master
netname = Name your master's network. With NIM you want to give objects names that are easy and descriptive. If I see "master_net" I know for that is my NIM master's network. Using a name like "NetworkA" doesn't really tell you anything just by the name itself.
The rest of the options are default options.

SMIT :

```
# smitty nimconfig
-or-
# smitty nim
=> Configure the NIM environment => Advanced configuration => Initialize the NIM master only

* Network Name                                  [master_net]
* Primary Network Install Interface             [en0]

  Allow Machines to Register Themselves as Clients?  [yes]
  Alternate Port Numbers for Network Communications
       (reserved values will be used if left blank)
    Client Registration                         []
    Client Communications                       []
```

Once this is complete you have a functioning NIM master.
Take a look at the following output and you'll see information about your master :

```
# lsnim -l master
```

Look at this next output and you'll see that by defining the NIM master you have some resources that have been pre-generated for you.

```
# lsnim -l |more
```

The "boot" resource created a /tftpboot directory to hold all of your boot images. There's also a nim_scripts" resource. That belongs to the master. Do not go into the /export/nim/scripts and mess with any files that get generated during an install. Finally, there's a "master_net" which represents the NIM network we created earlier. All NIM clients that are on the same subnet as this master will be assigned to the "master_net" network. If you add any NIM clients that are on a different network, then you will need to generate a new network name for that network. More on that a little later. Now we'll go into defining your lpp_source and SPOT resources.

Setting up your first lpp_source resource :

Before we get your lpp_source and SPOT defined we'll need to decide on a place to put them. One of the best things you can do in NIM is be neat. An organized NIM environment is a happy NIM environment. I recommend having separate filesystems for separate resource types. In other words I'll have a filesystem to hold my lpp_sources, one to hold my SPOT resources, one for my mksysb images.....etc. The "norm" is to use "/export/nim" filesystems.
For my lpp_sources I'll create a filesystem called /export/nim/lpp_source. A good rule on space is a little more than a ½ gig per volume you want to copy down to your lpp_source. I will be using all 8 volumes of my 5300-05 base AIX media. If you are using base AIX DVDs volume 1 is sufficient to perform minimal installs however if you have the space it would be best to add all volumes.

```
# crfs -v jfs2 -g nimvg -m /export/nim/lpp_source -a size=5G
```

This will create a jfs2 filesystem in nimvg with a size of 5gig and have a mountpoint of /export/nim/lpp_source. Again, this is just an example. Feel free to use rootvg or another volume group. If this command does not fit your environment you can go into :

```
# smitty crfs
```

...and create your own filesystem using whatever parameters you need.

We then mount up the filesystem :

```
# mount /export/nim/lpp_source
```

The lpp_source is now ready to be created. We'll need Volume 1 of your base media in the drive. The minimum you'll use is V1 of the media. You can put 1, 2, 3, or all volumes in the lpp_source. You're looking at a trade off of space and convenience. I recommend at least having volumes 1-3 if you're concerned about space. Ideally, and in this example environment, you want to create the lpp_source using all volumes of media.
If you are using base AIX DVDs volume 1 is sufficient to perform minimal installs however if you have the space it would be best to add all volumes.

From command line :

```
# nim -o define -t lpp_source -a location=/export/nim/lpp_source/53_05 -a server=master -a comments='5300-05 lpp_source' -a multi_volume=yes -a source=/dev/cd0 -a packages=all 5305_lpp
```

Yes, that would be 1 command. That is one of the reasons many NIM operations are done from SMIT. It's really easy to mistype something, especially if communicating over the phone with someone in a noisy server room. Now, to break down the command :
The only 2 required fields are the "location" (where we want it to be created) and "server" (which machine will hold this resource). You can hold resources on other NIM clients but for our purposes we will always hold resources on the NIM master. The rest of the "-a" flags are optional. You may think - 'wait a minute....the source has to be required, otherwise, where do you get the filesets from ?' You can "pre-generate" the lpp_source. If you've already copied the filesets down into a directory and want to use that as your lpp_source, then you have no "source", you just have a "location". At the end of the command, I named the resource "5305_lpp". This is what NIM uses to reference this resource. Next we'll use smit to do the same thing.

From SMIT :

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

Next you select "lpp_source" as the resource type.

* Resource Name                [5305_lpp]
* Resource Type                lpp_source
* Server of Resource                   [master]
* Location of Resource         [/export/nim/lpp_source/5305]
 Architecture of Resource             []
 Source of Install Images             [/dev/cd0]
 Names of Option Packages             [all]
 Show Progress                [yes]
 Comments                             [5300-05 lpp_source]
```

Notice there isn't an option for multiple volumes. For the most part smit and command line are the same, but occasionally there are differences. Doing it this way will only create the lpp_source from V1 of the media. If you wish to add other volumes you can do one of the following :
A) bffcreate the volumes into the lpp_source
B) use NIM to add the volumes

```
# smitty nim_res_op
-or-
# smitty nim
=> Perform NIM Administration Tasks => Manage Resources => Perform Operations on Resources

Select your lpp_source
Select "update"

TARGET lpp_source                                       5305_lpp
SOURCE of Software to Add                               /dev/cd0
SOFTWARE Packages to Add                        [all]
  -OR-
INSTALLP BUNDLE containing packages to add      []
gencopy Flags
DIRECTORY for temporary storage during copying  [/tmp]
EXTEND filesystems if space needed?             Yes
Process multiple volumes?                               Yes
```

Either way, you will get the same result.
How to we take a look at the lpp_source ? We use the 'lsnim' command.

```
# lsnim -l 5305_lpp
5305_lpp:
class           = resources
type            = lpp_source
comments        = 5300-05 lpp_source
arch            = power
Rstate          = ready for use
prev_state      = unavailable for use
location        = /export/nim/lpp_source/53_05
simages = yes
alloc_count     = 0
server          = master
```

Important lines :
Rstate = if this is not set to "ready for use" then you can not use this resource. Sometimes running a check on the lpp_source will allow you to clear this up.

```
# nim -o check
```

simages = This means that this lpp_source has the proper system images in order to properly build a SPOT resource. If any required system image filesets are missing from the lpp_source they will typically be listed at the bottom of the output.

Why wouldn't you want all lppsources have the "simages" attribute to be "yes" ?
A lpp_source can have pretty much anything you want in it. It doesn't have to be built from base installation media, nor does it have to be used to only build a SPOT. Let's say you have your 5300-05 lpp_source, build a SPOT from it, and build some systems.
You then order some service pack updates (5300-05-01) for example. You can update your current lpp_source, but if you do that all of your future installs using this will be at 5300-05-01. If you do not want this to be the case, you can create another lpp_source that only holds these updates. It is more of a preference issue than anything else.

Next, we move on to creating a SPOT resource.

Setting up your first SPOT resource :

Now we will create a filesystem for our SPOT. This does not have nearly the space requirement that an lpp_source does. 500meg should be plenty of space for your initial SPOT. If you recall, the SPOT is just like a /usr filesystem. When you install your system from CD-ROM not every single fileset gets installed to the system right - only what is necessary to run the system. The same applies for the SPOT.

```
# crfs -v jfs2 -g nimvg -m /export/nim/spot -a size=1G
```

We then mount up the filesystem :

```
# mount /export/nim/spot
```

From command line :

```
# nim -o define -t spot -a server=master -a source=5305_lpp -a location=/export/nim/spot -a auto_expand=yes -a comments='5300-05 spot' 5305_spot
```

Here you minimally have 3 required fields. You need to let the NIM master know who will be holding the resource (again, you can use a NIM client as a resource server, but that is rare, and for our purposes, it will always be the NIM master), you need to give it an lpp_source that contains the "simages=yes" attribute, and you need to give it a location to build the resource.

The "auto_expand=yes" is recommended because this allows the system to automatically expand the size of the filesystem if necessary (instead of failing the operation).

From SMIT:

```
# smitty nim_mkres
-or-
# smitty nim
=> Perform NIM Administration Tasks => Manage Resources => Define a Resource
Next you select "SPOT" as the resource type.

* Resource Name                         [5305_spot]
* Resource Type                         spot
* Server of Resource                            [master]
* Source of Install Images                      [5305_lpp]
* Location of Resource                  [/export/nim/spot]
Expand file systems if space needed?            Yes
Comments                                        [5300-05 spot]
```

This will take a while to create as it is typically installs 300+ filesets into the SPOT resource. Once this completes you can check the output of the lsnim command to see information about the SPOT.

```
# lsnim -l 5305_spot
5305_spot:
class           = resources
type            = spot
comments        = 5300-05 spot
plat_defined    = chrp
arch            = power
bos_license     = yes
Rstate          = ready for use
prev_state      = verification is being performed
location        = /export/nim/spot/5305_spot/usr
version         = 5
release         = 3
```

```
mod             = 0
oslevel_r       = 5300-05
alloc_count     = 0
server          = master
Rstate_result   = success
mk_netboot      = yes
```

Important lines :
Rstate = if this is not set to “ready for use” then you can not use this resource. The first thing you’ll want to do is run a force check against the SPOT. This forces the rebuild of the boot images and should return the “unavailable for use” back to “ready for use”.

```
# nim -Fo check
```

oslevel_r = this works just like the ‘oslevel -r’ if you ran that from an AIX command line. Knowing the level of the SPOT resource is extremely important in NIM operations we will go into later.

Your NIM master has officially been initialized and setup. The next 2 sections go into alternate (theoretically “easier”, but we’ll call it “less interactive”) ways of setting up the NIM master.

Feel free to review those and/or move on to defining NIM clients.

Setup of NIM using EZ-NIM

EZ-NIM is feature that allows as much of a “hands free” setup of NIM as possible. EZ-NIM also has a list of common operations that it can perform for you. Some people find the lack of interaction a blessing, however if there is an operation you need to perform that isn’t in the list provided for you, you might be lost as how to exactly get what you need done. It is best to determine what sort of environment you need before determining whether or not EZ-NIM is for you.

Installing the required filesets

You will need Volume 1 of your Base AIX Installation media CD. If you have a directory where you’ve used the bffcreate utility to copy down the contents of the media to disk, that is fine as well. What we’re looking for basically is the base level versions of the NIM filesets.
There are 3 filesets we will need to deliver the NIM software to our future NIM master.

1. bos.sysmgt.nim.master
2. bos.sysmgt.nim.client
3. bos.sysmgt.nim.spot

Put Volume 1 of your media in the drive and from and command line you can run the following command :

```
# installp -acgXd /dev/cd0 bos.sysmgt.nim.master bos.sysmgt.nim.client bos.sysmgt.nim.spot
```

Using SMIT :

```
# smitty install_all
* INPUT device / directory for software                 /dev/cd0
* SOFTWARE to install
PREVIEW only? (install operation will NOT occur)               no
COMMIT software updates?                                       yes
SAVE replaced files?                                           no
AUTOMATICALLY install requisite software?              yes
EXTEND file systems if space needed?                   yes
OVERWRITE same or newer versions?                      no
VERIFY install and check file sizes?                           no
DETAILED output?                                               no
Process multiple volumes?                                      yes
ACCEPT new license agreements?                         no
Preview new LICENSE agreements?                        no
```

EZ-NIM Setup of the NIM Master

```
# smitty eznim
Configure as a NIM Master
Setup the NIM Master environment

Select or specify software source to initialize environment            [cd0]
Select Volume Group for resources                                      [rootvg]
Select Filesytem for resources                                 [/export/eznim]

Options
  CREATE system backup image?                                          [yes]
  CREATE new Filesystem?                                               [yes]
  DISPLAY verbose output?                                              [no]
```

During script execution, lpp_source and SPOT resource creation times may vary. To view the install log at any time during nim_master_setup, run the command: tail -f /var/adm/ras/nim.setup in a separate screen.

What you’ll get after the process completes :

1. resources will be defined in rootvg.
2. mksysb resource 5300-05master_sysb
3. /tftpboot filesystem
4. resolv_conf resource master_net_conf
5. bosinst_data resource 5300-05bid_ow
6. lpp_source resource 530lpp_res
7. SPOT resource 530spot_res
8. resource group basic_res_grp

Here is a list of the predefined operations you can perform from the EZNIM utility :

Enable Cryptographic Authentication
Add fixes to the NIM Master environment
Add client to the NIM environment
Update clients
Backup a client
Reinstall clients
Reset clients
Show the NIM environment
Verify the NIM environment
Remove NIM environment

This concludes your EZ-NIM master setup. Further NIM operations will be discussed later in this guide, and will be done using command line and SMIT. No further EZ-NIM references will be made in regards to performing operations within the utility.

Setup of NIM using the "basic startup" method

This method is somewhere between a manual setup and EZ-NIM. You have more control over the master's setup options than you do in EZ-NIM, but the system still does some of the grunt work (like setting up filesystems) for you.

Installing the required filesets

You will need Volume 1 of your Base AIX Installation media cd. If you have a directory where you've used the bffcreate utility to copy down the contents of the media to disk, that is fine as well. What we're looking for basically is the base level versions of the NIM filesets.
There are 3 filesets we will need to deliver the NIM software to our future NIM master.

1. bos.sysmgt.nim.master
2. bos.sysmgt.nim.client
3. bos.sysmgt.nim.spot

Put Volume 1 of your media in the drive and from and command line you can run the following command :

```
# installp -acgXd /dev/cd0 bos.sysmgt.nim.master bos.sysmgt.nim.client bos.sysmgt.nim.spot
```

Using SMIT :

```
# smitty install_all
* INPUT device / directory for software                /dev/cd0
* SOFTWARE to install
PREVIEW only? (install operation will NOT occur)              no
COMMIT software updates?                                      yes
SAVE replaced files?                                          no
AUTOMATICALLY install requisite software?             yes
EXTEND file systems if space needed?                  yes
OVERWRITE same or newer versions?                     no
VERIFY install and check file sizes?                          no
DETAILED output?                                              no
Process multiple volumes?                                     yes
ACCEPT new license agreements?                        no
Preview new LICENSE agreements?                       no
```

Initializing the Master and Initial Lppsource / Spot Resources

From command line :

The command is forever long and actually could be longer. I'm only including it in here for those who might want to get crazy and script a NIM setup.

```
# /usr/lpp/bos.sysmgt/nim/methods/m_setup -w -x -y -S master -f 'en0' -i 'cd0' -l '5305_lpp' -D '/export/lpp_source' '-A' -J '1024' -G 'rootvg' -s '5305_spot' -E '/export/spot' '-B' -K '512' -H 'rootvg' '-v' '-o'
```

This initializes my NIM master with a gig for the lpp_source, and 512m for the SPOT. The lpp_source name is "5305_lpp" and the SPOT name is "5305_spot", both have had their own filesystems created for them.
The master's network is given the name "network1" by default.

From SMIT:

```
# smitty nim
Configure the NIM Environment
Configure a Basic NIM Environment (Easy Startup)

Initialize the NIM Master:
* Primary Network Interface for the NIM Master          [en0]

Basic Installation Resources:
* Input device for installation images                         [cd0]
* LPP_SOURCE Name                                      [5305_lpp]
* LPP_SOURCE Directory                                 [/export/lpp_source]
Create new filesystem for LPP_SOURCE?                 [yes]
Filesystem SIZE (MB)                                  [1024]
VOLUME GROUP for new filesystem                [rootvg]
* SPOT Name                                                   [5305_spot]
* SPOT Directory                                              [/export/spot]
Create new filesystem for SPOT?                       [yes]
Filesystem SIZE (MB)                                  [512]
VOLUME GROUP for new filesystem                [rootvg]
```

There are some other options for diskless / dataless clients and the resources associated with those, but since we're not discussing diskless/dataless we'll skip those options.

Once this process completes your master is setup and ready to start defining NIM clients and running installs. Again, feel free to take a look at your master and the resources with the following commands :

```
# lsnim -l master
# lsnim -l 5305_lpp
# lsnim -l 5305_spot
```

Nim Resources: What Are They / What Do They Do ? How Can I Make Them ?

....and other questions.
In this section we'll be discussing all sorts of different types of nim resources. We'll be revisiting the lpp_source and SPOT in greater detail, showing you how to create more resources than you've ever wanted to know about, and introducing you to some of the lesser used resource functions in NIM.
I've broken this section down into 2 areas with their subcategories :

Commonly Used

- lppsource
- SPOT
- mksysb
- bosinst_data
- image_data

Not-So-Commonly-Used

- machine groups
- resolv_conf
- script
- fb_script
- exclude_files
- installp_bundle

There are other resources that are available for use in NIM but most have to do with diskless and/or dataless clients.

Commonly used resources

Revisiting the lpp_source resource

By following through this guide you've built an lpp_source by now but lets take a closer look at exactly what it is, and how to treat your lpp_source in the future. For these examples we'll be using the lpp_source we've been using before : 5305_lpp

If you've forgotten the name of your lpp_source there's a handy command that will list out all of the lpp_source resources on your system.

```
# lsnim -t lpp_source
5305_lpp      resources       lpp_source
```

Once you have the name run an informational listing against it :

```
# lsnim -l 5305_lpp
5305_lpp:
Class                   = resources
Type                    = lpp_source
Comments                = 5300-05 lpp_source
Arch                    = power
Rstate                  = ready for use
Prev_state              = ready for use
Location                = /export/nim/lpp_source/53_05
Simages         = yes
Alloc_count             = 0
Server                  = master
```

Important information :
Rstate = This is the NIM state of the resource. If this is not set to "ready for use" then this lpp_source can not be used for any NIM operations. Typically running a check operation on the lpp_source name will reset this to a good state from a state such as "unavailable for use".

```
# nim -o check 5305_lpp
```

What this does is rebuild the .toc file and verify it has the "simages" attribute. If it does not have the "simages = yes" attribute that simply means that you can not use this lpp_source to build a SPOT or use it for a bosinst operation.

Alloc_count = This will give a numerical value as to how many NIM clients this specific lpp_source has been allocated to. You can find out the specific NIM client names by running the following command :

```
# lsnim -a lpp_source
lucidbso:
   lpp_source = 5305_lpp

-or-

# lsnim -a lpp_source -Z
#name:lpp_source:
lucidbso:5305_lpp:
```

Adding the -Z simply changes the format of the output - some find it easier to read. Personal preference really.

location = This is the top level directory where your filesets and RPMs are stored. In our case if we look at the contents of our lpp_source location : /export/nim/lpp_source/53_05

```
# cd /export/nim/lpp_source/53_05
# ls -al
total 0
drwxr-xr-x   5 root    system          256 Apr 11 13:18 .
drwxr-xr-x   4 root     system          256 Apr 15 13:02 ..
drwxr-xr-x   3 root     system          256 Apr 11 13:18 RPMS
drwxr-xr-x   3 root     system          256 Apr 11 13:18 installp
drwxr-xr-x   3 root     system          256 Apr 11 13:18 usr
```

Following the RPMS/ppc directory you'll see the system default provided RPMS.
Following the installp/ppc directory you'll see all of the filesets available in your lpp_source.
The /usr path follows down to the license agreement.

Next we'll look at all of the different operations you can perform on your lpp_source.
You can perform any of these operations through smit by running the following :

```
# smitty nim_res_op
-or-
#smitty nim
=> Perform NIM Administration Tasks => Manage Resources => Perform Operations on Resources
```

We'll go over the options individually :
showres = show contents of a resource

```
# nim -o showres 5305_lpp
```

This will show the contents of this resource. Unlike the 'lslpp' operation, you can use this operation to compare it to the master, another NIM client, or a SPOT resource. The output will be formatted with the filesets listed having a "+" or "@".
"+" = This fileset is not installed on the target
"@" = This fileset is installed on the target
*The option to compare filesets to another object is available in the SMIT menu only.

lslpp = list LPP information about an object

```
# nim -o lslpp 5305_spot
```

This is a simple listing of the lslpp output from your SPOT.
check = check the status of a NIM object

```
# nim -o check 5305_lpp
```

This command rebuilds the .toc file of the lpp_source and checks for the "simages" attribute. No output is displayed unless an error occurs.

lppmgr = eliminate unnecessary software images in an lpp_source

```
# nim -o lppmgr -a lppmgr_flags="rubxV" 5305_lpp
```

You should use this operation when you've added some maintenance/fixes to your lpp_source. Since base level filesets and fix level filesets can have the same names, the .toc file can get thrown off with what is really in the lpp_source. The lppmgr command allows you to remove any unnecessary duplicate filesets from your lpp_source.

update = add or remove software to or from an lpp_source

```
# nim -o update -a source=/dev/cd0 -a packages=all 5305_lpp
```

Use this option to add or remove software from the lpp_source (easy right). Typically used when copying volumes of CDs down to the lpp_source. In the example above you might use that if you received the latest service pack in and would like to add this to your lpp_source.
You can also just "cp" or "mv" filesets into your /<lpp_source>/installp/ppc directory manually. Remember that you do need to run an lppmgr command to update the .toc file. If you have any further RPMS that you would like to add to your lpp_source, make sure to put them in the <lpp_source>/RPMS/ppc location.

How to handle this lpp_source concerning upgrades and updates :
Given an example lpp_source : 5300-04_lppsource. Now you get a set of CDs in that contain 5300-05. Do you add this to your 5300-04_lppsource or do you create a new one ?
That all depends on your environment. It is a good idea to keep lppsources available to you as long as you have the need for clients to stay at that level. If in this example you are planning on upgrading all of your clients to 5300-05, then it might be best to update the existing lpp_source. If you have a number of clients for whatever reason that can not be upgraded, and you have the available disk space, you might be better off creating a separate lpp_source.
What it boils down to is environment management. How do you need to run your environment based on your needs and available space.....

In cases where you are dealing with different Version or Release levels you must create a different lpp_source. There is no upgrade path for lppsources.

Example : 5200-08_lppsource - you get in some 5300-05 base CDs. You will want to create a new lpp_source for that level.

Revisiting the SPOT resource

By following through this guide you've built a SPOT by now but lets take a closer look at exactly what it is, and how to treat your SPOT in the future. For all examples we will be using the SPOT we created earlier : 5305_spot

If you've forgotten the name of your SPOT there's a handy command that will list out all of the SPOT resources on your system.

```
# lsnim -t spot
5305_spot     resources       spot
```

Once you have the name run an informational listing against it :

```
# lsnim -l 5305_spot
5305_spot:
Class                   = resources
Type                    = spot
Comments                = 5300-05 spot
Plat_defined            = chrp
Arch                    = power
Bos_license             = yes
Rstate                  = ready for use
Prev_state              = ready for use
Location                = /export/nim/spot/5305_spot/usr
Version         = 5
Release         = 3
Mod                     = 0
Oslevel_r               = 5300-05
Alloc_count             = 0
Server                  = master
If_supported            = chrp.mp ent
Rstate_result           = success
```

Important information :

Rstate = This is the NIM state of the resource. If this is not set to "ready for use" then this SPOT can not be used for any NIM operations. Typically running a force check operation on the SPOT name will reset this to a good state from a state such as "unavailable for use".

```
# nim -Fo check 5305_spot
```

What this will do is have the SPOT check itself to make sure it has the proper support for, and actually recreate the boot images in the /tftpboot directory.

Alloc_count = This will give a numerical value as to how many NIM clients this specific SPOT has been allocated to. You can find out the specific NIM client names by running the following command :

```
# lsnim -a SPOT
lucidbso:
   spot = 5305_spot

-or-

# lsnim -a spot -Z
#name:spot:
lucidbso:5305_spot:
```

Adding the -Z simply changes the format of the output - some find it easier to read. Personal preference really.

location = This is the location the SPOT has been installed into. If you remember from earlier we discussed how a SPOT is essentially a /usr filesystem. If you cd down to that location and run a listing you'll see pretty much exactly what you'll see if you did a listing from your NIM master's /usr directory.

oslevel_r = This tells you the oslevel and latest TL that is completely installed. This is very important information that affects operations like maintenance boots of a NIM client, mksysb restore/clones, alt_mksysb operations, and others. We'll go into the specifics when we get into running installations later on.

Next we'll look at all of the different operations you can perform on your SPOT.
You can perform any of these operations through smit by running the following :

```
# smitty nim_res_op
-or-
# smitty nim
=> Perform NIM Administration Tasks => Manage Resources => Perform Operations on Resources
```

We'll go over the options individually :
reset = reset an object's NIM state / check = check the status of a NIM object

```
# nim -Fo reset 5305_spot
-or-
# nim -Fo check 5305_spot
```

The reset should only be ran if the current allocation count for that resource is equal to 0. It is preferable however to run a force check on the SPOT instead. The check operation will also rebuild the boot images it serves out to clients at boot time.
cust = perform software customization

```
# nim -o cust -a filesets=bos.atm -a lpp_source=5305_lpp 5305_spot
```

The customize operation typically is used to install additional support into the SPOT. If for example your NIM SPOT is missing the device driver for your target NIM client's ethernet card you will have problems with any operation where the client needs to be booted.

showres = show contents of a resource / lslpp = list LPP information about an object

```
# nim -o showres 5305_spot
-or-
# nim -o lslpp  5305_spot
```

This does exactly what it says. It shows the filesets that were installed to create the SPOT resource.
The showres is essentially the exact same as the lslpp operation. Used when you want to check the SPOT to see if a specific fileset or fileset level has been installed into it. It is helpful to use the "...| grep <filesetname>" on the end of the showres or lslpp commands to search for a specific fileset.

update_all = update all currently installed filesets

```
# nim -o cust -a fixes=update_all -a lpp_source=5305_lpp 5305_spot
```

In cases where you wish up update your SPOT with an individual APAR, a service pack, or an entire TL, you will use the customize operation and update_all. First of all you will add the desired filesets into the lpp_source, and as you can see above example command, use that as the "source of installation images".
There is a second way through smitty to perform the same operation :

```
# smitty nim_update_all
```

fix_query = perform queries on installed fixes
There are a few examples of using this feature that we'll look at from command line. I've only found using the SMIT option beneficial if I know the exact APAR number I am looking for. From command line we can run easier searches using 'grep', and get much more useful information.

Example 1 : When looking to see what the latest TL is that is installed to the SPOT

```
# lsnim -l 5305_spot
```

This will show an "oslevel_r" line that will give you the highest TL completely installed into the SPOT. If however you expect to see 5300-05 but only show 5300-04, we can use the fix_query operation to find out what is missing.

```
# nim -o fix_query -a fix_query_flags="cq" -a fixes=5300-05_AIX_ML 5305_spot |grep ":-:"
```

What this command is doing is listing out all filesets that are part of the 5300-05 TL and grep'ing for any ones that are installed at a lower level (:-: from the grep above) than required. The downlevel filesets will be listed out in the following format :

```
5300-05_AIX_ML::::-:AIX 5300-05 Update
```

Example 2 : When looking to see what the latest SP is that is installed to the SPOT

```
# lsnim -l 5305_spot
```

Though this shows the "oslevel_r" information, as of right now, it does not show service pack information. That may change in the future, but as of now we need to use the fix_query operation to check the Service Pack level.

```
# nim -o fix_query 5305_spot |grep SP
All filesets for 5300-03-CSP_SP were found.
All filesets for 5300-04-01_SP were found.
All filesets for 5300-04-02_SP were found.
All filesets for 5300-04-03_SP were found.
```

This will show all of the SP listings. You can also grep for a specific SP or range of Service Packs with the grep command. From the above listing you can see that there has been no service pack beyond 5300-05 added to the SPOT.

Example 3 : When looking to see if there is a specific APAR installed to the SPOT

```
# nim -o fix_query -a fixes=IY78180 5305_spot
All filesets for IY78180 were found.
```

If you get a return of "not all filesets found" for your APAR, refer back to Example 1 on how to determine which filesets are missing.

showlog = display a log in the NIM environment

```
# nim -o showlog 5305_spot
```

This is sort of like a special smit log just for NIM SPOTs. It logs activity or queries against the NIM SPOT. If you're doing most of your work through smit, then this will all be logged in your smit.log so you generally wouldn't need to bother with the showlog operation, however if running from command line, it can help determine causes for any failures.

lppchk = verify installed filesets

```
# nim -o lppchk -a lppchk_flags="v" 5305_spot
# nim -o lppchk -a lppchk_flags="l" 5305_spot
# nim -o lppchk -a lppchk_flags="c" 5305_spot
```

Since the SPOT is an installed entity, just like your NIM master or any other AIX system, it can run into cases where it has broken filesets, broken links, or missing/corrupt files. They are also fixed in the same manner as you would on any other system using the "Force Overwrite" or "Force Reinstall" options for -v errors, using the "-ul" flags for missing links from "-l" errors, and replacing bad files for any "-c" output.

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.

The mksysb resource

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
```

The bosinst.data resource

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.

The image_data resource

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
        PPs= 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.

Not-So-Commonly-Used Resources

The next series of resources I'll only briefly describe. Since this is more of a "getting started" guide you won't need to use any of these. By the time you're comfortable with NIM (hopefully after going through this guide), you should be able to come back and create, define, and work with these resources without too much trouble.

Machine Group

A machine group is exactly what it sounds like. Several machines can be defined into a "machine group" resource for convenience in installation. What you get in convenience you can pay for in performance however. Putting 20 machines in a machine group and running an installation to that machine group resource will obviously not be as fast as installing to 5 machines at once. This not only depends on your master's performance capabilities, but also your network's performance capabilities.

You may run into a case where you need 5 machines installed with a "gold image" mksysb. In a case like this, defining a machine group resource could save you some time.

Resolv_conf

This resource will look exactly like your /etc/resolv.conf file. It is simply a file containing nameserver and domain name information. Using this resource will, upon a successful bosinstall operation and reboot, configure that machine to use those services defined in the file.

Script

This is a user written shell script that can be allocated and executed after either a "cust" or bos_inst operation. This sort of resource is unique in that you can allocate multiple script resources to be executed on a client after an installation. The only downside however is that the order in which those scripts are executed is not a configurable option....typically it ends up random.
Note: The script resources must not point to files that reside in the /export/nim/scripts directory. This directory is used for the nim_script resource that is managed by NIM. NFS restrictions prevent defining multiple resources in the same location.Use your own location for holding your scripts, such as /export/nim/myscripts.

FB_script

A fb_script resource represents a file that is used to configure devices when a NIM client is booting for the first time after the BOS installation process is completed. During BOS installation, certain customization operations (such as device configuration) cannot be performed because they require certain daemons to be running. However, at this point in the BOS installation process, daemons are not available. As a result, certain devices may not be configured during system reboot, and have to be manually configured after the system has booted.

Exclude_files

This is a file with the names/locations of files/directories that you wish to exclude from a mksysb image. Do not confuse this with the target system's own local /etc/exclude file resource. This resource would be useful in cases where you are backing up multiple NIM clients and always want to use the same sort of exclusion for each of them.
Things to note with using exclude files :
1. Make sure all entries start with a "^./" otherwise you might end up excluding files you did not intend to. Using the "^./" makes the exclude start from / and look for your path.
Ex.

```
^./myfiles/
^./mytrash/
```

2. Do not use wildcards. They tend to exclude everything on the system.

3. Excluding files does **NOT** mean that you are excluding a filesystem. If you have a 200gig filesystem called /backups and you add that to your exclude list :
^./backups
All you've done is exclude all of the data in that filesystem. If you restore that mksysb, the /backups filesystem will still be recreated at 200gig, but it will be empty. The best way to exclude a filesystem from a mksysb backup and have it not show up upon restore, will be to unmount the filesystem before running your mksysb.
Alternatively you can put your filesystem entry in the /etc/exclude.rootvg and
edit the image.data and remove the entries for the lv and fs but be VERY careful. You can easily corrupt this file.

4. Be very careful with what you put in there. Do not either intentionally or unintentionally exclude files that will prevent the system from restoring successfully. If you have :

```
^./usr/sbin
```

...in your exclude file, you're not going to be able to restore that backup.

Installp_bundle

This allows you to install additional filesets to a NIM client during a bosinst operation. If you want all of your installed clients to have a specific set of filesets, you may want to create and allocate an installp bundle that will be added to the default set of filesets installed during an overwrite installation (just as an example).

Removing NIM resources

The last thing we need to know about NIM resources, is how to remove them. The command is quite easy, and the only thing you need to verify first is that it is not allocated to any clients.
You can check any resource's "Alloc_count" by running an 'lsnim -l' against that resource name.

From command line :

```
# nim -o remove
```

From SMIT :

```
# smitty nim_rmres
-or-
# smitty nim
=> Perform NIM Administration Tasks => Manage Resources => Remove a Resource
```

Define NIM Clients

Your NIM clients are defined to the master in one of two ways. The master can define clients to itself or a machine can tell a master that he is defining himself as a client. You can also disable the feature of allowing clients to define themselves to the NIM environment (by default this is turned on).

Since this is a NIM operation we will be using the 'nim' command to define clients. There is a new way of defining and installing multiple clients at once using the 'nim_clients_setup' command, but since this is an introduction/beginning guide, we will not be covering that here.

When a NIM master runs an operation to define a NIM client it does not actually go out and contact the client or provide any sort of information to the client letting it know that it is being defined. This is a purely informational operation that updates NIM. The only requirement you need is for the master to be correctly able to resolve the client's hostname.

From command line :

```
# nim -o define -t standalone -a platform=chrp -a if1="find_net lucidbso 0" -a netboot_kernel=mp -a connect=shell
```

Breaking down the command
 # nim -o define -t standalone : This is the NIM operation of defining an object of type=standalone.

... -a platform=chrp : The platform type of the target system. After AIX 5.2 this will always be chrp.

...-a if1="find_net lucidbso 0" : Used when a client is part of an existing NIM network. In this case my NIM client belongs to my master's NIM network (i.e. They're on the same subnet) so I don't need to give it further information. If this client needed to be on a new network then we would not use the "find_net" but rather the "net_definition= <value>....." options where you'd need to specify "networktype" "subnetmask" "clientgateway" "networkname" attributes.

*... -a netboot_kernel=mp :*This should always be set to mp for a multiprocessor and in the future they will probably have an mp64 option.

... -a connect=shell : This will be the connection type used. You can either choose shell (rsh) or nimsh.

... *<clientname> :* The clients resolvable hostname.

From SMIT :

```
# smitty nim_mkmac
-or-
# smitty nim
==> Administration Tasks ==> Manage Machines ==> Define a Machine
```

From here you are asked what the hostname of the NIM client is. If the master can resolve the client's hostname and he already has a NIM network associated with the client's network then it will "pre-generate" a final acceptance screen for you (see 1 below). If the master does not already have a NIM network defined for this client, then you will be prompted for more information and be asked to create a new network for this new client (see 2 on the next page).

```
* NIM Machine Name                              [clientname]
* Machine Type                                  [standalone]
* Hardware Platform Type                                [chrp]
Kernel to use for Network Boot                  [mp]
Communication Protocol used by client           []
Primary Network Install Interface
*   Cable Type                                          bnc
Network Speed Setting                           []
Network Duplex Setting                          []
*   NIM Network                                 master_net
*   Host Name                                           client_hostname
Network Adapter Hardware Address                        [0]
Network Adapter Logical Device Name             []
IPL ROM Emulation Device                                []
CPU Id                                          []
Machine Group                                   []
Comments                                                []
```

Notice that you actually don't have to fill in anything else as a required field. The master can resolve the client's hostname, and he has put the client on the same network (master_net) because in this case, the client and master are on the same network.
I recommend leaving the optional fields blank as they currently are (especially if your NIM client is 5.2 or below, as some options are not valid for 5.2 clients).
Pressing <enter> from here will define the client.

-or-

```
* NIM Machine Name                              [clientname]
* Machine Type                                  [standalone]
* Hardware Platform Type                                [chrp]
Kernel to use for Network Boot                  [mp]
Communication Protocol used by client           []
Primary Network Install Interface
*   Cable Type                                          bnc
Network Speed Setting                           []
Network Duplex Setting                          []
*   NIM Network                                 [10_14_20_Net]
*   Network Type                                        ent
*   Ethernet Type                                       Standard
*   Subnetmask                                  [255.255.254.0]
*   Default Gateway Used by Machine             [10.14.20.1]
*   Default Gateway Used by Master                      [9.3.58.1]
*   Host Name                                           clientname
Network Adapter Hardware Address                        [0]
Network Adapter Logical Device Name             []
IPL ROM Emulation Device                                []
CPU Id                                          []
Machine Group                                   []
Comments                                                []
```

Notice that there is much more to fill out here, because we're defining a new network within NIM. We'll say my new NIM client's ip = 10.14.20.34. I'm naming my NIM network 10_14_20_Net so that it is descriptive just by looking at the name. Also, any other client on this same network will now be auto-defined on this new network and the SMIT screens will look like example 1 above. We'll also need to enter my subnetmask and the gateway used by this client. The master's gateway is filled in for you. Once this successfully completes I'll now have 2 NIM networks.....master_net and 10_14_20_Net. (no I didn't do the math to see if that was a valid ip/snm combination).

Using 'niminit' from the client

Installing the required fileset :

Alternatively we can define a new client to the NIM environment from the client side as well. First of all you need to make sure that the NIM client fileset (bos.sysmgt.nim.client) is installed to the client system.
The client fileset should be installed now on all systems by default. Verify this by running the following
command :

```
# lsnim -l bos.sysmgt.nim.client
```

If the fileset is not installed proceed to the next step(s). If it is installed proceed to the "Using the NIM client to define itself to the master" section.

From command line :

Put V1 of the base AIX installation media in the drive on the client and run the following :

```
# installp -acgXd /dev/cd0 bos.sysmgt.nim.client
```

From SMIT :

```
# smitty install_all

* INPUT device / directory for software                 /dev/cd0
* SOFTWARE to install
PREVIEW only? (install operation will NOT occur)                no
COMMIT software updates?                                        yes
SAVE replaced files?                                            no
AUTOMATICALLY install requisite software?              yes
EXTEND file systems if space needed?                   yes
OVERWRITE same or newer versions?                      no
VERIFY install and check file sizes?                            no
DETAILED output?                                                no
Process multiple volumes?                                       yes
ACCEPT new license agreements?                         no
Preview new LICENSE agreements?                        no
```

*do not install any of the other 2 NIM filesets (bos.sysmgt.nim.master / bos.sysmgt.nim.spot) to a client. The client will then presume it is intended to be a NIM master and not allow any client functions to be ran against it.

Using the NIM client to define itself to the master

For this example the client name will be lucidbso. The NIM master's name will be shadoebso. In both cases we will be defining the client from the client side. Since hostname resolution is very important in NIM, it is a good idea to make sure your /etc/hosts file has correct information concerning the NIM master.

From command line :

```
# niminit -a name=lucidbso -a master=shadoebso -a pif_name=en0 -a platform=chrp
-a netboot_kernel=mp
```

To check this was successful we can run the following command on the client :

```
# nimclient -l -l lucidbso

lucidbso:
Class                   = machines
Type                    = standalone
Connect         = shell
Platform                = chrp
Netboot_kernel  = mp
If1                     = master_net lucidbso.austin.ibm.com 006094E93502 ent0
Cable_type1             = N/A
Cstate                  = ready for a NIM operation
Prev_state              = ready for a NIM operation
Mstate                  = currently running
Cpuid                   = 000890164C00
```

We will get the exact same output if we are on the NIM master and run the following command :

```
# lsnim -l lucidbso
```

From SMIT :

```
# smitty niminit
-or-
# smitty nim ==> Configure Network Installation Management Client Fileset

* Machine Name                          [lucidbso]
* Primary Network Install Interface          [en0]
* Host Name of Network Install Master   [shadoebso]
Hardware Platform Type                  chrp
Kernel to use for Network Boot          [mp]
Communication Protocol used by client   []
Ethernet Interface Options
Network Speed Setting                   []
Network Duplex Setting                  []
```

NIM Client Installations... How To Do Them Successfully

Finally we get to the good stuff.....how to install your newly created clients using your newly created resources. This section is broken down into 3 subcategories :

Installation Types
- rte
- mksysb
- spot
- update_all / single fileset install

Installation Methods
- push
- force push
- pull

Installation Types

Installation Type : rte

This will be either a Migration, Preservation, or New and Complete Overwrite install.

Installation Type : mksysb

This will be taking a mksysb image and either restoring it back to the system it came from or cloning that image to another system.
* A common problem when running a mksysb install is attempting to use a SPOT that is at a lower level than your mksysb image. NIM has a built in check now and will warn you if you are attempting to use a lower level SPOT, however it is a good idea to make it a habit to check this yourself.

Installation Type : SPOT

This is a very rarely used installation method. By doing this you take the SPOT that you are using for the installation, and copy it over. If you do not have the proper device support in your SPOT, the installation will complete, but the system will not boot.
*I will not be reviewing SPOT installs as they are so rarely used, however after going through the next section, you should have no problem setting one up should you wish to do so.

Installation Type: update_all

The last installation type we'll cover is the update_all. This can be used to update a master, client. or SPOT with individual APARs, technology levels, or service packs.
Most commonly you will use your lpp_source resource to run an update_all to a NIM client that is at a lower technology level. In this case you do not need a matching SPOT resource, as there is no initial boot required of the client. The master simply NFS mounts over the lpp_source location to the client which ends up running his own update_all operation.

*In all cases, it is always recommended to verify that your target system is up at the latest level of firmware before running any of these installations. Trust me, this isn't one of those, "eh I hate updating firmware, I'll just skip it this time" sorts of situations. This is a "seriously....make sure your firmware is up to the latest level" situations.
Below is the firmware download site. Instructions on installing the firmware are also found by clicking through to your firmware level.
FixCentral Firmware Download
http://www-933.ibm.com/support/fixcentral/

If planning to run an update_all to either your NIM master or a client machine you should refer to the DCF document Updating to a New Technology Level or Service Pack to go through recommended pre/post checks.
You can also use the search function at the DCF site to run a search for the document by using the same name.

From command line :

```
# nim -o cust -a lpp_source=5305_lpp -a accept_licenses=yes -a fixes=update_all ClientA
```

variation :
You don't have to necessarily run an update_all. You can also install individual filesets :

```
# nim -o cust -a lpp_source=5305_lpp -a accept_licenses=yes -a filesets= ClientA
```

From SMIT :

```
# smitty nim_update_all
-or-
# smitty nim
=> Perform NIM Software Installation and Maintenance Tasks => Install and Update Software => Update Installed Software to Latest Level (Update All)
```

You'll next select your target client (or SPOT) and the lpp_source you're using for the update.

```
* Installation Target                  lucidbso
* LPP_SOURCE                    5305_lpp
  Software to Install                  update_all
```

As you will see, there is no other option you need to set or change because they will all be filled in for you. If adding new filesets you may need to switch the option to accept licenses to "yes".

Installation Methods

Installation Method : Push

This method of installation is controlled by the NIM master. The client machine must have the bos.sysmgt.nim.client fileset installed and an /etc/niminfo file reflecting correct NIM master information. The NIM master will reset the bootlist to reflect the network adapter defined in NIM, allocate all appropriate resources, and initiate a system reboot. This installation method is commonly used when installing systems in a remote location using a bosinst_data resource to allow for non-prompted installations.
Provided are two examples of running a push install to a NIM client. Following the command line examples are their corresponding SMIT procedures on accomplishing the same tasks.

From command line : EXAMPLE I :

This command will initiate a non-prompted rte push install to ClientA. Depending on the current oslevel of ClientA, I can either run a migration install (if ClientA is running AIX 5.2 for example) or a Preservation (if ClientA is already running AIX 5.3) or a New and Complete Overwrite (regardless of the current level).

```
# nim -o bos_inst -a source=rte -a lpp_source=5305_lpp -a spot=5305_spot -a accept_licenses=yes -a boot_client=yes -a bosinst_data=bi_noprompt_ow ClientA
```

From command line : EXAMPLE II:
This command will initiate a mksysb install to ClientA using its own mksysb image. Since this is a mksysb restore back to the exact same system it was taken from, I do not need to use an lpp_source resource. If I were to be cloning this mksysb to **any other** system....even the same system type, I would need to allocate an lpp_source resource.

```
# nim -o bos_inst -a source=mksysb -a spot=5305_spot -a mksysb=ClientA_mksysb -a boot_client=yes ClientA
```

With this second command, we are not allocating the bi_noprompt_ow bosinst_data resource. Since this will be a prompted install, you'll need to have either yourself or someone else in front of the target system's console to go through the BOS Install Menus.

From SMIT : EXAMPLE I :

If you have decided to initiate a non-prompted rte push install to ClientA from smit you can run one of the following :

```
# smitty nim_bosinst
-or-
# smitty nim
=> Perform NIM Software Installation and Maintenance Tasks => Install and Update Software => Install the Base Operating System on Standalone Clients
```

After selecting your target client and the install type of "rte" you'll be asked to select your lpp_source and SPOT resources. Make sure these match, as you do not want to select an AIX 5.2 lpp_source and 5.3 SPOT. As you'll see there are a lot of options in the following screen. I will group these options into 3 different sections. Section 1 is generally the only section you'll use. In some certain rare circumstances you may want to use the installp flags for the addition of fixes, but it is recommended to leave those as they are for TL and ML updates. The final section on scheduling I've personally never seen anyone use, so it's there for you to play with but beyond that, we won't discuss it. In future SMIT options for installation methods I will only be showing section I output. The following SMIT entries will directly accomplish the same as the first command given earlier for the non-prompted NIM installation.
As you can see - this is a lot easier than command line and makes for fewer chances to have a spelling mistake.
Section I - The main options

```
* Installation Target                               ClientA
* Installation TYPE                                 rte
* SPOT                                              5305_spot
  LPP_SOURCE                                       [5305_lpp]
  MKSYSB

  BOSINST_DATA to use during installation          [bi_noprompt_ow]
  IMAGE_DATA to use during installation            []
  RESOLV_CONF to use for network configuration     []
  Customization SCRIPT to run after installation   []
  Customization FB Script to run at first reboot   []
    ACCEPT new license agreements?                 [yes]
  Remain NIM client after install?                 [yes]
  PRESERVE NIM definitions for resources on        [yes]
  this target?

  FORCE PUSH the installation?                     [no]

  Initiate reboot and installation now?            [yes]
    -OR-
  Set bootlist for installation at the             [no]
     next reboot?

  Additional BUNDLES to install                    []
    -OR-
  Additional FILESETS to install                   []
    (bundles will be ignored)
```

Section II - installp Flags
These would be used if you had a series of fixes you were adding on top of the base code and you wanted to for example have them installed in the "APPLIED" versus "COMMITTED" state. It is not recommended to install large groups of fixes or technology levels in the "APPLIED" state as the reject operation is not intended to backlevel entire technology level updates.
Bottom line - you're best off not relying on that as a backout method. Use alt_disk or multibos for that. Leave these as they are.

```
COMMIT software updates?                           [yes]
SAVE replaced files?                               [no]
```

```
AUTOMATICALLY install requisite software?       [yes]
EXTEND filesystems if space needed?             [yes]
OVERWRITE same or newer versions?               [no]
VERIFY install and check file sizes?                    [no]
ACCEPT new license agreements?                  [no]
  (AIX V5 and higher machines and resources)
Preview new LICENSE agreements?                 [no]
```

Section III - Scheduling
Not much I can say for this. I've never seen anyone use it. I would only recommend using it if you're positive your NIM setup is perfect because if you're counting on an install to be complete when you come in the next morning and all you see is a led 608 hang, you'll be pretty disappointed.

```
Group controls (only valid for group targets):
Number of concurrent operations                 []
Time limit (hours)                                      []

Schedule a Job                                  [no]
YEAR                                                    []
MONTH                                           []
DAY (1-31)                                              []
HOUR (0-23)                                             []
MINUTES (0-59)                                  []
```

From SMIT : EXAMPLE II:
This final smit screen will correspond to what the second NIM command accomplished. This will restore the mksysb image ClientA_mksysb back to ClientA.

```
# smitty nim_bosinst
-or-
# smitty nim
=> Perform NIM Software Installation and Maintenance Tasks => Install and Update Software => Install the Base Operating System on Standalone Clients
```

After selecting your target client and the install type of "mksysb", you'll be asked only for the SPOT to use. Remember, if you're restoring back to the exact same system, an lpp_source is not needed. If you're restoring to any other system, then you should also select an lpp_source at the same level as the SPOT and mksysb resources.

```
* Installation Target                                   ClientA
* Installation TYPE                                    mksysb
* SPOT                                          5305_spot
LPP_SOURCE                                      []
MKSYSB                                          ClientA_mksysb

BOSINST_DATA to use during installation         []
IMAGE_DATA to use during installation           []
RESOLV_CONF to use for network configuration    []
Customization SCRIPT to run after installation  []
Customization FB Script to run at first reboot  []
ACCEPT new license agreements?                  [yes]
Remain NIM client after install?                        [yes]
PRESERVE NIM definitions for resources on       [yes]
this target ?
FORCE PUSH the installation?                    [no]

Initiate reboot and installation now?                   [yes]
  -OR-
Set bootlist for installation at the next reboot ?      [no]

Additional BUNDLES to install                   []
  -OR-
Additional FILESETS to install                  []
(bundles will be ignored)
```

Installation Method : Force Push

This is similar to a push, as the master still initiates the bootlist change and reboots the client. In this case however, the client is not necessarily configured as a client of the NIM master....and does not have to have knowledge of the NIM environment. In order to accomplish this, you need to meet 2 requirements.
1. The NIM master must have 'rsh' permission to the client.
2. You need to have and allocate a bosinst_data resource. It doesn't have to be for a non-prompted install or pose any sort of configuration usage for the install, but one has to be allocated.

From command line :
The only difference between the push command we ran earlier and the force push is an additional attribute flag : *-a force_push=yes*

```
# nim -o bos_inst -a source=rte -a lpp_source=5305_lpp -a spot=5305_spot -a force_push=yes -a accept_licenses=yes -a boot_client=yes -a bosinst_data=bi_noprompt_ow ClientA
```

The same will apply to the second example with the mksysb push :

```
# nim -o bos_inst -a source=mksysb -a spot=5305_spot -a mksysb=ClientA_mksysb -a boot_client=yes -a bosinst_data=bi_prompt_mksysb -a force_push=yes ClientA
```

*notice in the above command I had to add a new bosinst_data resource because a force push install requires that one be allocated, even if it is not going to be used. Any options you select in the BOS menus will override the options set in the allocated bosinst_data resource or the bosinst.data file that is contained within the mksysb.

From SMIT :
Again, there is just one main difference here. You simply switch the following entry in your smit screen from "no" to "yes". Of course, we have to make sure we have a bosinst_data resource allocated.

```
BOSINST_DATA to use during installation [bi_prompt_mksysb]
FORCE PUSH the installation?            [yes]
```

Again, since this is just an attribute change, not a change in the methodology of the install. If you choose to allocate a non-prompted bosinst_data then the install will proceed on its own using the configurations you selected in the file, otherwise you will have to be at the system console to provide the necessary selections.

Installation Method : Pull

Part I - Setting up the nim master
This is the most common sort of NIM install that we see today. In this case, the NIM master will setup for the install, but not initiate it. All resources will be allocated - and the master will wait for a bootp request from the NIM client. In order to initiate this request, the NIM client must be booted to SMS (System Management Services). At the SMS menus you can enter the IP address information, set the bootlist, and initiate the boot. This method is required for systems that are unable to boot into normal mode, or for those systems that have no operating system on them.

From command line :
The setup command again does not change much from the push install. All we are doing is changing an attribute to the bosinst operation telling it not to reboot the NIM client, thus changing the install type to a pull install : *-a boot_client=yes* to *-a boot_client=no*

```
 # nim -o bos_inst -a source=rte -a lpp_source=5305_lpp -a spot=5305_spot -a boot_client=no -a accept_licenses=yes -a bosinst_data=bi_noprompt_ow ClientA
```

The same will apply to the second example with the mksysb

```
# nim -o bos_inst -a source=mksysb -a spot=5305_spot -a mksysb=ClientA_mksysb -a boot_client=no ClientA
```

From SMIT :
Again, there is just one main difference here. You simply switch the following entry in your smit screen from "yes" to "no".

```
Initiate reboot and installation now?           [no]
```

In this case we are changing the methodology of the install even though it is just an attribute of the overall bosinst command that is altered.

Installation Method : Pull

Part II -The master is set up....now what ?
The NIM master is now ready and waiting for a bootp request from the client. In order to do this we need to boot the client machine into SMS (System Management Services). Depending on your machine type there are a few different ways to boot to SMS.

1. Using a graphical terminal, during system boot you'll see icons going across the bottom of the screen. Anytime after the "keyboard" icon and before the "speaker" icon you can press the F1 key to indicate that you wish to boot to SMS.

2. Using an ASCII terminal, during system boot you'll see words come across the bottom of the screen. Anytime after the word "keyboard" and before the word "speaker" you can press the 1 key to indicate you wish to boot to SMS. This will be the 1 key on the number line, not on the number pad.

3. If you have an HMC managed environment, you can simply boot your target system using the SMS profile.

If you are one of those whose terminal seems to not come up in time to be able to see the icons or words come across the screen, the LED display during the time you have to press the appropriate button is E1F1. Also, sometimes you'll notice on graphical systems it specifically asks for 1 instead of F1....it should display on the screen the correct button to press.

Your firmware level can depend on what sort of options you see in SMS. I don't have access to all variations of the menu options, so I'll just use the one my NIM master has. You should be able to navigate closely enough, as the wording should be similar.

```
SMS - SYSTEM MANAGEMENT SERVICES -
1. Select Language
2. Change Password Options
3. View Error Log
4. Setup Remote IPL (RIPL (Remote Initial Program Load))
5. Change SCSI Settings
6. Select Console
7. Select Boot Options
```

First of all we need to provide the IP addresses necessary to tell the client who he is, and where to boot from.

```
4. Setup Remote IPL (RIPL (Remote Initial Program Load))
```

Next you'll have a selection of adapters to use. Select your adapter that corresponds to the adapter/host you defined in NIM. You will not see "ent0/ent1...etc" options. You will however see the hardware addresses and slots.

We then are brought to 3 options :

```
1. IP Parameters
2. Adapter Configuration
3. Ping Test
```

Now we need to set our IP Parameters.

```
1. Client IP Address            [###.###.###.###]
2. Server IP Address            [###.###.###.###]
3. Gateway IP Address   [###.###.###.###]
4. Subnet Mask          [###.###.###.###]
```

Enter the correct addresses for each field.
VERY IMPORTANT
If your NIM master and client are on the SAME subnet, then you will set the **Gateway IP Address** to be the NIM master's ip address.....(usually). There were a few firmware levels that made you use 0.0.0.0 if the master and client were on the same network, but running into that is very rare. Use the NIM master's ip.

If you are unsure or need to verify if your master and client are on the same network you can always go back to your NIM master and check to see what NIM thinks. Of course, NIM is only correct if the information provided to it was correct originally.

On the NIM master you'll want to compare the "if1=" lines on the master and client :

```
# lsnim -l master |grep if1
if1                 = master_net shadoebso

#lsnim -l ClientA |grep if1
if1            = master_net dipperbso
```

So we know the master and client are defined on the same subnet. If the client is defined on a different NIM network than the master, then use the client's gateway as the Gateway IP Address entry.

After setting the IP Addresses use 'M' to return to the main menu. You typically do not want to go into the "Adapter Parameters" (option 2 on the previous screen) to change the adapter parameters or disable spanning tree. Spanning Tree goes out and wakes up parts of the network that might have been disabled since they're not in use. This can increase performance but also block bootp if it is not expecting to receive a request through that portion of the network.

Also, with the Ping Test....the ping test is not a reliable way to determine if bootp is going to work. In fact, we use it the other way around. If bootp is failing for some reason we may come back and check the ping test to see if that is having a problem as well. Do not presume bootp will fail if the ping test fails. It may, but it is not a reliable indication.

With our IP Parameters set we should now be back at the main menu.

SMS - SYSTEM MANAGEMENT SERVICES -

```
1. Select Language
2. Change Password Options
3. View Error Log
4. Setup Remote IPL (RIPL (Remote Initial Program Load))
5. Change SCSI Settings
6. Select Console
7. Select Boot Options
```

Now we're ready to select our boot device.

```
7. Select Boot Options
```

The next menu should come up :

```
1. Select Install or Boot Device
2. Configure Boot Device Order
3. Multiboot Startup
```

You can take either option 1 which will make this a 1 time boot device selection, or option 2 which will permanently change the boot device order until either you or the system changes it back to your boot drive after the install is complete.

```
1. Select Install or Boot Device
```

Select Device Type :

```
1. Diskette
2. Tape
3. CD/DVD
4. IDE
5. Hard Drive
6. Network
7. List all Devices
```

Trust me, make your life easier and select :

```
7. List all Devices
```

The system will go out for you and scan itself to determine which devices are available to boot from. All of your available boot devices will be displayed here. The menu can be a little tricky here. If you have a device pre-selected it will have a 1 next to it under the “Current Position” column. Use the “Select Device Number” listing to choose the device you want to boot from.
The next screen will offer you three choices :

```
1. Information
2. Normal Mode Boot
3. Service Mode Boot
```

Select :

```
2. Normal Mode Boot
```

It shouldn’t really matter if you select normal or service mode, I always select normal mode.
Finally it asks if you’re sure you want to exit from SMS. Select ‘yes’ and let the boot go.

What you SHOULD see :
You’ll likely see a brief splash screen then the bootp request attempt.
Ideally you’ll see something like :

```
BOOTP : S=1             R=1
```

Which indicates it sent and received a request successfully. If you start counting up like this :

```
BOOTP : S=7             R=0
```

...then you’ve got a bootp problem. We’re not covering troubleshooting yet, so we’ll presume bootp was successful.

Next the master will tftp the boot image over so you should see a number rapidly increasing to roughly 24000-ish. Once that is complete the client will go through all of the necessary LED codes to establish proper communication and NFS mounts. After that you will be presented with the BOS menu screens (if running a prompted install).

Troubleshooting : When Things Go Wrong

Ok so let’s be real. This is the part you really care about right. What happens when the perfect world of NIM that I’ve presented goes up in smoke. This section will cover the main problems that people come across during NIM installs, namely - the LED hangs. First of all let’s look at a bootp issue.

There are 7 LED hangs that are most common. We’ll go through them one at a time, discuss what causes them, and I’ll provide a brief set of checks/steps that you can run to help resolve those LED hangs on your own.

Bootp

This is the first thing that can go wrong. If you don’t have bootp connectivity, then forget the rest of the install because you’re not going anywhere. The most common causes of a bootp failure :

Bootp failure : Forgetting about the NIM master
Trust me, this does happen. Sometimes you’re so into getting your system installed that you forget the most important thing…letting the master know you’re running an install. When running a pull install don’t forget to initiate the bosinst operation on the NIM master. Unless the master knows that a client will be sending a bootp request, he’ll ignore it.
You can check by running an ‘lsnim’ listing on the master side against the target client. Make sure that the “Cstate” of the client reflects that you have initiated the bosinst operation.

```
# lsnim -l ClientA
ClientA:
Class                   = machines
Type                    = standalone
Connect         = shell
Platform                = chrp
Netboot_kernel  = mp
if1                     = master_net ClientA 0
Cable_type1     = bnc
Cstate                  = BOS installation has been enabled
Prev_state              = ready for a NIM operation
Mstate                  = currently running
Boot                    = boot
Lpp_source              = 5305_lpp
Nim_script              = nim_script
Spot                    = 5305_spot
Cpuid                   = 000ACD3A4C00
Control         = master
```

Bootp failure : Make sure it’s running
Sneaky network admins might turn off bootp on your NIM master. Typically this is done for security/auditing reasons. If bootp isn’t running, even if you initiate the bosinst operation, the master will not start it for you.

```
# lssrc -t bootps
Service         Command         Description             Status
bootps          /usr/sbin/bootpd        bootpd /etc/bootptab    active
```

If this is set to “inoperative” try starting it from the command line :

```
# vi /etc/inetd.conf  ==> make sure 'bootps' and 'tftp'  are uncommented ==> save the file.
# refresh -s inetd
# lssrc -t bootps
```

Bootp failure : The /etc/bootptab file
This is where the master gets the information on which client should be making a request. This file should be empty (except for the commented informational section) unless a bootp operation has been initiated. Once initiated the most recent entry should be for the target client. It will look similar to this :

```
ClientA:bf=/tftpboot/ClientA:ip=#.#.#.###:ht=ethernet:sa=#.#.#.###:sm=###.###.###.#:
```

You’ll want to make sure all of the IP address entries are correct. If any of them are wrong, then you have a definition issue with NIM.

Bootp failure : The number 1 rule, nobody knows about
The most common reason for bootp failure is not knowing what to put in the SMS menus for a correct “Client’s Gateway” entry.

Earlier during the “installation” portion of the guide we briefly discussed the SMS menu entries. If your NIM master and client are on the **SAME** NIM network then you need to use the master’s ip address as the gateway entry.
Ex.
Client IP Address [###.###.###.###]
Server IP Address **[###.###.###.###]**
Gateway IP Address **[###.###.###.###]**
Subnet Mask [###.###.###.###]

Only use the client’s actual gateway address if the client is on a different NIM network than the master.

Bootp failure : Routers / Network Admins / Access Denied
If your master and client are on different NIM networks - and everything else is correct - and you’re still getting bootp failures - your network admins may be at fault here. The router between your client and master may have bootp packet forwarding turned off. Again, this is in some environments considered to be a security risk.

One thing you can do to test this theory is to set bootp into debug. What we’re doing here is stopping the bootp daemon and restarting it with debugging turned on.
- This will lock the window that is currently open on the master so make sure you have a free window open.
- Doing this will also comment out bootp from your /etc/inetd.conf file, so make sure you uncomment it and manually restart it after your testing is complete.

This process below is what we would follow if all else above checks out correctly.
We’ll first reset and deallocate everything so we can start from scratch.
To put bootp into debug on the NIM master :

Window 1

```
# nim -Fo reset
# nim -o deallocate -a subclass=all
# stopsrc -t bootps
# ps -ef |grep bootp          (just to make sure nothing is running)
# bootpd -s -d -d -d -d
```

At this time your window will be locked. Then we are now listening for any bootp requests being made. If the master hears one it will display the information on this screen.

Window 2

```
# smitty nim_bosinst      (and we setup for the install)
```

On the NIM client we run through the SMS screen as normal and kick off the boot from the adapter.

Two possibilities :
1. Failure - If the master does NOT display any bootp request information in the locked window either the adapter you’re using is bad, or the router between the client and master is set to not forward this broadcasted bootp request.

2. Failure - If the master receives it but does not send it back, you most likely have a problem with the ip addressing, the bosinst wasn’t successful during the setup, or the master has a mac address issue with the client.

Remember, once you’ve completed your testing, just Ctrl-C out of Window1 and restart bootp.

The Fatal LEDs
There are 7 LED hangs that we run into quite often. Again, we’ll briefly discuss them, the causes, and I’ll give the most common way to fix them. I can’t guarantee that the solution will work every time, as sometimes LED hang resolutions take more than a few simple commands....but this should give you a good starting point for problem determination.

LED 605 : Device Driver Problems :
This indicates that the SPOT you have allocated does not have the correct device support to configure the NIM client’s chosen adapter.

To check your NIM SPOT for a specific fileset :

```
# nim -o lslpp  |grep
```

The driver is likely missing or in a bad state. If you have a missing device driver you’ll want to add that fileset into your NIM SPOT resource (see earlier in the guide for instructions).

LED 607 : Device Driver Problems II :
This is similar to a 605 in that it is a failure with your network card. In this case however it is the ‘ifconfig’ command that failed. The adapter may not be supported with the oslevel being used, the device driver in the SPOT may be too far downlevel, or the system or adapter firmware may need to be updated.

LED 608 : Bad Network Information :
Probably the most common NIM related LED hang. Somewhere in your NIM environment, you’ve given it some bad IP/NETWORK information. Start by checking hostname resolution, check the client’s definition, the client’s network definition, the master’s network definition.....somewhere, something is wrong. Unfortunately, I can’t really be more specific than that. Remove and readd the client to make sure NIM still believes it should be defined on the network it currently is defined on.
A few years a go a 608 could also be caused by a downlevel SPOT, or a SPOT with low level filesets. This hasn’t caused a 608 in a long time, but sometimes causes for LEDs change.

LED 611 : NFS
Either (most commonly) something hosed up in NFS, or you’ve got a hostname resolution problem. NIM tends to get really picky about what is NFS exported, and how it got that way. Typically running a full NFS reset and recycle will clear up a 611.

To run a quick NFS cleanup simply execute the following commands on the NIM master.
As always, run a reset and deallocate of the NIM client first.

```
# nim -Fo reset
# nim -o deallocate -a subclass=all
# cd /etc
# stopsrc -g nfs
```

```
# rm exports        (alternatively you can just rename the exports file if you wish to keep it)
# touch exports
# rm rmtab xtab (removing cache files)
# cd /var/statmon
# rm -rf ./         (here you're removing 2 directories and a file. If you don't like rm -r commands                    feel free to remove them however you feel comfortable. Just make SURE you're                    in /var/statmon)
# startsrc -g nfs
# smitty nim_bosinst      (setup for your install again)
```

Typically, all other things being good, this will clear up a 611. If not, start focusing on a hostname resolution issue.

LED 613 : Routing incorrectly
You have bad routing information in your setup. Unlike an LED 608, this is at least more specific. Check the "routing1" information listed for your client.

```
# lsnim -l
```

It's probably wrong. You can also check your client's network information and your master's network information.

```
# lsnim -l
```

LED 622 : Speaking at Different Speeds
You'll not actually notice a failure here, because the install WILL proceed.....it'll just take forEVER to do so.
Your NIM master, router (if one exists), and/or client are probably talking at different speeds.....(usually). Make sure that your master, client, and router are all set at the same adapter speed. Sometimes you can have them all matched up and you're still going dog slow during your install.

LED 888 : Access Denied!!
Your NIM client does not have root access to the SPOT that has been allocated. Ideally when you kick off your NIM install, the entry for the exports in /etc/exports should read similar to :

```
/export/nim/spot/5305_spot/usr -ro,root=ClientA,access=ClientA:
```

A bad entry can look similar to :

```
/export/nim/spot/5305_spot/usr -ro,anon=ClientA
```

To resolve this :

```
# nim -Fo reset
# nim -o deallocate -a subclass=all
# nim -Fo reset
# nim -o deallocate -a subclass=all
# cd /etc
# stopsrc -g nfs
# rm exports        (alternatively you can just rename the exports file if you wish to keep it)
# touch exports
# rm rmtab xtab (removing cache files)
# cd /var/statmon
# rm -rf ./         (here you're removing 2 directories and a file. If you don't like rm -r commands                    feel free to remove them however you feel comfortable. Just make SURE you're                    in /var/statmon)
# startsrc -g nfs
# stopsrc -s nimesis     (recycle the nimesis daemon)
# startsrc -s nimesis
# smitty nim_bosinst     (setup for your install again)
```

Kicking off the bosinst operation should export the SPOT and all other resources properly.

Thank you very much for taking the time to read through this guide. I hope it has been not only helpful but an easy read. If you feel you have found any mistakes or inconsistencies please don't hesitate to email me at *storm@us.ibm.com*. If you have any *technical questions* concerning the document, please follow normal support procedures and open a PMR. The first available technician in the Install team will be happy to assist you.

Rate this page:

Rated by 5 users