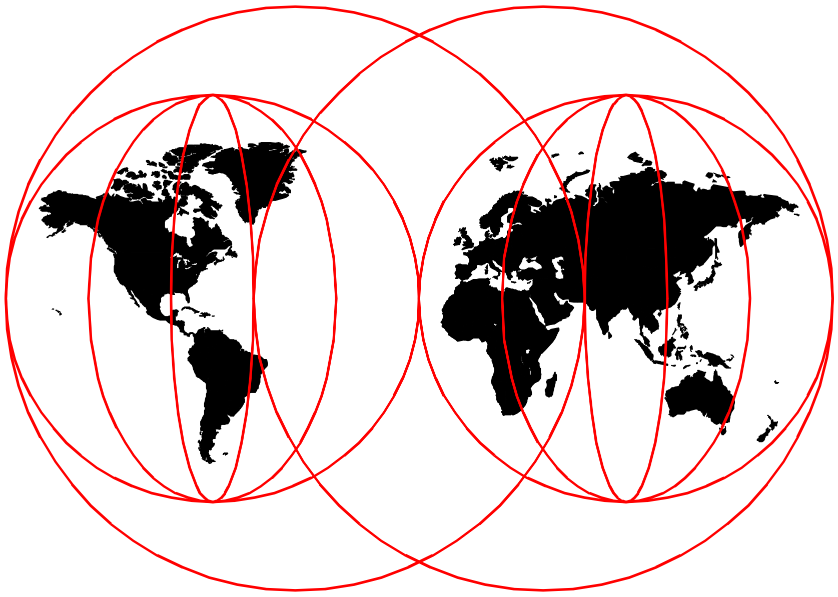
NIM: From A to Z in AIX 4.3

KyeongWon Jeong, Karl Heinz Uhl, Maura Prendiville, Wayne Andrews



International Technical Support Organization

www.redbooks.ibm.com

SG24-5524-00

SG24-5524-00

International Technical Support Organization

NIM: From A to Z in AIX 4.3

February 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 293.

First Edition (February 2000)

This edition applies to IBM RS/6000 for use with the AIX Operating System Version 4 and is based on information available in October, 1999.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures . . . vii

Tables . . . xi

Preface . . . xiii
The team that wrote this redbook . . . xiii
Comments welcome . . . xiv

Chapter 1. NIM basics . . . 1
1.1 Overview . . . 1
  1.1.1 History . . . 1
  1.1.2 Features . . . 2
  1.1.3 Benefits . . . 2
  1.1.4 Five easy NIM steps for beginners . . . 3
1.2 Environment . . . 4
  1.2.1 Machines . . . 4
  1.2.2 Networks . . . 8
  1.2.3 Resources . . . 10
  1.2.4 Groups . . . 22
1.3 Network boot . . . 26
  1.3.1 Booting a machine over the network . . . 26
  1.3.2 Overall process . . . 32
  1.3.3 IPL-ROM . . . 38
1.4 Setup . . . 40
  1.4.1 Planning . . . 40
  1.4.2 Master setup . . . 42
  1.4.3 Network definition . . . 49
  1.4.4 Routing information . . . 53
  1.4.5 Machine definition . . . 58
  1.4.6 Resource definition . . . 67
  1.4.7 Resource allocation . . . 69
  1.4.8 Installation . . . 71
  1.4.9 Quick setup . . . 85
1.5 Administration . . . 87
  1.5.1 Customization . . . 87
  1.5.2 Maintenance . . . 91
  1.5.3 Client operations . . . 93
  1.5.4 Other NIM operations . . . 98
1.6 Differences between AIX versions . . . 102
  1.6.1 Machines . . . 102
  1.6.2 Networks . . . 103

1.6.3 Resources 103
1.6.4 Groups 106
1.6.5 smitty and Web-Based System Manager 106
1.6.6 Commands, operations and attributes 107
1.6.7 Other enhancements 107
1.7 Managing resources 108
1.7.1 Maintaining software in an lpp_source 108

Chapter 2. Working with NIM and advanced NIM topics 113
2.1 Cloning an RS/6000 with an mksysb image 113
2.1.1 Prerequisites for cloning an RS/6000 113
2.1.2 Cloning considerations 114
2.1.3 Cloning: Step by step 114
2.2 Cloning scripts for AIX Version 4.1 117
2.3 Alternate disk installation (alt_disk_install) 117
2.3.1 Prerequisites for alternate disk installation on a NIM client 118
2.3.2 Alternate disk installation: Step by step 119
2.4 Firmware and ROS menus 122
2.4.1 System Management Services menu 122
2.4.2 Remote Initial Program Load (RIPL) 123
2.4.3 Changing the bootlist to network device 124
2.5 Managing resources 125
2.5.1 Maintaining software in an lpp_source 127
2.6 NIM over ATM 130
2.6.1 Prerequisites for booting an ATM NIM client 131
2.6.2 Converting generic networks into ATM networks 132
2.6.3 System recovery after ATM boot failed 132
2.7 NIM in a DHCP environment 133
2.8 Managing diskless/dataless station 136
2.8.1 Adding a diskless or dataless client to the NIM environment. . . 136
2.8.2 Initializing and booting a diskless or dataless client 137
2.8.3 Uninitializing a diskless or dataless client 139
2.9 NIM name resolution 140
2.10 NIM troubleshooting 140
2.10.1 NIM relevant LED codes 140
2.10.2 Debugging a network boot problem 142
2.10.3 Viewing NIM logs 150
2.11 NIM debugging and network debugging 154
2.11.1 Producing debug output from the bootp procedure 155
2.11.2 Producing debug output from a network boot image 158
2.11.3 Producing debug output from the BOS install program 160
2.11.4 Attach an RS/6000 as a tty 161
2.11.5 Producing debug output by using an IP-trace 162

2.12 NIM installation performance considerations 166
2.12.1 NIM clients as resource servers 166
2.12.2 Multithreaded nimesis daemon (AIX 4.3.3 or later) 168
2.12.3 Using the NIM group (AIX 4.2 or later) 169
2.12.4 Using global exports for NIM resources (AIX 4.3 or later) 170
2.12.5 Network options 171
2.13 Case studies 173
2.13.1 Implement NIM preload environment 174
2.13.2 Upgrade operating system 175
2.13.3 NIM in a large environment with multiple networks 176
2.13.4 Software update and maintenance 180
2.13.5 Installation multiple level of AIX 197
2.13.6 Different physical networks on the same logical subnet 198
2.13.7 Using NIM script resource for customization. 200

Chapter 3. NIM in an SP environment 205
3.1 General configuration. 205
3.1.1 NIM objects 207
3.1.2 Differences with SP 207
3.2 setup_server 224
3.2.1 Structure 224
3.3 NIM commands in SP. 228
3.3.1 Managing NIM objects 228
3.3.2 Reviewing NIM information. 243
3.4 NIM problems in SP 246
3.4.1 SPOT creation problem 246
3.4.2 Corruption of an lppsource 250
3.4.3 LED 231 problem 255
3.4.4 LED c48 hang during migration 259
3.4.5 Resource maintenance, creation, allocation, and deallocation . 262
3.5 NIM debugging on the SP 272
3.5.1 Basic problem determination steps. 272
3.5.2 Setting NIM to debug mode 275
3.5.3 An example of using NIM debug in an LED 611 hang. 280
3.5.4 SP LEDs. 284

Appendix A. Key NIM scripts used in the network boot process. . . . 289

Appendix B. Cloning script 291

Appendix C. Special notices 293

Appendix D. Related publications. 297
D.1 IBM Redbooks publications 297

D.2 IBM Redbooks collections 297
D.3 Other resources 298
D.4 Referenced Web sites 299

How to get IBM Redbooks 301
IBM Redbooks fax order form 302

Glossary 303

Index 305

IBM Redbooks evaluation 313

Figures

1. Example bosinst.data file 19
2. Boot sequence 32
3. Network boot flow diagram 36
4. bi_main flow diagram 38
5. wsm nim fast path 43
6. Example niminfo file of NIM master 48
7. Example niminfo file of NIM client 49
8. Communication flowchart. 57
9. Example client definition file. 66
10. smitty screen of a bosinst operation type mksysb 116
11. smitty screen of an alternate disk installation type clone rootvg 121
12. Example /etc/dhcpsd.cnf file 134
13. Starting the DHCP server and examining the address pool 135
14. Checking the infinite lease 136
15. Entries of the bosinst log shown with the showlog command 153
16. Single entries of the boot log shown with the showlog command 154
17. Output from a bootp debug without errors 156
18. Output from a bootp debug with NIM client IP address error. 157
19. Output from a bootp debug with gateway IP address error 158
20. Debug output when the LED 611 hang occurred. 160
21. Diagram of the NIM environment 164
22. IP trace packet 1 165
23. IP trace packet 2 165
24. IP trace packet 3 166
25. IP trace packet 4 166
26. Strategic placement of NIM resource servers 167
27. NIM environment planning diagram 178
28. Contents of an installp_bundle file 182
29. Example of a NIM definition stanza file 196
30. Example of a logical network including two different physical networks . 199
31. Script to configure DNS after AIX 4.1 installation
201
32. Parts of a customization script 203
33. A typical single-frame SP system 206
34. Entering the Site Environment Information 211
35. Entering the SP Ethernet information 213
36. Acquiring the nodes ethernet MAC addresses 215
37. A typical /spdata directory structure. 217
38. Example of the Hardware Perspectives GUI 220
39. Example spmon -d output 221

40. Example of node conditioning logfile 222
41. setup_server flowchart. 225
42. delnimclient flowchart. 230
43. mknimclient flowchart. 231
44. mknimres flowchart 232
45. allnimres flowchart 234
46. unallnimres flowchart 236
47. mknimmast flowchart 237
48. delnimmast flowchart 239
49. mknimint flowchart 240
50. Example lsnim output. 243
51. Example output from the splstdata -b command 244
52. Spot creation error 247
53. Checking the spot.out file. 247
54. Examining the lppsource object with lsnim -l 248
55. Example nim -o check command. 248
56. Checking the lppsource for the simages attribute 249
57. Looking at the filesets required for the Semites attribute. 250
58. mknimres failure. 250
59. Requisite failures in the spot.out file 251
60. Examining the .toc file 252
61. A failing bootp request 255
62. Entering the IP information for the ping test. 257
63. Checking the ping test results 257
64. Checking the /etc/bootptab file. 258
65. Correcting the boot/install server information 259
66. lsnim -l of migrating node. 260
67. Locating the bos_inst migration script 260
68. Examining the migration script. 261
69. Checking the console in a LED c48 hang 261
70. First portion of files archived in the bos image. 262
71. Using bffcreate to copy over AIX 4.2.1 images 263
72. Changing the volume group SDR objects 264
73. Taking an mksysb of a node 266
74. Setting nodes to install from another mksysb resource 267
75. Choosing the update_all option on the CWS. 268
76. Exporting the lppsource directory 269
77. Examining the /smit.script 270
78. Installing PTFs on a node using dsh 271
79. Using dsh to install PTFs on all the nodes in the SP 272
80. exportfs showing the correct exports for an install on f01n01 274
81. Checking the files in /tftpboot. 274
82. Freeing up the SPOT resource 276

83. Checking the enter_dbg attribute 277
84. Waiting for the Trap Instruction interrupt line 278
85. Start of the NIM debug output 279
86. NIM debug console output 280
87. Using lsnim -l to check for errors 281
88. Checking the NFS exports 282
89. Examining further NIM debug information 283

Tables

1. Client operations for different machine types. 5
2. Possible attributes associated with the define option of the nim command. 6
3. Relevant attributes when defining a network object. 8
4. Possible resources available to machines in the NIM environment. 10
5. Possible attributes to choose in defining a SPOT resource. 13
6. Possible attributes to choose when defining an lpp_source resource 15
7. Possible attributes to select in an mksysb definition 16
8. Possible attributes when configuring master 47
9. Attributes that can be defined in the client definition file 63
10. Flag choices when running the nimdef command 66
11. Minimum required and optional resources by machine 70
12. Troubleshooting some network problems 81
13. Attributes that can be used with the bos_inst operation 82
14. Control states. 94
15. Possible attributes to choose from when using the niminit command 95
16. Flags that can be used with the nimclient 97
17. Operations used in conjunction with the nim command. 98
18. Resources available to dataless and diskless machines. 100
19. Overview of NIM relevant LED codes 141
20. Different kind of NIM logs. 151
21. MTU values of network interfaces 172
22. NIM environment summary table 178
23. Minimum required PSSP filesets 218
24. Flags available in spbootins. 241
25. Flags available in spchvobj 242
26. Understanding the output from splstdata -b. 244
27. Flags available in splstdata 245
28. Understanding the .toc file 252
29. pssp_script LED/LCD codes 285
30. psspfb_script LED/LCD codes 287

Preface

NIM is an excellent feature of the AIX operating system and is very important for teams or companies that have a need to install or upgrade many RS/6000 machines with the same images at the same time. NIM features continue to improve as new versions of AIX appear.

This redbook will provide users with an understanding of NIM and its application in today's large-scale multiplatform distributed network environments. Our focus is on the capabilities of NIM, and, with that in mind, we have devoted a section to case studies of NIM usage in AIX Version 4.3 on multiple RS/6000 and SP systems. This redbook will be a valuable tool for system administrators and other technical support personnel and is tailored to their needs.

This redbook is a definitive guide to NIM. It begins with the design concepts, implementation, and benefits of using NIM for managing networked AIX systems. It then explains in detail (and with a pronounced practical bias) how to install, configure, and use NIM to manage connected machines. This redbook contains many step-by-step examples of typical implementation scenarios.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

KyeongWon Jeong is a Senior Software Engineer at the International Technical Support Organization, Austin Center. He writes extensively about AIX and BI as well as producing educational materials. Before joining the ITSO, he worked in IBM Global Learning Services of IBM Korea as a Senior Education Specialist and was a class manager of all AIX classes for customers and interns. He has many years of teaching and development experience.

Karl Heinz Uhl is a staff engineer at the IBM development lab in Boeblingen, Germany. He has two years of experience in the field of administrating large-scale computing environments based on the AIX operating system. He holds a degree in Automation Engineering from the Fachhochschule fuer Technik und Wirtschaft in Reutlingen, Germany. His areas of expertise include AIX system management.

Maura Prendiville is an IT specialist in Dublin, Ireland and has worked for IBM for two years. Her first year was spent supporting a manufacturing PC environment before moving to AIX support of multiple applications running on RS/6000 systems. She holds a degree in Commerce from University College, Cork.

Wayne Andrews is an Advisory IT specialist in the AIX Systems Support Center in Basingstoke, England where he specializes in SP systems and AIX networking support. Now in his fifth year with IBM, for the first two years, he worked as an RS/6000 salesperson before pursuing a more technical role where his areas of expertise include Firewalls and TCP/IP-related problem determination. Prior to joining IBM, he worked as a programmer and systems administrator on DEC VAX and HP/3000 systems.

Thanks to the following people for their invaluable contributions to this project:

Julie Craft, Kevin Fought, Suanne Lowe
IBM Austin

Jean-Michel Berail, IBM France

Volker Haug, IBM Germany

Special thanks to the editing team for their help in finalizing the text and publishing this book:

Milos Radosavljevic
IBM ITSO, Austin center

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks evaluation” on page 313 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. NIM basics

The basics of Network Installation Management (NIM) will form the building blocks of any NIM environment. These basics can be related to the parts or objects within NIM. By maximizing networks, machines, resources, and groups, we will be able to approach NIM’s full potential.

1.1 Overview

The first question to be answered is: What is NIM? Once users have gained an understanding of what NIM is and what it does, we can proceed to its setup, administration, and maintenance. Where procedures are detailed, the user has the option of choosing the Web-Based System Manager GUI, the smitty panels, or the command line. This chapter will close by focusing on how NIM has progressed since AIX Version 3.2.5 and the enhancements in the latest version.

The second question to be answered is: Who would want to use NIM? If any of the following apply to you, you may want to use NIM:

- If you have more than two RS/6000 machines to manage
- If you are tired of using *sneakernet* to install your machines
- If you like using unusual acronyms, such as “SPOT”, “LPP_SOURCE”, and “BOS_INST” or if you like saying “nimesis”
- If you want to impress your boss by installing machines without physically touching them

1.1.1 History

If we look at the NIM family tree, we can trace its origins to Diskless Workstation Management (DWM) and NetInstl, which were available with AIX Version 3.2. NetInstl allowed the installation of the base operating system. DWM was used to minimize the disk requirements on individual dataless/diskless workstations by allowing most system, application, and data files to be stored on remote servers. AIX Version 4.1 built on these offerings, but its major advantage over its predecessor was the capability to push centrally-managed resources. AIX Version 4.2 saw enhancements in usability primarily in the smitty interface. Usability was further enhanced in AIX Version 4.2.1 by the VSM GUI for NIM. The users could use the VSM interface to control NIM, including drag and drop support. ATM networks were supported. Major headway was made in AIX Version 4.3.0. A Web-Based System Manager was introduced and alternate disk installation support were introduced; the user had the option of controlling clients joining the NIM environment, and the global resource exporting feature was added. Checking of the CPU-ID was made optional. The ability to create boot image types corresponding to the defined clients during SPOT creation was added. This prevents NIM from creating all the boot images whether they will be needed or not. If new clients that will need other boot images that were not created earlier are defined, they will be created during SPOT allocation. In Version 4.3.1, support for secure rcmds was added to NIM. NIM locks were changed in Version 4.3.2 to allow more parallelism; smitty and Web-Based System Manager were given the IEEE802.3 attribute, and changes were made to allow concurrent execution control. This brings us to the present and Version 4.3.3. This heralded improvements to NIM itself. Security was improved, influenced no doubt by SPs. Scalability was improved in terms of the multithreaded nimesis daemon, the nim_script resource location, and the ability to propagate resources.

1.1.2 Features

NIM permits the installation and maintenance of AIX, its basic operating system, and additional software and fixes that may be applied over a period of time over token-ring, ethernet, FDDI, and ATM networks. NIM also permits the customization of machines both during and after installation. As a result, NIM has eliminated the reliance on tapes and CD-ROMs for software installation; the onus, in NIM’s case, is on the network. NIM will allow one machine to act as a master in the environment. This machine will be responsible for storing information about the clients it supports, the resources it or other servers provide to these clients, and the networks on which they operate.

1.1.3 Benefits

Some of the benefits of NIM are:

- Manageability - It allows central localization of software installation images, thus, making backup and administration easier.
- Central Administration - Administrators can install remote AIX machines without having to physically attend them.
- Scalability - You can install more than one machine at a time, implement a group strategy of machines and resources, and choose how many machines to install at a time.
- Usability - VSM GUI for NIM has been improved so that, now, it can be used to configure NIM groups.

- Availability - Where server down time means loss of profits, NIM provides you with a backup image of all your servers. A new server can be set up and running in just over an hour.
- Non-prompted installation - NIM provides a function to install systems without having to go to the machine, thus, avoiding the *sneakernet* method.
- Installations can be initiated by either the client or master at a convenient time. For example, if a client is unavailable at the time of the install, you can initiate an install when it is back on line, or, if there is less traffic on your network at a certain time, you can initiate the installations to occur then.
- It is a relatively faster means of installation than tape or CD-ROM.
- NIM provides greater functionality than CD-ROM or tape. Among other things, it allows you to customize an install, initiate a non-prompted install, or install additional software.

1.1.4 Five easy NIM steps for beginners

If you are a beginner with NIM, you may want to follow these five easy steps:

1. Prepare AIX, and install CD-ROMs that are the same levels as your system.
2. Select a master and install bos.sysmgt.nim.master fileset.
3. Configure a master and define resources. NIM will do all of this for you from the `smitty nim_config_env` fastpath.
4. Define clients. If the client is not running, define it on the master with the `smitty nim_mkmac` fastpath. However, if the client is running, install bos.sysmgt.nim.client, and then run the `smitty niminit` fastpath.
5. Install clients using the `smitty nim_bosinst` fastpath on the master machine. If the clients are not running, set Initiate reboot and installation now? to NO and press **Enter**. Then, go to the clients and boot into a firmware menu. If the client is running, set Initiate reboot and installation now? to YES and press **Enter**. It will be rebooted; the install menus will be shown, and you can then proceed with the install.

Don’t worry if you are not yet familiar with some terms, such as master and client. The following section will explain the basic NIM attributes and these terms. You may find that there are many other ways to make it work in your environment.

1.2 Environment

A NIM environment is typical of any client-server environment. You have client machines accessing resources that are remotely held on servers. In the NIM environment, there is also the additional requirement that these resources bring stand-alone, dataless, and diskless machines to a running state. It is obvious, then, that certain resources are required to support the operation of systems within the NIM environment. This capability is dependent upon the functionality of the network.

All information about the NIM environment is stored in three ODM databases (this data is located in files in the /etc/objrepos directory):

- nim_object - Each object represents a physical entity in the NIM environment.
- nim_attr - Stores individual characteristics of physical entities.
- nim_pdattr - Contains predefined characteristics.

The objects that compose the ODM database are machines, networks, resources, and groups. When we speak of their characteristics, we are referring to their attributes that are part of their initial definition. In this definition, we also assign the objects a name. This name is for NIM purposes only and may be totally different from any defining physical characteristic it may have. To have a functioning environment, the following conditions must be met:

- NFS and TCP/IP must be installed.
- TCP/IP must be configured.
- TCP/IP communications must be established between machines.
- Name resolution must be configured.

We will cover the different objects, their considerations, and a strategy for choosing them in subsequent sections.

1.2.1 Machines

There are three basic roles that a machine can assume in the NIM environment: A machine can be a master and a client but also a resource server. In a NIM environment, the master, or some clients, can be a server of some resources. In that case, they must have enough disk space to store the mksysb or lppsource resource. For the master and client, it is not as simple as the client having the bos.sysmgt.nim.client fileset installed and the master having the bos.sysmgt.nim.master fileset installed. The master's role is much more complex than that of the client. It will manage the installation of all the machines in its database. In any environment, we can have only one master but many client machines defined. Client machines can be stand-alone machines, which have the ability to self-boot, dataless machines, which have a local disk drive but are not capable of booting themselves, or diskless machines, which, as their name suggests, have no disk drive and cannot perform a boot themselves. The master is limited to being a stand-alone machine. The method of installation also determines a machine's role. A master can push an installation onto a client, but a client can only pull an installation from the resource server. Because these resources are stored locally, only stand-alone clients can initiate a pull installation. This is another differentiating feature of stand-alone machines versus dataless or diskless machines. As we mentioned before, a client may also be a server (in the client-server sense) of resources to other clients. For the same reason that only stand-alone clients can do pull installations, only stand-alone machines can act as resource servers.

Machine objects have the following information in the database:

- User-defined name of the object
- Hardware address of client network interface(s)
- Hostname of the client's network interface(s)
- Network object names to which the client interfaces connect
- Cable type (for ethernet) or ring speed (for token-ring)
- Location of Initial Program Load Read-Only Memory (IPL-ROM) emulation program, if needed

The platform that client machines run on determines its level of support as well. The platform types that are supported include:

- chrp - Common Hardware Reference Platform
- rspc - IBM Power PC computers
- rs6k - MicroChannel-based RISC System/6000

There are unique operations allowed on each client configuration to initialize the machine. Table 1 shows the operations that can be performed on the different client configuration types.

Table 1. Client operations for different machine types

NIM operation	**Stand-alone machine**	**Diskless machine**	**Dataless machine**
bos_inst	x		

NIM operation	Stand-alone machine	Diskless machine	Dataless machine
dkls_inst		x	
dtls_init			x
diag	x	x	x
cust	x		
fix_query	x		
lppchk	x		
maint_boot	x		
reset	x	x	x
check	x	x	x
showlog	x	x	x
reboot	x	x	x

To define a machine in the NIM environment, we use the following command:

```
# nim -o define -t MachineType -a Attribute=Value... MachineName
```

where the attributes in Table 2 can be assigned:

Table 2. Possible attributes associated with the define option of the nim command

Attribute	Description
-t Machine Type (required)	Specifies the type of machine being defined. Valid values are stand-alone, dataless, or diskless.
-a if=Value(required)	Stores network interface information for a NIM client.
-a ring_speed=Value(optional)	Specifies the ring speed of the client's token-ring adapter.
-a cable_type=Value (optional)	Determines the cable type of the client's ethernet adapter.
-a platform=Value (optional)	Details the platform of the machine being defined. Run the `bootinfo -p` command on a running machine to determine its platform.

Attribute	Description
-a netboot_kernel=Value (optional)	Specifies the kernel type of the client. Valid values are up for uniprocessor machines and mp for multiprocessor machines.
-a iplrom_emu=Value (optional)	Establishes the device that contains the IPL-ROM emulation software. IPL-ROM emulation is required for machines that do not have bootp-enabled IPL-ROM.
-a net_definition=Value (optional)	Defines a NIM network to be associated with the client being defined. When specifying the net_definition attribute to create or change a machine definition, the find_net keyword must be specified as the first component of the if attribute for the machine. The net_definition attribute may also be specified when defining additional NIM interfaces for machine definitions.
-a cpuid=Value (optional)	Specifies the CPU ID of the machine being defined.
-a master_port=Value (optional)	Enumerates the port number that NIM master uses for communication with NIM clients.
-a registration_port=Value (optional)	Enumerates the port number used by clients to register themselves with the NIM master.
-a group=Value (optional)	Gives a machine group that the client should be a member of.
-a comments=Value (optional)	For more reference information or further clarification.
-a verbose=Value (optional)	Displays information for debugging.

Master strategy

The deciding factors in choosing the master machine should cover the following:

- The master machine must have a running version of AIX installed on it; for our purposes, we are using AIX Version 4.3.3.
- The master machine has to be a stand-alone machine because it will store resources locally.

- The master machine must be able to communicate with all the clients in the NIM environment. It should be able to perform a basic `ping` test. The master must also be able to execute commands on the client. In order to do this, the $HOME/.rhosts file on the client must contain an entry for the master root. The file permissions on the .rhosts file must be set to 600.
- The master machine and resource servers should have sufficient disk space and memory in order to serve the resources.
- The master machine should be located in a secure easily-accessible area.
- Considerations in choosing the master machine should include the platform it is running on, the level of AIX installed on it, and disk access throughput.

1.2.2 Networks

The underlying network is what allows the machines in a NIM environment to communicate with each other. If the master cannot talk to its clients, how can it be expected to execute commands on them? Therefore, the performance of the network will impact the performance of our NIM environment. If the network is a simple local area network (LAN), the definition of our network object is simplified. However, possible scenarios could see our token-ring master running on an ethernet network serving resources from a server on a token-ring LAN to clients on an FDDI network accessed by hopping from gateway to router to bridge, and so on. The purpose of the network object is to depict our network topology. If we make changes to our physical network, for example, moving a client onto a token-ring LAN, we must reflect this change in our NIM database. The types of network objects that we support include: Ethernet, token-ring, FDDI, and ATM. All other types are grouped together as Generic. Generic network objects represents a type of network that has no support for a network boot. It's not possible, therefore, to perform the bos_inst and diag operations on it. However, you can use the cust and maint operations.

To define a network object, we use the following command:

```
# nim -o define -t NetworkType -a Attribute=Value... NetworkName
```

where the attributes listed in Table 3 can be specified.

Table 3. Relevant attributes when defining a network object

Attribute	Description
-a net_addr=Value (required)	Specifies the IP address of the network being defined.

Attribute	Description
-a snm=Value (required)	Establishes the subnet mask of the network.
-t NetworkType (required)	Details the type of network being defined. Valid values are atm, tok, ent, fddi, and generic.
-a comments=Value (optional)	This field is used for comments that you may wish to put in to help you clarify what, exactly, you are doing.
-a ieee_ent=Value (optional)	Specifies the IEEE 802.3 ethernet configuration. Only valid for ent networks.
-a other_net_type=Value (optional)	Specifies another network type that applies to this logical network. Relevant if bridges and routers present.
-a routing=Value (optional)	Stores NIM routing information for a network. It can consist of three values: Value 1 specifies the NIM name of the destination network for this route. Value2 specifies the host name of the gateway to use in order to communicate with the destination network. Value3 specifies the host name of the gateway used by the destination network to get back to this network; it can be used to define a default or static route.
-a verbose=Value (optional)	Displays information for debugging.

A network object in the ODM database will contain the following information:

- The user-defined name of the network object
- The network interface type, which can be token-ring, ethernet, FDDI, or ATM
- The IP address of the network
- The subnet mask of the network
- NIM routing information for the network

Routing information is required where the master is trying to communicate with clients on a different local area network. For example, in our hypothetical scenario, the master will need a route to its clients on the FDDI and token-ring network. Routes can be either static or default.

Network strategy

When considering a strategy to achieve maximum performance from our network, we need to have a good picture of our network topology in its present state. This will provide us with our starting point. If changes are made to the physical network topology, we can see what machines have been impacted, and we can update the NIM database accordingly. Obviously, it does not make sense to implement NIM in a network environment that is constantly changing.

Because the master is crucial to the NIM environment, its location is essential to the success of our NIM environment. It should possibly be placed on a low-impact high-throughput network. If that option is not open to you, you may want to consider placing some of the resources on different servers throughout the environment, thereby, spreading the master's load. If we have multiple networks in our environment, this option is also viable to prevent bottlenecks from occurring on the different LANs.

The timing of installations can also help to distribute the load. If we schedule jobs to occur when there is less traffic on the network, this will alleviate problems that can result from network overload. Another possibility is to have a client initiate the installation at a convenient time.

Name resolution is vital for the master to be able to manage the machines in its NIM environment. If these machines are not on our DNS server, we require an entry for them in the /etc/hosts file. Name resolution is also vital when mounting file systems from the resource servers. If we are having DNS server problems, these file systems will not be available to us.

Note

When first defining NIM routes, it is more intuitive to use the smitty interface.

1.2.3 Resources

NIM resource objects represent files, directories, and devices that are used in order to support some type of NIM operation. These resources are generally NFS mounted on the client machine. The available resources are listed in Table 4.

Table 4. Possible resources available to machines in the NIM environment

Resource	Description
* boot	Boot image to support the network boot of a client

Resource	Description
* nim_script	Script to support NIM operations
* SPOT	Share Product Object Tree equivalent to the /usr file system
lpp_source	Source device for optional product images
mksysb	AIX mksysb image
exclude_files	List of files to be excluded when creating an mksysb image
script	Executable file that is run on a client.
bosinst_data	Configuration file used during base system installation
image_data	Configuration file used during base system installation
installp_bundle	Installp bundle file
fix_bundle	Fix (keyword) input file for the cust or fix_query operations
resolv_conf	Configuration file for name server information
root	Parent directory for client / (root) directories
paging	Parent directory for client paging files
dump	Parent directory for client dump files
home	Parent directory for client /home directories
shared_home	/home directory shared by clients
tmp	Parent directory for client /tmp directories

To obtain information about any resource, enter the following command:

```
# lsnim -Pa ResourceType
```

Resources defined in the database contain information detailing:

- The location of resources
- The number of clients using it

• Its state

Different resources are applicable to the different machines. The resources marked with an asterisk (*) in Table 4 on page 10 are required by all machines.

1.2.3.1 Minimum resources

The minimum resources required to bring a machine to a running state depend on the type of machine. All three machine types require the following basics:

• SPOT

• boot

• nim_script

Dataless and diskless machines require root and dump resources, but only dataless machines require the paging resource. Lpp_source resource becomes a requirement when the SPOT resource is using it as its source for installation images. Another option for SPOT is an lpp_source that sources a CD-ROM or tape for installation images.

Shared Product Object Tree (SPOT) resource

This is the fundamental resource in the NIM environment. It is a directory that contains AIX code and is equivalent in content to a /usr file system. A SPOT can be a converted /usr file system or it may be a separate directory, generally, off the /export directory. This type of SPOT is referred to as a non-/usr SPOT. The images that the SPOT needs to perform an installation will be placed within this directory when a SPOT is created. If we use a converted /usr file system, the SPOT will inherit all the LPPs that are already installed on that server. For more information, refer to Section 1.4.6.2, “Defining the SPOT resource” on page 68.

A SPOT may physically reside on any client, but it is created, controlled, and maintained by the master. Spreading resources will alleviate network load and will increase the throughput of the master as it is not being hit for SPOT-copy BOS installation. During a bos_inst operation, when the `installp` command is invoked, the filesets present in the SPOT will be installed.

A SPOT will also create and maintain network boot images. These boot images reside in the /tftpboot directory of the SPOT server. tftp will get them using the /etc/tftpaccess.ctl access method.

SPOTs are used to support all NIM operations that require a machine to boot over the network. These operations are as follows:

- bos_inst
- maint_boot
- diag
- dkls_init
- dtls_init

When the SPOT is defined, the following events occur:

- The BOS image is retrieved from archive or, for /usr conversion, just the root directory is retrieved from archive.
- The device support required to support the NIM operations is installed.
- Network boot images are created in the /tftpboot directory.

To define a SPOT, we use the following command:

```
# nim -o define -t spot Attribute=Value... SpotName
```

The attribute choices are shown in Table 5.

Table 5. Possible attributes to choose in defining a SPOT resource

Attribute	Description
-a location=Value (required)	Specifies the parent directory under which the SPOT is to be created
-a comment=Value	Describes the SPOT
-a server=Value (required)	Cites the name of the machine where the SPOT is to be created
-a source=Value (required)	Identifies the source device for installation images to create and install the SPOT
-a auto_expand=Value (optional)	Expands the file system as needed when installing the SPOT
-a debug=Value (optional)	Builds debug-enabled network boot images
-a installp_flags=Value (optional)	Specifies the flags that describe how installp should install software into the SPOT
-a show_progress=Value (optional)	Shows installp output as the SPOT is installed
-a verbose=Value (optional)	Displays information for debugging

To list the software installed in a SPOT, use the following command:

```
# nim -o lslpp SpotName
```

We can tell what the default AIX packages required to perform a BOS installation are from the /usr/lpp/bos.sysmgt/nim/methods/c_sh_lib script. They include:

bos.64bit	bos.up	bos.mp
bos.net.nfs.client	bos.net.tcp.client	bos.diag
bos.sysmgt.sysbr	bos.sysmgt.smit	bos.terminfo
devices.all		

Note

You should not create a non-/usr SPOT in a subdirectory of /usr.

Network boot resource

The boot resource is an internally-managed NIM resource. It is assigned automatically to machines; you do not see it being allocated. Its purpose is to bring a machine to a specific execution state. When the machine has rebooted, it is automatically deallocated. It is located in the /tftpboot directory on each SPOT server.

Nim_script

This resource is also internally managed by the NIM master. It indicates that a script should be run by NIM as part of a NIM operation to customize the NIM environment in some way. The nim_script resource is automatically deallocated when the operation completes. For a better understanding of what the nim_script is and how it behaves, refer to Section 1.3.2.1, “Network boot process” on page 33.

1.2.3.2 Non-install lpp_source

An lpp_source resource acts as a repository for optional software packages. It is used to support the cust operation. It may also support a SPOT resource by providing it with updated software. If we need an lpp_source resource to support a BOS installation, the lpp_source will have to contain a minimum number of file images. If an lpp_source meets this requirement, it will be marked with the simages attribute. This attribute is assigned by performing a check operation on the lpp_source resource, which updates the .toc file, or, when the lpp_source is first defined, the master will inspect the lpp_source to see whether it meets the simages requirement. The following is a list of images required for an lpp_source to be awarded the simages attribute:

bos	bos.up	bos.mp
bos.diag	bos.net	bos.sysmgt
bos.terminfo	bos.terminfo.all.data	devices.base.all
devices.buc.all	devices.common.all	devices.mca.all
devices.rs6ksmp.base	devices.scsi.all	devices.sio.all
devices.sys.all	devices.tty.all	xlC.rte

The lpp_source resource is mounted on the client when performing bos_inst and cust_operations. When completed, this resource is deallocated.

You can define an lpp_source in several ways:

- If a directory containing installation images already exists, it can be directly defined as an lpp_source.
- If a directory should be created and populated by NIM with the default set of support images for a BOS install, use the source attribute when defining the resource. This attribute specifies the name of the device that contains the installation images. NIM will copy the software images from this device into the location specified for the lpp_source. The images copied will include those from the simages list, all available device support, as well as some additional software that is generally installed.
- If you require an lpp_source to be created from a source device using a list of software other than the default set of images, specify the options attribute when defining the lpp_source. Use the options attribute to list the alternative set of software images to copy.

To define an lpp_source resource, we use the following:

```
# nim -o define -t lpp_source -a Attribute=Value... lpp_sourceName
```

The attributes tabulated in Table 6 are the attributes that can be specified when defining an lpp_source resource.

Table 6. Possible attributes to choose when defining an lpp_source resource

Attribute	Description
-a location=Value (required)	Specifies the directory that will contain the installation images.

Attribute	Description
-a server=Value (required)	Establishes the name of the machine where the lpp_source is to be created.
-a comments=Value (optional)	Describes the lpp_source.
-a group=Value (optional)	Determines the name of a resource group to which this resource should be added.
-a packages=Value (optional)	Specifies a list of filesets to copy into the lpp_source if the default list of images is not desired.
-a source=Value (optional)	Identifies the source device for copying installation images when defining the lpp_source. Not required if the location of the lpp_source already contains installation images.

1.2.3.3 Mksysb resource

Instead of a SPOT or lpp_source, an mksysb resource can be used to provide the image files for a BOS installation. The mksysb resource is intended to be used to install a client machine from an original mksysb image. The image must reside on the hard disk of a machine in the NIM environment (we can ftp it from one machine to another, for example). The image may already exist, or we have the option of creating it when defining the mksysb resource.

To define an mksysb resource, enter the following:

```
# nim -o define -t mksysb -a Attribute=Value... mksysbName
```

Attributes can be chosen as shown in Table 7.

Table 7. Possible attributes to select in an mksysb definition

Attribute	Description
-a location=Value (required)	Specifies the full path name of the mksysb image
-a server=Value (required)	Identifies the machine on which the mksysb image resides
-a comments=Value (optional)	Describes the mksysb resource
-a exclude_files=Value (optional)	Details an exclude_files resource to use to exclude files and directories from the system backup

Attribute	Description
-a group=Value (optional)	Establishes the name of a resource group that should be added to an mksysb resource.
-a mk_image=Value (optional)	Specifies the flag to use to create an mksysb image from a machine in the NIM environment
-a mksysb_flags=Value (optional)	Sets the flags to use to tell the command how to create the backup
-a size_preview=Value (optional)	Specifies the flag to verify that space is available before creating an mksysb image
-a source=Value (optional)	Identifies the machine to be backed up in mksysb image. The *source* attribute for creating an mksysb image should be a machine if the mk_image attribute is set to *yes*. Although the support to replicate the mksysb image was added in AIX V4.3.3, the value of the *source* attribute may be another mksysb resource if the mk_image attribute was not specified or set to *no*.
-a verbose=Value (optional)	For debugging purposes

To limit the size of our mksysb image, we have a few options:

- Any unwanted file systems should be unmounted to prevent them from getting backed up.
- On the client being backed up, we can use an incl.excl list, which, as the name suggests, will specify the file systems to include and exclude.
- We can create an exclude_files resource.

Exclude_files resource

The advantage of using this resource is that if we are installing an identical image onto a number of clients and wish to exclude the same files or directories, this resource points to a file that lists the files and directories to exclude. It is mounted prior to initiating the mksysb creation process.

We can define an example exclude_file resource as follows:

```
# nim -o define -t exclude_files -a location=/nim/resources/exclude.files \
-a server=master exclude_user_apps
```

Image_data resource

This is another resource relevant to our mksysb resource. It provides NIM access to an image data file for BOS installation. The image.data files should not be created manually. Instead, use the `mkszfile` command to generate the image.data for the machine for which an mksysb will be created. Alternatively, extract the /image.data from an mksysb image modify and reference to the resource. The only fields that should be edited are the SHRINK and paging fields. SHRINK is useful if the disks listed in rootvg before an mksysb resource is created exactly match, in all respects including location, the disks found on the target machine during install. It is recommended to leave the ppsize blank since the BOS install will select one.

```
# nim -o define -t image_data -a location=/nim/resources/image.data \
-a server=master image_data_res
```

1.2.3.4 Non-prompted install

The less manual intervention is required, the quicker an installation will be. To this end, we will use a bosinst_data resource. This resource will point to a preformatted file that specifies how and where the BOS is to be installed on the client machine. An example of the bosinst.data file is found in the /var/adm/ras directory. You can copy this to another directory and edit it to your requirements. The top of the file contains comments on how to fill in the variables that control the BOS installation. The file is ordered into stanzas that should make it simpler to follow. The most important variable for our purposes is the PROMPT variable in the control_flow stanza, which should be changed to no for a non-prompted install. For a non-prompted install, CONSOLE value also needs to be specified. An example bosinst.data file is shown in Figure 1 on page 19.

```
# cp /var/adm/ras/bosinst.data /export/nim/resources/bosinst.data
# vi /export/nim/resources/bosinst.data
control flow:
CONSOLE=/dev/lft0
INSTALL_METHOD=overwrite
PROMPT=no
...

tgt disk data:
PVID=
CONNECTION=
LOCATION=
...

locale:
BOSINST_LANG=en_US
MESSAGES=
KEYBOARD=
...
#
```

Figure 1. Example bosinst.data file

To perform a non-prompted install using Web-Based System Manager, perform the following steps:

1. Type `wsm` at the command line to open the main Web-Based System Manager window.
2. From the NIM container, open the resources container.
3. From this, you can choose the **Add New Resource TaskGuide**.
4. Select the **Advanced** button followed by the bosinst_data resource. Fill in the appropriate fields. The task guide will create a bosinst.data file, which can be used as-is or modified.

To perform a non-prompted install using smitty, perform the following steps:

1. The assumption is that the bosinst.data file exists.
2. Enter the `smitty nim_mkres` fast path.
3. Select **bosinst_data** from the list. Supply the values for the required fields.
4. After the bosinst_data resource has been defined, when performing the installation, make sure that the *bosinst_data to use during installation* points to the bosinst_data that you have just created.

To perform a non-prompted install using the command line, enter the following:

```
# nim -o define -t bosinst_data -a location=/nim/resources/bosinst.data \
-a server=master bosinst_data_res
```

Note

- The bosinst_data resource will serve as a valuable facility for a push installation.
- If there are errors during the installation, while processing the bosinst.data file, BOS will revert to prompt mode.
- The fields pvid and connection, in the bosinst.data file, are really only necessary for SSA disks.

1.2.3.5 Script resource

The script resource points to a program that you want to run on the NIM client after the installation of the base operating system. You can use the script resource to perform any additional tasks on the client that are not normally performed by NIM. It is used in conjunction with the cust or bos_inst operations. Multiple scripts can be allocated to a client, but the order of processing is not predictable. It can be used for creating users on newly-installed machines, setting passwords, and configuring IP addresses and available printers. In addition, it will act as a check to ensure the NIM environment is set up properly so that the client machine can contact the server upon reboot.

To define an example script resource, use the command:

```
# nim -o define -t script -a location=/export/nim/resources/inst.script \
-a server=master inst_script
```

1.2.3.6 Other resources

We will briefly describe the other resources available to support the NIM environment. The resources applicable to dataless and diskless machines will be covered in Section 1.5.4.2, “Diskless/dataless client” on page 100.

installp_bundle resource

This represents the location of a bundle file that can be used with the `installp` command. The file contains a list of fileset names, each on a separate line. These additional filesets will get installed when the operation is invoked. It is used with the bos_inst and maint operations, and, once completed, NIM deallocates it. The installp_bundle can also be used with the cust operation. It is probably used most often with the cust operation. The command line syntax would read:

```
# nim -o define -t installp_bundle \
-a location=/export/nim/resources/instl.bdle \
-a server=master -a group=mac_grp1 instl_bdle
```

fix_bundle resource

This represents a file containing fixed keywords to be used by the `instfix` command, which is called by the NIM cust and fix_query operations. NIM mounts the fix_bundle resource on the client so that it can be used by the local `instfix` command. NIM automatically unmounts the resource when the operation has completed. A fix can include either a single fileset update or multiple fileset updates that are related in some way. Filesets can be identified with an Authorized Program Analysis Report (APAR) number.

```
# nim -o define fix_bundle -a location=/export/nim/resources/fix.bdle \
-a server=master fix_bdle
```

resolv_conf bundle

This resource was introduced and is available from AIX Version 4.2 onwards. It represents a configuration file to define Domain Name Services. It can be used to configure TCP/IP on the NIM client after a BOS installation. It supports the bos_inst, dkls_init and dtls_init operations. Upon successful installation and reboot, the machine will be configured to use the domain name services defined by the resource.

The following are sample entries in a resolv_conf resource file:

```
nameserver 129.35.143.253
domain test.ibm.com
```

To define a resolv_conf bundle, we use:

```
# nim -o define -t resolv_conf \
-a location=/nim/export/resources/resolv.conf \
-a server=master resolv_conf1
```

1.2.3.7 Resource strategy

In a simplified environment, all resources would reside on the master machine. In certain cases, this is not feasible. In these circumstances, we will choose to distribute resources throughout our environment. Points to consider would be:

- How much disk space is available on the master or clients in our NIM environment?
- Do we have a requirement to create an AIX 4.3 SPOT in an AIX 4.2 environment? Spot creation is not supported when the level of AIX installed in the SPOT is higher than the level of AIX running on the server.
- Would doing so balance the network load? This way, we will avoid bottlenecks that may arise on gateway machines between the master on one subnet and clients on another.

- Reduce the redundancy time if servers have to be taken off-line for any period of time.
- Would grouping resources be advantageous?

1.2.4 Groups

Group object types are used to represent sets of resources or machines. The use of groups allows the same NIM task to be performed on more than one machine in a single operation. This is useful for repetitive administrative tasks.

1.2.4.1 Group strategy

How many machines are there in a group? There is no limit to the number of machines that can be added to any one group. You will, however, be restricted by how labor-intensive the operation is.

A further restriction is the throughput of the network. Certain operations may lead to bottlenecks on the network.

The performance and platform type of our servers will need to be considered.

Another consideration is NFS. The maximum number of hosts to which a file or directory may be exported is limited by NFS to 256.

1.2.4.2 Machine group

A machine group is composed of a number of machines that are of the same type: Stand-alone, dataless, or diskless. The first machine to join a group sets the machine type of that group. Any NIM operation that can be performed on a given type of machine can also be performed on a group of machines of that type. A machine can be a member of several groups. It is possible to exclude group members when performing operations on that group. The machine is simply excluded; so, when NIM is performing an operation on the group, it simply skips over that machine.

Defining a machine group

The following are the steps necessary to define machine groups under different conditions.

To define a machine group using Web-Based System Manager (available from AIX Version 4.3.3), perform the following steps:

1. Open the NIM Container.
2. Select **NIM --> New Group**.

3. Assign a name to the group, the type of member machines, and available machines that you want to add to this group.

To define a machine group using smitty, perform the following steps:

1. Enter the smitty `nim_mkgrp` fast path.
2. Select the type of group you want to add.
3. Enter the name of the group, and identify the members.

To define a machine group using the command line, enter:

```
# nim -o define -t mac_grp -a add_member=MemberName GroupName
```

For example, to create a machine group named MacGrp1 containing the previously-defined machines, Standalone1, Standalone2, and Standalone3, enter:

```
# nim -o define -t mac_grp -a add_member=Standalone1 \
-a add_member=Standalone2 -a add_member=Standalone3 \
-a comments="Machines for finance dept." MacGrp1
```

Adding members to a group

The following are the steps necessary to add members to a group under different conditions.

To add members to a group using Web-Based System Manager, perform the following steps:

1. Open the NIM container.
2. Highlight the group to which you want to add other machines.
3. Right-click to bring up the property dialog box.
4. Select **Administration --> Add Members to Group**.
5. In the next screen, select the machines to be added.

To add members to a group using smitty, perform the following steps:

1. Enter the `smitty nim_chgrp` fast path.
2. Select the machine group to modify
3. On the next screen using LIST select the machines that you want to add.

To add members to a group using the command line, enter:

```
# nim -o change -a add_member=MachineName GroupName
```

For example, to add the diskless client, diskless5, to the machine group, diskless_grp, run the command as follows:

```
# nim -o change -a add_member=diskless5 diskless_grp
```

Removing members from a group

The following are the steps necessary to remove members from a group under different conditions.

To remove members from a group using Web-Based System Manager, perform the following steps:

1. From the WSM nim dialog box, highlight the group from which you wish to remove a machine.
2. From the menu bar, choose **Selected --> properties**.
3. The general properties are displayed. Remove machines by transferring them from the “NIM Machines in Group” to “NIM Machines available for group”.

To remove members from a group using smitty, perform the following steps:

1. Use the `smitty nim_chgrp` fast path.
2. Select the machine group to modify.
3. Using LIST, remove machines from the group.

To remove members from a group using the command line, enter:

```
# nim -o change -a rm_member=MachineName GroupName
```

For example, to remove the diskless client, diskless5, from the machine group, diskless_grp, run the command as follows:

```
# nim -o change -a rm_member=diskless5 diskless_grp
```

include or exclude a member machine from operations

The following are the steps necessary to include or exclude a member machine from operations under different conditions.

To include or exclude a member machine from operations using Web-Based System Manager, perform the following steps:

1. In the NIM container, `wsm nim`, highlight the group from which you wish to include or exclude a machine.
2. From the menu bar, choose **Selected --> Properties**.
3. Click on the **Include/Exclude** tab.

4. Use the transfer container to include and exclude members from the operation being performed on the group.

To include or exclude a member machine from operations using smitty, perform the following steps:

1. Enter the `smitty nim_grp_select` fast path.
2. Select the name of the group from which you want to include or exclude members.
3. Select the members to include or exclude.

To include or exclude a member machine from operations using the command line, enter:

```
# nim -o select -a include_all=Value -a exclude_all=Value \
-a include=MemberName -a exclude=MemberName GroupName
```

As an example, to exclude the machine, Standalone2, from further operations on the machine group, MacGrp1, and to include a previously excluded machine, Standalone3, enter:

```
# nim -o select -a exclude=Standalone2 -a include=Standalone3 MacGrp1
```

The attributes, include_all and exclude_all, when assigned a value of yes, can be used to include or exclude all members in a group.

To display information about the status of group members, we use the command:

```
# lsnim -g GroupName
```

Operations that are performed on members of a machine group are done randomly and asynchronously. NIM does not wait for an operation to complete on one member before starting it on another. If you want to turn this option off, use the async=no attribute when running the `nim` command.

1.2.4.3 Resource group

Resource groups allow us to allocate multiple resources as one unit at one time. Resource groups can only have one of each type of resource, except for script and installp_bundle resources, which may occur multiple times within a resource group. When a resource group is allocated for a NIM operation, all resources within that group that are not currently allocated will be allocated. Resource groups and their definition and maintenance are currently not supported by Web-Based System Manager.

To define a resource group using smitty, we proceed as follows:

1. Enter the `smitty nim_mkgrp_resource` fast path.
2. Enter the name of the group with member information.

The command line syntax for defining a resource group is:

```
# nim -o define -t res_group -a default=Value \
-a ResourceType=ResourceName... ResourceGroupName
```

By specifying default=yes, the resource group will be automatically allocated for all operations that require resources.

To allocate a resource group using smitty:

1. Use the `smitty nim_alloc` fast path.
2. Select the machine or machine group from the list of defined machines.
3. A list of resource groups is displayed. Select the resource group you want to allocate.

Using the command line, enter:

```
# nim -o allocate -a group=ResGroupName MachineName
```

Individual allocation of resources to members of the resource group will override the group resource attributes.

1.3 Network boot

The main feature of NIM revolves around being able to install clients over the network. To do this, the relevant client must be able to boot over the network. If it can do this, the master can contact it and send its relevant network boot image. The boot image resides on the SPOT server and is built from the SPOT, but it exists in the /tftpboot directory on the SPOT server. It does not get NFS mounted but rather gets TFTPd to the client by use of the BOOTP process when the client attempts to boot using a network device, thereby, allowing the client to assume the network boot image as its own and then boot. This is the lifeline of dataless and diskless machines because they will depend upon it for all booting activity.

1.3.1 Booting a machine over the network

It is the platform and kernel type of a client that determine the procedure required to boot the machine over the network. To determine the platform of a running machine, use the `bootinfo -p` command. For AIX Version 4.1 and earlier, the `bootinfo -T` command is used instead. To determine the kernel type of a running machine, use the `bootinfo -z` command. If you are using an rs6k machine with an up kernel, use Method A. If you are booting an rs6k machine with an mp kernel, use Method B. For models of rspc machines, you may use Method C. For all other platform and kernel types, follow the procedures in your hardware documentation to perform the network boot.

Method A (booting an rs6k uniprocessor machine)

1. Begin with your machine powered off.
2. If your client requires IPL-ROM emulation, insert the media into the appropriate drive of the client, and turn on the machine with the hardware key in the Service position. When the bootp menus display, continue with step 3.

   If your client does not require emulation, turn the key to the Secure position and turn on the machine. Note the LEDs on the front of the machine. They will eventually stop changing and display `200`. When this happens, turn the key to the Service position and quickly press the yellow **Reset** button. When the **bootp** menus display, continue with step 3.
3. From the **bootp** main menu, choose the **Select BOOT (Start-up) Device** option.
4. In the next menu that appears, select the boot device.
5. Select the network adapter to be used. Choose the adapter with the correct network type (ethernet, token-ring, and so on) and adapter characteristics (thick cable, twisted pair for ethernet, 4 MB and 16 MB data rates for token-ring, and so on).
6. Set or change the network addresses. Specify the IP addresses of:

- The client machine you are booting.
- Your SPOT server in the bootp server address field.
- Your client’s gateway in the gateway address field.
- The subnet mask value getting set in the IPL_ROM.

After you determine the addresses and save the addresses, return to the main menu.

Note

You do not need to type the ‘.‘ characters in the IP addresses, but you must specify any leading ‘0’ characters that make up parts of the addresses.

7. From the main menu, select the **Send Test Transmission (PING)** option.

8. Verify that the displayed addresses are the same as the addresses you specified for your boot device.

   If the addresses are incorrect, return to the main menu. Then, go back to step 3.

   If the addresses are correct, select the **Start the ping test** option.

   If the ping test fails, verify that the addresses are correct, and perform network problem determination if necessary. If the ping test completes successfully, return to the main menu.

9. From the main menu, select the **Exit Main Menu and Start System (BOOT)** option.

10. Turn the hardware key to the **Normal** position, and press **Enter** to boot your client over the network.

Method B (booting an rs6k multiprocessor machine)

1. Begin with the machine switched off.
2. Turn the key mode switch to the **Secure** position.
3. Turn the power switch on the system unit to the **On** position.
4. When the LED displays 200, turn the key mode switch to the **Service** position.
5. Press the **Reset** button once.
6. When the SMS menu appears, select the **System Boot** option.
7. Select the **Boot from Network** option from the sub-menu.
8. Choose the **Select BOOT (Start-up) Device** option.
9. Select the network adapter from which the machine will boot. If there are multiple network adapters displayed, press the **Enter** key to view the other entries. Type a number from the list and press the **Enter** key.
10. If a network adapter is selected, the Set or Change Network Addresses screen is displayed next. The hardware address for the network adapter is displayed in the hardware address field. Record the hardware address for defining the NIM machine object.

    If you want to attempt the *broadcast* style install, leave the IP address fields as zeros for the bootp request over the LAN. If there are multiple bootp servers on the LAN or the client is on a different network than the server, enter the client and server IP addresses. Type in the IP addresses using leading zeros to pad the network address fields, for example, 009.166.133.004. If this machine must use a gateway to reach the server, enter the IP address for the gateway. Save the address information and return to the main menu.

11. Select the **Sent Test Transmission (PING)** option on the main menu to test the network connection between the client and the server systems. Press the **Enter** key to start the ping test. If the ping test was not successful, check that the IP addresses are correct and that the physical network connections are sound. If the ping test was successful, return to the main menu.
12. Select the **Exit Main Menu and Start System (BOOT)** option.
13. Follow the instructions on the screen to turn the key mode switch to the **Normal** position and press the **Enter** key.

The bootp request will be issued, followed by a TFTP transfer of the network boot image.

Method C (booting a rspc machine)

1. Begin with your machine powered off.
2. Bring the machine up to System Management Services using the SMS diskette, or, once the graphic images start appearing on the screen, press the **F1** key.

Note

- For ASCII terminals, press the **F4** key as words representing the icons appear. The relevant function key will depend on the type and model of rspc machine; so, refer to your User Guide.
- If the last icon or keyword is displayed prior to pressing the **F4** or **F1** key, the normal mode boot list is used instead of the Systems Management Services diskette.
- For later models of rspc, the functionality of the SMS diskette is incorporated into the firmware, which is accessed by pressing the **F1** key.

3. The System Management Services menu is displayed. Select the **Utilities** option.
4. From the Utilities menu, select the **Remote Initial Program Load Setup** option.
5. From the Network Parameters screen, select the **IP parameters** option.
6. Set or change the values displayed so they are correct for your client system.
7. Specify the IP address of:

- The client machine you are booting in the client address field.
- Your SPOT server in the bootp server address field.
- Your client’s gateway in the gateway address field.

Note

You do not need to type any leading `0` characters that make up parts of the IP addresses, but you must specify ‘.’ characters in the IP addresses.

8. Specify the subnet mask for your client machine if you are prompted for one in the subnet mask field.
9. After you determine the addresses, press **Enter** to save the addresses and continue.
10. The Network Parameters screen is displayed. Select the **Ping** option.
11. Select the network adapter to be used as the client’s boot device and verify that the displayed addresses are the same as the addresses you specified for your boot device. If the addresses are incorrect, press **Esc** until you return to the main menu. Then, go back to Step 5.
12. If the addresses are correct, press **Enter** to perform the ping test. The ping test may take several seconds to complete.
13. If the ping test fails, verify that the addresses are correct, and perform network problem determination if required. If the ping test completes successfully, you will see a success sign and will be returned to the SMS menu.
14. From the Systems management services menu, choose the **Select Boot Devices** option.
15. Select the network adapter to be used for the network boot list from the list of displayed bootable devices. Be sure to select the correct network type and adapter characteristics. Once you are happy with the devices listed in

the boot list, exit from SMS and continue the boot process. Sometimes, you may find it better to power the machine off and then back on again.

Note

- When performing a BOS installation on a NIM client with an rspc platform, the machine may fail to boot if network traffic is heavy.
- If the network boot was initiated from the NIM Master, the machine will eventually boot from the disk. If the network boot was initiated from the SMS menus on the NIM client, the machine will return control to the SMS menus.
- For multiple interfaces, select the interface that has been specified in the NIM client definition so that NIM master can allocate the correct boot image.

1.3.2 Overall process

The boot sequence shown in Figure 2, also referred to as the Initial Program Load (IPL), is, in theory, similar on all computer systems.

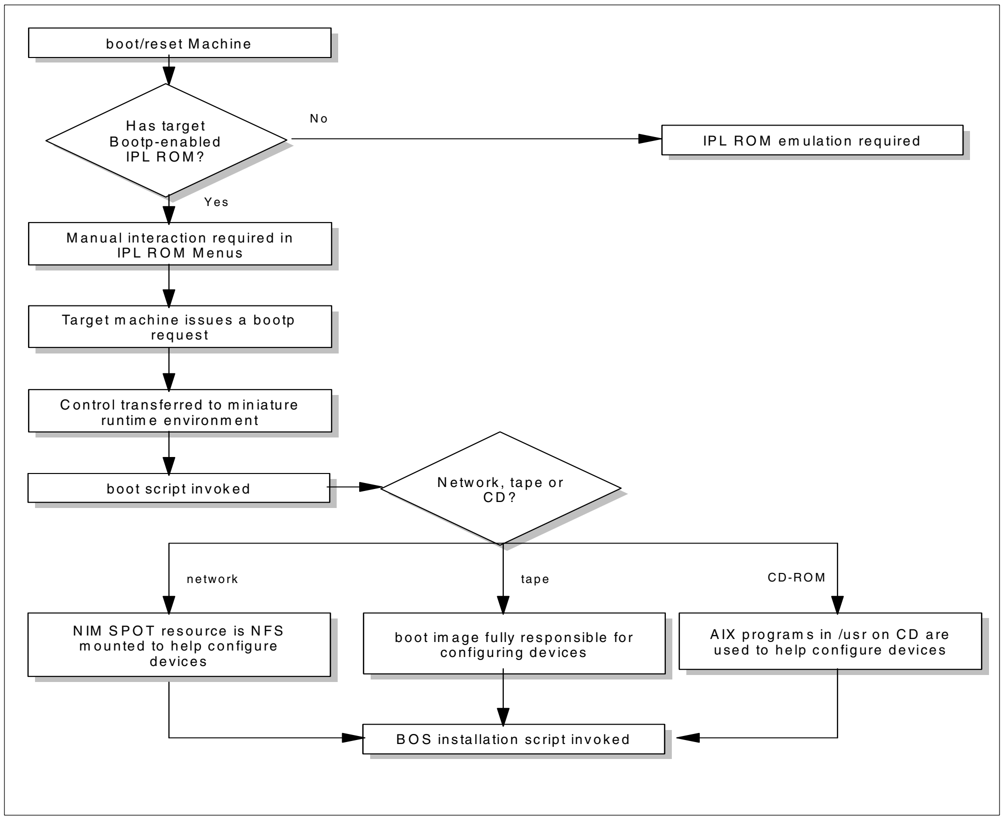

boot/reset Machine
Has target Bootp-enabled IPL ROM?
No
IPL ROM emulation required
Yes
Manual interaction required in IPL ROM Menus
Target machine issues a bootp request
Control transferred to miniature runtime environment
boot script invoked
Network, tape or CD?
network
tape
CD-ROM
NIM SPOT resource is NFS mounted to help configure devices
boot image fully responsible for configuring devices
AIX programs in /usr on CD are used to help configure devices
BOS installation script invoked

Figure 2. Boot sequence

The basic steps carried out by the hardware are:

1. Checks the power, CPU, and memory systems
2. Looks for the operating system image
3. Loads and runs the operating system image

It is this final stage that concerns us. As you are aware, computer systems can boot from a variety of devices: Tape, CD-ROM, hard disk, and the network. Refer to Figure 2. The deciding factor is held by the Non-Volatile Random Access Memory (NVRAM). It decides how the computer system is going to boot from its list. The IPL Read-Only Memory (ROM) locates and loads the boot image. When this image is read, AIX starts to load. This is indicated when the `software starting please wait` message appears on the screen. It may happen that the boot list becomes corrupt, is deleted, or, for our purposes, the order of bootup needs to be changed. After the boot image is loaded into memory, the IPL-ROM’s role is complete.

To obtain the network boot image, clients must contact the server in the NIM environment using the BOOTP protocol. These boot images are created and maintained by the SPOT resource and are stored in the /tftpboot directory on our SPOT server. Each network boot image supports a specific network type, a specific hardware platform, and a processor type. Each image will take up about 4 MB in the file system. An image is created using a permutation of the above; the syntax is as follows:

<*SpotName*>.<*PlatformType*>.<*KernelType*>.<*NetworkType*>

The image is not machine-specific; so, it should not contain any data pertinent to a specific client. Machine-specific data is held in a configuration file, known as the /tftpboot/<clienthostname>.info file.

1.3.2.1 Network boot process

The network boot process proceeds as follows:

1. The client boots to IPL-ROM/firmware menus
2. The client sends a bootp request to the server from the IPL-ROM/firmware menus.
3. The server only matches it against a MAC address during broadcast installs. If it is not a broadcast install, it matches against a client IP address.
4. The boot image contains a miniature runtime environment (kernel, file systems, libraries, and key programs) to which control is then transferred.
5. Control is eventually passed to a boot script, rc.boot, which configures devices that will be needed for the installation.
6. The client then retrieves the relevant /tftpboot/<clienthostname>.info file with tftp. It copies it to its /etc/niminfo. This file contains a set of environment variables that will define the BOS installation environment.
7. Finally, the BOS installation program is invoked by the boot script to install AIX from images residing on the CD-ROM, tape, or network.

The miniature runtime environment contained within the boot image, which is actually the executable /usr/lib/boot/ssh disguised as init, invokes the rc.boot for different phases of network boot processing.

1.3.2.2 rc.boot

What is important here is the order in which rc.boot can call the network boot scripts. This will depend on the reasoning behind a network boot. The outcome is different if we are running diagnostics or doing a maintenance boot or installation.

For simplicity, there are two phases to the rc.boot script: Phase 1 and Phase 2. During Phase 1, the rc.boot script will invoke the following:

1. The cfgmgr is invoked in a Phase 1 state by specifying the -f flag. In this phase, the cfgmgr will configure the base devices.
2. The ifconfig defines the network address of each interface present on a machine.
3. Using tftp, it transfers /tftpboot/<clienthostname>.info from the SPOT server.
4. It creates an /etc/hosts file based on information it has in its /tftpboot/<clienthostname>.info file, and, using this file, it will also be able to define and add routes ($NIM_HOSTS and $ROUTES).

   It also mounts the SPOT from data contained in the /tftpboot/<clienthostname>.info file.
5. It then invokes Phase 1 of the script referenced in $RC_CONFIG, again defined by the /tftpboot/<clienthostname>.info file.

It is at this point that the reasoning behind our network boot will determine our next step. If you perform an installation, $RC_CONFIG in the /etc/niminfo file takes on the value of rc.bos_inst. In this case, the script /usr/lib/boot/network/rc.bos_inst is copied from the SPOT into the local /etc directory. If you perform a network boot for diagnostic reasons, $RC_CONFIG assumes the value of rc.diag and the file /usr/lib/boot/network/rc.diag is copied into the /etc directory. The final value that $RC_CONFIG can preempt is rc.dd_boot. This is relevant to dataless and diskless machines and is what allows them to boot over the network.

We will concentrate on following an installation boot. Refer to Figure 3 on page 36. Phase 1 will invoke the rc.bosinst script. This script is not part of the boot image; it is executed from the mounted SPOT, which occurs during phase 1 of rc.boot. This script can also be broken into two phases: Phase 1 and Phase 2.

1. During Phase 1, some file systems and files are mounted from the installation server, as defined by the $NIM_MOUNTS environment variable in the /etc/niminfo file. It will also invoke the cfgmgr to configure other needed devices.

Once phase 1 is completed, control is returned to the rc.boot script.

The rc.boot script will then, under Phase 2, reexport the variables in the /tftpboot/<clienthostname>.info file and invoke Phase 2 of the script referenced in $RC_CONFIG. For our purposes, we will only trace the actions of the bos_inst script.

2. During the second phase, rc.bos_inst will call the /usr/lpp/bosinst/bi_main script. It is this script which triggers the installation process. At this stage, the `nimclient -R success` is run which tells the master to update the C_state of the client to running bos_inst.

1.3.2.3 bi_main script

The bi_main script procedure is executed directly from the mounted SPOT. It is shown in Figure 3 on page 36 and proceeds as follows:

1. It determines whether the bosinst_data resource exists. If it does not exist, bi_main will prompt the user for the necessary information.
2. It installs the BOS image.
3. It installs support for detected devices, system bundles, and any bundle defined in BUNDLES variable of the bosinst.data file.
4. It invokes the $NIM_BOSINST_RECOVER script to recover some information that may have been lost during the installation.

   This variable is defined in the /tftpboot/<clienthostname>.info file. Refer to Appendix , “NIM_BOSINST_RECOVER” on page 289, for this script.

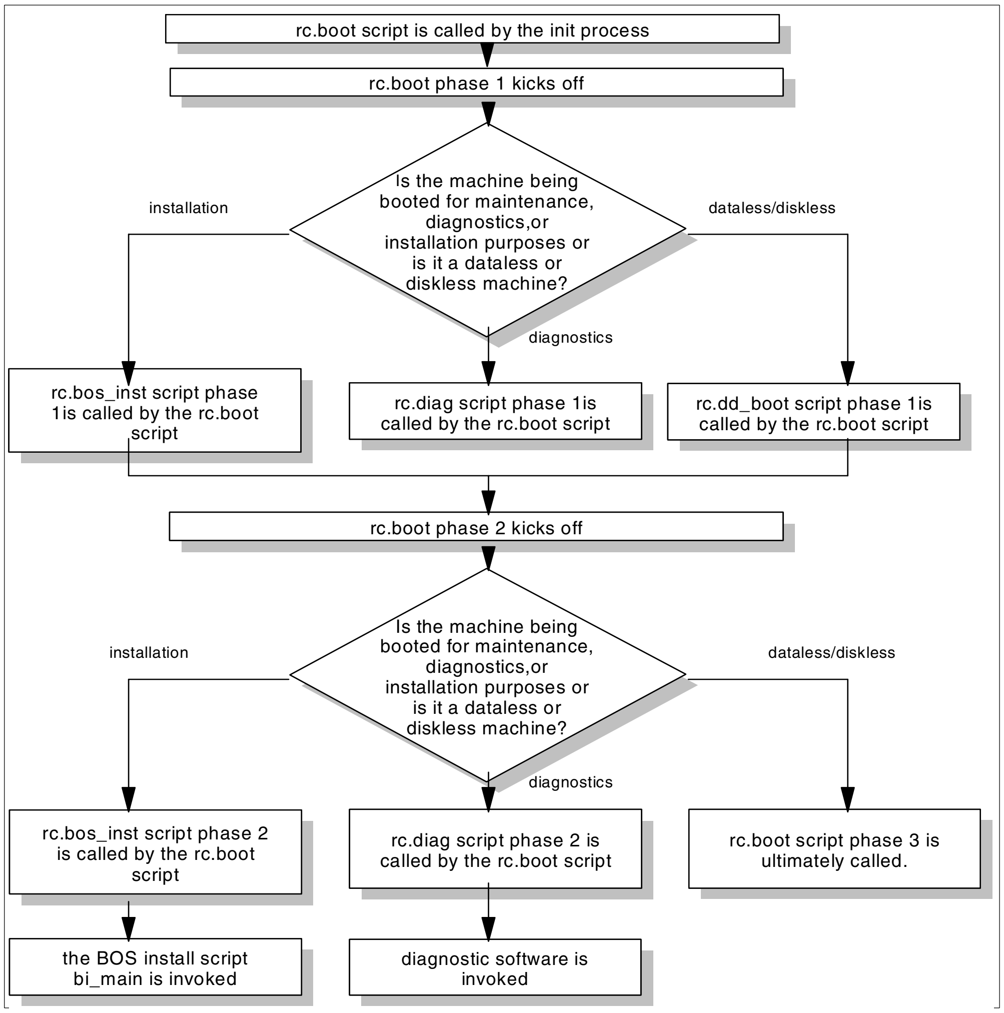

rc.boot script is called by the init process
rc.boot phase 1 kicks off
Is the machine being booted for maintenance, diagnostics,or installation purposes or is it a dataless or diskless machine?
installation
dataless/diskless
diagnostics
rc.bos_inst script phase 1is called by the rc.boot script
rc.diag script phase 1is called by the rc.boot script
rc.dd_boot script phase 1is called by the rc.boot script
rc.boot phase 2 kicks off
Is the machine being booted for maintenance, diagnostics,or installation purposes or is it a dataless or diskless machine?
installation
dataless/diskless
diagnostics
rc.bos_inst script phase 2 is called by the rc.boot script
rc.diag script phase 2 is called by the rc.boot script
rc.boot script phase 3 is ultimately called.
the BOS install script bi_main is invoked
diagnostic software is invoked

Figure 3. Network boot flow diagram

5. It invokes NIM customization scripts as defined by the $NIM_CUSTOM variable in the /tftpboot/<clienthostname>.info file. It is set to /SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script. Refer to Appendix , “NIM customization scripts” on page 289, for more details on this script, its purpose, and what it does.

6. It invokes the `bosboot` command.

   The `bosboot` command will create the device boot image that interfaces with the machine boot Read-Only Storage (ROS) Erasable Programmable Read-Only Memory (EPROM). It uses the file system mounted and the kernel to create the boot file image. When the machine is powered on or rebooted, the ROS loads the boot image into memory. Control is then transferred to the loaded image's kernel. It does not, however, update the list of boot devices in the NVRAM. This can be modified with the `bootlist` command. To run this command, you will need some space in the /tmp directory. To create a boot image file for an ethernet boot, enter:

```
# bosboot -ad /dev/ent0 -M both
```

To create a token-ring boot image for a machine whose hardware platform type is rspc while you are running on a machine whose hardware platform is an rs6k, enter:

```
# bosboot -ad /dev/tok0 -T rspc
```

7. Invoke the CUSTOMIZATION script if there is an entry for CUSTOMIZATION_FILE in the bosinst.data file and the bosinst.data resource has been allocated.
8. Sets the order of the boot list to boot off the hard disk at the next reboot.

   To do so it will invoke the `bootlist` command. This command is used to alter the list of boot devices. When a system is powered on or rebooted, the `bootlist` command will scan the devices in the list and attempt to boot from the first device it finds containing a boot image. If no boot file is detected on the first device, the system will move on to the next device and so on until the list is exhausted. To display the current bootlist on a machine, use the following:

   ```
   # bootlist -m Mode -o
   ```

   where `Mode` can be service, normal, both (service and normal), and previous boot. It also depends on the model and type of the machine. If a device specified in the boot list is no longer available a '-' is specified instead of a name. It is also possible to alter the bootlist of a remote client using the following example for explanatory purposes:

   ```
   # bootlist -m normal tok0 gateway=9.3.187.yyy bserver=9.3.187.yyy \
   client=9.3.240.yyy
   ```

9. It reboots the machine.

   If the reboot was for the purpose of a maintenance boot, bi_main takes on a value of maint for BOS_FORMAT, and, when this occurs, the user is prompted on the console for further action. If the reboot was for diagnostic purposes, rc.diag Phase 2 will invoke the diagnostics software. On a dataless or diskless machine, control is returned to rc.boot, which will ultimately be invoked for Phase 3 to finish booting the machine to a running state.

   This can be represented graphically as shown in Figure 4 on page 38:

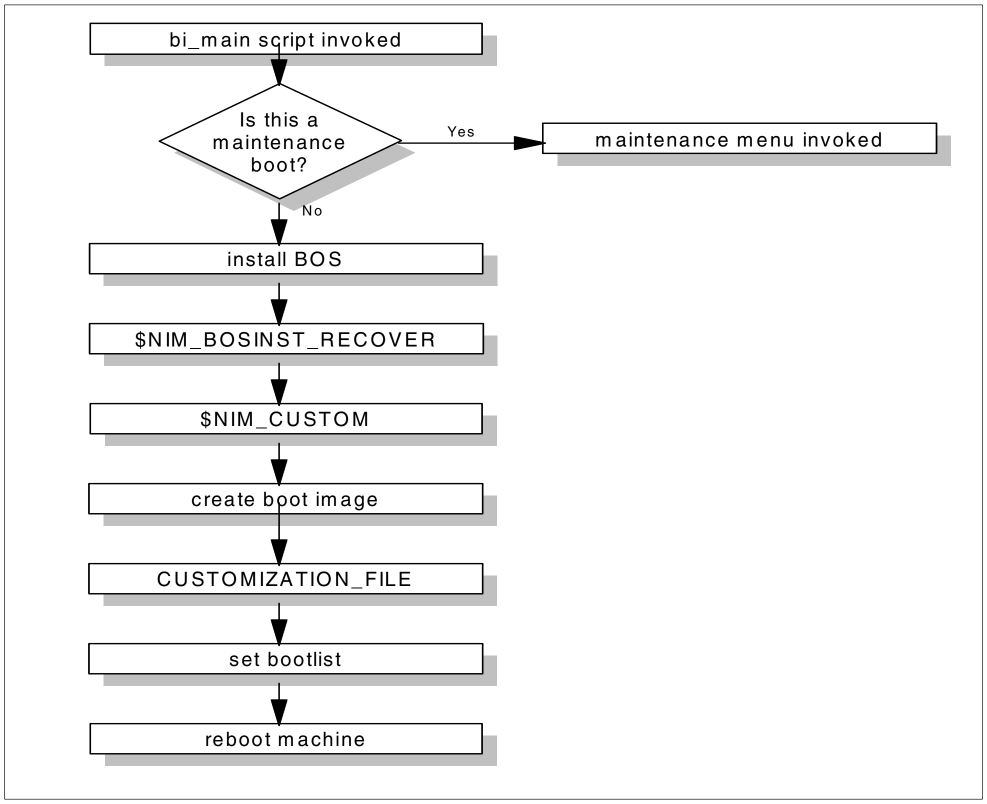


Figure 4. bi_main flow diagram

1.3.3 IPL-ROM

All versions of IPL-ROM can search local devices for an AIX boot image. However, only versions known as BOOTP-enabled IPL-ROM can use a network interface to search for a remote boot image. If it finds a network boot image using bootp, it retrieves it using tftp. Generally speaking, all machines manufactured before June 1993 do not have BOOTP-enabled IPL-ROM (except for models 220, 230, 340, and 350).

To determine if a running machine requires IPL-ROM emulation, run the command:

```
# bootinfo -q <network adapter name>
```

If the response is a 1, the adapter is network-boot enabled, and no IPL-ROM emulation is required; otherwise, IPL-ROM intervention is required. If the machine is not running, it is possible to determine if emulation is required by booting the machine with the key turned to **Secure**. If the LEDs eventually stop at 200, turn the key to **Service** and quickly push the **Reset** button. If the LEDs alternate persistently between 260, 261, and 262, press **Enter**. IPL-ROM intervention is not needed. If the LEDs do not display 260, 261, or 262, the system must use IPL-ROM emulation. The IPL-ROM emulation program enables the machine to search for a remote boot image. This program is available in the bos.sysmgt.nim.master fileset and, consequently, must be created on the NIM master.

If your NIM environment has an FDDI network that you need to access, the machines that connect to the FDDI network must use IPL-ROM emulation.

To create the IPL-ROM emulation media using the command line, perform the following steps:

1. Insert a formatted diskette or tape into the appropriate drive on the NIM master.
2. Type the following:

   ```
   # bosboot -T rs6k \
   -r /usr/lpp/bos.sysmgt/nim/methods/IPL-ROM.emulation \
   -d DeviceName -M both
   ```

   where DeviceName can be fd0, /dev/fd0, rmt0, or /dev/rmt0. This operation requires that the devices.base.rte fileset be installed on the machine upon which the emulation is being created.
3. Insert the IPL-ROM emulation media in the appropriate drive on the target machine.

To create the IPL-ROM emulation media using Web-Based System Manager, perform the following steps:

1. From the NIM container in the NIM menu, select **Create IPL-ROM Emulation Media**.
2. Use the dialog to complete the task. All fields are required.

To create the IPL-ROM emulation media using smitty, perform the following steps:

1. Enter the `smitty IPL-ROM` fast path.
2. Select the target device on which the emulation will be placed, - fd0 or rmt0, for instance. The path name of the IPL-ROM emulation image will default to /usr/lpp/bos.sysmgt/nim/methods/IPL-ROM.emulation. We also take the default value for the boot mode. You have a choice of normal, service or both.

1.4 Setup

This section describes the procedure for the installation of NIM on the client and server machines.

1.4.1 Planning

The following issues should be considered when planning for the installation of NIM in your environment:

- What form of installation is suited to our needs?

  With NIM, we explore the different ways of installing the base operating system. The options available include overwrite, preservation, and migration.

  Overwrite

  Installing an operating system in overwrite mode will, basically, overwrite whatever version of AIX is on the machine eliminating all user and group definitions and any existing directories. We also choose this option if it is a first time installation of the operating system.

  Migration

  On the other hand, a migration installation will preserve the /usr, /var, and /(root) file systems and will save user and group definitions, configuration files, and logical volumes. It will remove files in the /tmp directory.

  Preservation

  Preservation installation differs from migration installation in that the contents of /, /usr, /var, and /tmp will be deleted, but it will retain the previous paging space, /home, and other user-created file systems in rootvg.

- Is the user familiar with their network topology?
  - Is it well administered?
  - Is it frequently changing?

    If this is the case, we will constantly be updating the database to reflect these changes, and that is an ongoing exercise that you can do without.
  - Can you pinpoint routers, gateways, and subnets within your environment? Is it just one type of network or is your environment a combination of a few types?
- What type of machines will we be using?

  - Are they all based on a similar hardware platform? If not, we will have more considerations in developing our NIM environment.
  - What level of AIX are they running?
  - Do they have multiple network adapters, and do we need IPL-ROM emulation to boot them?
- Can we pinpoint a machine to act as our master machine?

  The master should have sufficient disk space if we intend to store resources locally on it. It should be efficient; the greater the number of processors and the more memory, the better.
- How many operations do we intend to process at a time?
  - The greatest determining factor is the location of the resources to support these.
  - We must also consider how well the master can handle simultaneous requests to it.
  - If our network is composed of many subnetworks, the network is going to be greatly impacted by an operation, such as installing a number of machines at once.
- Security
  - Is our master located in a secure area?
  - If we are considering global exports of our resources, everyone on the network will have read access to them.
- NFS issues

  NFS makes remote objects stored in a file system appear to be local, as if they reside on the local host. This is referred to as *transparent* access to remote files. To be able to perform a mount, clients will need read authority to the file system being mounted and write authority to the directory being mounted over.
  - You will require an exports file. This is a stream file that is used to store information about file systems or directories to be exported. For each directory to be exported, you will need an entry in the /etc/exports file.
  - If you are experiencing DNS problems or if the DNS server is slow or down, you will be unable to perform the mounts.
  - The more memory on the NFS server the better, because this will increase the ability to cache the files that are being read from the server during the install process. The data is in memory and is much quicker to access. This will increase the throughput of the server since potential disk operations have been alleviated.

- If you have one server doing all the installs, with only one interface in use, it will create a bottleneck.

One thing to remember in planning for our installation is that expectations should be set based on available resources. All of the above points should be depicted in a diagram. This will give us a better understanding of our environment and it's requirements. We cannot emphasize enough how much work a good diagram will save you.

1.4.2 Master setup

Configuring the single master per environment will create the limited basic installation resources required for NIM client installation and allow us to manage resources for dataless and diskless machines.

Prerequisites

The NIM master has a working AIX operating system installed on it: For our purposes, it is Version 4.3. The NIM master must have at least 750 MB of available disk space. If such space is not available, options open to us include:

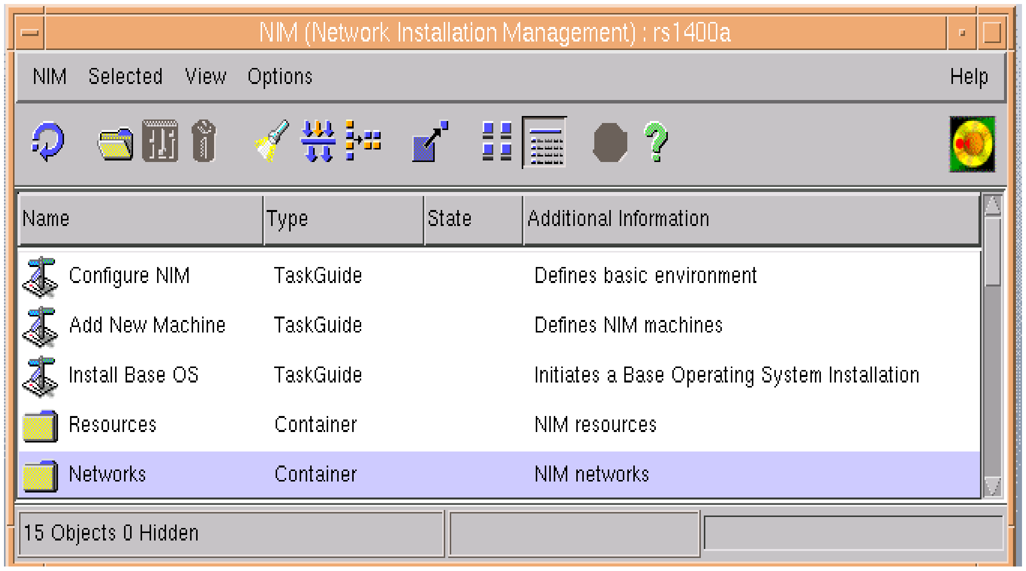
- Using client machines as resource servers.
- Defining /usr versus non-/usr SPOT's.
- Defining an lpp_source on CD-ROM versus disk.

There are three ways to perform the master setup: Using Web-Based System Manager, smitty, or the command line.

To perform the master setup using Web-Based System Manager:

1. Insert the AIX Version 4.3 CD-ROM or tape into the appropriate drive of the designated master machine.
2. Start the Web-Based System Manager software application by entering the `wsm software` fast path.
3. From the Software menu, select **New Software (Install/Update) --> Install Additional Software (Custom)**.
4. In the Install Software dialog, select **/dev/cd0** or **/dev/rmt0** as the software source.
5. Specify **bos.sysmgt.nim.master** as the software to install.
6. Additional options are available by clicking **Advanced**. Exit the Software application.

7. Start the Web-Based System Manager NIM application by entering the `wsm nim` fast path at the prompt as shown in Figure 5.
8. In the NIM container, double-click the **Configure NIM TaskGuide**.
9. Follow the TaskGuide instructions to guide you through the configuration.



Name	Type	State	Additional Information
Configure NIM	TaskGuide		Defines basic environment
Add New Machine	TaskGuide		Defines NIM machines
Install Base OS	TaskGuide		Initiates a Base Operating System Installation
Resources	Container		NIM resources
Networks	Container		NIM networks

15 Objects 0 Hidden

Figure 5. wsm nim fast path

To perform the master setup using smitty:

1. Insert the AIX Version 4.3 CD-ROM or tape into the appropriate drive of the designated master machine.
2. Install the software using the smitty `install_latest` fast path.
3. To install the bos.sysmgt.nim fileset, enter the `smit install_latest` fast path.
4. Using the LIST option, select **/dev/cd0 or /dev/rmt0** for the input device/ directory for software.
5. Specify **bos.sysmgt.nim** as the software to install.
6. Accept the default values for all other fields on this screen. After successful completion of this installation, exit smitty.
7. To configure the NIM master, enter the `smitty nim_config_env` fast path.
8. Using the **LIST** option, select the **Primary Network Interface** for the NIM master.

9. Using the LIST option, select **/dev/cd0 or /dev/rmt0** for the Input device for installation images field.
10. Specify the **lpp_source** resource; it is advisable to use relevant names, for instance, **lpp_source411** or **lpp_source433** and the location of the lpp_source.
11. Do the same for the SPOT resource.
12. If you will be supporting diskless and dataless clients, select **yes** in the Create Diskless/Dataless Machine Resources? field, and supply the names for the resources to be created. Resources to be created include root, dump, paging, home, shared_home, and tmp.
13. Defining system bundles relates to installp_bundle resources, which are created for the installation of additional software.
14. For complex environments to speed up client definition (if the system to be installed is identical from system to system), you can create a client definition file that will install the clients based on stanzas defined in this file.
15. Select **yes** at the Remove all newly added NIM definitions and file systems if any part of this operation fails? field. This will make it easier to restart this procedure if failures occur.
16. If the NIM environment is basic, you can simply accept the default values.

Note

- Depending on the speed of your machine, this procedure could take some time.
- As you develop a more in-depth understanding of configuration tasks, you may prefer to not automatically undo all configuration when failures occur (as in step 10 above). Continuing from the last point of failure will speed up the installation process.

To perform the master setup using the command line:

1. Insert the AIX Version 4.3 CD-ROM into the CD-ROM drive of the designated master machine, or insert the tape into the tape drive of the designated master machine.
2. If installing from a tape, go to step 5.

   To create a mount point for the CD-ROM, enter:

   ```
   # mkdir /cdfs
   ```
3. To create a CD-ROM file system, enter:

   ```
   # crfs -v cdrfs -p ro -d'cd0' -m'/cdfs'
   ```

4. To mount the CD-ROM, enter:

```
# mount /cdfs
```

5. To install the bos.sysmgt.nim fileset from the CD-ROM, run:

```
# installp -agX -d /cdfs/usr/sys/inst.images bos.sysmgt.nim
```

   or to install the bos.sysmgt.nim fileset from a tape, run:

```
# installp -agX -d /dev/rmt0 bos.sysmgt.nim
```

6. If installing from CD-ROM, to unmount the cdrom file system, enter:

```
# unmount /cdfs
```

7. To configure the NIM master using the `nimconfig` command, enter:

```
# nimconfig -a attr1=value1 -a attr2=value2 \
      ...
```

   For example, to configure a NIM master with the following configuration:

```
master host name = master1
primary network interface = tr0
ring speed = 16
platform = rspc
kernel type = mp
```

   enter the following command sequence:

```
# nimconfig -a netname=network1 -a pif_name=tr0 \
-a ring_speed=16 -a platform=rspc -a netboot_kernel=mp
```

   To initialize the NIM master using an ethernet network interface, issue the following command:

```
# nimconfig -a pif_name=en0 -a master_port=1058 -a netname=net2 -a
cable_type=bnc
```

Note

Note that pif_name is the physical name as opposed to the logical name of our network adapter.

8. To create a file system in the rootvg volume group with 400 MB of space with a mount point of `/export/lpp_source`, enter:

```
# crfs -v jfs -g rootvg -a size=$((2000*400)) \
-m /export/lpp_source -A yes -p rw -t no \
-a frag=4096 -a nbpi=4096 -a compress=no
```

9. To mount the file system, enter:

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

10. The lpp_source contains the installation images copied from the source device (in this example, the CD-ROM). The server of the lpp_source will

be the NIM master. For this example, we assume the images will be stored in the /export/lpp_source/lpp_source1 directory. To create the lpp_source resource named lpp_source1, enter:

```
# nim -o define -t lpp_source -a source=/dev/cd0 \
-a server=master -a location=/export/lpp_source/lpp_source1 \
lpp_source1
```

11. To create a file system in the rootvg volume group with 200 MB of space with a mount point of /export/SPOT, type:

```
# crfs -v jfs -g rootvg -a size=$((2000*200)) \
-m /export/SPOT -A yes -p rw -t no \
-a frag=4096 -a nbpi=4096 -a compress=no
```

12. To mount the file system, enter:

```
# mount /export/SPOT
```

The SPOT resource will be installed from images in the image source (in our case, the lpp_source that was created in the previous step). The server of the resource will be the NIM master, and the SPOT will be stored in the /export/SPOT/SPOT1 directory. To create the SPOT resource, in this example, we will name it SPOT1. Enter the following:

```
# nim -o define -t spot -a source=lpp_source \
-a server=master -a location=/export/SPOT SPOT1
```

13. If you are not supporting diskless and dataless clients, you do not need to continue with this procedure. If you are supporting diskless and dataless machines, create and mount a file system for their resources. To create a file system in the rootvg volume group with 150 MB of space and a mount point of /export/dd_resource:

```
# crfs -v jfs -g rootvg -a size=$((2000*200)) \
-m /export/SPOT -A yes -p rw -t no \
-a frag=4096 -a nbpi=4096 -a compress=no
```

1.4.2.1 nimconfig command

Once the bos.sysmgt.nim fileset is installed, our master installation is complete. Before our master is initialized, we must configure it using the `nimconfig` command. The `nimconfig` command activates the NIM master by making resources available to the NIM environment. The `nimconfig` command requires the following information to initialize the database:

- The primary network interface
- The name of the network object, that is, the network to which the NIM master's primary interface is connected

- The port numbers that the nimesis daemon runs on and the registration port of the client.
- The ring speed if the network is token-ring
- The cable type if the network is ethernet.
- The defaults for lpp_source, SPOT, dataless/diskless resources.

The syntax is as follows:

```
# nimconfig -a pif_name=PrimaryInterfaceUsed -a netname=ObjectName\
-a master_port=PortNumber -a registration_port=PortNumber\
-a [ring_speed=Speed | cable_type=CableType]
```

The values 1058 and 1059 are the default TCP/IP port numbers used by NIM for communication between the NIM master and NIM clients (as appears in the /etc/services file). These values are not set in stone and can be changed using the attributes to the `nimconfig` command.

For example, to initialize the NIM master using an ATM network interface, enter:

```
# nimconfig -a pif_name=at0 -a master_port=1058 -a netname=ATMnet
```

Flags

-r Rebuilds the /etc/niminfo file on the master, if it has been deleted, using information that already exists in the NIM database

-a Assigns the following attribute=value pairs shown in Table 8.

Table 8. Possible attributes when configuring master

Attribute=value pair	Description
pif_name=Pif	Designates the primary network interface for the NIM master. This value must be a logical interface name, such as tr0 or en0, which is in the available state.
master_port=PortNumber	Specifies the port number of the nimesis daemon used for NIM client communication.
netname=ObjectName	Determines the name of the network object you want the `nimconfig` command to use, which will represent the network to which the master's primary interface connects.

Attribute=value pair	Description
ring_speed=Speed	Speed in Mbps. When the pif_name refers to a token-ring network, this value must be given. Acceptable values are 4, 16, and autosense.
cable_type=CableType	Specifies the ethernet cable type. When the pif_name refers to an ethernet network, this value must be given. Acceptable values are bnc, dix, and n/a.
registration_port=PortNumber	Cites the port number used for NIM client registration.

The `nimconfig` command activates the NIM master by performing the following tasks:

1. Defines a network object
2. Logically connects the NIM master to the network
3. Starts the NIM communication daemon, nimesis, using the default port and creates an entry for it in the /etc/inittab file
4. Automatically defines a resource object to represent the network boot resource
5. Defines a resource object to represent the customization scripts that NIM automatically builds to perform customization

1.4.2.2 /etc/niminfo

This is created when a NIM master or client is initialized. This file contains variables used by NIM. Figure 6 shows an example of a niminfo file found on the test environment master.

```
export NIM_NAME=master
export NIM_CONFIGURATION=master
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_MASTER_HOSTNAME=rs1230.ibm.com
```

Figure 6. Example niminfo file of NIM master

If the NIM environment has been initialized from the NIM client (client runs the `niminit` command), the /etc/niminfo file will not contain as many variables. All a client needs to know is the hostname of the master to be able to join the NIM environment. From /etc/niminfo, on one of our clients that follows, we can tell that our NIM client was initialized from within the NIM environment. Figure 7 on page 49 shows an example niminfo file of NIM client.

```
#------------------------------- Network Install Manger-----------------------
#warning - this file contains NIM configuration information and should only be
# updated by NIM.
export NIM_NAME=rs1230b
export NIM_HOSTNAME=rs1230b.ibm.com
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=rs1230a.ibm.com
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION=1059
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_RECOVER="/./SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env
-a hostname=rs1230a.ibm.com"
export SPOT=rs1230a.ibm.com:/export/SPOT/SPOT1/usr
export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
export NIM_CUSTOM="/./SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
location=rs1230a.ibm.
com:/export/nim/scripts/rs1230b.script"
export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
export NIM_BOS_FORMAT=mksysb
```

Figure 7. Example niminfo file of NIM client

If the NIM client has been installed by the NIM master, the /tftpboot/<client hostname>.info file is transferred to the client and becomes the /etc/niminfo file.

Note

Be aware of potential conflicts if you have these ports (1058 and 1059) already assigned.

1.4.3 Network definition

A definition of our network should give us a map of the topology of our physical network.

1.4.3.1 Network definition

Every subnet participating in a NIM environment must be represented by a network object. Creating a network object is not required if all machines are residing on one simple network or if the master has only one interface and all NIM clients are connected to the same subnet to which the master's primary network interface is connected.

Enter the `lsnim -l master` command and notice that `nimconfig` defined "*if1*" for the master.

This is the first primary interface of the master. It was created automatically when the master fileset was configured during the initial setup of the NIM environment. If you have NIM clients residing in one subnet trying to access resources on the master in another subnet, you will need to define additional network interfaces for the master.

Before we do this, however, we will need to create a new network definition representing the subnet to which the other interface is connected. There are three ways to define a network: Using the Web-Based System Manager, smitty, or the command line.

To define the network using the Web-Based System Manager, perform the following steps:

1. From the NIM container, select the networks container.
2. Double-click on **Add New Network TaskGuide**. The Add New Network TaskGuide displays.
3. Follow the TaskGuide instructions to create a network by choosing the type of network, subnet mask, and IP address of a gateway in this subnet. You are given the option of creating an additional route, but we will cover this in a subsequent section.

To define the network using smitty, perform the following steps:

1. To create an additional network object, enter the `smitty nim_mknet` fast path.
2. Select one of the following network types: **tok**, **ent**, **fddi**, **generic**, or **atm**.
3. Select a relevant network name for your NIM environment.
4. Enter the network IP address, which refers to the IP address of the gateway on our subnet.
5. Enter the subnet mask and any comments to further clarify this network definition.

To define the network using the command line, enter the following:

```
# nim -o define -t NetworkType -a net_addr=IPAddress \
-a snm=SubnetMask NetworkObjectName
```

Note

If you do not know the name of the NIM network to which the interface is attached or if a network corresponding to the interface has not been defined, use the find_net keyword and net_definition attribute as described in the previous example. The algorithm it uses is:

```
if matching definition found
        use in definition of NIM client
else
        prompt for:
        type of network
        subnet mask
        default gateway
define network with default route.
```

1.4.3.2 Creating additional interface attributes

The primary interface or the first interface (*if1*) is created when the master is activated, and a sequence number is used to identify the additional interfaces (*if2, if3*) in the machine object definition. To create an additional *if* attribute for the master object, use either Web-Based System Manager, smitty, or the `nim -o change` command operation.

To create an additional *if* attribute for the master object using Web-Based System Manager:

1. Double-click on the machine (**master**, **stand-alone**, **dataless**, or **diskless**). The Machine Properties notebook for the selected machine is displayed.
2. Click the NIM Interfaces tab.
3. Use the NIM Interfaces page to complete the task.

To create an additional *if* attribute for the master object using smitty:

1. Enter the `smitty_nim_mac_if` fast path.
2. Select the **Define a Network Install Interface** option.
3. Select the machine object name. In the example, this is **master**.
4. Enter the host name for the interface.
5. Complete the network-specific information in the entry fields on the Define a Network Install Interface screen.

Note

If a NIM network corresponding to the IP address of the host name specified for the interface (if the first step has not been undertaken) does not already exist, additional network information will be requested so that the network can be defined. Both steps are combined. For a better understanding of what is happening, it is recommended to define a network object in two steps.

To create an additional *if* attribute for the master object from the command line,

for token-ring, enter:

```
# nim -o change -a ifseq_no='NetworkObjectName AdapterHostName \
AdapterHardwareAddress' -a ring_speedseq_no=Speed master
```

for ethernet, enter:

```
# nim -o change -a ifseq_no='NetworkObjectName AdapterHostName \
AdapterHardwareAddress' -a cable_typeseq_no=Type master
```

for FDDI, enter:

```
# nim -o change -a ifseq_no='NetworkObjectName AdapterHostName \
AdapterHardwareAddress' master
```

for other networks, enter:

```
# nim -o change -a ifseq_no='NetworkObjectName AdapterHostName \
AdapterHardwareAddress' master
```

In the example, the following command is run:

```
# nim -o change -a if2='Network2 srv1_ent 0' -a \
cable_type2=bnc master
```

With this syntax, another *if* attribute is created for the master. This tells NIM that the master has an ethernet interface that uses the host name, `srv1_ent`, and the ethernet's adapter hardware address is 0 (not used). It also tells NIM that the master connects to the Network2 network object.

We will, therefore, see that multiple interfaces have been defined; so, multiple networks exist.

In order to perform communication between the net1 and net2 network objects, a NIM route must be established. It defines the gateway to use to go from one network to the other. Additionally, NIM routing provides the gateway entry to /etc/bootptab when the boot resource is allocated during a bos_inst operation. The routing information is subsequently transferred to a booting client and used to establish TCP/IP routing. Then, the master will have a route to its potential client so it can transfer the images and resources.

1.4.4 Routing information

We will show you how to set up NIM routing between NIM networks. Network routes must be defined in the NIM environment because a NIM client must know how it can contact its NIM master if it is in another network. Network traffic may go through several gateways, but it is only necessary for NIM to know about the first and last gateway to communicate with the other NIM machine. Intermediate gateways between the originating network and the destination networks are irrelevant for NIM routing purposes.

There are two types of routes: Default and static. Default more closely models the network configuration of common network environments, and they also allow resources that are distributed throughout the environment to be more easily accessed by any client.

1.4.4.1 Establishing a default NIM route between networks

Default NIM routes are used when no special routing information is found to reach the destination machine. All unspecified IP packets are routed into the network that is defined in the default route.

This procedure describes how to create default NIM routes for two networks (net1 and net2 for example).

To create default NIM routes using Web-Based System Manager, perform the following steps:

1. In the NIM Networks container, double-click on the network icon. The Properties notebook for the selected network is displayed.
2. Click the **NIM Routes** tab.
3. Use the NIM Routes page to add the default route.

You can also use the Add New Network TaskGuide to establish the default network route:

1. In the NIM Network container, double- click on **Add New Network**. The Add New Network TaskGuide displays.
2. Follow the TaskGuide instructions to create the network route as the network is being defined.

To create default NIM routes using smitty, perform the following steps:

1. Enter the `smitty nim_mkdroute` fast path.
2. In the displayed dialog fields, supply the values or accept the defaults. Use the help information and the LIST option to help you.

To create default NIM routes using the command line, enter:

```
# nim -o change -a routingseq_no='default <Gateway>' NetworkObject
```

where `default` is the reserved keyword used by NIM to indicate a default route, and `Gateway` is the host name (or IP address) of the interface that clients on `NetworkObject` use to contact other networks in the NIM environment.

For example, to establish default NIM routes for `net1` and `net2`, type:

```
# nim -o change -a routing1='default gw1_tok' net1
# nim -o change -a routing1='default gw1_fddi' net2
```

where `gw1_tok` is the hostname of the default gateway for machines on `net1`, and `gw_fddi` is the hostname of the default gateway for machines on `net2`.

The detailed information for the network objects now shows the added default routes. Detailed information for the two networks can be displayed as follows:

```
# lsnim -l net1 net2
```

1.4.4.2 Establishing a static route between networks

This procedure describes how to create a static NIM route between two networks. A static route is a means of explicitly defining the next hop from a machine to a particular destination.

To create default NIM routes using Web-Based System Manager, perform the following steps:

1. In the NIM Networks container, double-click on the network icon. The Properties notebook for the selected network is displayed.
2. Click the **NIM Routes** tab.
3. Use the NIM Routes page to add the static route.

You can also use the Add New Network TaskGuide to establish the static network route:

1. In the NIM Network container, double-click on **Add New Network**. The Add New Network TaskGuide is displayed.

2. Follow the TaskGuide instructions to create the network route as the network is being defined.

To create default NIM routes using smitty, perform the following steps:

1. Enter the `smitty nim_mkdroute` fast path.
2. In the displayed dialog fields, supply the values or accept the defaults. Use the help information and the LIST option to help you.

To create default NIM routes using the command line, enter:

```
# nim -o change -a routingseq_no='DestinatioNetworkObject \
Gateway1 Gateway2' NetworkObject
```

where `Gateway1` is the hostname of the interface that clients on `NetworkObject` use to get to `DestinationetworkObject`, and `Gateway2` is the hostname that clients on `DestinationNetworkObject` use to get back to `NetworkObject`.

For example, we can establish a NIM route between `net1` and `net2` using:

```
# nim -o change -a routing1='net1 gw1_tok gw1_fddi' net2
```

where `gw1_tok` is the hostname of the gateway that machines on net1 use to communicate with machines on net2, and `gw_fddi` is the hostname of the gateway that machines on net2 use to communicate with machines on net1.

NIM routing serves two purposes:

- When allocating resources, it checks for connectivity between the server of a resource and the client of a resource.

- In the network boot environment, during installation of the Base Operating System, the routing information defined for NIM networks is used for tcp routing.

Note

To figure out what your gateway is from machine to machine or subnet to subnet, do a traceroute to each machine from both machines. That is, on the master, issue a `traceroute client` command, and, on the client, do a `traceroute master` command. You can use the `netstat -rn` command to determine what your gateway is and, thus, your default route definition, on the master. For a machine installed with two network interfaces and masquerading as a gateway, ensure that ipforwarding has been switched on. From the command line, enter:

```
# no -o ipforwarding=1
```

This allows the kernel of the configured machine to forward on IP packets. A value of 0 prevents forwarding. By default, ipforwarding assumes a value of 0.

So, in summary, we turn to the flowchart in Figure 8 on page 57, which depicts the steps performed by the connectivity algorithm:

1. Get the definition of the client's primary interface (the *if1* attribute).
2. Find the server of the resource (the *server* attribute).
3. Get one of the interfaces for the server (probably the first *if1* attribute).
4. For each interface specified for the server of the resource, test the following conditions:
    a. Determine whether the client's primary interface is connected to the same network object as the server's interface.

       If the client and the server are connected to the same network object, NIM communication is possible.
    b. Determine whether there is a NIM route between the network objects to which the client's primary interface and the server's interface are connected.
    c. If there is a NIM routing definition for the network objects, and the server interface is not being used as a gateway, NIM communication is possible. Otherwise, try another interface on the server.
5. Otherwise, NIM communication is not possible.

Possible checklist items would include ipforwarding and availability of the network.

1.4.4.3 Configuration of a second network adapter

Since NIM configures only one network interface during the NIM client installation, it is necessary to manually configure any further network devices of the NIM client after the installation.

NIM allows you to execute a script on the NIM client after the installation has been made. This kind of script is called a customization script and must be allocated before starting the installation of the NIM client. With this script, you can automatically configure your additional network interfaces after the installation.

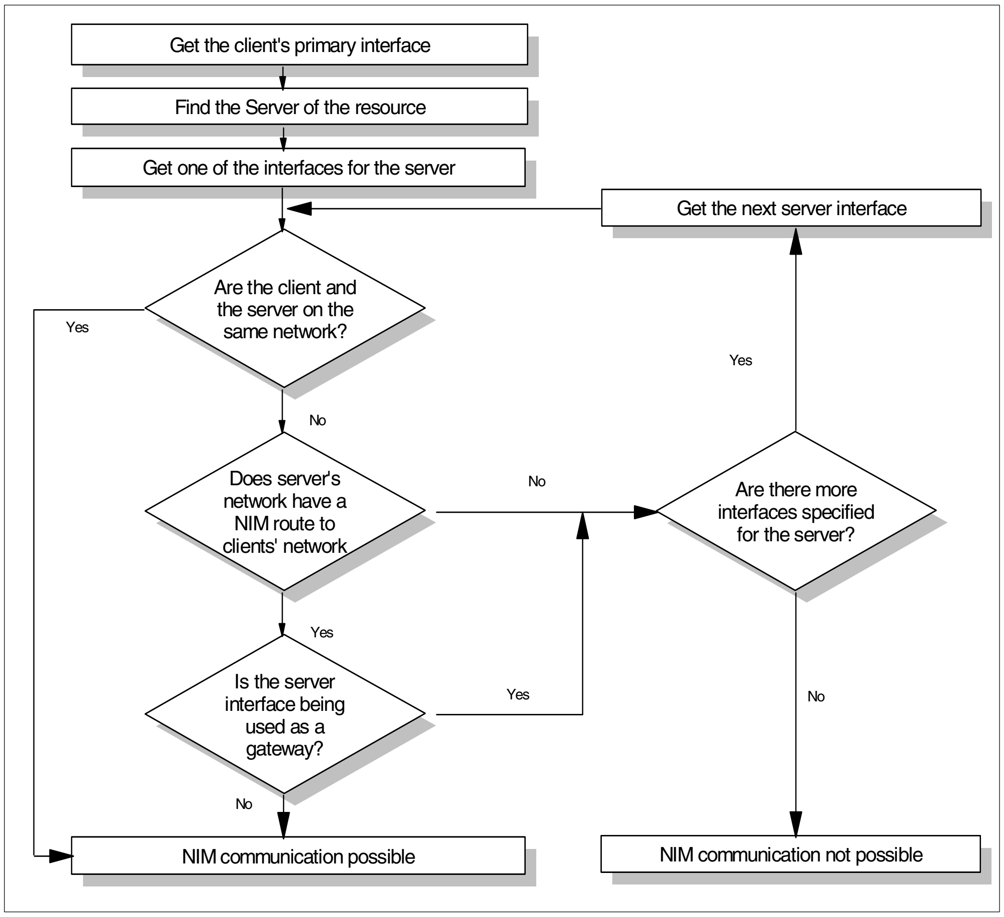

Get the client's primary interface
Find the Server of the resource
Get one of the interfaces for the server
Get the next server interface
Are the client and the server on the same network?
Yes
No
Yes
Does server's network have a NIM route to clients' network
No
Are there more interfaces specified for the server?
Yes
Is the server interface being used as a gateway?
Yes
No
No
NIM communication possible
NIM communication not possible

Figure 8. Communication flowchart

1.4.5 Machine definition

In order for clients to be part of the NIM environment, they must be registered and known to the NIM master. Entries are created in the NIM database for the NIM clients.

1.4.5.1 Adding a NIM client to the NIM environment

From one of the following interfaces, use Method A if the client machine is not running or if the client does not have AIX Version 4 installed. Method A can also be used if BOS is to be installed on the client and the client is to be network-booted manually or to initiate the install from a force-push operation. From one of the following interfaces, use method B if the client machine already has AIX Version 4 installed.

If the NIM client being defined is on a network that is not currently defined in the NIM environment, the `niminit` command will fail. If this is the case, use method A of these procedures to define the client on the NIM master, and then follow the steps in method B to complete the configuration.

Prerequisites

- The NIM master must be defined and initialized.
- You must know the subnet mask, the default gateways for the client machine, and the default gateway for the NIM master.
- Name resolution must be set up.

To add a NIM client to the NIM environment using Web-Based System Manager:

Method A (client machine not running)

1. From the NIM container, open the **Add New Machine TaskGuide**.
2. Follow the TaskGuide instructions to guide you through the configuration.

Method B (client machine is a running AIX machine)

1. To start the Web-Based System Manager software application, enter the fast path:

   ```
   # wsm software
   ```

2. From the software menu, select **New Software (Install/Update) --> Install Additional Software (Custom)**.
3. In the Install Software dialog, specify **bos.sysmgt.nim.client** as the software to install.
4. From the Software menu, select **Nim Client --> Join NIM Environment**.

5. Use the dialog to complete the task.

To add a NIM client to the NIM environment using smitty:

Method A

1. To add a client to the NIM environment, enter the `smitty nim_mkmac` fast path.
2. Specify the host name of the client.
3. The next smitty screen displayed depends on whether NIM already has information about the client's network. Supply the values for the required fields or accept the defaults. Use the help information and the LIST option to help you specify the correct values to add the client machine.

Note

For the machine type field requirement, it is here that we specify what type of machine we wish to include in our NIM environment. The options open are stand-alone, dataless, and diskless. Their definitions and differences have been spelled out earlier.

Method B

1. Install the bos.sysmgt.nim.client fileset on the running machine.
2. Enter the `smitty niminit` fast path.
3. Supply the values for the required fields or accept the defaults. Use the help information and the LIST option to help you specify the correct values to add the client machine.

Note

If the List option is used to display valid platforms for the client definition, only platforms currently supported by SPOT's defined in the NIM environment are displayed. If no SPOTs are currently defined, only rs6k and rspc are displayed as selectable platforms.

To add a NIM client to the NIM environment using the command line:

Method A (client machine not running)

```
# nim -o define -t MachineType -a platform=PlatformType \
-a netboot_kernel=NetbootKernelType -a if1=InterfaceDescription \
-a net_definition=DefinitionNmae -a iplrom_emu=DeviceName MachineName \
[-a ring_speed=SpeedValue | -a cable_type=TypeValue]
```

Example 1:

To add the machine with host name, machine1, with the configuration

```
host name=machine1
platform=rspc
kernel=up
network type=ethernet
subnet mask=255.255.240.0
default gateway=gw1
default gateway used by NIM-master=gw_master
cable type=bnc
network boot capability=yes (no emulation needed)
```

enter the following command sequence:

```
# nim -o define -t standalone -a platform="rspc" \
-a netboot_kernel="up" \
-a if1="find_net machine1 0"\
-a net_definition="ent 255.255.240.0 gw1 gw_master" \
-a cable_type="bnc" machine1
```

Example 2:

To add the machine with host name, machine2, with the configuration

```
host name=machine2
platform=rs6k
kernel=up
network type=token-ring
subnet mask=255.255.225.0
default gateway=gw2
default gateway used by NIM-master=gw_master
ring speed=16
network boot capability=no (emulation needed)
```

you can enter the following command sequence:

```
# nim -o define -t standalone -a platform="rs6k" \
-a netboot_kernel="up" \
-a if1="find_net machine2 0"\
-a net_definition="tok 255.255.225.0 gw2 gw_master" -a ring_speed="16" \
-a iplrom_emu="/dev/fd0" machine2
```

Note

- Prior to AIX Version 4.2, specific network objects must be defined in addition to the steps provided in this procedure. In this procedure, NIM networks are added automatically when needed.
- If the find_net keyword in the *if* attribute causes NIM to successfully match a network definition to the client definition, `net_definition` is ignored.
- Failure to specify the correct platform could result in network boot failures on start-up.

Method B (client machine is a running AIX machine)

1. Install the bos.sysmgt.nim.client fileset on the client machine.
2. From the machine being defined as a client, enter:

```
# niminit -a name=ClientDefinitionName -a master=MasterName \
-a pif_name=Interface -a platform=PlatformType\
-a netbook_kernel=NetboodKernelType -a iplrom_emu=DeviceName \
[-a ring_speed=SpeedValue | -a cable_type=TypeValue]
```

Example 1:

To add the machine with host name, machine1, with the configuration

```
host name=machine1
NIM master's host name=master_mac
primary interface adapter=en0
platform=rspc
kernel=up
cable type=bnc
network boot capability=yes (no emulation needed)
```

enter the following command sequence:

```
# niminit -a name=machine1 -a master=master_mac \
-a pif_name=en0 -a platform=rspc \
-a netbook_kernel=up \
-a cable_type=bnc
```

Example2:

To add the machine with host name, machine2, with the configuration

```
host name=machine2
NIM master's host name=master_mac
```

```
primary interface adapter=tr0
platform=rs6k
kernel=up
ring_speed=16
network boot capability=no (emulation needed)
```

enter the following command sequence:

```
# niminit -a name=machine2 -a master=master_mac \
-a pif_name=tr0 -a platform=rs6k \
-a netbook_kernel=up -a ring_speed=16 \
-a iplrom_emu="/dev/fd0"
```

By examining the output that the system returns from the `lsnim -l clientname` command, you can see whether the client was created successfully. If you see any errors in this output, validate all of your data, checking for accurate spelling, non duplication of NIM names, and so forth, and rerun the command.

Be sure to coordinate this operation with the system administrator on the NIM master, and ensure that all NIM object names are unique in the entire NIM environment.

A configured NIM client has the following properties:

- bos.sysmgt.nim.client is installed
- /etc/niminfo file exists
- An entry in the root user's $HOME/.rhosts file granting the master root rsh permissions
- Corresponding definition in the master's NIM database.

To speed up client installation, a client definition file can be used.

1.4.5.2 Client definition file

This works in tandem with the `nimdef` command**.** The `nimdef` command can also create NIM networks and NIM machine groups automatically in the NIM environment to support the new client definitions.

This is useful in large and complex environments with a large number of clients, and it makes the process of client installation less time-consuming. It acts as a work-around solution if the client has not given the master rsh permission.

The client definition file is structured in stanza format. Each stanza relates to the intended NIM client. Included in the stanza is information pertaining to the machine's network adapter and routing configuration.

1.4.5.3 Client definition file rules

The format of the client definition file must comply with the following rules:

- After the stanza header, attribute lines take the form Attribute=Value.
- If you define an attribute value multiple times within the same stanza, only the last definition is used unless the attribute is machine_group. However, if you specify multiple machine_group attributes, all are applied to the machine definition.
- Invalid attribute keywords are ignored.
- Each line of the file can have only one header or attribute definition.
- Only one stanza may exist for each machine hostname.
- If the stanza header entry is the keyword default, this specifies to use it for the purpose of defining default values.
- You can specify and change default values at any location in the definition file. After a default value is set, it applies to all definitions following it.
- To turn off a default value for all following machine definitions, set the attribute value to nothing in a default stanza.
- To turn off a default value for a single machine definition, set the attribute value to nothing in the machine stanza.
- You can include comments in a client definition file. Comments begin with the pound (#) character.
- When parsing the definition file for header/attribute keywords and values, tab characters and spaces are ignored.

1.4.5.4 Client definition file keywords

The client definition file uses the keywords listed in Table 9 to specify machine attributes:

Table 9. Attributes that can be defined in the client definition file

Keyword	Description
cable_type (required)	Determines the cable type of the machine. Required if network_type is ent.

Keyword	Description
gateway (required)	Specifies the hostname or IP address of the default gateway used by the machine. If the machine does not use a gateway, specify the value 0 (zero) for this attribute.
machine_type (required)	Details the type of the machine: Stand-alone, dataless, or diskless.
network_type (required)	Specifies the machine's network adapter: Ent (for ethernet), tok (for token-ring), fddi, atm, or generic
ring_speed (required)	Establishes the ring speed of the machine. Required if the network_type is tok (token-ring).
subnet_mask (required)	Specifies the subnet mask used by the machine.
nim_name (optional)	Sets the NIM name to use for a machine. Use this attribute if something other than the hostname of the machine with any domain information is stripped off. If you use non-unique hostnames in different domains, a conflict occurs because the same NIM name is used for both machines. In such an environment, define this attribute for the affected machine definitions.
platform (optional)	Determines the machine hardware platform: rs6k for the RISC System 6000 architecture and rspc for IBM Power PC computers. If you do not specify this attribute, the default is rs6k. To determine what type of platform you are using, run the following command: `# bootinfo -p`
net_adptr_name (optional)	Establishes the name of the network adapter used by the machine (tok0, ent0, and so on.)
netboot_kernel (optional)	Gives the type of kernel to use when booting the client over the network. The netboot_kernel values are *up* for uniprocessor or *mp* for multiprocessor.

Keyword	Description
ipl_rom_emulation (optional)	Specifies the device to use for IPL-ROM emulation (/dev/fd0, /dev/rmt0, and so on.)
primary_interface (optional)	Specifies the hostname used for the original machine definition. Use this attribute if the current stanza is only to define an additional interface to a machine that is defined in the NIM environment.
master_gateway (optional)	Gives the gateway the NIM master uses to reach this machine if this machine is on a different network. This attribute is not necessary if this machine is defined on a network that is already defined in the NIM environment or if the NIM master network has a default gateway specified.
machine_group (optional)	Lists the group or groups to which to add the machine when it is defined.
comment (optional)	Specifies a comment to include in the machine definition. The comment string should be enclosed in double quotes(“).

1.4.5.5 Client definition file stanza errors

A definition stanza is incorrect under any of the following conditions:

- The hostname used in the stanza header is unresolvable.
- A required attribute is missing.
- You specify an invalid value for an attribute.
- An attribute mismatch occurs. For example, you cannot specify a group for a machine if the group includes stand-alone machines and you specify machine_type=diskless.
- Machine definitions occur multiple times for the same hostname.
- A machine definition occurs for a machine that is already defined in the NIM environment.
- The primary_interface value in a machine definition does not match the hostname of any defined machine or stanza definition.
- The primary_interface value in a machine definition matches the hostname of another machine definition, but that definition is incorrect.

Figure 9 shows a sample client definition file.

```
#Set default values
default:
machine_type=standalone
subnet_mask=255.255.240.1
gateway=gwtok1
network_type=tok
ring_speed=16
platform=rs6k
machine_group=standalone_grp

#define machines using defaults unless otherwise stated

machine1:
machine2:
platform=rspc
machine3:
default:
machine_type=dataless
```

Figure 9. Example client definition file

To process the client definition file use the `nimdef` command:

```
# nimdef [-p preview] [-c command] [-d defines] [-f filename] Name
```

The `nimdef` command checks for inconsistencies and invalid values in the definition file. It is highly recommended that the `-p` option be used on a definition file.

Exit Status

0... Success

(1-9)... Error State

Table 10 lists flag choices when running the `nimdef` command.

Table 10. Flag choices when running the nimdef command

Flag choice	Description
-c	Generates commands from the client definition file. This flag processes the definition file and generates the commands to add the definitions. The commands are not invoked but displayed as a ksh script that you can redirect to a file and invoke at a later time.

Flag choice	Description
-d	Defines machines from a client definition file.This flag processes the definition file and invokes the commands to add the definitions to the NIM environment.
-p	Checks for errors before using the file to add client and network definitions. It reports the valid definitions, invalid definitions, reasons for failure, additional interfaces, machine groups and their members to add, and all the commands to invoke each definition.
-f Name	Specifies the filename of the client definition file.

1.4.6 Resource definition

We already know that the `nimconfig` command activates the NIM master. In activating the master it also defines a resource object to represent the network boot resource, which is managed automatically by NIM, and resource objects for the lpp_source resource and the SPOT resource. Resources defined in the NIM ODM database contain information detailing the location of the resource, the number of clients using it, and the state it is in.

A resource can be in one of the following three states:

- Ready for use
- Unavailable for use
- Verification being performed

1.4.6.1 Defining the lpp_source resource

The creation of the lpp_source resource for client machine installation is done primarily for stand-alone machines. This resource contains a collection of install images uploaded from CD-ROM or tape into the hard drive (for example, into the /exports/lpp_source file system).

To define the lpp_source resource using Web-Based System Manager, perform the following steps:

1. Open the NIM container; select and open the resources container.
2. Click on the **Add New Resource TaskGuide**.
3. Choose **advanced** to create the lpp_source.

4. On the following screen, you will be prompted for a name for this resource, the server that it is residing on (this need not necessarily be the master), the source (determine whether the install images are on the hard disk, tape, or CD-ROM), and the path to these images.

To define the lpp_source resource using smitty, take the smitty `nim_mkres` fast path and configure the new resource.

To define the lpp_source resource using the command line, enter:

```
# nim -o define -t lpp_source -a location=/inst.images \
-a server=client1 -a source=/dev/rmt0 my_images
```

1.4.6.2 Defining the SPOT resource

There are two types of SPOT's: /usr and non-/usr. The location attribute indicates how the SPOT should be created. The string, /usr, specifies that a machine's /usr file system should be used and it is here that the prototype root files from the BOS image are put. The SPOT will inherit all the lpp files that are already installed on that server. As for a non-/usr file system in the directory you specify, generally, the /export/spot file system and the BOS image in the lpp_source are archived into this new directory. This allows you to customize an installation without affecting the current master machine setup.

The definition of a SPOT requires an lpp_source that has an simages attribute defined. When a SPOT is created, the lpp_source object will be sourced for a set of optional packages that are required by the SPOT.

To create a separate SPOT directory, enter:

```
# nim -o define -t spot -a source=my_images \
-a server=client1 -a location=/export/spot/433_spot my_SPOT
```

In this case, the BOS image in the lpp_source is unarchived into the new directory.

To convert /usr file system into SPOT, enter:

```
# nim -o define -t spot -a source=my_images \
-a server=client1 -a location=/usr my_SPOT
```

Here, the /usr/lpp/bos/inst_root directory is initialized with prototype root files from the BOS image.

1.4.6.3 Defining dataless/diskless resources if applicable

Dataless and diskless machines will have their required resources on remote servers in the NIM environment. These resources are the minimum needed to bring them up to a running state. Even though these files may be located on various servers throughout our environment, they are created and managed from the NIM master. It is the location attribute that defines, by file system, their actual location. The following are the required resources for both dataless and diskless machines:

- SPOT
- root
- dump

In addition, diskless machines require a paging resource to be defined.

The root resource can be defined as follows:

```
# nim -o define -t root -a location=/export/roots \
-a server=master my_roots
```

The paging resource can be defined as follows:

```
# nim -o define -t paging -a location=/export/swapfiles \
-a server=master my_swaps
```

The dump resource can be defined as follows:

```
# nim -o define -t dump -a location=/export/dumps \
-a server=master my_dumps
```

The home resource can be defined (optional) as follows:

```
# nim -o define -t home -a location=/export/homes \
-a server=master my_homes
```

1.4.6.4 Defining the boot resource

The boot resource is defined automatically when you configure the NIM master. It is allocated automatically when you perform a bos_inst, maint_boot, or diag operation for the target client. Then, the link to a network boot image, a <client name>.info configuration file, adds the entry into the /etc/bootptab.

1.4.7 Resource allocation

Resources must be allocated to clients before a BOS installation can be started. Resource allocation does not have to be done in a separate command; it can be done during the bos_inst cust or during whatever NIM operation you are trying to perform. The resource server is granting the client permission to use the resources resident on it. The minimum required and optional resources that can be allocated depend on the type of machine and are presented in Table 11.

Table 11. Minimum required and optional resources by machine

	Stand-alone	Diskless	Dataless
Required	SPOT, lpp_source	SPOT, root, paging, dump	SPOT, root, dump
Optional	fix_bundle, bosinst_data, image_data, makesysb, script, resolv_conf, exclude files, installp_bundle	tmp, resolv_conf, home or shared_home	tmp, paging, home or shared_home

For stand-alone clients, lpp_source and SPOT resources are necessary. If the install is a force-push or a non-prompted install, the bosinst_data resource must also be allocated to stand-alone machines.

We use `# lsnim -q <operation> -t <machine type>` to determine the optional and required resources for a given operation on a specific type of machine.

When we say allocate resources, it means that the file system on which the resource resides is NFS exported to the relevant clients. Allocate operation performs additional bookkeeping beyond the export operation by checking for compatibility between the client machine configuration type and the resource, if the resource is available to a particular client and there is connectivity between the client and server.

Once resources have been allocated and BOS Install enabled, clients may issue a bootp request for initiating the install operation.

To allocate resources for stand-alone clients using smitty, you can use the `smitty nim_alloc` fast path.

To allocate resources for stand-alone clients from the command line, enter:

```
# nim -o allocate -a spot=my_SPOT -a bosinst_data=my_specs1 \
-a lpp_source=my_images stand_client
```

To allocate resources for a diskless client from the command line, enter the following:

```
# nim -o allocate -a spot=my_SPOT -a root=my_roots -a dump=my_dumps \
-a paging=my_swaps -a home=my_homes disk_client
```

To allocate resources to a dataless client is similar, except the paging resource is optional:

```
# nim -o allocate -a spot=my_SPOT -a root=my_roots \
-a dump=my_dumps -a home=my_homes data_client
```

1.4.8 Installation

The Base Operating System can be installed from images that hold the BOS run time files. These images are used to populate the client's /usr file system. This image can come from three different resources: rte, SPOT, or mksysb.

rte

- Run Time Environment (rte) BOS install is the runtime image part of lpp_source resource that has been allocated to the client.
- Only the base files are loaded. It is similar to installing from CD.
- It supports all three forms of installation.
- It is a slower means of install because device support tailored to each machine must be installed.
- The result is a system that is smaller in terms of required space.

SPOT

- The BOS run time image results in a client whose /usr file system has all the files that exist in the SPOT.
- Since it is not tailored to individual systems, it is quicker, but the resulting system is larger.
- It supports the overwrite and preservation forms of installation.

mksysb

- Provides a backup image of an existing machine that can be used as a recovery measure or else to clone one machine onto another. Not only is the BOS installed, but any optional software, paging space information, and other file systems hanging off rootvg are installed; however, an mksysb install only supports the overwrite installation method.

1.4.8.1 Install the BOS from the lpp_source resource

Using installation images to install BOS on a NIM client is similar to the traditional BOS installation from a tape or CD-ROM device because the BOS image is installed from the installation images in the lpp_source resource.

Prerequisites

- The NIM master must be configured, and lpp_source and SPOT resources must be defined.
- The NIM client to be installed must already exist in the NIM environment.

There are two methods of installing BOS on a NIM client using the Web-Based System Manager: An easy installation and an advanced installation.

Easy installation

1. From the NIM container, select the **Install Base Operating System TaskGuide**.
2. Follow the prompts to continue the installation.

Advanced installation

1. From the NIM container, select a target stand-alone machine for the install.
2. From the Selected menu, select **Install Base Operating System (BOS)**.
3. Select **rte - Install from Installation Images**, and fill in the required fields.
4. Click **OK**.
5. If the client machine being installed is not already a running configured NIM client, NIM will not automatically reboot the machine over the network for installation. If the client was not rebooted automatically, initiate a network boot from the client to install it.
6. After the machine boots over the network, the display on the client machine will begin prompting for information about how to configure the machine during installation. Specify the requested information to continue with the installation.

To install BOS on a NIM client using an rte install, perform the following steps:

1. Enter `smitty nim_bosinst` from the NIM master.
2. Select the target machine for the operation.
3. Select **rte** as the installation type.
4. You will be prompted to select the SPOT to use for the installation.
5. Select the lpp_source to use for the installation. In the displayed dialog fields, supply the correct values for the installation options.

6. On the displayed screen, you have the option of allocating a bosinst_data resource that will provide for a non-prompted install. You can also perform customization by allocating an image_data resource during the installation or a script resource after the install. Additional software can be installed if it has been defined in an installp_bundle resource.
7. The other options open to you are: Force the installation, or, if the machine isn't a running machine, postpone the reboot to some time in the future.
8. If all else fails, you can accept the default values. However, the help and LIST option should simplify matters.
9. As we have seen, if the client machine being installed is not already a running configured NIM client, NIM will not automatically reboot the machine over the network for installation. If the client was not rebooted automatically from smitty, the client itself will have to initiate a reboot.
10. After the machine boots over the network, the client machine will begin prompting for information about how the machine should be configured during installation. Supply the relevant information to continue with the installation.

To install BOS on a NIM client from the command line, perform the following steps:

1. To initiate the **bos_inst** operation, enter:

   ```
   # nim -o bos_inst -a source=rte -a lpp_source=Lpp_Source433 \
   -a spot=SPOTName -a boot_client={yes|no} ClientName
   ```

   Specify the resources to be used to support the installation and any additional options for customizing the installation. To perform a simple rte installation, specify the lpp_source and SPOT resources. Note that source=rte is not necessary. rte is the default; so, if source=<value> for bos_inst operation is not specified, rte is assumed.

   If the client machine being installed is not already known to the NIM master, NIM will not automatically reboot the machine. A network boot will have to be performed manually on the machine. If that is the case, supply the boot_client=no attribute to the `bos_inst` command. If the boot_client attribute value is not specified, it defaults to boot_client=yes.
2. If the client was not rebooted automatically, initiate a network boot from the client to install it.
3. After the machine boots over the network, the display on the client machine will begin prompting for information about how to configure the machine during installation. Specify the requested information to continue with the installation.

For example, the client machine, mac1, is not a running configured NIM client. You should specify boot_client=no. To install the client using the lpp_source named lpp_source1 and the SPOT named SPOT1, enter:

```
# nim -o bos_inst -a source=rte -a lpp_source=Lpp_Source1 \
-a spot=SPOT1 -a boot_client=no mac1
```

Note

For an install to succeed, the master must be able to communicate with the client. It is worth ensuring that the client's hostname and ip address are correct in the /etc/hosts file on the master.

1.4.8.2 Install the BOS by restoring an mksysb image

An mksysb installation restores BOS and additional software to a target from an mksysb image in the NIM environment.

Prerequisites

- The NIM master must be configured, and lpp_source and SPOT resources must be defined. Mksysb resource will use SPOT for the boot images and device configuration and will install any additional software that exists in the lpp_source.
- The NIM client must exist in the NIM database.
- The mksysb must be defined on some resource server's hard disk or the mksysb image will be created during this operation from either the NIM master or a running NIM client.
- Only NIM clients running Version 4.2 or later can be used as source machines when creating mksysb images for resources.
- The SPOT and mksysb resources should be at the same level of AIX when used for NIM BOS installations.

Many applications, particularly databases, maintain data in sparse files. These are files with empty spaces to which disk blocks have not been assigned. It is recommended that you have enough free space in the file system for future allocation of the blocks, or, basically, not to have any sparse files on your system.

To install the BOS by restoring an mksysb image using Web-Based System Manager, perform the following steps:

1. If the mksysb resource has already been created, skip to step 4. Otherwise, from the Resources Container, double-click the **Add New Resource** TaskGuide.

2. Follow the TaskGuide instructions to add the mksysb resource to the NIM environment.
3. Upon successful completion of this task, return to the main NIM container.
4. You can select the Install Base OS TaskGuide and follow the prompts to install an mksysb image, or continue with this procedure.
5. From the NIM container, select a target stand-alone machine for the install.
6. From the selected menu, select **Install Base Operating System**.
7. Select **mksysb - Install from a mksysb image** and fill in the required fields.
8. Click OK.
9. If the client was not rebooted automatically, initiate a network boot from the client to install it.
10. After the machine boots over the network, the display on the client machine will begin prompting for information about how to configure the machine during installation. Specify the requested information to continue with the installation.

To install the BOS by restoring an mksysb image using smitty, perform the following steps:

1. If the mksysb resource has already been created, skip to step 6. Otherwise, to create the mksysb resource, enter the `smitty nim_mkres` fast path.
2. Select **mksysb** from the list of resources that can be defined.
3. In the displayed dialogs, supply the values for the required fields. Use the help information and the LIST option to help you specify the correct values for defining your mksysb resource.
4. If the mksysb image does not exist, create it by supplying the values for the fields under System Backup Image Creation Options.
5. Upon successful completion of this task, exit smitty.
6. To use the mksysb resource to install a NIM client, enter the `smitty nim_bosinst` fast path.
7. Select a target machine for the operation.
8. Select **mksysb** as the installation type.
9. You will be prompted to select the SPOT to use for the installation.
10. Select the lpp_source to use for the installation.

11. In the displayed dialog fields, supply the correct values for the installation options or accept the default values. Use the help information and the LIST option to help you.
12. Run the smitty dialog to install the NIM client.
13. If the client was not rebooted automatically, initiate a network boot from the client to install it.
14. After the machine boots over the network, you will be prompted for information about how to configure the machine during installation.

To install the BOS by restoring an mksysb image from the command line, perform the following steps:

1. If the mksysb resource has already been created, skip to step 2. If not, to create the mksysb resource, enter:

   ```
   # nim -o define -t mksysb -a server=ImagesServer -a size_preview=value \
   -a location=PathNameOfImages -a mk_image=yes \
   -a excludefiles=Exclude_FilesResource \
   -a source=NIMClienttoBackup ResourceName
   ```

   Specify the server name and the location of the mksysb image. The mk_image and source attributes are used to create the mksysb image if it does not already exist.

   Example 1:

   To define an mksysb resource, mksysb_res1, from an existing mksysb image located in /export/backups/client_mksysb on the master, enter:

   ```
   # nim -o define -t mksysb -a server=master \
   -a location=/export/backups/client_mksysb mksysb_res1
   ```

   Example 2:

   To create an mksysb image of the client machine, client1, in /export/resources/new_mksysb on the master, and to define an mksysb resource, mksysb_res2, enter:

   ```
   # nim -o define -t mksysb -a server=master \
   -a location=/export/resources/new_mksysb -a mk_image=yes \
   -a source=client1 mksysb_res2
   ```

2. You can initiate the bos_inst operation as follows:

   ```
   # nim -o bos_inst -a source=mksysb -a mksysb=mksysb_res2 \
   -a lpp_source=Lpp_SourceName -a spot=SpotName \
   -a boot_client={yes|no} ClientName
   ```

Specify the resources to be used to support the installation and any additional options for customizing the installation. To perform a simple mksysb installation, specify the mksysb, lpp_source, and SPOT resources.

If the client machine being installed is not already a running, configured NIM client, NIM will not automatically reboot the machine over the network for installation. A network boot must be performed manually on the machine. If that is the case, supply the boot_client=no attribute to the `bos_inst` command. If the boot_client attribute value is not specified, it defaults to boot_client=yes.

3. If the client was not rebooted automatically, initiate a network boot from the client to install it.
4. After the machine boots over the network, the display on the client machine will begin prompting for information about how to configure the machine during installation. Specify the requested information to continue with the installation.

Example 3:

To perform an mksysb installation using the mksysb, mksysb1, the lpp_source, lpp_source1, and the spot, SPOT1, on client machine, machine1, which is not a running configured NIM client, enter:

```
# nim -o bos_inst -a source=mksysb -a mksysb=mksysb1 \
-a lpp_source-Lpp_source1 -a spot=SPOT1 \
-a boot_client=no machine1
```

Note

- It is not recommended to use an mksysb resulting from a sysback image backup because there is additional overhead that NIM and the BOS install process do not support.
- You can use /etc/exclude.rootvg to omit files from the mksysb, use the -a mksysb_flags=-e, or create and specify an exclude_files resource. Also, ensure that unmounting of JFS file systems must be specified when you are defining/creating the mksysb.
- If you are using a non-prompted install, install_method in the bosinst.data file must be set to overwrite.

1.4.8.3 Using a SPOT-copy installs the BOS image on a NIM client

A SPOT-copy installs the BOS image on a machine by copying the BOS files from a SPOT resource.

Prerequisites:

- The NIM master must be configured, and lpp_source and SPOT resources must be defined.
- The NIM client to be installed must already exist in the NIM environment.

To install BOS on a NIM client using a SPOT-copy install with Web-Based System Manager, perform the following steps:

1. From the NIM Container, select a target stand-alone machine for the install.
2. Select **Install Base Operating System**.
3. Select **SPOT- Install from a SPOT Copy Image**, and fill in the required fields.
4. Click **OK**.
5. NIM will attempt to boot the client. If it is not successful, the client will have to be booted manually.
6. After the machine boots over the network, specify the requested information to continue with the installation and configuration of that client.

To install BOS on a NIM client using a SPOT-copy install with smitty, perform the following steps:

1. Enter the `smitty nim_bosinst` fast path. To perform a simple SPOT-copy installation, specify the lpp_source and SPOT resources.
2. Select a machine for the operation.
3. Select **SPOT** as the installation type.
4. You will be prompted to select the SPOT to use for the installation.
5. Select the lpp_source to use for the installation.
6. In the displayed dialog fields, supply the correct values for the installation options or accept the default values. Use the help information and the LIST option to help you.
7. Run the smitty dialog to install the NIM client.
8. NIM will attempt to boot the client. If it is not successful, the client will have to be booted manually.
9. After the machine boots over the network, specify the requested information to continue with the installation and configuration of that client

To install BOS on a NIM client using a SPOT-copy install from the command line, perform the following steps:

1. To initiate the bos_inst operation, enter:

```
# nim -o bos_inst -a source=mksysb -a source=spot \
-a lpp_source-Lpp_Source -a spot=SpotName \
-a boot_client={yes|no} ClientName
```

Specify the resources to be used to support the installation and any additional options for customizing the installation. To perform a simple mksysb installation, specify the mksysb, lpp_source, and SPOT resources.

If the client machine being installed is not already a running configured NIM client, NIM will not automatically reboot the machine over the network for installation. A network boot must be performed manually on the machine. If that is the case, supply the boot_client=no attribute to the `bos_inst` command. If the boot_client attribute value is not specified, it defaults to boot_client=yes.

2. If the client was not rebooted automatically, initiate a network boot from the client to install it.
3. After the machine boots over the network, the display on the client machine will begin prompting for information about how to configure the machine during installation. Specify the requested information to continue with the installation.

For example, the client machine, mac1, is not a running configured NIM client. You should specify boot_client=no. To install the client using the lpp_source, lpp_source1, and the SPOT, SPOT1, enter:

```
# nim -o bos_inst -a source=spot \
-a lpp_source=Lpp_source1 -a spot=SPOT1 \
-a boot_client=no mac1
```

1.4.8.4 /tftpboot/<ClientName>.info

On the SPOT server, a /tftpboot/<client_hostname> link is created to the network boot image that will be used for the installation, and a configuration file is created at /tftpboot/<client_hostname>.info. It is created when the boot resource is allocated (for example, bos_inst operation or maint_boot or diag), which is done automatically by NIM. It is tailored to each client; so, a different /tftpboot/<client_hostname>.info is created for each specific machine, specific in terms of the platform type, kernel type, and network adapter present. It contains information, such as the client's hostname, mount points for NIM resources, and other installation details that are specified as environment variable assignments. This configuration file is transferred from the SPOT server to the client as part of the boot environment configuration process after the boot image has been transferred. The rc.boot program envelopes this file to control processing during network boot.

The Trivial File Transfer Protocol (TFTP) enables you to transfer files to and from remote systems. TFTP is used to transfer the configuration files from a device to your system via the network. You must verify that the TFTP daemon is enabled, the TFTP environment variable set correctly, and that a tftpboot directory exists. If you do not perform these tasks, you will not be able to remotely configure a device, and a message will appear on the console screen stating that TFTP was not enabled.

Sometimes, the existence of the file /etc/tftpaccess.ctl may hamper the functionality of tftp, whereby clients may not be able to tftp to the SPOT server. The .ctl file is used to restrict access to the system through tftp. When this file exists, access is denied to files unless allowed by the control statements within this file. It must be world readable and the first uncommented line must be an allow statement. In algorithm format:

```
If /etc/tfptaccess.ctl exists
    add allow/tftpboot
    remove deny/tftpboot
else
    if NIM didn't uncomment the tftp entry in /etc/inetd.conf while creating this
                            SPOT
    do nothing
else
create /etc/tftpaccess.ctl with allow/tfptboot entry.
```

Look in the file, `/etc/inetd.conf`, for the line that invokes tftpd. If the line is commented out (starts with a pound sign (#)), remove the pound sign.

```
tftp dgram udp wait root /user/etc/in.tftpd in.tftpd -s /tftpboot
```

Verify that TFTP is enabled by typing:

```
# netstat -a | grep tftp
```

The output should be similar to the following:

```
*.tftp Idle
```

Note

On the line relating to tftp in the `/etc/inetd.conf` file, change nowait to wait for slower networks.

We summarize this information in Table 12.

Table 12. Troubleshooting some network problems

Possible Problem	Solution
Network is disconnected	Use firmware to ping the server. Is there a path from client to server? If a path exists (`traceroute`), a disconnected network is not the problem. If no path exists, make sure that a path is available before again attempting to netboot.
TFTP Server is down	Check to see if the server is up and running. You can do this by attempting to make a TFTP connection from the boot server to itself. The connection will be successful if the TFTP Server is running. If the TFTP Server is not running, initialize it (Change the `inetd.conf` file or refresh the `inetd` daemon if tftpd is running).
Image in wrong directory	Look at the server configuration file to see if it points to the directory in which the image is loaded. Make sure that the `/tftpboot` directory is reachable over the network. It may be necessary to edit the `/etc/tftpaccess.ctl` file to allow access to the `/tttpboot` directory.
Image file permissions are incorrect	Check the permissions of the system image file. If necessary, change the permissions for the file to 600.
Bad protocol address	Check the server configuration file to make sure the IP address of the host is correct. Change the configuration if it is incorrect.

1.4.8.5 bos_inst operation

The command line syntax for the `bos_inst` operation is as follows:

```
# nim -o bos_inst -a source=Value -a Attribute=Values... TargetName
```

The target of a `bos_inst` operation can be a stand-alone NIM client or a group of stand-alone NIM clients.

The required and optional attributes that can be specified for the `bos_inst` operation to install and customize a machine are shown in Table 13.

Table 13. Attributes that can be used with the bos_inst operation

Attribute	Description
-a lpp_source=Value (required)	Identifies the lpp_source resource to be used. The lpp_source specified must have the simages attribute. The lpp_source provides software for machine customization. It also provides the BOS image for installation if the source attribute is rte.
-a spot=Value (required)	Determines the SPOT resource to be used. The SPOT provides support for network boot and operations in the boot environment. It also provides the BOS run-time files if the source attribute is SPOT.
-a source=Value	Identifies the source for BOS run-time files. Valid values are rte (installs from a BOS image in the lpp_source), SPOT (copies BOS run-time files from the SPOT), and mksysb (installs the machine from an mksysb image). If not specified, the default is rte.
-a async=Value	Specifies whether NIM should perform operations on group members asynchronously and not wait for the operation to complete on one member before beginning the operation on the next. The default values is async=yes.
-a auto_expand=Value	Indicates whether or not to expand file systems for a force_push installation. The default value is auto_expand=yes.
-a boot_client=Value	Determines whether or not NIM should attempt to reboot the client immediately for BOS installation. The boot_client attribute is the opposite of the no_client_boot attribute. The default is boot_client=yes, indicating that NIM should attempt to reboot the client.
-a bosinst_data=Value	Allocated the bosinst_data resource to use for non-prompted installation.

Attribute	Description
-a concurrent=Value	Details the maximum number of machines from the selected group that should be installing at any given time. This attribute is only valid when the target of the operation is a machine group. If specified, NIM will monitor the progress of all machines in the group and attempt to keep no more or less than the number specified installing until all machines in the group are installed.
-a filesets=Value	Specifies a list of filesets to install on target after BOS installation.
-a force_push=Value	Indicates whether or not a force_push installation should occur. A force_push should be used for installing machines that are running but are not configured with the NIM client fileset.
-a group=Value	Gives the name of a resource group to use for installation. A resource group can be specified as an alternative to specifying multiple resources as separate attributes. If a resource group is specified and it contains a SPOT and lpp_source, the SPOT and lpp_source attributes are no longer required.
-a image_data=Value	Specifies an image_data resource to describe how physical and logical data is organized on the client.
-a installp_bundle=Value	Specifies an installp_bundle resource that lists filesets to install on the target after BOS installation.
-a installp_flags=Value	Tells installp how to apply the filesets specified by the filesets or installp_bundle attributes. The default value is installp_flags=-agQX.
-a mksysb=Value	Provides the run-time files for BOS and other filesets if the source attribute is mksysb. The level of BOS run-time files in the mksysb must be equal to the level of the SPOT used for the installation.

Attribute	Description
-a no_client_boot=Value	Indicates whether or not NIM should attempt to reboot the client immediately for BOS installation. The no_client_boot attribute is the opposite of the boot_client attribute. The default value is no_client_boot=no, indicating that NIM should attempt to reboot the client.
-a no_nim_client=Value	Indicates whether the target should remain in the NIM environment after installation completes. The default value is *no*, indicating that the target system should remain in the NIM environment
-a preserv_res=Value	Indicates whether or not resources in non-rootvg file systems should be preserved on the client system being installed. The default value is reserve_res=no.
-a resolv_conf=Value	Specifies the resolv_conf resource to use for configuring domain and name resolution on a client.
-a script=Value	Specifies the script resource to be run on the target system after all software has been installed.
-a set_bootlist=Value	Indicates whether or not NIM should set the bootlist of the client so that the client boots over the network on the next reboot. Usually, set_bootlist is yes if the client is not going to be rebooted immediately for installation (no_clinet_boot=yes or boot_client=no). The default value is set_bootlist=no.
-a show_progress=Value	Indicates whether group's status should be displayed for each group member when the installation target is a group of machines. The default value is show_progress=yes.

Attribute	Description
-a time_limit=Value	Specifies the maximum number of hours that should elapse before ceasing to initiate installation of additional members of the selected group of machines. This value can only be specified when limiting the number of concurrent operations on a group.
-a verbose=Value	Displays information for debugging. Valid values are 1-5. Use verbose=5 to show maximum detail. The default is to show no debugging output.

1.4.8.6 push -v- force_push

A push install is where the BOS installation has been initiated from the master in the NIM environment. It is the default in the bos_inst operation. The master will attempt to push the installation by allocating the boot resource and attempting to rsh script on client to initiate the boot operation. If boot_client=no is specified, the bos_inst operation will enable the installation but not attempt to initiate the network boot. If boot_client=no and set_bootlist=yes are specified, the bos_inst operation will not attempt to initiate the network boot, but will set the normal mode bootlist to boot off the network so that the installation will commence when the machine is next rebooted. The converse is a *pull* installation whereby the client will initialize itself (`niminit -o bos_inst` command) and will pull the BOS image from the master. On the other hand, a force_push can be used under the following circumstances:

- If the target has granted the master root permission in roots $HOME/.rhosts
- If the bosinst_data resource has been allocated
- If the machine is an rs6k type (however, the service key must be in the normal position).

It tells NIM that the target machine does not necessarily have the bos.sysmgt.nim.client fileset installed and configured. It can be specified by setting the force_push attribute to yes.

1.4.9 Quick setup

For a quick setup of the NIM environment, we can use the **easy startup menu**. It is available to help us configure the master. It makes the NIM environment appear less complex than it is. Much of the NIM configuration is hidden from the user because the default behavior is to supply logical and reasonable names for the configuration. This allows a less experienced systems administrator to set up and configure a NIM environment much more easily than before.

The new smitty easy startup panel has a fast path: `nim_config_env`. It allows the systems administrator to set up a basic NIM environment by looking for a minimum of two pieces of information, specifically:

- Input device for installation images.
- Primary network interface.

Default values are provided for the remaining options. Once this smitty panel has completed successfully, the following actions will have been completed:

- NIM master initialized on the primary network interface
- NIM daemons running
- lpp_source created and available
- SPOT resource created and available

When the user selects the default action of creating file systems for the lpp_source and SPOT resources, the new file systems are created with the specified sizes. The size set for these file systems can be overwrote if the user wants. During creation of the resources, the file systems will be expanded automatically as required. Less interaction and therefore less awareness is needed by the user.

Creating a file system for each resource makes system storage management easier, particularly in environments where resources are added and removed frequently. It is easier to expand a file system if the resource grows, and it is also easier to back up a whole file system rather than individual files.

We recommend creating one large file system (for example, /export) for easier disk space management. Because when you have several different file systems for each type of resource, there are times when you have allocated too much and it is not needed there anymore, but there is no more free PPs to allocate to the appropriate file system you need. For example, you have a large /export/dd_resource, and you still support diskless and dataless; so, you still need those files/directories/file systems around, but you need to create another mksysb resource in /export/mksysb (which is a separate file system), but it fails because you do not have enough free space and you have no more free PPs, but /export/dd_resource is huge and is hardly using any of its allocated disk space.

Setting up resources for diskless and dataless machines and configuring the list of NIM clients from a stanza file can also be carried out at the same time from this smitty panel. If you try to perform either of these tasks, after you have already used the smitty easy startup panel, you will get an error because the machine will assume that you are trying to configure an already configured master. They should be performed from the smitty advanced configuration panel (`smitty nim_config_env`). NIM_config_adv is similar to the method used in 4.1 but has been enhanced to group related tasks together. You will be allowed to create a SPOT and lpp_source at the same time and define multiple NIM clients in one operation by processing a stanza file.

1.5 Administration

A generic NIM environment does not exist. Each NIM environment is tailored to meet the needs of its machines, resources, and networks; so, how you define your environment will determine how well NIM works for you.

It may happen that we need to install or update software on clients or in defined SPOTs. We need to consider the administration of local and remote clients and the probability of clients, which may act as resource servers, being unavailable. Two operations will aid us in administering our environment:

- cust
- maint

Both operations make use of the `installp` command. When these operations are initiated, the users select the desired functionality by specifying the corresponding installp flags (`-a installp_flags=<installp_flags>`) to NIM.

You cannot, however, use the `installp` command directly on a SPOT. If you attempt to do so, NIM will inform you that NIM must be used instead.

These operations can be initiated from both the NIM client and master (as long as the bos.sysmgt.nim.client fileset is installed on the client). When a master initiates these operations, we refer to it as a push operation. However, for certain reasons, the client may initiate the operation, and, when this happens, we call it a pull operation.

1.5.1 Customization

Customization is the installation of supplementary software and fixes on SPOTs and machines in the NIM environment. It can be used to update a machine to a new AIX level. It is also possible to execute customization scripts on clients, even if no installation operation is to be performed.

When considering how to customize our clients and resources, we need to determine the source of the images. For a SPOT cust operation, we can use tape, CD, or lpp_source. However, a machine cust operation is limited to an lpp_source resource as its image source. Be aware that any changes to a lpp_source must be reflected in the table of contents (.toc) of that directory. This is done by performing the check operation. The command, `# nim -o check lpp_source` ..., is needed before the cust operation to update the .toc after modifying the lpp_source.

To install additional/optional software on a running NIM stand-alone client, you must first allocate the lpp_source that will provide the installation images. To add software to an existing lpp_source, use smitty *copy software to hard disk for future installation*, and then issue the `cust` option of the `nim` command. Allocation can be done in the cust operation by simply specifying *-a lpp_source=LppSourceName* in the command.

If you wish to install software using Web-Based System Manager, perform the following steps:

1. Open the NIM container using `wsm nim`.
2. Select the machine or SPOT to be updated.
3. From the menu bar, choose **Selected** --> **Install/Update Software** --> **Install additional software (Custom)**.

This can be done from smitty using the `smitty nim_task_inst` fastpath:

1. Select the machine or SPOT to be updated.
2. You will be prompted to choose the lpp_source resource.
3. Using the LIST option, select the software to be installed.
4. You are given the option of customizing the software installed using the customization script. If you wish, you can also force the customization.
5. The cust operation uses the installp flags of the `installp` command so that we can preview, commit, or overwrite the software.
6. If you are customizing a group of machines in Version 4.3, by using this menu, you can determine the number of machines to install at one time, and you can stop the installation after a given period of time has elapsed.

From the command line, for example, to install the X11.Dt software on the machine client1, first allocate the lpp_source resource containing the desired software, then issue:

```
# nim -o cust -a filesets=X11.Dt client1
```

APARs can be installed on SPOTs or clients in the NIM environment with the fixes attribute of the cust operation.

```
# nim -o cust -a fixes=IX12345 client1
```

Software installed in a SPOT or machine in the NIM environment can be updated by software in an lpp_source by specifying the fixes attribute with the special update_all keyword. First, allocate the lpp_source resource containing the update images. Then enter

```
# nim -o cust -a fixes=update_all clientname
```

1.5.1.1 Customizing a machine

The following rules apply to the cust operation:

1. NIM will require access to a source for the software to install. Therefore, before any installation can take place on the client, an lpp_source resource containing the desired software must first be allocated to the corresponding machine object. For installp to execute, the software must be in bff form (backup file format).
2. The installp flags attribute may be supplied to the cust operation. The installp_flags attributes specifies the flags which the `installp` command uses to perform the installation. If no installp_flags attributes are supplied, NIM uses *-agX* by default.
3. The filesets or installp_bundle attributes may also be supplied on the command line, but not at the same time. These attributes specify the specific software packages installp should install from the allocated lpp_source resource:

   - The filesets attributes must contain a string with names of the software packages separated by white spaces or tabs.

   - The installp_bundle attribute must point to an installp_bundle resource that is allocated to the client. An installp_bundle resource specifies the location where a file containing names of software packages can be found. The `installp` command will import this file with the *-f* flag. The names in this file, optionally followed by a level, should be one per line of text, with any text following the second set of white spaces or tabs on a line being ignored. Output from the `installp -l` command is suitable for input to the installp_bundle resource file.

If neither filesets nor an installp_bundle are supplied, NIM defaults to installing all software from the lpp_source resource.

4. If a script resource has been allocated to the client when the cust operation is performed, NIM will execute the corresponding shell script on the client when the software installation has completed. This function can be used to configure software that was installed during the cust operation. Specifying -a script=<ScriptResource> along with the cust operation instead of having to do separate commands for the allocation.

   The following example applies and commits the afs.rte fileset on a client named Standalone1. Software is taken from an lpp_source resource that has been allocated to the client. If a script resource was also allocated to the client, it will automatically be executed subsequent to the installation:

   ```
   # nim -o cust -a installp_flags="-acgX" -a filesets="afs.rte"
   standalone1
   ```

   The following example will execute the shell script in the script resource on a client named Standalone1, given that the resource has been allocated to the corresponding machine object:

   ```
   # nim -o cust standalone1
   ```

1.5.1.2 Customizing a SPOT

The following rules apply to this operation:

1. The SPOT resource cannot be allocated to machine objects. You can check this by looking at the alloc_count attribute: *# lsnim -a alloc_count -t spot...*. If the alloc_count is not zero, run *# lsnim -a spot -t standalone*. This will list the stand-alone clients that have a spot allocated to it.
2. The installp flags attribute may be supplied to the cust operation. The default for NIM is *-ag.* (The *X* flag, which informs the `installp` command to automatically expand the file systems, is not used on SPOTs.)
3. Similarly, for SPOTs, the filesets or installp_bundle attributes may also be supplied on the command line, but not both at the same time.
4. The script resource cannot be allocated to a SPOT.
5. The following example will apply and commit the afs.rte fileset on a SPOT named SPOT1. Software is taken from an lpp_source resource that has been allocated to the client:

   ```
   # nim -o cust -a lpp_source=my_images -a installp_flags="-acg" \
   -a filesets="afs.rte" SPOT1
   ```

Note

- It is not possible to create a resource in a subdirectory of another.
- You cannot use a SPOT to support installation of a newer level of AIX. The level of SPOT can be greater than or equal to the modification level of images being installed, but it cannot be less.
- Refer to Section 1.7, “Managing resources” on page 108, for more information.

1.5.2 Maintenance

Maintenance operations are used to perform other *non-install* installp operations, such as commit, reject, deinstall, and cleanup, on stand-alone clients and SPOTs. For diskless/dataless clients, maintenance operations are performed on the SPOT resource.

To perform maintenance operations using Web-Based System Manager, perform the following steps:

1. Open the NIM container, using `wsm nim`.
2. Select the machine or resource to be updated.
3. From the menubar, choose **Selected** -->**Software Utilities**--> **Commit/Reject Updates** or **Remove Software**.

To perform maintenance operations using smitty, perform the following steps:

1. Enter the `smitty nim_task_maint` fastpath.
2. Choose the item that corresponds to the intended action.

To perform maintenance operations from the command line, enter the following:

```
# nim -o maint -a installp_flags=InstallpFlags \
[-a filesets=FilesetNames | -a installp_bundle=InstallpBundleName] \
ClientName
```

1.5.2.1 Maintaining a machine

The following rules apply to this operation:

1. The installp flags attribute must be supplied to the maint operation.
2. The filesets or installp_bundle attributes may also be supplied on the command line, but not at the same time. These attributes specify the

specific software packages upon which installp should perform maintenance.

- The filesets attributes must contain a string with the names of the software packages separated by white spaces or tabs.
- The installp_bundle attribute points to an installp_bundle resource that is allocated to the client. It will specify the location of a file containing the names of software packages. The `installp` command will import this file with the *-f* flag. The names in this file, optionally followed by a level, should be one per line of text, with any text following the second set of white spaces or tabs on a line being ignored. Output from the `installp -l` command is suitable for input to the installp_bundle resource file.

If neither filesets nor an installp_bundle are supplied, NIM defaults to use *all*, thereby, apply the operations in the installp_flags on all software in the client.

3. The following example will deinstall the afs.rte fileset on a client named Standalone1:

```
# nim -o maint -a installp_flags="-u" -a filesets="afs.rte" standalone1
```

1.5.2.2 To perform maintenance on a SPOT

```
# nim -o maint -a installp_flags=InstallpFlags \
[-a filesets=FilesetNames | -a installp_bundle=InstallpBundleName] SpotName
```

The maint operation on SPOT objects performs maintenance on existing software in SPOTs using the `installp` command. It is used to perform commit, reject, or deinstall of software packages. The following rules apply to this operation:

1. The installp flags attribute must be supplied to the maint operation. The installp_flags attributes specifies the flags that the `installp` command uses to perform the maintenance.
2. The filesets or installp_bundle attributes may also be supplied on the command line, but not at the same time. These attributes specify the specific software packages on which the `installp` command should perform maintenance:
   - The filesets attributes must contain a string with names of the software packages separated by white spaces or tabs.
   - The installp_bundle attribute must point to an installp_bundle resource that is allocated to the client.

If neither filesets nor installp_bundle are supplied, NIM defaults to use *all*, thereby, applying the operations in the installp_flags to all software in the SPOT.

3. The following example will deinstall the afs.rte fileset on a SPOT named SPOT1.

```
# nim -o maint -a installp_flags=”-u” -a filesets=”afs.rte” SPOT1
```

The following example will remove the FDDI support from a SPOT named SPOT1:

```
# nim -o maint -a installp_flags=”-u” -a filesets=”devices.mca.8ef4”
SPOT1
```

This operation will cause the network boot images associated with the SPOT to be re-created. The FDDI network boot images will no longer be available, which can save space on the / file system.

1.5.3 Client operations

In this section, we will cover the operations that can be initiated from the client. Clients, however, have limited ability to manipulate the NIM ODM database. A client cannot define resources, but it can change its own definition.

The first relevant operation to a client is the ability to add itself to the NIM environment. We use the term *initialize itself*. If an installed and running machine wants to add itself to the NIM environment, it can do so from the local machine so long as it has the bos.sysmgt.nim.client fileset installed.

1.5.3.1 Client control

Control of machines and resources can be a matter of contention in the NIM environment. Control of machines relates to the master’s ability to remotely execute commands on the client - This can be enabled or disabled. On the NIM master, you can check which system has control by looking at the control attribute in the output generated:

```
# lsnim -l ClientName
```

The four states of control are listed in Table 14.

Table 14. Control states

Control state	Explanation
Control attribute is not set	If the control attribute is not displayed when listing the machine object attributes, neither the master nor the stand-alone client has control.
control=master	The master has allocated resources to the client and is ready to initiate an operation.
control=ClientName	The stand-alone client has allocated resources and can now initiate NIM operations on itself.
control=ClientName push_off	The stand-alone client has prohibited the NIM master from allocating resources or initiating operations on the client. The client itself can still control the allocation of NIM resources and the initiation of NIM operations.

Even though the the master's ability to execute commands on the local client machine has been disabled (by removing the master and root from the .rhosts file), the client can still access resources by pulling them from the resource server. This only applies if the client is still defined in the NIM database and has not been removed by the master.

Control of resources relates to the power machines have over resources to prevent them from being reallocated or deallocated until the operation completes. When a client allocates resources before or during a NIM operation, the client has control over those resources, and any attempt to allocate the resources from the master will fail. Once the master or client allocates a resource, the other cannot deallocate that resource until the operation and/or the original system (master or client) actually deallocates the resources. Other clients can still allocate that same resource. However, being a master, the master machine can enforce an allocation using the -F option. Other clients should use caution. If they are not careful, they could corrupt NIM DB or the client system performing an operation at the time they do a deallocate or a reset with the -F flag.

1.5.3.2 The niminit and nimclient commands

This section explains the niminit and nimclient commands.

niminit command

The `niminit` command configures the NIM client package. This must be done before the `nimclient` command can be used. When the required attributes are supplied to the `niminit` command, a new machine object will be created to represent the machine where the `niminit` command is being executed. When the `niminit` command completes successfully, the machine will be able to participate in the NIM environment. Once the NIM client package has been successfully configured, the `niminit` command may be run again to rebuild the /etc/niminfo file on the client. The /etc/niminfo file is used by the `nimclient` command and must be rebuilt if it is accidently removed by a user. The syntax for its use is:

```
# niminit {-a name=Name -a pif_name=Pif -a master=Hostname} \
[-a master_port=PortNumber] [-a registration_port=PortNumber] \
[-a ring_speed=Speed | -a cable_type=CableType] [-a iplrom_emu=Device] \
[-a platform=PlatformType] [-a netboot_kernel=NetBootKernelType] \
[-a adpt_addr=AdapterAddress]
```

To rebuild the /etc/niminfo file:

```
# niminit -a name=Name -a master=Hostname -a master_port=PortNumber
```

To verify that the `niminit` command completed successfully, enter the following command at the client:

```
# nimclient -l -L MachineObjectName
```

The smitty equivalent can be accessed by using the `smitty niminit` fast path.

Attributes that can be used with the `niminit` command are listed in Table 15.

Table 15. Possible attributes to choose from when using the niminit command

Attribute	Description
-a name=Value (required)	Specifies the name that NIM will use to identify the workstation.
-a pif_name=Pif (required)	Defines the name of the network interface for all NIM communications.
-a master=HostName (required)	Specifies the hostname of the NIM master. The client must have the ability to resolve this hostname to an IP address.
-a master_port=PortNumber (optional)	Specifies the port number of the nimesis daemon used for NIM communications.

Attribute	Description
-a cable_type=CableType (optional)	Used when pif_name references an ethernet network. It determines the type of cable. Acceptable values are bnx, dix, and n/a.
-a ring_speed=RingSpeed (optional)	Used when pif_name refers to a token-ring network. It is the ring speed in MBps and acceptable values are 16, autosense, and 4.
-a iplrom_emu=Device (optional)	Identifies the device that contains a ROM emulation image. This image is required for models that do not have internal support for booting via network interface.
-a platform=PlatformType (optional)	Specifies the platform that corresponds to the client's machine type. The supported platforms are rspc, rs6k, and chrp.
-a adpt_add=AdapterAddress (optional)	Specifies the hardware address that corresponds to the network adapter.
-a registration_port=PortNumber (optional)	Specifies the port number used for NIM client configuration.
-a netboot_kernel=NetbootKernelType (optional)	Specifies the type of kernel to use when booting the client over the network. The netboot_kernel values are up for uniproccessor machines and mp for multiprocessor machines.

nimclient command

The `nimclient` command allows NIM operations to be performed from a NIM client. It can be likened to the `lsnim` command of the master. The `nimclient` allows the clients to pull NIM resources. It can be used to enable or disable the NIM master's ability to initiate workstation installation and customization for the machine. The `nimclient` command can be used to generate a list of available NIM resources or display the NIM resources that have already been allocated to the client. A limited set of NIM operations can also be performed by the `nimclient` command using the -o flag.

Consider the following contexts:

• If you want to enable or disable the NIM master's push permissions, use
```
# nimclient {-p} | {-P}
```

- To list information about the NIM environment, use
  `# nimclient -l lsnim`Parameters

If you want to set the date and time to that of the NIM master, use

```
# nimclient -d
```

To perform a NIM operation, use

`# nimclient -o` Operation [`-a Attribute=`Value]

Note

The nimclient `-d` command would be a useful addition to the user-defined customization script as part of the initial NIM installation

Table 16 lists flags that can be used with the nimclient.

Table 16. Flags that can be used with the nimclient

Flag	Description
-a Attribute=Value	Passes information to NIM operations. From the master: Use the `lsnim -q Operation -t Type` command to get a list of valid attributes for a specific operation. From the Client: Use the `nimclient -l -q Operation -t Type` command to get a list of valid attributes for a specific operation.
-d	Sets the client's date and time to that of the master.
-l Lsnim parameters	Executes the `lsnim` command on the master using the lsnim parameters that you specify. All the parameters you use with this option must adhere to the syntax rules of the `lsnim` command. Note that some lsnim syntax requires the use of a NIM object name. To find out what the NIM name is for your machine, look in the /etc/niminfo file.

Flag	Description
-o operation	Performs the specified operation. The possible operations are: Allocate - Allocates a resource for use. bos_inst - Performs a bos installation. Change - Changes an object’s attributes. Check - Checks the status of a NIM object. Cust - Performs software customization. Deallocate - Deallocates a resource. diag - Enables a machine to boot a diagnostic image. maint_boot - Enables a machine to boot in maintenance mode. (This operation only applies in 4.2 or later). reset - Reset’s an object’s NIM state. showres - Displays the contents of a NIM resource. (This operation applies only to AIX Version 4.2 or later).
-p	Enables the NIM master to push commands.
-P	Removes the NIM master’s permissions to push commands.

1.5.4 Other NIM operations

This section covers other NIM operations.

1.5.4.1 nim command operation options

Other NIM operations that are used in the NIM environment with the `nim -o <operation><objectname>` command are displayed in Table 17.

Table 17. Operations used in conjunction with the nim command

Operations	Machines	Networks	Resources
change - Changes or adds information to an existing NIM object	x	x	x
define - Used to create a NIM object	x	x	x
remove - Removes an object from the NIM environment	x	x	x
check - Checks the status of a NIM object	x		x
fix_query - Lists the fix information for a given APAR or keyword	x		x

Operations	Machines	Networks	Resources
lppchk - Verifies installed filesets on NIM machines and SPOTs	x		x
lslpp - Displays installp information about the specified object	x		x
maint - Performs software maintenance on a SPOT or NIM client	x		x
maint_boot - Enables a machine to boot in maintenance mode	x		
showlog - Displays NIM client or SPOT logs	x		x
allocate - Performs resource allocation	x		
bos_inst - Initializes the NIM environment to perform a BOS installation on a stand-alone client.	x		
cust - Performs software customization or installation on a SPOT or NIM client	x		x
deallocate - Deallocates resources from NIM clients	x		
dkls_init - Initializes the NIM environment resources required to boot a diskless machine	x		
dtls_init - Initializes the NIM environment resources required to boot a dataless machine	x		
reset - Resets an objects NIM state	x		
unconfig - Unconfigures the NIM master fileset	x		
diag - Enables the diagnostic boot resource for a client to use	x		
sync_roots - Synchronizes root directories for diskless and dataless clients for a specific SPOT			x
showres - Displays the contents of a NIM resource			x

1.5.4.2 Diskless/dataless client

The underlying mechanism used by NIM to manage dataless and diskless clients is fundamentally the same as that for the installation of stand-alone machines.

The reasons for defining a machine as dataless or diskless are basically cost driven. Only a small hard disk is needed for the dataless machines, and none is needed for the diskless machine. It is easier to manage what is installed on these types of machines because they all mount the same file systems (resources) allocated by the master. It is also easier to back up user data if it is all confined to one machine rather than having to back up the individual clients.

Resources must be allocated prior to or during the initialization of the client. The parallel of the bos_inst operation is the dkls_init or dtls-init operation.

Managing software, maintenance, and customization is done primarily on the machines serving the resources to the dataless and diskless clients. Because the dataless and diskless clients mount the /usr and / (root) file systems from a server, the installation or deinstallation of software on a dataless or diskless client must be done on the resource server; in this case, that is the SPOT server. Software updates will also need to be done on the resource server. Any changes to SPOT will then be propagated throughout our environment. The sync_roots operation can be used to ensure that the client root directories match the root parts stored in the SPOT.

The resources listed in Table 18 are managed by NIM to support diskless and dataless clients.

Table 18. Resources available to dataless and diskless machines.

Resource	Description
boot (required)	Defined as network boot image for NIM clients. The boot resource is managed automatically by NIM and is never explicitly allocated or deallocated by users.

Resource	Description
SPOT (required)	Defined as a directory structure that contains the AIX run-time files common to all machines. These files are referred to as the usr parts of the fileset. The SPOT resource is mounted as the /usr file system on dataless and diskless machines. Contains the root parts of filesets. The root part of a fileset is the set of files that may be used to configure the software for a particular machine. These root files are stored in special directories in the APOT, and they are used to populate the root directories of diskless and dataless clients during initialization. The network boot images used to boot clients are constructed from software installed in the SPOT.
root (required)	Defined as a parent directory for client "/" (root) directories. The client root directory in the root resource is mounted as the "/" (root) file system on the client. When the resources for a client are initialized, the client root directory is populated with configuration files. These configuration files are copied from the SPOT resource that has been allocated to the same machine.
dump (required)	Defined as a parent directory for client dump files. The client dump file in the dump resource is mounted as the dump device for the client.
paging (required for diskless but optional for dataless)	Defined as a parent directory for client paging files. The client paging file in the paging resource is mounted as the paging device for the client.
home (optional)	Defined as a parent directory for client /home directories. The client directory in the home resource is mounted as the /home file system on the client.
shared_home (optional)	Defined as a /home directory shared by all clients. All clients that use a shared_home resource will mount the same directory as the /home file system.

Resource	Description
tmp (optional)	Defined as a parent directory for client /tmp directories. The client directory in the tmp resource is mounted as the /tmp file system on the client.
resolv_conf (optional)	Contains name server IP addresses and a network domain name. Unlike the other resources, this resource does not remain mounted by the client. Instead, it is copied to the /etc/resolv.conf file in the client's root directory.

1.6 Differences between AIX versions

The NIM improvements came with each release of AIX. In covering these differences, that is, differences in terms of NIM objects, NIM's application, maintenance, and administration and the tools to support it from one level to the next, we will start with AIX Version 4.1 and trace its progress up to AIX Version 4.3.3.

1.6.1 Machines

In AIX Version 4.1, configuring a NIM environment was a time-consuming and laborious exercise requiring considerable expertise. This was true for a simple network install environment where the flexibility offered was not always there. Configuring the master and creating resources was done in separate steps. Similarly, clients had to be defined in a number of steps. The procedure for client definition was as follows:

1. If the network that the client was attached to had not been previously defined, define it.
2. Define the route between the client network and the network interface defined on the master.
3. Define the client by selecting the correct NIM network.

All in all, it took three smitty panels to set up one client machine.

When installing a client using an mksysb installation, if the source mksysb was created on a machine with a different hardware configuration to the target, all the necessary device support needed to be installed on the target system prior to creating the mksysb resource. Obviously, this was not a very efficient exercise. Starting with AIX Version 4.2, cloning was built into the BOS installation program so that all necessary additional device support was automatically installed at the time the mksysb image was installed. What made it a little bit easier was that fact that NIM allowed the creation of the image while defining the NIM resource. In AIX Version 4.2, with large environments in mind, the client definition file together with the `nimdef` command allowed you to define multiple NIM clients in a single operation. A restriction was placed on the clients in AIX Version 4.3: They were prevented from adding themselves to the NIM environment. To prevent clients from initializing themselves, you use the smitty fastpath, `smitty nim_client_reg`, or run the following command:

```
# nim -o change -a client_reg=no master
```

1.6.2 Networks

We mentioned that prior to defining a machine in AIX Version 4.1, the network and static routes had to be defined. In AIX Version 4.2, NIM had the ability to resolve the fully-qualified hostname of the client machine to an IP address. It could then determine whether a NIM network had been defined for the subnet the client was connected to. If the network had been defined, details of it were displayed. If the network had not been defined, the user was prompted for the type of network the client was connected to. Once subnet mask and default route were supplied, the network was defined as part of the client definition process. The introduction of default routes improved the speed of defining clients in the environment. They more closely match the underlying network topology.

Generic networks were introduced in AIX Version 4.1.5 to support non-boot-dependent NIM operations on networks that did not support network boot at the time, such as ATM networks. In AIX Version 4.2.1, ATM support was offered. For ATM, ATM does not support Network boot; it was added as a valid NIM network type, but, for ATM installs, the client must be up and running because the boot image to support ATM install is actually written to the clients' /dev/hd5; it is not using bootp to get the boot image from the SPOT server. FDDI support was also offered to rspc machines.

1.6.3 Resources

It appears that the functionality of resources within the NIM environment was fairly restricted. This is due, in part, to the incompatibilities between the kernel codes of the different versions of AIX. If you wanted to have an AIX Version 4.2 SPOT or AIX Version 4.3 SPOT installed on a machine running AIX Version 4.1, you would have to have a 4.2 machine in the environment to serve the SPOT resource to the other clients.

In AIX Version 4.1, network boot images were defined according to the following syntax:

<SPOTname>.<platform>.<networktype>

In AIX Version 4.2, it was edited to contain the processor type. For versions prior to AIX Version 4.3, a network boot image was created for every type of machine and network for which support was available in the SPOT. To conserve space in AIX Version 4.3, the number of network boot images created in /tftpboot is limited to the machine/network configuration of the client (this is done when a SPOT resource is allocated and is defined if NIM clients have been defined previously); so, if a new client joins and no network boot image exists for it, just run a check on the SPOT, and it will create a link to the relevant boot image for that client.

Again, the process of defining and allocating resources was time consuming. In AIX Version 4.2, NIM permits resources to be allocated automatically by allowing you to specify the resources when the operation is invoked.

The resolv_conf resource was introduced in AIX Version 4.2. It represents a configuration file used to define the Domain Name Servers. Note that it is not supported if it is allocated to a machine running AIX Version 4.1.

For environments in which it is not necessary to worry about who has access to the NIM resources, you can export resources globally, thereby, eliminating the repeated updates to the /etc/exports file. It is not possible to export the resources used exclusively by dataless and diskless machines globally. These global resources are readable by any machine in the network; it does not have to be a NIM client. They are available when allocated, and, when deallocated, they are unexported from all clients. This feature is only supported on resource servers that have the NIM client set at AIX Version 4.3.0 or higher.

In an environment in which access to resources is to be restricted, you can control the behavior of clients. Default behavior allows them to allocate and use any resources in the NIM environment. To change this from smitty, use the `smitty nim_control_alloc` or `nim -o change -a client_alloc=no clienthostname` command.

As we have seen, the mksysb is a resource that has improved with each version of AIX. In AIX Version 4.3, it can be used to support alternate disk installation. This allows you to copy the mksysb image onto another disk on the same machine while running. This is ideal in today’s cost-driven environment where downtime means loss of profits.

AIX Version 4.3 added the variables PVID= and Connection= to the file to support targeting a specific SSA drive during a BOS install. AIX Version 4.3.3 added new features to reduce possible errors arising from this file. There exists a command in `/usr/lpp/bosinst/bicheck <filename>` that will perform a check on the file to see if it contains any errors. The `bicheck` command verifies the existence of the control_flow, target_disk_data, and locale stanzas as needed. If a non-prompted install is specified, the existence of values for required fields is confirmed. If the `bicheck` command returns a 0, there are no errors in the file. The `mksysb` and `savevg` commands can be used to check the file if it corresponds with the hardware that is actually present amending the target_disk_data if necessary.

The nim_script resource defines the directory containing customization scripts created by NIM. AIX Version 4.3.3 supports the nim_script resource to reside on NIM servers other than the NIM master. This will reduce the resource contention on the NIM master during installation of large quantities of machines. It will also impact on network contention by placing the resource in an optimal position in the network. Depending on the operation, NIM will use the following rules to determine on which NIM server to place the nim_script resource:

- If it is a bos_inst operation, the nim_script resource will be located on the SPOT server.
- If it is a cust operation and the lpp_sources is allocated, the nim_script resource will be placed on the lpp_source resource.
- If it is a cust operation, no lpp_source is allocated, and a script resource is allocated, the nim_script will be placed on the script server.
- In all other circumstances, the nim_script will be placed on the NIM master

In an attempt to make NIM more scalable, better facilities for distributing the install resources to other NIM servers have been added in AIX Version 4.3.3. NIM already provides the capability to propagate SPOT and lpp_source resources, and, starting with AIX Version 4.3.3, all install resources are able to be propagated to other NIM servers. The resources for which this feature is adding support are mksysb, script, bosinst_data, image_data, installp_bundle, fix_bundle, resolv_conf, and exclude_files. This allows you to have replica copies of these resources throughout your NIM environment. The idea is to minimize network load, permitting you to locate them on the same subnet or at least closer to your clients. These replicas are just copies; the onus is on the administrator to keep them in sync. The smitty panels have a new Source for Replication field to reflect this change.

1.6.4 Groups

In AIX Version 4.1, it was necessary to configure the environment for each NIM client separately and then perform NIM operations on each client in turn. In AIX Version 4.2, groups can be set up for groups of client systems with similar hardware configurations on which identical operations need to be performed at the same type. Further enhancement was made in AIX Version 4.3.2 to alleviate resource and network constraints. Two new settings are available and allow the administrator to determine how many members of the group should be operated on at once and after what period of time should the master cease initiating the operations. For example, this would allow the administrator to install 100 machines. No more or no less than 10 machines should have the operation in progress at any one time. This ensures the network bandwidth is not exhausted. This option is only valid for certain operations when a NIM group is used as the target. The NIM operation will fail with an error message if the options are used for an individual machine, such as lpp or SPOT targets. The options appear near the end of the NIM smitty panels that initiate operations likely to generate large amounts of network traffic, such as installation and cloning.

Resource groups are similar in concept to client groups. They are handled as a single entity and assigned to client systems or groups for NIM operations. In AIX Version 4.1, allocating five resources to five machines would require twenty five allocations. In AIX Version 4.2, once the resource group and the machine group exist, it now only takes one allocation.

1.6.5 smitty and Web-Based System Manager

Web-Based System Manager was introduced in AIX Version 4.3. New functions and task guides have been added to ease NIM operations. Even though machine groups were present in AIX Version 4.3.0, due to time constraints, support for them was only added in AIX Version 4.3.3; so, from the NIM container, you can create and manipulate machine groups. Installation of NIM on a machine is now even simpler because it makes use of the taskguide Install Base Operating System. In its current form, however, it still lacks some of the capabilities of the smitty screens or the command line. For instance, creation and maintenance of resource groups is not yet possible.

In AIX Version 4.1, NIM adopted an object-oriented style interface to software installation and maintenance, whereas, the standard smitty interface is task-oriented. In AIX Version 4.2, NIM smitty panels mimic the standard smitty interfaces as closely as possible, and, for the equivalent NIM install or update operation, it will produce the same results as the standard install operation. Version 4.2 also added the easy startup menu in smitty. This has greatly reduced the complexity of NIM. It has in turn speeded up the definition of a NIM environment. AIX Version 4.2 also added the options to back up the database, re-create the /etc/niminfo file, and unconfigure the master from the smitty interface. Previously, these all had to be done using the NIM command from the command line. In AIX Version 4.3.2 smitty and Web-Based System Manager added support for ATM networks and the IEEE802.3 interface, which was previously just a command line. The smitty fastpath, `smitty nim_env_opts`, was introduced in AIX Version 4.3.

1.6.6 Commands, operations and attributes

The `nim` and `nimclient` commands have been enhanced in AIX Version 4.2 to allow resource allocation and NIM operation invocation at the same time. It is no longer necessary to allocate resources in a separate step prior to performing the NIM operation, although this is supported for compatibility with AIX Version 4.1. The nimclient had the following attributes added, which are used with the -o flag:

- maint_boot
- reset
- showres

It also added the -p and -P flags.

The `nimdef` command was available from AIX Version 4.2 onwards.

The `nimconfig` command added the -r flag and the attribute registration_port, which is used with the -a flag in AIX Version 4.2.

The `lsnim` command added the -g and -m flags in AIX Version 4.2 and later. It also added the maint_boot, showlog, and showres attributes to the -o flag. In AIX Version 4., the operation attribute, alt_disk_install, was added, and the -t flag now supported exclude_files, resolv_conf, atm, mac_group, and res_group.

1.6.7 Other enhancements

The lock granularity of the NIM subsystem has been improved to allow more operations in parallel. Previous versions of NIM would lock an object for the duration of some operations, thus, preventing any other operation on the same object. From AIX Version 4.3.2 onwards, the locking methodology has been changed to only lock the object for critical parts of the operation. This will allow other operations to complete when, in the past, they may have waited or timed out.

In conjunction with this enhancement and to make the environment more scalable, the nimesis daemon is now a multi-threaded process. This is the daemon, running on the master, that funnels updates and state changes from the clients to the master. Therefore, when installing a large number of machines at once, the master can very easily become swamped. Since AIX Version 4.3.3, the nimesis daemon will be able to handle nimclient operations to support installations of around 200 client machines simultaneously, with better performance than previous releases. To set the number of threads needed to support NIM activities in your environment, you can use the `smitty nim_tune_nimesis` fastpath or the max_nimesis_threads attribute in the `nim` command.

In terms of security, NIM's method of running commands on remote clients is based on standard AIX authentication and, starting with AIX Version 4.3.1, Kerberos 5. In SP environments, Kerberos 4 is used for authenticating remote commands. In order to improve security, AIX Version 4.3.3 also supports Kerberos 4 authentication in NIM operations.

1.7 Managing resources

The commands for managing software on stand-alone clients and SPOT resources are generally the same. Specify the name of the machine, group, or SPOT as the target of the option.

If the SPOT is currently allocated on a NIM client, NIM prevents the change to the SPOT.

Software updates to a SPOT can cause the SPOT's network boot images to be rebuilt when necessary. If you think the boot images are bad, you can force them to be rebuilt using the NIM check operation.

Software updates to a SPOT may also cause software updates to occur in the root parts of diskless and dataless clients of the SPOT. This will occur automatically. You can force a synchronization using the NIM sync_roots operation on the SPOT.

1.7.1 Maintaining software in an lpp_source

To add or remove software in an lpp_source, simply add or remove the installation image from the lpp_source directory, and then initiate the NIM check operation on the lpp_source.

1.7.1.1 Copying software to an lpp_source

The following steps will copy software to an lpp_source under different conditions.

To copy software to an lpp_source using Web-Based System Manager, perform the following steps:

1. From the resources container, double-click the **lpp_source**.
2. Identify the location of the resource.
3. Select **Resources --> Copy Software to Directory**, and specify, as the destination directory, the location to the resource identified in the notebook.
4. After the copy is completed, select the **lpp_source**, highlight it, and, from the menu bar, select **Check NIM state**. This action updates the table of contents (.toc) file for the lpp_source.

To copy software to an lpp_source using smitty, perform the following steps:

1. To copy software from installation media to an lpp_source, insert the installation media in the appropriate drive of the lpp_source server.
2. To copy the software to the lpp_source directory, enter `smitty bffcreate` from the resource server.
3. Enter the input devic/directory for software.
4. In the displayed dialog fields, supply the correct values or accept the default values. Specify the lpp_source location for the directory to store the installation images. Use the help and the LIST option to help you.

To copy software to an lpp_source from the command line, perform the following steps:

1. Copy the software from the media to the lpp_source directory.
2. Perform the NIM check operation on the lpp_source by entering the following command:

```
# nim -o check Lpp_SourceName
```

1.7.1.2 Removing software from an lpp_source

To remove software from an lpp_source using the command line, perform the following steps:

1. Remove the installation image from the lpp_source directory.
2. Perform the NIM check operation on the lpp_source by entering the command `# nim -o check lpp_sourceName`.

1.7.1.3 Running the NIM check operation

After adding or updating software, you must run the NIM check operation on the lpp_source to update the installation table-of-contents file for the resource.

In addition, updating the .toc for the lpp_source, the check operation also updates the simages attribute for the lpp_source, which indicates whether or not the lpp_source contains the images necessary to install the Base Operating System images on a machine.

To run the NIM check operation using Web-Based System Manager, perform the following steps:

1. In the NIM Resources container, select a target lpp_source resource.
2. From the selected menu, select **Check NIM state**.

To run the NIM check operation using smitty, perform the following steps:

1. To run the NIM check operation, enter the `smitty nim_res_op` fast path.
2. Select the lpp_source for the operation.
3. Select **check** for the operation to be performed.

To run the NIM check operation from the command line:

On the lpp_source, enter

```
# nim -o check lpp_sourceName
```

If the lpp_source is currently allocated to a client, use the Force option as follows:

```
# nim -F -o check lpp_sourceName
```

1.7.1.4 Verifying installation with the lppchk operation

You can use the lppchk operation to check the installed software. This is particularly useful when troubleshooting.

To verify installation with the lppchk operation using Web-Based System Manager, perform the following steps:

1. From the NIM container, select a target stand-alone machine, or, in the NIM Resources container, select a target SPOT.
2. From the Selected menu, select **Troubleshooting --> Verify Installed Software**.

3. Use the dialog box to select whether to verify all or some installed software on the selected machine or SPOT. If you want to verify file existence and length (fast check), initiate the verify action.

   If you want to perform another type of verification, click **Advanced**. Select the type of verification to perform and choose additional options as needed.

To verify installation with the lppchk operation using smitty, perform the following steps:

1. Enter the `smitty nim_mac_op` fast path to check the software on a machine, or enter `smitty nim_res_op` to check the software on a SPOT.
2. Select the target of the lppchk operation.
3. Select the desired verification mode.

To verify installation with the lppchk operation from the command line, enter:

```
# nim -o lppchk -a filesets=FilesetName \
-a lppchk_flags="lppchkFlags" ObjectName
```

where FilesetName is the name of a single fileset and ObjectName is the name of the machine or SPOT that is the target of the lppchk operation. Valid lppchk_flags are defined as follows:

-f	Fast check (file existence, file length)
-c	Checksum verification
-v	Fileset version consistency check
-l	File link verification.
-u	Update inventory (only valid with -c or -l).
-m n	Controls the detail of messages, n equals 1 to 3, where 3 is the most verbose.

Only one of the flags -f, -c, -v, or -l may be specified.

There is a limitation with the /usr SPOT in that it cannot be created on a machine that has been updated from one level of AIX to another if the level of BOS (bos.rte) on the installation media is different from the original level of BOS used to install the machine.

For example, if a machine has been updated from 4.3.1 to 4.3.2, the installation media for creating the /usr SPOT must contain AIX Version 4.3.1

base level images plus Version 4.3.2 updates. In a planning sense, you should check `lslpp -h` before creating an lpp_source.

Any changes made to an lpp_source should be reflected in its .toc. It is advisable to update or re-create a different SPOT that will reference the new lpp_source.

The same is true of the mksysb resource. The level of AIX installed on it should be identical to the level that is in the SPOT.

Software is managed using the cust and maint operations as well. Refer to Section 1.5.1, “Customization” on page 87, and Section 1.5.2, “Maintenance” on page 91, for further information.

Chapter 2. Working with NIM and advanced NIM topics

In this chapter, we will describe NIM topics that are a normal part of a NIM environment. We will describe how to clone either machines or only the root volume group. You will find information about how to use the firmware of a machine and how to boot a machine over ATM. There is also a lot of information about how to debug a NIM task and many hints and tips for troubleshooting. We will close this chapter with some case studies that may be helpful in some situations. You can apply these case studies to your own situation and, perhaps, find helpful hints and tips to solve your own problem.

2.1 Cloning an RS/6000 with an mksysb image

Cloning is a procedure that duplicates one previously-installed RS/6000 onto another, thereby, restoring the whole BOS and additional software. It is possible to clone various hardware types from a single mksysb. This powerful mechanism can be used to standardize and simplify your environment to a *Common Image*.

2.1.1 Prerequisites for cloning an RS/6000

The following are the prerequisites for cloning an RS/6000 system:

- There must be a configured NIM master. The resources, lpp_source and SPOT, must be available.
- The RS/6000 to be installed must be a NIM client.
- The mksysb must be on the hard disk of the NIM masters or a NIM resource server (any stand-alone NIM client). The mksysb NIM resource cannot be created during the installation procedure (for example, during a bos_inst operation), but the resource can be created while the mksysb image itself is being created or when there is already an mksysb image on hard disk.
- An mksysb with AIX Version 4.1.5 can only be used if it is located on the hard disk of the NIM master. Only NIM clients running AIX Version 4.2 or later can be used as source machines when creating mksysb images for resource.
- The resources, SPOT and lpp_source, must be at the same level for a BOS installation.

2.1.2 Cloning considerations

Cloning an RS/6000 means putting the BOS and other installed software from one hardware platform to another. The target system may not contain the same hardware, adapters, or devices. It may not require the same kernel because it may be a uniprocessor or multiprocessor machine. Moreover, it may not be the same hardware platform (rs6k, rspc, or chrp) as the source machine of the mksysb.

At the end of the NIM installation, TCP/IP is configured. We recommend that you allocate a bosinst_data resource for cloning with an mksysb. In this bosinst_data resource, you should set the field, RECOVER_DEVICES, to no. This will prevent the cloning process from restoring the devices as they were on the mksysb.

Many applications, particularly databases, maintain data in sparse files. A sparse file is a file with empty space in it for future use. Restoring these sparse files may cause problems when the file system is too small. We recommend that you not have sparse files in your mksysb or that you have enough space left in your file system.

If you are using a LUM-based compiler, such as C/C++ Version 3.6, C Version 4, or Fortran Version 5.1, we recommend that you check the Licence Use Management (LUM) configuration after cloning.

OpenGL or graPHIGS use special graphic adapter-specific filesets. If you have OpenGL or graPHIGS installed in your source machine and you will be using it on your target machine, you should check whether there are different adapter types. If there are, you must install these LPPs on your target system.

2.1.3 Cloning: Step by step

Use the steps described below to clone an RS/6000 system. It is possible to clone a machine by using the Web-Based System Manager, smitty, or the command line.

2.1.3.1 From Web-Based System Manager (WSM)

Perform the following steps to clone your system from WSM:

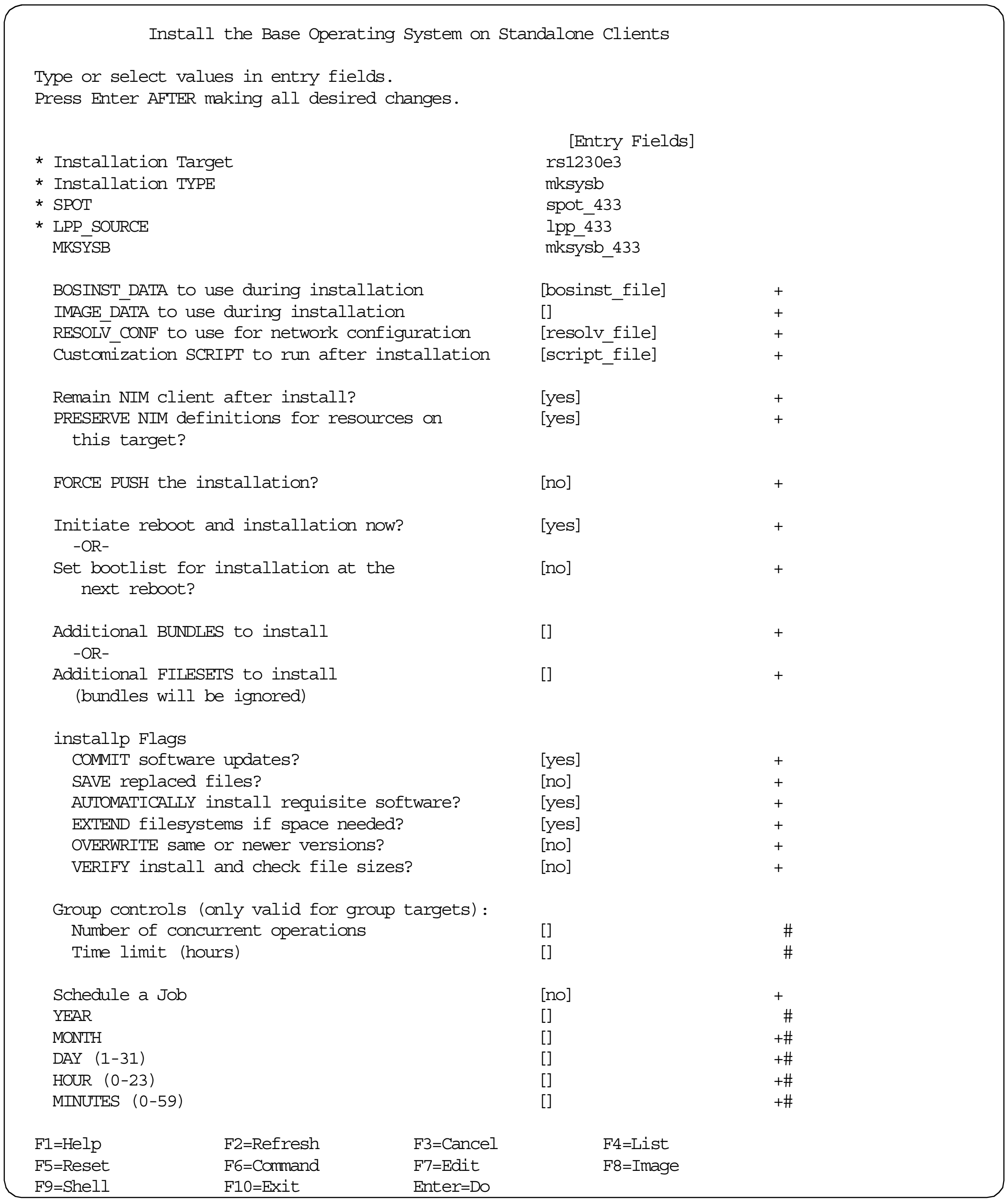
1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Select the machine that you want to install. If you cannot find the target machine in this list, the machine is not actually a NIM Client. To set up a NIM client, see Section 1.4.5.1, “Adding a NIM client to the NIM environment” on page 58.

4. Go to **Selected** --> **Install Base Operating System**.
5. Select the **Installation Type** --> **mksysb - Install from a mksysb**.
6. Select a system backup image from the list.
7. If you want to install a NIM client running AIX Version 4.1, you should allocate a cloning script. Therefore, you must select a script from NIM resources in the Optional selections frame, and click **OK**.
8. After the NIM client boots over the network, it will prompt you for information about how to set up the NIM client. When you have typed in the requested information, the machine will continue to install. To avoid prompting for information, see Section 1.2.3.4, “Non-prompted install” on page 18.

2.1.3.2 From smitty

Perform the following steps to clone your system from smitty:

1. Open smitty with the `nim_bosinst` fastpath.
2. Select the target machine that you want to install. If you cannot find the target machine in this list, the machine is not actually a NIM client. For information about how to set up a NIM client, see Section 1.4.5.1, “Adding a NIM client to the NIM environment” on page 58.
3. Select **mksysb** as the installation type.
4. Select the mksysb you want to use for the installation. To create an mksysb resource, see Section 1.2.3.3, “Mksysb resource” on page 16.
5. Select the SPOT you want to use for the installation.
6. Select the lpp_source you want to use for the installation. A dialog menu will appear as shown in Figure 10 on page 116. Fill in the correct values, or accept the default values.
7. If you want to clone an mksysb including AIX Version 4.1, you should allocate a cloning script at the Customization SCRIPT field to run after installation.
8. Run the smitty dialog to start the installation.
9. After the NIM client boots over the network, it will prompt you for information about how to set up the NIM client. After you have typed in the requested information, the machine will continue to install. To avoid prompting for information, see Section 1.2.3.4, “Non-prompted install” on page 18.



```
            Install the Base Operating System on Standalone Clients

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Installation Target                                 rs1230e3
* Installation TYPE                                   mksysb
* SPOT                                                spot_433
* LPP_SOURCE                                          lpp_433
  MKSYSB                                              mksysb_433

  BOSINST_DATA to use during installation            [bosinst_file]          +
  IMAGE_DATA to use during installation              []                      +
  RESOLV_CONF to use for network configuration       [resolv_file]           +
  Customization SCRIPT to run after installation     [script_file]           +

  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)

  installp Flags
    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]                    +

  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)                                     []                      +#

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 10. smitty screen of a bosinst operation type mksysb

2.1.3.3 From the command line

To clone a machine starting from the command line, only one command is necessary. The second step is only needed for cloning without non-prompted bosinst_data resource.

1. To start the installation, type in the following command:

   ```
   # nim -o bos_inst -a source=mksysb -a mksysb=mksysb_name \
   -a lpp_source=lpp_source_name -a spot=SPOT_name \
   -a boot_client=yes clientname
   ```

   If you want to clone a mksysb including AIX Version 4.1, you should allocate a cloning script as well; so, enter the following command:

   ```
   # nim -o bos_inst -a source=mksysb -a mksysb=mksysb_name \
   -a lpp_source=lpp_source_name -a spot=SPOT_name \
   -a script=script_name -a boot_client=yes clientname
   ```

2. After the NIM client boots over the network, it will prompt you for information about how to set up the NIM client. When you have typed in the requested information, the machine will continue to install. To avoid prompting for information, see Section 1.2.3.4, “Non-prompted install” on page 18.

2.2 Cloning scripts for AIX Version 4.1

For later AIX versions, such as AIX Version 4.2 or AIX Version 4.3, a cloning script is not needed.

This cloning script will change the boot kernel from the one located in the mksysb to the one needed for these machines. This is only necessary if you clone one hardware type to another, such as rs6k to rspc.

Save the cloning script on the NIM master. Do not forget to make it executable and to create a NIM resource for the cloning script. Use the resource type script.

You can find an example of a cloning script for AIX Version 4.1 in Appendix B, “Cloning script” on page 291.

2.3 Alternate disk installation (alt_disk_install)

With the alternate disk installation feature, you can install a second version of the BOS or clone the existing one on an alternative disk on the same machine. This installation can be done while the machine is running. With a simple reboot, you can switch from the old BOS to the new installed one. To switch back and forth, you must set the bootlist to the specific hdisk and then reboot. By default, the `alt_disk_install` command sets the bootlist to boot off the alternate rootvg. However, to go back to the original rootvg, one has to run the `bootlist` command. This mechanism can be very useful for testing updates without a long machine downtime.

There are two ways to perform an alternate disk installation: By installing a new mksysb image or tape or by cloning the existing root volume group onto another hard disk.

Installing a new mksysb image or tape
This method can only be used for mksysb images or mksysb tapes with AIX Version 4.3. The level of mksysb that should be installed must match the level of the bos.alt_disk_install.boot_images fileset.

Alternate Disk Installation is a good way to install a new BOS version on a running system without deleting the old one.

This mksysb can only be restored onto a hard disk that is currently not assigned to a volume group.

Cloning the existing root volume group onto another hard disk
Cloning the existing root volume group onto another hard disk means creating a backup copy of the root volume group. This backup copy can be used to install additional updates or to test modifications.

The existing root volume group can only be copied to a hard disk that is currently not assigned to a volume group. This option is only available in AIX Version 4.1.4 and higher.

2.3.1 Prerequisites for alternate disk installation on a NIM client

Before you can perform an alternate disk installation, you should ensure that the following prerequisites have been met:

- There must be a configured NIM master.
- The machine to be installed must be a NIM client and must be running.
- The fileset bos.alt_disk_install.rte must be installed on the NIM client.
- There must be at least one free hard disk that is not currently assigned to a volume group.

2.3.2 Alternate disk installation: Step by step

To perform the alternate disk installation, follow the steps described below. The progress of the installation can be shown with the `lsnim` command. Additionally a log is kept in the /var/adm/ras/nim.alt_disk_inst.log file on the target system including all progress messages and any error or debug messages that might be occur.

2.3.2.1 From Web-Based System Manager (WSM)

You can use the Web-Based System Manager to do an alternate disk installation.

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Select the machine you want to install. If you cannot find the target machine in this list, the machine is not actually a NIM client. For information on how to set up a NIM client, see Section 1.4.5.1, “Adding a NIM client to the NIM environment” on page 58.
4. Go to **Selected** --> **Alternate Disk Installation**.
5. Select **Clone the Rootvg to Alternate Disk** or **Install Mksysb on an Alternate Disk**.
6. Choose one or more hard disks as Target Disk(s).
7. There may be no hard disk left to choose because all hard disks are assigned to a volume group already. In that case, an Alternate Disk Installation is not possible. Add another hard disk to the system, and retry the operation.
8. In that dialog, you can change the settings for the installation in the **Optional selections** frame.
9. If you want to automatically reboot the machine from the new installed disk after installation, you have to check **Reboot when complete** in the Advanced Settings.

2.3.2.2 From smitty

You can also use smitty to perform an alternate disk installation:

1. Open smitty with the `nim_alt_mksysb` or `nim_alt_clone` fastpath on the NIM master.
2. Select the target machine that you want to install.
3. If you cannot find the target machine in this list, the machine is not actually a NIM client. For information on how to set up a NIM client, see Section 1.4.5.1, “Adding a NIM client to the NIM environment” on page 58.

4. Select the target hard disk(s).
5. There may be no hard disk left to choose because all hard disks are assigned to a volume group already. In that case, an Alternate Disk Installation is not possible. Add another hard disk to the system, and retry the operation.
6. If you have started smitty with nim_alt_mksysb, you have to choose a mksysb. A menu will appear as shown in Figure 11 on page 121.
7. Accept the default installation options, or type in different ones in the displayed dialog field.

```
                     Clone the rootvg to an Alternate Disk

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Target Machine / Group to Install                 [rs1230b]                 +
* Target Disk(s) to install                         [hdisk2]
  Phase to execute                                   all                      +
  IMAGE_DATA resource                               []                        +/
    EXCLUDE_FILES resource                          []                        +/
     (leave blank to include all files in backup)

  BUNDLE to install                                 []                        +
    -OR-
  Fileset(s) to install                             []

  FIX_BUNDLE to install                             []                        +
    -OR-
  FIXES to install                                  []

  LPP_SOURCE                                        []                        +
  (required if filesets, bundles or fixes used)

  installp Flags
    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                       +

  Customization SCRIPT resource                     []                        +/
  Set bootlist to boot from this disk
  on next reboot                                     yes                      +
  Reboot when complete?                              no                       +
  Verbose output?                                    no                       +
  Debug output?                                      no                       +

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

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 11. smitty screen of an alternate disk installation type clone rootvg

2.3.2.3 From the command line

When using the command line, you must select the correct command to do an alternate disk installation.

Installing an mksysb onto an alternate disk

To start the installation, enter the following:

```
# nim -o alt_disk_install -a source=mksysb -a mksysb=mksysb_name \
-a disk='diskname(s)' client name
```

Cloning the root volume group onto an alternate disk

To start the installation, enter the following:

```
# nim -o alt_disk_install -a source=rootvg \
-a disk='diskname(s)' client name
```

2.4 Firmware and ROS menus

The System Management Services (SMS) menu is used to manually boot the RS/6000 over the network. The SMS menu has many more functions, but here, we will describe only the NIM related one. To boot the machine over the network, you must perform the following steps:

1. Enter the SMS menu.
2. Set up the Remote Initial Program Load (RIPL) menu.
3. Perform a ping test.
4. Set the bootlist to the network device.

You will find more information in the next three sections about the individual steps described above.

2.4.1 System Management Services menu

The way to enter the System Management Services (SMS) menu depends on the hardware type of the machine to be booted over the network. There may be rspc-, rs6k-, or chrp-based hardware. Old rspc-based machines might need an SMS diskette, and old rs6k machines might need special IPL-ROM devices because the SMS code is not integrated into the firmware.

To determine which hardware type you have, do the following. For machines running AIX Version 4.2 or later, use the `bootinfo -p` command, and for machines running AIX Version 4.1, use the `bootinfo -T` command.

2.4.1.1 Power PC and Common Hardware Reference Platform

There are two different kind of SMS: Text-based SMS and the graphic-based SMS. If you have an ASCII terminal attached to your RS/6000, you must use the text-based SMS. The graphic-based SMS must be used if you have a graphic terminal attached to your RS/6000. To enter the SMS, you must press the **F1** key (for graphics-based) or the **1** key (for text-based) after the first icon appears and before the last icon appears during the startup or reboot of the machine. At this point, you might need an SMS diskette. If so, insert the SMS diskette and press **Enter**. After the SMS is started, you will see the SMS menu. The next step is to set up the RIPL.

2.4.1.2 Micro channel

First, you should know whether your machine needs an IPL ROM device or not; so, you should issue the `bootinfo -q NetworkDevices` command. If you use Token Ring, enter `bootinfo -q tok0`. If the machine does not need an IPL ROM, the return value of the `bootinfo -q` command will be 1.

If your machine requires an IPL ROM, insert the disk into the diskette drive, turn the key switch to Service, and turn the power on. The SMS menu will appear.

If your machine does not need an IPL ROM, turn the key switch to secure and power the machine on. Your machine will boot until the LED shows 200. Now, turn the key switch to service and quickly press the reset button. The SMS menu for uniprocessor and the Maintenance menu for multiprocessor will appear.

The next step is to set up the Remote Initial Program Load.

2.4.2 Remote Initial Program Load (RIPL)

In this menu, you should set the correct IP addresses of the NIM master, the NIM client (this machine), the Gateway, and the Subnet mask. Afterwards, you should perform a ping test whether your input and the network connection are working or not. If the Ping test was OK, set your boot device.

2.4.2.1 Micro channel uniprocessor machine

Perform the following steps to set up the RIPL menu.

1. Go to the Select BOOT (Startup) Device menu.
2. Select the network adapter, **device to BOOT**.
3. Type in the IP addresses of the NIM client into the client’s address field; enter the NIM master into the server address field, and enter the NIM clients gateway into the gateway address field.
4. When you are finished, save the IP addresses, and go back to the MAIN menu.
5. Do a ping test to verify the network connection to your master.

2.4.2.2 Micro channel multiprocessor machine

Perform the following steps to set up the RIPL menu:

1. Select **System Boot** --> **Boot from Network**. The MAIN MENU will appear.
2. Select the **Select BOOT (Startup) Device** menu.
3. Select the network adapter, **device to BOOT**.
4. Enter the IP addresses of the NIM client into the client address field; enter the NIM master into the server address field, and enter the NIM clients gateway into the gateway address field.
5. When finished, save the IP addresses and go back to the MAIN menu.
6. Do a ping test to verify the network connection to the master.

2.4.2.3 Power PC and Common Hardware Reference Platform

Type the information into the RIPL menu by performing the following steps:

1. Select **Utilities** --> **Remote Initial Program Load Setup**. If you are using an Enterprise server, the Remote Initial Program Load Setup is part of the MAIN menu.
2. Select **IP Parameters**. For Enterprise server, select **Set Address**.
3. Specify the IP addresses for the NIM client (this machine) into the client address field; specify the IP addresses for the NIM master into the server address field, and specify the IP addresses for the clients gateway address into the gateway address fields.
4. Select a network adapter.
5. For a ping test, select **Ping**. For Enterprise server, select **Ping Remote System**.

2.4.3 Changing the bootlist to network device

To start the installation over the network, the machine must be booted over the network. One exception is ATM. ATM does not boot over the network even though we support network install.

2.4.3.1 Micro channel

There is no network device to set specially for the network boot because it has been set during configuration of the RIPL menu.

2.4.3.2 Power PC and Common Hardware Reference Platform

Perform the following steps to change the bootlist to network device with a Power PC and Common Hardware Reference Platform.

1. From the SMS menu, select the **Select Boot Device** option. If you have an Enterprise server, from the SMS menu, go to **Multiboot** --> **Select boot Device** --> **Configure 1st Boot Device**.
2. Select the network adapter to be used for the network boot. Be sure to select the correct one.
3. Exit the SMS menu, and the machine starts booting over the network.

Note

For some hardware types, a network boot may fail because the machine has installed an older version of microcode. To cover a problem that could occur during network boot, you should install the latest available microcode (firmware) on the machine before initiating a network boot.

To get the newest microcode/firmware, visit the URL `http://www.rs6000.ibm.com/support/micro/download.html`

2.5 Managing resources

The commands for managing software on stand-alone clients and SPOT resources are generally the same. Specify the name of the machine, group, or SPOT as the target of the option.

If the SPOT is currently allocated on a NIM client, NIM prevents the change to the SPOT. You can the Force (-F) option to force the operation. Again, they should use caution. Using the -F flag when a resource is allocated by a NIM client could potentially corrupt the client and/or NIM DB.

Software updates to a SPOT can cause the SPOT’s network boot images to be rebuilt when necessary. If you think the boot images are bad, you can force them to be rebuilt using the NIM check operation.

Software updates to a SPOT may also cause software updates to occur in the root parts of diskless and dataless clients of the SPOT. This will occur automatically. You can force a synchronization using the NIM sync_roots operation on the SPOT.

To see what software is installed on a stand-alone client or SPOT using the Web-Based System Manager, perform the following steps:

1. Select a target machine or a target SPOT resource.
2. From the selected menu, select **List Installed Software --> All installed.**

To see what software is installed on a stand-alone client or SPOT using smitty, perform the following steps:

1. Enter the `smitty nim_list_installed` fast path.
2. Select the menu item that you want to perform the operation on.
3. Select a target for the operation.
4. In the displayed dialog fields, supply the required values.

To see what software is installed on a stand-alone client or SPOT from the command line, enter:

```
# nim -o lslpp [-a lslpp_flags=LslppFlags] TargetName
```

If you wish to list software updates that are installed on a stand-alone client or SPOT, we can use Web-Based System Manager as follows:

1. Select a target machine or a target SPOT resource.
2. From the selected menu, select **List Installed Software --> Fix (APAR) Status.**
3. Use the dialog to list the installation menus of specific installed fixes.

To see what fixes have been installed on a stand-alone client or SPOT using smitty, perform the following steps:

1. Enter the `smitty nim_mac_op` or `smitty nim_res_op` fast path.
2. Select the machine or SPOT resource object.
3. Select the fix_query operation.
4. Supply the required values to the fix_query flags. Specify the fix_bundle object name, or, to check the installation status of an APAR, specify the fix APAR numbers. If you leave both blank, all known fixes are displayed.

To see what fixes have been installed on a stand-alone client or SPOT from the command line, enter:

```
# nim -o fix_query[-a fixes="FixKeywords"][-a fix_bundle=BundleName] \
[-a fix_query=FixQueryFlags] TargetName
```

where `FixKeywords` are APAR numbers; `FixBundlename` is the object name of the `fix_bundle` resource; `FixQueryFlags` are optional flags to the `fix_query` operation, and `TargetName` is the client, group, or SPOT for which to display fix information.

Valid FixQueryFlags are defined as follows:

-a	Displays symptom text

-c	Displays output in colon-separated format
-F	Returns failure unless all filesets associated with a fix are installed
-q	Quiet option; if -q is specified, no heading is displayed
-v	Verbose option

2.5.1 Maintaining software in an lpp_source

To add or remove software in an lpp_source, simply add or remove the installation image from the lpp_source directory, and then initiate the NIM check operation on the lpp_source.

2.5.1.1 Copying Software to an lpp_source

The following are procedures for copying software to an lpp_source.

To copy software to an lpp_source using Web-Based System Manager, perform the following steps:

1. From the resources container, double-click the **lpp_source**.
2. From the General page, identify the location of the resource.
3. Select **Resources --> Copy Software to Directory**, and specify the location to the resource identified in the notebook as the destination directory.
4. After the copy is completed, select the **lpp_source**, highlight it, and, from the menu bar, select **Check NIM state**. This action updates the table of contents (.toc) file for the lpp_source.

To copy software to an lpp_source using smitty, perform the following steps:

1. To copy software from installation media to an lpp_source, insert the installation media in the appropriate drive of the lpp_source server.
2. To copy the software to the lpp_source directory, enter `smitty bffcreate` from the resource server.
3. Enter the INPUT device/directory for software.
4. In the displayed dialog fields, supply the correct values or accept the default values. Specify the lpp_source location for the directory to store the installation images. Use the help and the LIST option to help you.

To copy software to an lpp_source from the command line, perform the following steps:

1. Copy the software from the media to the lpp_source directory.

2. Perform the NIM check operation on the lpp_source by entering the following command:

```
# nim -o check Lpp_SourceName
```

2.5.1.2 Removing software from an lpp_source

To remove software from an lpp_source, delete the installation image from the lpp_source directory.

To remove software from an lpp_source from the command line, perform the following steps:

1. Remove the installation image from the lpp_source directory.
2. Perform the NIM check operation on the lpp_source by entering the following command:

```
# nim -o check lpp_sourceName
```

2.5.1.3 Running the NIM check operation

After adding or updating software, you must run the NIM check operation on the lpp_source to update the installation table-of-contents file for the resource.

In addition, updating the .toc for the lpp_source, the check operation also updates the simages attribute for the lpp_source, which indicates whether or not the lpp_source contains the images necessary to install the Base Operating System images on a machine.

To run the NIM check operation using Web-Based System Manager, perform the following steps:

1. In the NIM Resources container, select a target lpp_source resource.
2. From the selected menu, select **Check NIM state**.

To run the NIM check operation using smitty, perform the following steps:

1. To run the NIM check operation, enter the `smitty nim_res_op` fast path.
2. Select the lpp_source for the operation.
3. Select **check** for the operation to be performed.

From the command line, to initiate the NIM check operation on the lpp_source, enter:

```
# nim -o check lpp_sourceName
```

If the lpp_source is currently allocated to a client, use the Force option as follows:

```
# nim -F -o check lpp_sourceName
```

2.5.1.4 Verifying installation with the lppchk operation

You can use the lppchk operation to check the installed software. This is particularly useful when troubleshooting.

To verify installation with the lppchk operation using the Web-Based System Manager, perform the following steps:

1. From the NIM container, select a target stand-alone machine, or, in the NIM Resources container, select a target SPOT.
2. From the Selected menu, select **Troubleshooting --> Verify Installed Software**.
3. Use the dialog box to select whether to verify all or some installed software on the selected machine or SPOT. If you want to verify file existence and length (fast check), initiate the verify action.

   If you want to perform another type of verification, click **Advanced**. Select the type of verification to perform, and choose additional options as needed.

To verify installation with the lppchk operation using smitty, perform the following steps:

1. Enter the `smitty nim_mac_op` fast path to check software on a machine, or enter `smitty nim_res_op` to check software on a SPOT.
2. Select the target of the lppchk operation.
3. Select the desired verification mode.

To verify installation with the lppchk operation from the command line, enter:

```
# nim -o lppchk -a filesets=FilesetName \
-a lppchk_flags="lppchkFlags" ObjectName
```

where `FilesetName` is the name of a single fileset and `ObjectName` is the name of the machine or SPOT that is the target of the `lppchk` operation. Valid `lppchk_flags` are defined as follows:

-f	Fast check (file existence, file length).
-c	Checksum verification.
-v	Fileset version consistency check.

-l	File link verification.
-u	Update inventory (only valid with -c or -l).
--m n	Controls the detail of messages. n equals 1 to 3, where 3 is the most verbose.

Only one of the flags, -f, -c, -v, or -l, may be specified.

There is a limitation with the /usr SPOT in that it cannot be created on a machine that has been updated from one level of AIX to another if the level of BOS (bos.rte) on the installation media is different than the original level of BOS used to install the machine

For example, if a machine has been updated from 4.3.1 to 4.3.2, the installation media for creating the /usr SPOT must contain AIX Version 4.3.1 base level images plus Version 4.3.2 updates. In a planning sense, you should check `lslpp -h` before creating an lpp_source.

Any changes made to an lpp_source should be reflected in its .toc. It is advisable to re-create a different SPOT that will reference the new lpp_source.

The same is true of the mksysb resource. The level of AIX installed on it should be identical to the level that is in the SPOT.

Software is managed using the cust and maint operations as well. Refer to Section 1.5.1, “Customization” on page 87, and Section 1.5.2, “Maintenance” on page 91, for further information.

2.6 NIM over ATM

An ATM network adapter cannot be used to boot a machine directly over the network. To perform a BOS installation over ATM, special processing is needed.

Network boot with non-ATM adapters

Normally, the IPL-ROM or the firmware are configuring the network adapter when a machine is performing a network boot. Then, a boot image is transferred from the boot server to the client using tftp. After the transfer is completed, the boot image performs the additional steps of configuring the NIM client. The network installation resources are mounted, and the installation begins.

Network boot with ATM adapters

An ATM adapter cannot be configured by IPL-ROM or firmware, and, therefore, no boot image can be transferred from the NIM server to the NIM client. Thus, the only solution can be to copy the boot image to the NIM client's hard disk before the NIM client is rebooted. Some ODM information is also saved on the NIM client.

There are no executables installed on the NIM client to support the special processing required for installation over ATM. These requirements are the directories, /usr/lib/boot/bin/ and /usr/lpp/bos.sysmgt/nim/methods, which are mounted from the NIM master. These directories contain executables that are running during the setup performed by the NIM bos_inst operation.

When this preboot setup is complete, the machine is rebooted with an *at* job after one minute. During the boot, the machine is able to configure its ATM adapter and mount the resources from the master. From this point on, the ATM-attached NIM client performs a normal installation until the customization phase. During customization, the ATM adapter is not reconfigured with a `mktcpip` command.

2.6.1 Prerequisites for booting an ATM NIM client

If you want to boot a machine over ATM, make sure to fulfil the following prerequisites:

- The ATM-attached machine to be installed must be a NIM client and must be running.
- The ATM adapter, at0, will be used during the BOS installation.
- The NIM master fileset installed must be at least Version 4.3.0.0 with the update for ATM install or any superceding level.
- The NIM resource, SPOT, which will be used to install the ATM-attached NIM client, must be at least Version 4.3.0.0 with the update for ATM install or any superceding level.
- The following filesets must be installed on the NIM master to support the various platforms to be installed:

rs6k	devices.base.rte
rspc	devices.rspc.base.rte
chrp	devices.chrp.base.rte

2.6.2 Converting generic networks into ATM networks

With older versions of AIX (before 4.3), it was necessary to define an ATM network as a generic network because there was no support for ATM network in the NIM environment. Now, it is possible to change this generic network into an ATM network with the following command:

```
# nim -o change -a new_type=atm (network_name)
```

The adapter name for the client interface of the ATM network will automatically be set to at0 in the NIM database.

The name of the network can also be changed:

```
# nim -o change -a new_name=(new network name) (old network name)
```

2.6.3 System recovery after ATM boot failed

To boot an ATM-attached machine, it is necessary to change the boot image in this machine. If the BOS installation over ATM fails, the original boot image will be lost, and the machine will not be able to perform a normal boot. In this case, you must create a new boot image on the machine.

2.6.3.1 System recovery for non-rebooted machines

If the machine has not rebooted already, you should perform the following steps to prevent the machine from rebooting and to recover the boot image:

1. List the *at* jobs by entering the following command:

   ```
   # at -l
   ```

2. The output will look like the following:

   ```
   root.940949400.a Tue Oct 26 09:50:00 CDT 1999
   ```

   The first field of the output is the *at* job number. Remove the *at* job by entering the following command:

   ```
   # at -r (name of the job)
   ```

3. Run the following commands to recover the boot image:

   ```
   # bosboot -ad /dev/ipldevice
   # BLVDISK=$(lslv -l hd5 | grep hdisk | head -1 | cut -d' ' -f1)
   # bootlist -m normal $BLVDISK
   # sync
   # sync
   # sync
   ```

2.6.3.2 System recovery for already rebooted machines

If the machine has rebooted already, you must recover the boot image and change the bootlist. Perform the following steps.

1. Boot the machine from CD.
2. Select the system maintenance mode from the installation options menu.
3. Access the root volume group.
4. You will get a maintenance shell. Run the following commands:

```
# bosboot -ad /dev/ipldevice
# BLVDISK=$(lslv -l hd5 | grep hdisk | head -1 | cut -d' ' -f1)
# bootlist -m normal $BLVDISK
# sync
# sync
# sync
# reboot -q
```

2.7 NIM in a DHCP environment

Using both NIM and DHCP within the same environment should not generally cause any extra concern if the NIM Master and DHCP server are separate machines.

The only thing we must ensure within this environment is that our NIM clients use a directed boot; that is, when the client netboots, we should have the server IP address and the client IP address populated with the correct data (and a gateway address if appropriate). If we leave these addresses as zeros, that is, 0.0.0.0, the bootp packet will be broadcast over the network, and we risk the DHCP server picking up the bootp packet first and generating an incorrect response.

We only need to take special action if we decide to combine the role of a DHCP server and a NIM master in one box. In this situation, we must let DHCP take care of the bootp requests and responses and, so, must stop the bootp daemon from running so that the dhcpsd daemon can bind to the port.

We can stop bootp from running by editing the /etc/inetd.conf file and commenting out the following line:

```
bootps dgram   udp     wait     root     /usr/sbin/bootpd       bootpd
/etc/bootptab
```

Once we have made the change and saved the file, we need to stop and restart the inetd subsystem, which we can do with the following command:

```
# refresh -s inetd
```

Now that bootp is no longer running, we can start our DHCP server daemon (dhcpsd). This takes its configuration from the /etc/dhcpsd.cnf file. Figure 12 shows the example file we created in order to serve our network.

Our example network has our NIM Master/DHCP server at address 9.3.187.229 and serves hosts on an interconnecting network, 9.3.240.0. The hosts on the 9.3.240.0 network need to use the router at address 9.3.240.1 in order to reach the server. The default route is passed to the client using option 3; we also give the client the address of our name server with option 6.

```
numLogFiles             4
logFileSize             100
logFileName             /usr/tmp/dhcpsd.log
logItem                 SYSERR
logItem                 OBJERR
logItem                 PROTERR
logItem                 WARNING
logItem                 EVENT
logItem                 ACTION
logItem                 INFO
logItem                 ACNTING
logItem                 TRACE

leaseTimeDefault                30 minutes
leaseExpireInterval             5 minutes
supportBOOTP                    yes
supportUnlistedClients          yes

network 9.0.0.0 255.255.255.0
{
        subnet 9.3.240.0        9.3.240.51-9.3.240.55
        {
                option 1        255.255.255.0
                option 3        9.3.240.1
                option 6        9.3.240.2
        }
}
```

Figure 12. Example /etc/dhcpsd.cnf file

Now, we can start the dhcpsd daemon and check the status of our address pool as shown in Figure 13 on page 135.

```
# startsrc -s dhcpsd
# dadmin -s

PLEASE WAIT....Gathering Information From the Server....PLEASE WAIT


IP Address      Status  Lease Time Start Time  Last Leased Proxy ClientID

9.3.240.51      Free
9.3.240.52      Free
9.3.240.53      Free
9.3.240.54      Free
9.3.240.55      Free
```

Figure 13. Starting the DHCP server and examining the address pool

If we now go ahead and perform a bos_inst operation on one of our clients, NIM will add some lines into our dhcpsd.cnf file. It does this by generating the entry in the /etc/bootptab file as normal and then uses the `/usr/sbin/bootptodhcp` command to convert the client entry in the bootptab file into the corresponding entry in the dhcpsd.cnf file. In our example, the following lines were appended to the end of our /etc/dhcpsd.cnf file:

```
# BOOTP CLIENT: rs1230a.itsc.austin.ibm.com
client 6 10005ab174ec 9.3.240.51
{
        option 1 255.255.255.0
        option 3 9.3.240.1
        option sa 9.3.187.229
        option bf "/tftpboot/rs1230a.itsc.austin.ibm.com"
}
```

Because our IP addresses for the clients are not set anywhere, it is, obviously, important that any NIM reference to the client be done via its hostname. At the time that we perform the bos_inst operation, the name is resolved to an IP address using normal resolution methods. In order to not risk picking up an address that has already been allocated, we need to either use DDNS to keep a name server updated with the correct address information or assign a *temporary* hardcoded address we know will be free. We can even assign an address to a host outside the subnet range of the DHCP server. For example, in this case, we could use the address of 9.3.240.56 without a problem.

Once we netboot the client and it sends its bootp request to our server, the server passes back the necessary tftpboot file information, and the install continues normally from this point. The IP address that has been leased to the client will not expire because it’s on an infinite lease. We can confirm this with the `dadmin` command as shown in Figure 14.

```
# dadmin -s

PLEASE WAIT....Gathering Information From the Server....PLEASE WAIT


IP Address      Status  Lease Time Start Time  Last Leased Proxy ClientID

9.3.240.51      Leased  INFINITE   11/16 15:54 11/16 15:54 FALSE 6-10005ab174ec
9.3.240.52      Free
9.3.240.53      Free
9.3.240.54      Free
9.3.240.55      Free
#
```

Figure 14. Checking the infinite lease

Once the installation is complete, the extra lines are removed from the dhcpsd.cnf file, and the lease is released. As it stands, the client will be configured with the static address, 9.3.240.51. Because this is a DHCP environment, we should create a script file and allocate it as a resource that will configure the machine as a DHCP client so that it will be allocated an IP address dynamically.

2.8 Managing diskless/dataless station

A dataless or diskless NIM client can be added to the NIM environment by providing the required information to the NIM database on the NIM master. Diskless clients must mount all file systems from a remote server. Dataless clients must have paging space on the local hard disk. Dataless and diskless clients do not have a local boot image; so, this kind of NIM client must boot over the network.

Since the price of hard disks is no longer very high, we recommend that you not work with dataless or diskless NIM clients because these kinds of NIM clients depend on the network as well as on a resource server.

2.8.1 Adding a diskless or dataless client to the NIM environment

There are different ways of adding a diskless or dataless NIM client to your NIM environment. Use the information in this section to perform this task.

2.8.1.1 Prerequisites to add a diskless or dataless client

Ensure that the following prerequisites for adding a diskless or dataless machine to your NIM environment are fulfilled:

- The NIM master must be configured. To set up the NIM master, see Section 1.4.2, “Master setup” on page 42.
- The resources to support diskless and dataless NIM clients must be available.

2.8.1.2 From the Web-Based System Manager (WSM)

The following are the steps for adding a diskless or dataless NIM client to your NIM environment:

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Double-click on **Add New Machine**.
4. Enter the hostname of the client.
5. Follow the Task Guide to add the dataless or diskless NIM client to the NIM environment.

2.8.1.3 From smitty

To add the diskless or dataless client using smitty, perform the following steps.

1. Start smitty with the `nim_mkmac` fastpath.
2. Type in the hostname of the client.
3. Enter the requested information into the smitty menu, or accept the default values.

2.8.1.4 From the command line

On the command line, enter the following command on the NIM master:

```
# nim -o define -t diskless|dataless \
-a platform=PlatformType -a netboot_kernel=NetBootKernelType \
-a if1=InterfaceDescription -a net_definition=DefinitionName \
-a ring_speed=SpeedValue -a cable_type=TypeValue \
-a iplrom_emu=DeviceName MachineName
```

2.8.2 Initializing and booting a diskless or dataless client

After adding the client, you can initialize and boot the diskless or dataless client. In the following sections, you will find information about how to perform this task.

2.8.2.1 Prerequisites for booting a diskless or dataless client

Check the prerequisites before starting a boot of a dataless or diskless client.

- The NIM master must be configured. For information about how to set up a NIM master, see Section 1.4.2, “Master setup” on page 42.
- The resources to support diskless and dataless NIM clients must be available.
- The diskless or dataless NIM client must already exist in the NIM environment.

2.8.2.2 From Web-Based System Manager (WSM)

Use the Web-Based System Manager in the following way:

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Go to **Selected** --> **Initialize Machine Resources**.
4. Follow the Task Guide to specify or select the resources to use for initialization. You can specify the Home resource or the Shared Home resource, but not both. These resources are not required resources.
5. Perform a network boot on the client.
6. After the client boots, it will display a menu where you will type in the requested information.

2.8.2.3 From smitty

You can also use the smitty by performing the following steps:

1. Start smitty with the `nim_dd_init` fastpath.
2. Select the client from the list.
3. Fill in the requested information in the smitty menu, or accept the default values.
4. Perform a network boot on the client.
5. After the client boots, it will display a menu were you will type in the requested information.

2.8.2.4 From the command line

It is also possible to perform a boot of a dataless and diskless NIM client from the command line.

1. On the command line, type in the following commands on the NIM master:

   For a diskless client:

```
# nim -o dkls_init -a spot=SpotName \
-a root=RootName -a dump=DumpName \
-a paging=PagingName ClientName
```

For a dataless client:

```
# nim -o dtls_init -a spot=SpotName \
-a root=RootName -a dump=DumpName ClientName
```

2. Perform a network boot on the client.
3. After the client boots, it will display a menu in which you will type the requested information.

2.8.3 Uninitializing a diskless or dataless client

Diskless and dataless NIM clients can be uninitialized by performing the reset operation. With this operation, only the boot image will be deallocated. This action also provides the option of deallocating all resources for the machine. Deallocating all resources removes all root data from the machine as well.

2.8.3.1 From the Web-Based System Manager (WSM)

Use these steps to uninitialize a dataless or diskless client:

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Select the NIM client to uninitialize.
4. Go to **Selected** --> **Uninitialize Machine Resources**.
5. Use the dialog to uninitialize, and, if desired, deallocate the resources for the client.

2.8.3.2 From smitty

smitty can also be used to uninitialize a dataless or diskless client.

1. Start smitty with the `nim_dd_uninit` fastpath.
2. Select the client from the list.
3. If you want to remove the root data, change the filed DEALLOCATE to yes.

2.8.3.3 From the command line

To uninitialize the client, enter the following command on the NIM master:

```
# nim -F -o reset ClientName
```

To deallocate all the resources of the client, enter the following command on the NIM master:

```
# nim -o deallocate -a subclass=all ClientName
```

2.9 NIM name resolution

NIM added functionality to use the NIM clients if1 attribute value for the hostname instead of /etc/hosts or DNS. If *-a use_clients_if_host* is set to yes, it will look at the second value for the hostname.

Basically, NIM is using the standard name resolution techniques from AIX. Whatever order is specified in the network environment to resolve the hostnames, NIM will use the same order to resolve hostnames.

For example, if the network configuration of the machine is set up to resolve hostnames by using NIS, then BIND/DNS and, at least, the local /etc/hosts file, NIM, will also use that order to resolve hostnames.

This order is defined in the /etc/netsvc.conf file, and all processes and applications on this machine depend on this order. To set up a different order only for testing or to perform a special NIM command, it is possible to set a different order by setting the NSORDER environment variable to another value. For example, to use only the local resolution functions, you can set the variable to NSORDER=local.

There may be problems when NIM master and NIM clients are using different orders to resolve hostnames. This may occur when a service is available to one machine but not to another machine. Mixing up the case-sensitive BIND/DNS and the case-insensitive NIS may also result in problems.

In order to avoid problems, we recommend that you always use the same hostname resolving mechanism on all machines that belong to the NIM environment.

2.10 NIM troubleshooting

This section describes the NIM-relevant LED codes and suggests solutions for possible NIM problems. Furthermore, we will describe how to view NIM logs.

2.10.1 NIM relevant LED codes

The progress of a NIM operation is displayed on the LED display of the machine. If a problem occurs within that process, the machine will display a problem-specific LED code. See Table 19 on page 141 for the meaning of the LED codes.

If your NIM client does not have an LED display, you can get the LED information by issuing the `lsnim -l NimClientName` command on the NIM master. The Web-Based System Manager (WSM) will also display the value of the LED while a NIM process is in progress.

Table 19. Overview of NIM relevant LED codes

LED	Description
299	Boot image successfully received at the NIM client.
600	Starting network boot (portion of /sbin/rc.boot).
602	Configuring network parent devices.
603	Script defsys, cfgsys, or cfgbus located in /usr/lib/methods/ failed.
604	Configuring physical network boot device.
605	Configuration physical network boot device failed.
606	Running /usr/sbin/ifconfig on logical network boot devices.
607	/usr/sbin/ifconfig failed.
608	Attempting to retrieve the client.info file with tftp from the SPOT server. A flashing 608 indicates multiple attempts to retrieve the client.info file are occurring.
609	The client.info file does not exist or could not be accessed, or it is zero length.
610	Attempting to mount a remote file system using NFS.
611	The client is unable to mount a remote file system (NIM resource) using NFS.
612	Accessing remote files. Unconfiguring network boot devices.
613	The `route` command failed.
614	Configuration of logical paging devices.
615	Configuration of logical paging device failed.
616	Converting from diskless to dataless configuration.
617	Diskless to dataless configuration failed.
618	Configuring remote (NFS) paging device.
619	Configuration of remote (NFS) paging space failed.
620	Updating special device files and ODM in permanent file system.
622	Control returned to the /sbin/rc.boot program.

LED	Description
623	The BOS installation program has encountered a fatal error.
624	Control passed to the BOS installation Program.

2.10.2 Debugging a network boot problem

Usually, all errors encountered during NIM network boot processing can be split into three main parts:

1. Establishing network communication between NIM master and NIM client
2. Obtaining the boot image from the server
3. Running the boot image on the client

The following problem determination step descriptions may be helpful for finding the failure during each part.

2.10.2.1 Establishing communication between client and server

Perform the following steps to determine the problem:

1. Perform a ping test from the remote initial program load (RIPL) menu before starting the network boot process. If the bootp menu is not accessible during a manually-started network boot either from the IPL-ROM or the Firmware on a PCI-based client, new or bad attached devices may be the cause of the problem. If the ping test is OK, continue testing with the steps described in Section 2.10.2.2, “Boot image transfer to client” on page 143.
2. If the ping test fails, verify that all IP addresses in the RIPL menu are specified correctly.
3. If the IP addresses are correct, try to ping the NIM master from another machine in the client’s subnet.
4. If the NIM master can be pinged from another machine in the same subnet, the network adapter of the client may be damaged. If the client is an MCA-based and bootp-enabled machine, try to perform the network boot by using an IPL ROM device.
5. If the NIM master cannot be pinged from a machine in the NIM client’s subnet but rather from a machine in the NIM master’s subnet, there may be routing problems between the NIM master and the NIM client.
6. If the NIM master cannot be pinged from the other machine in the NIM master’s subnet, the network communication on the NIM master may be faulty. Perform network debugging procedures to determine the problem.

You can find help to do this in Section 2.11.5, “Producing debug output by using an IP-trace” on page 162.

2.10.2.2 Boot image transfer to client

If the ping test is successful, perform the following steps to determine the network boot problem:

1. If bootp packets are sent but not received, the boot server may not be responding to the bootp request.
2. View the /etc/bootptab file on the boot server. It should contain an entry for the client machine including the following information:

   ```
   NimClientName
   bf=BootFileName
   ip=ClientIPAddress
   ha=HardwareAddress
   sa=BootServerAddress
   sm=SubnetMask
   gw=GatewayAddress
   ```

   If this entry does not exist, the NIM command to set up the current operation failed, or the machine was reset before the boot operation could occur.

   If the entry exists, verify that the specified data is correct. If a field contains incorrect data, the information that was used to define the client machine or network was wrong or has changed since the creation of that NIM resources. In that case, reset the client, correct the incorrect data, retry the NIM operation, and boot the client machine.
3. If the data in the /etc/bootptab file are correct, verify that the inetd daemon is running. Use the following command:

   ```
   # lssrc -s inetd
   ```

   If the daemon is not running, the status of the output line will be inoperative as shown in the following:

   ```
         Subsystem     Group   PID     Status
           inetd       tcpip   30136   inoperative
   ```

   If this is the case, start the inetd daemon, and retry the network boot of the NIM client. To start the inetd daemon, issue the following command:

   ```
   # startsrc -s inetd
   ```

If the inetd daemon is running, it should automatically start the bootpd daemon when the first bootp request is received at the server. The output of the `lssrc` command should look like the following:

```
Subsystem    Group   PID     Status
inetd        tcpip   30136   active
```

4. If the bootpd daemon is not started, check whether the entry for the bootpd daemon in the file /etc/inetd.conf is commented or not. If it is commented, uncomment the entry, restart the inetd daemon with the `refresh -s inetd` command, and try to boot the client over the network again.
5. If the bootp reply is still not received at the NIM client, you should start the bootpd daemon in debug mode. Therefore, perform the following steps (see Section 2.11.1, “Producing debug output from the bootp procedure” on page 155):
   a. In the /etc/inetd.conf file on the server, you should comment the entry for the bootps.
   b. Restart the inetd daemon with the `refresh -s inetd` command.
   c. Create an output file for the debug information.
   d. Start the bootpd manually from the command line with the following command:

   ```
   # /usr/sbin/bootpd -s -d -d -d
   ```

   The three -ds after the `bootpd` command are not a typo. They tell the command how detailed the output should be.
6. Perform a network boot from the NIM client again. If no debug output is displayed, the bootp request from the NIM client is not reaching the NIM master. In that case, verify that the addresses specified in the NIM clients bootp menus are correct. If these addresses are correct, you should perform a network debugging procedure to determine the problem.
7. If the bootp request was received from the server, debug output containing the clients information from the /etc/bootptab file is displayed. Because this information is sent back to the client in the bootp replay, you should verify that it is correct.
8. If the bootp replay is not reaching the client, perform network debugging procedures to determine the problem.
9. After the client has received the bootp replay from the server, it will download the boot image from the server using tftp.
10. The number of received tftp packets will be shown on the client’s screen.

11. The boot image download has successfully ended if the client machine shows the LED 299.
12. If the boot image was not successfully transferred to the client with tftp, the clients definitions on the NIM master may be wrong, and the client tries to get the wrong boot image. Verify that the information for the kernel type and the hardware platform of that client are correct in the NIM master's database. If not, correct the information, perform a reset on the NIM client, rerun the NIM operation, and do a network boot of the NIM client again.
13. Verify that in the /tftpboot directory of the server is a link with the name of the NIM client and that this link points to the correct boot image for this client. If not, reset the NIM client, rerun the NIM operation, and perform a network boot on the client.
14. If the link in the /tftpboot directory is pointing to the correct boot image but the tftp of the boot image does not complete successfully, this boot image may be corrupted. To cover this, you should re-create the boot image by using the `nim -o check -F SpotName` command.
15. If the NIM client is not an rs6k platform machine, verify that the NIM client has installed the latest version of the firmware.

    It may also be useful to put additional debug code into the boot script files. Therefore, put the `showled` command into the file rc.boot to isolate the problem. In the rc.bos_inst file, a command can be added that updates the info or err_info filed in the NIM master's database for this NIM client. Therefore, enter the following into the rc.bos_inst file:

    ```
    /../SpotName/usr/sbin/nimclient -o change -a force=yes \
     -a ignore_lock=yes \
     -a info="Debug Step 1: begin mounting NIM resource"
    ```

    This additional debug information would be shown with the `lsnim -a info ClientName` command like this:

    ```
    # lsnim -a info client1

    client1:

       info = Debug Step 1: begin mounting NIM resource
    ```

    These files should be modified in the SPOT resource, which will be used to boot the client.

    Note

    Adding debug code to the rc.boot file will take effect after recreating the boot image. Changing the rc.bos_inst file does not require a rebuild of the boot image.

2.10.2.3 Running the boot image on the client

If the boot image is successfully transferred to the client, the machine tries to run the boot image. During this procedure, the most common errors are hangs with the LED codes 605, 608, 611, or 613. When using machines without an LED display, you should use a debug-enabled boot image to get more information about the problem. The most common errors are described below.

Hang with the LED 605

This LED code indicates that the NIM client is unable to identify the boot devices.

At this point in the NIM procedure, the client has successfully received the boot image and booted into the BOS install mode. The next step, which failed in that procedure, was to configure the primary network adapter to perform the network boot. This failed step caused the hang with LED code 605. To solve this problem, perform the following steps:

1. Make sure that the device driver to support the specified network device is in the SPOT resource.

   If it is not, deallocate the resources, install the device driver into the lpp_source, and re-create the SPOT resource. Allocate the new resources and rerun the NIM operation.

   We recommend that you install all available additional device drivers into the lpp_source to avoid any problems caused by missing device drivers.

2. If you are using two different network types connected by a bridge, this may also cause the problem. The NIM master will define the network type of the NIM client like the one the NIM master is using. So, if the NIM client is on a different physical (but the same logical) network, its network type will be set wrong. In that case, you must set the network type on the command line manually. Therefore, see Section 2.13.6, “Different physical networks on the same logical subnet” on page 198.

Hang with the LED 608

This LED code indicates that the NIM client was not able to get the ClientName.info file from the SPOT server.

At this point in the NIM procedure, the client has successfully received the boot image, booted into the BOS install mode, and configured the primary network adapter. The next step, which failed in that procedure, is to get the ClientName.info file from the SPOT server. This failed step caused the hang with the LED code 608. The machine will stay in this loop until the tftp process gets the CilentName.info file successfully. The flushing 608 indicates that the client tries to get that file again and again. At this point, you can leave the client and work on the problem by performing the following steps:

1. Check the permissions of the file, ClientName.info, in the /tftpboot directory of the SPOT server. If you are not sure which machine the SPOT server is, you can find out by entering the `# lsnim -a server SpotName` command at the command line of the NIM master. Check the permissions and the owner/group of that file. The permissions should be 644, and the owner/group should be root/system. Change the permissions and owner/group if it is not correct. If the file does not exist, reset the NIM client, and try the `NIM` command again.

   If the ClientName.info file was still not created, use local problem reporting procedures.

2. If the ClientName.info file is available, make sure that the file can be downloaded from the SPOT server using tftp onto another machine in the NIM client's subnet. For this test, run

   ```
   # tftp -g /tmp/ClientName.info ServerName \
   /tftpboot/ClientName.info image
   ```

   on the command line of the machine in the clients subnet. The output should look like the following:

   ```
   Received 1120 bytes in 0.1 seconds
   ```

   If the test was not successful, you should verify that this directory is not restricted in the /etc/tftpaccess.ctl file. The entry for the /tftpboot directory should like this: `allow:/tftpboot`

3. It may be that the network adapter was not configured properly when the boot image was run.

4. If the NIM client is not an rs6k-platform machine, make sure that it has the latest version of the firmware installed.

Hang with the LED 611

This LED code indicates that the NIM client was not able to mount the allocated resources using NFS from the server.

At this point in the NIM procedure, the client has successfully received the boot image, booted into the BOS install mode, configured the primary network adapter, downloaded the ClientName.info file by using tftp, exported all NIM environment variables, and set up any routing. The next step, which failed in that procedure, was to mount the allocated resources from the server using NFS. If that is not possible, the machine will hang with the LED code 611. One or more of the following reasons may cause the hang:

1. Make sure that the filesets bos.net.tcp.client and bos.net.nfs.client are in the committed or applied state. To check this, issue the

   ```
   # lslpp -l | grep net | grep client
   ```

   command on the command line on the master. Check the fix levels of these filesets as well. The fix levels should be the same on both filesets or, if not, even very close to each other. This means that if there is a big gap between these two fileset levels, you should update the older one to a newer version. If you are using Version 4.1, make sure the fileset levels are not between 4.1.4.8 and 4.1.4.13.

2. If you have the same AIX release level on the NIM master as in the SPOT resource you want to install, you should check these fileset levels also. Consequently, use the

   ```
   # nim -o lslpp -a lslpp_flags=-l SpotName | grep net | grep client
   ```

   command on the NIM masters command line. Make sure that the fix levels of the filesets in the SPOT and the filesets on the NIM master are very close or the same. By very close, we mean that the gap between the fix level should not be more than one or two fix levels. If you are using AIX 4.1, make sure that the fileset levels are not between 4.1.4.8 and 4.1.4.13.

   If you are using an incorrect file set level in the SPOT, you need to match the level that is installed on the NIM master. If you have to down level the fileset, you should go back to the base fileset using the command

   ```
   # nim -o cust -a lpp_source=lpp_sourceName \
   -a installp_flags=acFd \
   -a filesets="fileset level fileset level ..." SpotName
   ```

   with an update to the correct level.

   To upgrade the fileset or filesets in a SPOT resource, you must have these fileset(s) in an lpp_source. We recommend that you not use the lpp_source including the base filesets. Create a new lpp_source for the fixes. For information about how to create an lpp_source, see Section 1.4.6.1, “Defining the lpp_source resource” on page 67.

   If the fixes are already in an lpp_source, use the cust operation to upgrade the filesets:

   ```
   # nim -o cust -a lpp_source=lpp_sourceName \
   -a installp_flags=acgXd \
   -a filesets="fileset level fileset level ..." SpotName
   ```

3. Make sure that NFS is running on the server. Check this with the `# lssrc -s nfsd` command. The output should look like the following:

```
Subsystem         Group         PID         Status
nfsd              nfs           11108       active
```

4. Verify that all requested resources are exported properly. Use the following command to find out the exported resources:

   ```
   # lsnim -Fl ClientName | grep exported
   ```

   The output should look like the following:

   ```
   exported= /export/lpp_source/lpp_aix433

   exported= /export/spot/spot_433/usr
   ```

   You should see a line for each resource you have allocated for the NIM operation. If there is an exported resource missing, you have to deallocate the resources and allocate the resources again. Make sure that you are not trying to export a child directory of an already exported parent directory or to export a parent directory of an already exported child directory. Therefore, check the entries in the file /etc/exports if there is an actually exported directory that can cause the problem. Also, check the /etc/xtab file.

   If there is an error in the file /etc/exports, correct it, and issue the following commands:

   ```
   # exportfs -ua

   # exportfs -va
   ```

   Check the permissions of these files as well. The permissions should be 644 and the owner/group should be root/system.

   Check the exported resources again. If there are still allocated resources missing, you should perform a global export of all resources. Therefore, deallocate all resources, and put the directories into the file /etc/exports without any options. Run the `exportfs -ua` and `exportfs -va` commands.

5. General software inconsistencies in the used SPOT may also cause the hang. Create a new SPOT resource temporarily, and try the NIM operation again.

   If you are using an lpp_source including the newest fixes and a SPOT resource created by using this lpp_source, you should create a new lpp_source temporarily without the newest fixes. Create a new SPOT resource as well, and try the NIM operation with these new resources again.

6. Incorrect routing may cause the hang. Perform an IP-trace described in Section 2.11.5, “Producing debug output by using an IP-trace” on page 162, to find the problem.
7. Bad hardware may also cause the problem. Perform a hardware test to find the problem.
8. If there is still a problem, you should perform a NIM debug described in Section 2.11, “NIM debugging and network debugging” on page 154, to find the problem.

Hang with the LED 613

This LED code indicates that the A route is incorrectly defined for a network in the NIM database.

At this point in the NIM procedure, the client has successfully received the boot image, booted into the BOS install mode, configured the primary network adapter, and received the client.info file from the SPOT server. The next step, which failed, was to generate the /etc/hosts file and configure the IP route. This failed step caused the LED code 613.

To solve this problem do the following:

1. Make sure that the routing information in the NIM database for that NIM clients NIM network is correct. Use the command

   ```
   # lsnim -l NetName
   ```

   to get the information from the NIM database. If the information is not correct, correct it, reset the NIM clients state and retry the NIM operation again.
2. If the info for the routes is correct, make sure that all gateways between the SPOT server and the NIM client are functional.

   Verify that the correct gateways are set between the networks and that all gateways are functional. To find out which route could not be defined use a debug enabled network boot image.

Hang with other LED codes

It may be possible, that other LED hangs occur. To find out the main problem which causes the hang, see Table 19 on page 141.

2.10.3 Viewing NIM logs

Many NIM operations are writing log information in NIM logs during the execution of the operation. This information can be reviewed after the NIM operation has finished. Use the showlog operation to view the NIM logs.

There are several log types, see Table 20 for the valid attributes you can use to specify which kind of log you want to see by using the showlog operation. An example to show the NIM log containing the boot log information for the NIM client rs1230e3 is

```
# nim -o showlog -a log_type=boot rs1230e3
```

To view the other kind of logs, you must specify a another value to the log_type attribute.

Table 20. Different kind of NIM logs

Valid attribute for showing NIM logs	Description
devinst	Output from the installation of key system and device-driver software.
niminst	Output from the installation of user-sepecific software (including installation of NIM client software during a bos_inst operation).
bosinst	Output from the BOS installation program.
boot	The machine's boot log.
lppchk	A log of the output from the lppchk operation executed on a stand-alone NIM client. (AIX 4.2 and later)
script	Output from any configuration script resources allocated for a bos_inst operation. (AIX 4.2 and later)
nimerr	Errors encountered during execution of the nim command.
alt_disk_install	Output from the alternate disk installation command. (AIX 4.3 and later)

You can view the NIM logs by using different ways. By default, the showlog operation without specifying the log_type applied to a NIM client machine shows the output of the niminst log from the last software which was installed. The last output entry is shown from the script and lppchk logs by default. You can view the entry contents of the niminst, script, and lppchk log by assigning the full_log attribute a value of yes when executing the showlog command. For all other log types, the entry log is shown by default.

From Web-Based System Manager:

Start the view from the Web-based system Manager like the following steps.

1. Start the Web-Based System Manager.
2. Open the NIM container.

3. Select a machine or got to **Resources** and select a SPOT resource.
4. Go to **Selected** --> **Troubleshooting** --> **Show NIM logs**.
5. Select the logs you want to view.
6. If you want to see more than the last entry, you must uncheck the box. Only show the last entry in the log when applicable in the bottom of the menu.

From smitty:
Use the smitty as described in the following steps to view the NIM logs.

1. Start smitty with the `nim_mac_op` fastpath for viewing machines log data or with the `nim_res_op` fastpath for viewing SPOT log data.
2. Select a target machine, or SPOT.
3. Select the showlog operation.
4. Select a the log type.
5. If you want to see more than the last entry, change the Only Show The Last Entry in Log to no.

From the command line:
Use the following command to view the last entry in a NIM log:

```
# nim -o showlog -a log_type=log_type_value NIMobjectName
```

To show the hole NIM log add the attribute -a full_log=yes to the command:

```
# nim -o showlog -a log_type=log_type_value \
  -a full_log=yes NIMobjectName
```

A example of a bosinst log is shown in Figure 15 on page 153. You can review the BOS installation process for this single NIM client. If there was a problem during the BOS installation, you may can see it in that bosinst log.

```
# nim -o showlog -a log_type=bosinst rs1230e3
Preparing target disks.
rootvg
Making boot logical volume.
hd5
Making paging logical volumes.
hd6
Making logical volumes.
hd8
hd4
hd2
hd9var
hd3
hd1
Forming the jfs log.
Making file systems.
Mounting file systems.
Restoring base operating system.
Initializing disk environment.

Over mounting /.
rs1230e3
Copying Cu* to disk.
Installing additional software.
rs1230e3
lft0 changed
Initializing dump device.
primary             /dev/hd6
secondary           /dev/sysdumpnull
copy directory      /var/adm/ras
forced copy flag    TRUE
always allow dump   FALSE
dump compression    OFF
Network Install Manager customization.
Creating boot image.

bosboot: Boot image is 6674 512 byte blocks.

                               Running Customization

        Please wait...
#
```

Figure 15. Entries of the bosinst log shown with the showlog command

Single steps of a boot log are shown in Figure 16 on page 154. The shown single steps are the configuration of the ethernet and the token-ring network adapter and the output of the final steps from the boot process.

```
# nim -o showlog -a log_type=boot rs1230e3
...
----------------
Time: 11LEDS: 0x539
Number of running methods: 0
----------------
attempting to configure device 'ent0'
Time: 11LEDS: 0x742
invoking /usr/lib/methods/cfgkent -2 -l ent0
Number of running methods: 1
----------------
Completed method for: ent0, Elapsed time = 1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------
...
----------------
Time: 12LEDS: 0x539
Number of running methods: 0
----------------
attempting to configure device 'tok0'
Time: 12LEDS: 0x750
invoking /usr/lib/methods/cfgstok -2 -l tok0
Number of running methods: 1
----------------
Completed method for: tok0, Elapsed time = 9
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------
...
devices.pci.86802912
Configuration time: 26 seconds
Starting AIX Windows Desktop..........
Saving Base Customize Data to boot disk
Starting the sync daemon
Starting the error daemon
System initialization completed.
Starting Multi-user Initialization
 Performing auto-varyon of Volume Groups
 Activating all paging spaces
swapon: Paging device /dev/hd6 activated.
/dev/rhd1 (/home): ** Unmounted cleanly - Check suppressed
 Performing all automatic mounts
Multi-user initialization completed
#
```

Figure 16. Single entries of the boot log shown with the showlog command

2.11 NIM debugging and network debugging

Problems may occur when a NIM client is booted/installed. To cover these problems and to find their causes, it is very helpful to produce debug information. There are different kinds of debug information. A Bootp debug may be helpful if the boot image could not be transferred to the NIM client without failures. If the NIM client is not able to boot with the boot image, it may be helpful to produce a debug output from the network boot image. When the NIM client has successfully booted over the network but the install process stops with a failure, it may be helpful to produce a debug output from the BOS install program.

2.11.1 Producing debug output from the bootp procedure

In order to produce a bootp debug, start the bootpd daemon in debug mode. Perform the following steps:

1. Perform a NIM install on a NIM client without booting the NIM client automatically.
2. Stop all currently running bootp daemons:
   a. Comment the filed bootps in the file /etc/inetd.conf.
   b. Refresh the inetd daemon with the command
   ```
   # refresh -s inetd
   ```
3. Put the debug output in file, for example, /tmp/bootpd.log:
   a. Add the line
   ```
   *.debug/tmp/bootpd.log
   ```
   into the file /etc/syslog.conf.
   b. Create the file with
   ```
   # touch /tmp/bootpd.log
   ```
   c. Stop and then start the syslog daemon with
   ```
   # stopsrc -s syslogd
   # startsrc -s syslogd
   ```
4. Start the bootpd daemon manually from the command line with the following command:
   ```
   # /usr/sbin/bootpd -s -d -d -d
   ```
5. Start the NIM client, and perform a network boot.

The bootp daemon is now running in debug mode, and all output will be written into the file /tmp/bootpd.log. You can afterwards analyze this file to resolve bootp problems or watch what happens during the bootp process.

The output of a bootp debug looks like that shown in Figure 17 on page 156. In this case, no bootp error occurred.

```
# tail -f /tmp/bootpd.log
...
Nov  1 15:03:28 rs1400a bootpd[17474]: bootptab mtime is Fri Oct 29 14:54:51 1999
Nov  1 15:03:28 rs1400a bootpd[17474]: Received boot request.
Nov  1 15:03:28 rs1400a bootpd[17474]: request from IP addr 1.1.2.2
Nov  1 15:03:28 rs1400a bootpd[17474]: found 1.1.2.2 rs1230e3
Nov  1 15:03:28 rs1400a bootpd[17474]: bootfile = /tftpboot/rs1230e3
Nov  1 15:03:28 rs1400a bootpd[17474]: vendor magic field is 0.0.0.0
Nov  1 15:03:28 rs1400a bootpd[17474]: RFC1048 vendor data ( bp_vend[64] )
Nov  1 15:03:28 rs1400a bootpd[17474]: sending RFC1048-style reply
Nov  1 15:03:28 rs1400a bootpd[17474]: The following addresses are included in the bootp reply
Nov  1 15:03:28 rs1400a bootpd[17474]: Client IP address (bp->bp_ciaddr) = 1.1.2.2
Nov  1 15:03:28 rs1400a bootpd[17474]: Server IP address (bp->bp_siaddr) = 9.3.187.229
Nov  1 15:03:28 rs1400a bootpd[17474]: Gateway IP address (bp->bp_giaddr) = 1.1.2.1
Nov  1 15:03:28 rs1400a bootpd[17474]: Finished processing boot request.
Nov  1 15:03:28 rs1400a bootpd[17474]: bootptab mtime is Fri Oct 29 14:54:51 1999
#
```

Figure 17. Output from a bootp debug without errors

Examples for bootp errors are shown below. The numbers in front of each line are normally not shown. We have added these lines into the graphic to be able to explain what happened.

If the NIM clients IP address is wrong in the NIM database, the entry in the file /etc/bootptab will be created wrong too. As shown in the graphic Figure 18 on page 157, you can see what happens, when the IP address of the NIM client is not correct.

- line 6: The NIM clients bootp request is received from the NIM master.
- line 7: The NIM master tries to resolve the hostname of the received IP address by using the name resolution mechanism set in the network environment.
- line 8: If the IP address is wrong, the NIM master can not find the hostname and generates an error message.
- line 9: The NIM master stops processing the bootp request from the client. The waiting NIM client will get no response from the NIM master.
- line 11: After a moment, the NIM client sent a second bootp request to the NIM master. This request will also not be processed because of the wrong IP address in the /etc/bootptab file.

The NIM client will send a couple of bootp requests to the NIM master. If the NIM master will not process one these requests, the NIM client hangs.

```
# tail -f /tmp/bootpd.log
...
1) Nov  2 14:44:22 rs1400a bootpd[23012]: bootptab mtime is Tue Nov  2 14:39:24 1999
2) Nov  2 14:44:22 rs1400a bootpd[23012]: reading "/etc/bootptab"
3) Nov  2 14:44:22 rs1400a bootpd[23012]: read 1 entries from "/etc/bootptab"
4) Nov  2 14:44:22 rs1400a bootpd[23012]: dumped 1 entries to "/etc/bootpd.dump".
5) Nov  2 14:45:00 rs1400a bootpd[23012]: bootptab mtime is Tue Nov  2 14:39:24 1999
6) Nov  2 14:45:00 rs1400a bootpd[23012]: Received boot request.
7) Nov  2 14:45:00 rs1400a bootpd[23012]: request from IP addr 1.1.2.2
8) Nov  2 14:45:00 rs1400a bootpd[23012]: IP address not found: 1.1.2.2
9) Nov  2 14:45:00 rs1400a bootpd[23012]: Finished processing boot request.
10) Nov  2 14:45:12 rs1400a bootpd[23012]: bootptab mtime is Tue Nov  2 14:39:24 1999
11) Nov  2 14:45:12 rs1400a bootpd[23012]: Received boot request.
12) Nov  2 14:45:12 rs1400a bootpd[23012]: request from IP addr 1.1.2.2
13) Nov  2 14:45:12 rs1400a bootpd[23012]: IP address not found: 1.1.2.2
14) Nov  2 14:45:12 rs1400a bootpd[23012]: Finished processing boot request
#
```

Figure 18. Output from a bootp debug with NIM client IP address error

If the IP address for the NIM clients gateway is wrong in the NIM database, the entry in the file /etc/bootptab will be created wrong as well. As shown in Figure 19 on page 158, you can see what happens when the gateway IP address of the NIM client is not correct.

- line 2: The NIM client’s Bootp request is received from the NIM master.
- line 3: The NIM master tries to resolve the hostname of the received IP address by using the name resolution mechanism set in the network environment.
- line 4: If the IP address is correct, the hostname will be shown, and the NIM master begins to process the bootp request.
- line 8: The NIM master sends a bootp replay to the NIM client including the information shown in lines 9 to 12.
- line 12: This bootp replay will never reach the NIM client because of the wrong IP address of the gateway.

After a moment, the NIM client will send a second bootp request to the NIM master. The NIM master will process all of these bootp requests, but no bootp replay will reach the NIM client. The NIM client will hang.

```
# tail -f /tmp/bootpd.log
...
1) Nov  2 14:53:25 rs1400a bootpd[23012]: read 1 entries from "/etc/bootptab"
2) Nov  2 14:53:25 rs1400a bootpd[23012]: Received boot request.
3) Nov  2 14:53:25 rs1400a bootpd[23012]: request from IP addr 1.1.2.2
4) Nov  2 14:53:25 rs1400a bootpd[23012]: found 1.1.2.2 rs1230e3
5) Nov  2 14:53:25 rs1400a bootpd[23012]: bootfile = /tftpboot/rs1230e3
6) Nov  2 14:53:25 rs1400a bootpd[23012]: vendor magic field is 0.0.0.0
7) Nov  2 14:53:25 rs1400a bootpd[23012]: RFC1048 vendor data ( bp_vend[64] )
8) Nov  2 14:53:25 rs1400a bootpd[23012]: sending RFC1048-style reply
9) Nov  2 14:53:25 rs1400a bootpd[23012]: The following addresses are included in the bootp reply
10) Nov  2 14:53:25 rs1400a bootpd[23012]: Client IP address (bp->bp_ciaddr) = 1.1.2.2
11) Nov  2 14:53:25 rs1400a bootpd[23012]: Server IP address (bp->bp_siaddr) = 9.3.187.229
12) Nov  2 14:53:25 rs1400a bootpd[23012]: Gateway IP address (bp->bp_giaddr) = 1.1.2.9
13) Nov  2 14:53:25 rs1400a bootpd[23012]: Finished processing boot request.
#
```

Figure 19. Output from a bootp debug with gateway IP address error

2.11.2 Producing debug output from a network boot image

To produce debug output from a network boot image, follow these steps:

1. The network boot images must be re-created. Therefore, use the following command:

   ```
   # nim -Fo check -a debug=yes SpotName
   ```

   The Web-Based System Manager or smitty can also be used to recreate the network boot images.

2. Get the address for entering the debugger with the following command:

   ```
   # lsnim -a enter_dbg SpotName
   ```

   You will get output similar to the following:

   ```
   spot433:
           enter_dbg = "rs6k.up.tok 0x001fa0b8"
           enter_dbg = "rspc.up.ent 0x001fa0b8"
   ```

   Write down the `enter_dbg` address for the client you are going to boot. For example, if your client is an rscp uniprocessor with ethernet, write down the address `1fa0b8`.

3. Attach a tty to your NIM client on port 1, or use another RS/6000 as a tty for the NIM client. If you use another RS/6000, you will be able to log the debug output into a file. This may be helpful for problem determination. For information on how to use an RS/6000 as a tty, see Section 2.11.4, "Attach an RS/6000 as a tty" on page 161.

4. Perform the NIM operation to boot/install the NIM client without an automatic reboot of the client. Boot the NIM client over the network.

5. After the NIM clients get the network boot image from the SPOT resource server, the debug screen will appear on the ttys screen. At the > prompt, enter:

   ```
   > st enter_dbg_value 2
   ```

   The number you wrote down in step 2 is the `enter_dbg_value` number. The number `2` behind the enter_dbg_value specifies the debug to print the output to the tty.

6. Enter`> g` and press **Enter** to start the boot process.

7. You can use the Ctrl-s keys to temporarily stop the process.

   With the keys, Ctrl-q, you can resume the process.

8. After the debugging, you can rebuild your boot images to non debug mode with the following command:

   ```
   # nim -Fo check SpotName
   ```

Note

If you do not rebuild your boot images for non-debug mode, every time a NIM client is booted from these boot images, the process will stop and wait for the command at the debugger > prompt. If no tty is attached to these NIM clients, it will look like a hang.

Example debug output is shown in Figure 20 on page 160. This debug output was produced because of the LED 611 hang of the machine.Many lines were cut at the front of the output because the reason for the hang is shown in the last lines.

As you can see in this example, the NIM client tried to mount the NIM resources from the server. This NFS mount failed, and the exit code of that command was `Exec format error`. This caused a `showled` command to show error code 611 on the LED display of the NIM client.

```
# more /tmp/debug.log
...
...
+ echo 9.3.91.26 hdc013.hallo.com
+ 1>> /etc/hosts
+ [ -n  ]
+ 1> /etc/filesystems
+ /usr/lib/methods/showled 0x610
exec(/usr/lib/methods/showled,0x610)
showled LED{610}
+ mount -r hdc013.hallo.com:/export/spot/spot_415/usr /SPOT/usr
exec(/usr/sbin/mount,-r,hdc013.hallo.com:/export/spot/spot_415/usr,/SPOT/usr)
exec(/sbin/helpers/nfsmnthelp,M,0,hdc013.hallo.com,/export/spot/spot_415/usr,/SPOT/usr,ro)
nfsmnthelp: Exec format error
+ [ 1 -ne 0 ]
+ loopled 0x611
exec(/usr/lib/methods/showled,0x611)
showled LED{611}
#
```

Figure 20. Debug output when the LED 611 hang occurred

2.11.3 Producing debug output from the BOS install program

The debug output can be produced by two different methods. One method is to use a bosinst_data resource and the other method is to enter a special value into the installation menu. Instead of using a tty as the console of the NIM client, you can also use another RS/6000 as a tty. This enables you to log the debug output. To do this, see Section 2.11.4, “Attach an RS/6000 as a tty” on page 161.

2.11.3.1 Without using a bosinst_data resource

To produce debug output from the BOS install program without using a bosinst_data resource, perform the following steps:

1. Start the NIM process to install the client.
2. Because you are not using a bosinst_data resource, you will be prompted to enter install information to the BOS install program.
3. Select your console and then your language.
4. The Base Operating System Installation and Maintenance menu will be displayed. Do not select one of the displayed options. Type in the value `911` at the prompt and press **Enter**.
5. The following steps are the same as those used to install a NIM client without debugging the BOS install program. The debug output will be sent to the display when the installation begins.

2.11.3.2 By using a bosinst_data resource

To produce debug output from the BOS install program by using a bosinst_data resource, perform the following steps:

1. In order to enable debugging of the BOS installation, you must set the parameter BOSINST_DEBUG = yes in the bosinst_data resource file, which will be used to install the NIM client.
2. If you have not set other values in the bosinst_data resource file, you will be prompted to supply the install information to the BOS installation program.
3. The following steps are the same as those used to install a NIM client without debugging the BOS install program. The debug output will be sent to the display when the installation begins.

Note

A minimum bosinst_data resource file for debugging purposes would look like the following:

```
control_flow:

    BOSINST_DEBUG = yes
```

No other values must be set in the debug bosinst_data resource.

2.11.4 Attach an RS/6000 as a tty

To interface an RS/6000 as a tty, follow these steps:

1. To connect the hardware you will need
   - Two RS232 serial cables
   - One gender changer (female on both sides to get the right connection)
   - One interposer (null modem)

   Connect the hardware as shown in the following schematic:

   [S1]--[X]---[R]---[I]-[G]---[R]---[X]--[Sx]

where	S1	= the first serial port on the machine to be installed
	Sx	= any serial port on the machine attached as a tty
	X	= any extra cables needed to connect the RS232 cable to the serial port
	R	= RS232 cable
	I	= interposer (null modem)
	G	= gender changer

2. To set up the interface system, perform these steps. In the listed commands, the serial port 0 is used.

   Make sure that the fileset bos.net.uucp is installed.

   Set up the Sx port. You must create a tty on the Sx port. Use the following command to set up the tty serial port 0:

   ```
   # mkdev -c tty -t 'tty' -s 'rs232' -p 'sa0' -w '0'
   ```

   you will get a response, such as:

   ```
   tty0 available
   ```

   Add the following line to the file /etc/uucp/Devices by using the vi:

   ```
   Direct tty0 - 9600 direct
   ```

3. Run the following commands to capture the debug output. The script command will log all output to the /tmp/debug.log file, and the `cu` command connects the current session to the serial port

   ```
   # script /tmp/debug.log
   # cu -ml tty0
   ```

   To disconnect from the NIM client and stop logging, enter the following:

   ```
   ~.
   exit
   ```

2.11.5 Producing debug output by using an IP-trace

In order to produce an IP trace, you must know whether your NIM client is in the same network as the NIM master. If the two machines are in the same network, the IP trace must only be set up on the NIM master. If NIM client and NIM master are on different networks, the IP trace must be set up on the NIM master and on a different machine in the NIM client's network.

2.11.5.1 NIM master and NIM client in the same network

In this case, it is very simple to produce an IP trace. Perform the following steps on the NIM master:

1. Start the iptrace subsystem with the following command on the NIM master:

   ```
   # startsrc -s iptrace -a "-a -b -d ClientName OutputFile.bin"
   ```

2. Now, the IP trace is running. Remember that the output file `OutputFile.bin` can grow very quickly. If the error occurs, stop the IP trace.
3. To stop the IP trace, issue the following command on the NIM master:

   ```
   # stopsrc -s iptrace
   ```

4. Now, the IP trace is done, and it must converted into an IP report; so, enter the following on the NIM master:

```
# ipreport -srn OutputFile.bin > OutputFile.rpt
```

5. The file, `OutputFile.rpt`, includes a report you can use to determine the problem.

2.11.5.2 NIM master and NIM client in different networks

In this case, you must set up two machines to create the ip traces. Perform the following steps:

1. Start the iptrace subsystem with the following commands:

   On the NIM master, enter:

   ```
   # startsrc -s iptrace -a "-a -b -d ClientName OutputFile.bin"
   ```

   On the machine in the NIM clients network, enter:

   ```
   # startsrc -s iptrace -a "-a -e -b -d \
   ClientName OutputFile.bin"
   ```

2. Now the IP traces are running. Please remember, the output files `OutputFile.bin` on both machines can be grow up very fast. If the error occurs, stop the ip traces.
3. To stop the IP traces, enter the following commands:

   On the NIM master, enter:

   ```
   # stopsrc -s iptrace
   ```

   On the machine in the NIM clients network, enter:

   ```
   # stopsrc -s iptrace
   ```

4. Now, the IP traces are done, and they must be converted into an IP report. Therefore, do the following:

   On the NIM master, enter:

   ```
   # ipreport -srn OutputFile.bin > OutputFile.rpt
   ```

   On the machine in the NIM clients network, enter:

   ```
   # ipreport -srn OutputFile.bin > OutputFile.rpt
   ```

5. The file, `OutputFile.rpt`, includes reports you can use to determine the problem.

An example of an IP trace output is shown in Figure 22 on page 165 to Figure 25 on page 166. In that case, the bootpd daemon was not running on the NIM master; so, the NIM client did not get the bootp reply. That caused a hang of the NIM client. In Figure 21, the NIM environment in which the bootp error occurred is shown.

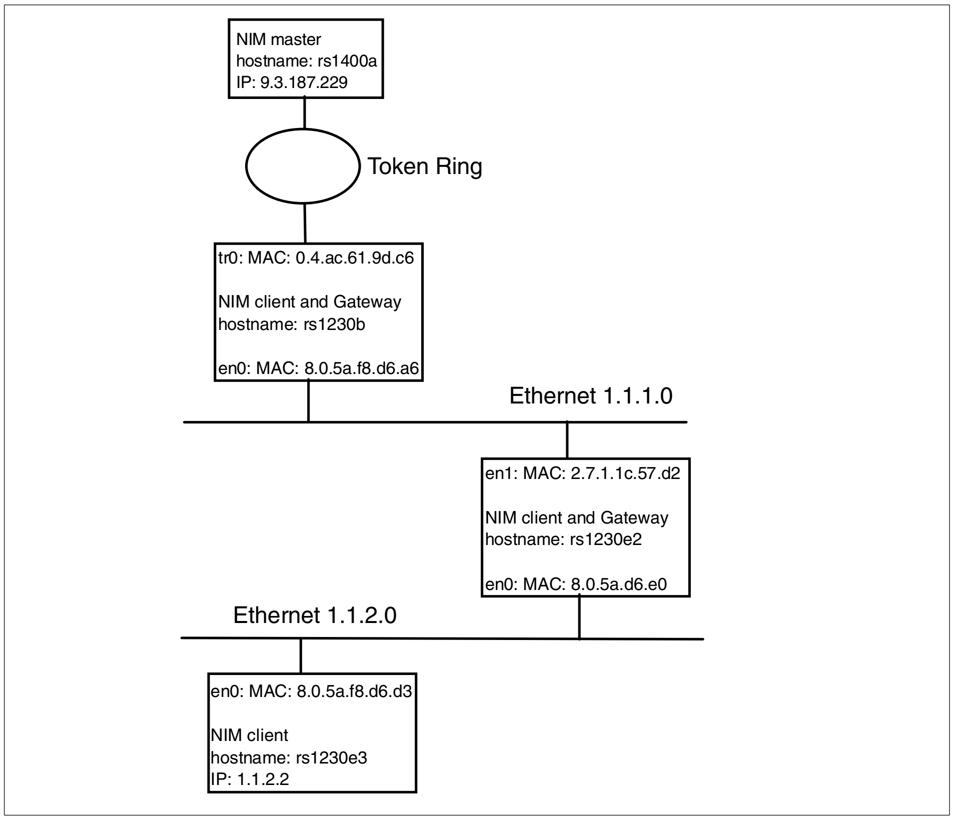


Figure 21. Diagram of the NIM environment

The numbers in front of each line of the IP trace are normally not shown. The IP trace output is usually one file and not, as shown here, different parts for each IP packet. We have done these changes to be able to explain what happened. The IP trace was made on the NIM client named rs1230e2, which is also a gateway to get from the ethernet 1.1.1.0 into the ethernet 1.1.2.0.

In Figure 22 on page 165, an IP packet that contains the bootp request from the NIM client is shown. This IP packet reached the gateway on the network adapter en0 (see line 2 in Figure 22 on page 165). In line 3 of Figure 22, the MAC addresses (NIM client, rs1230e3, Gateway, and rs1230e2) are shown. This is the direct connection from one network adapter to the other one. The source and the destination of the IP packet are shown in lines 4 (NIM client) and 5 (NIM master) of Figure 22.

```
1) Packet Number 1
2) ETH: ====( 342 bytes received on interface en0 )==== 16:44:24.054829066
3) ETH:    [ 08:00:5a:f8:d6:d3 -> 08:00:5a:f8:d6:e0 ]  type 800  (IP)
4) IP:  < SRC =         1.1.2.2 >  (rs1230e3)
5) IP:  < DST =     9.3.187.229 >  (rs1400a.itsc.austin.ibm.com)
6) IP:  ip_v=4, ip_hl=20, ip_tos=0, ip_len=328, ip_id=0, ip_off=0
7) IP:  ip_ttl=30, ip_sum=d3ba, ip_p = 17 (UDP)
8) UDP: <source port=68(bootpc), <destination port=67(bootps) >
9) UDP: [ udp length = 308 | udp checksum = a9a9 ]
10) UDP: BOOTP header breakdown:
11) UDP: Request:
12) UDP: htype:1(10Mb Ether.), hlen:6, hops:0
13) UDP: xid:0x479a, secs:0, flags:0x0
14) UDP: ciaddr:1.1.2.2, yiaddr:0.0.0.0
15) UDP: siaddr:0.0.0.0, giaddr:0.0.0.0
```

Figure 22. IP trace packet 1

In line 2 of Figure 23, you can see how the gateway sends the IP packet through the second network adapter into the next network. This is called routing. In line 3 of Figure 23, the MAC addresses of this and the next gateway are shown. The IP packet will be routed to the NIM master. The NIM master will send a replay, or an error will be returned from the NIM master.

```
1) Packet Number 2
2) ETH: ====( 342 bytes transmitted on interface en1 )==== 16:44:24.054862999
3) ETH:    [ 02:07:01:1c:57:d2 -> 08:00:5a:f8:d6:a6 ]  type 800  (IP)
4) IP:  < SRC =         1.1.2.2 >  (rs1230e3)
5) IP:  < DST =     9.3.187.229 >  (rs1400a.itsc.austin.ibm.com)
6) IP:  ip_v=4, ip_hl=20, ip_tos=0, ip_len=328, ip_id=0, ip_off=0
7) IP:  ip_ttl=29, ip_sum=d4ba, ip_p = 17 (UDP)
8) UDP: <source port=68(bootpc), <destination port=67(bootps) >
9) UDP: [ udp length = 308 | udp checksum = a9a9 ]
10) UDP: BOOTP header breakdown:
11) UDP: Request:
12) UDP: htype:1(10Mb Ether.), hlen:6, hops:0
13) UDP: xid:0x479a, secs:0, flags:0x0
14) UDP: ciaddr:1.1.2.2, yiaddr:0.0.0.0
15) UDP: siaddr:0.0.0.0, giaddr:0.0.0.0
```

Figure 23. IP trace packet 2

In this case, the NIM master did not replay the bootp request because the bootpd daemon of the NIM master was not active. In lines 4 and 5 of Figure 24 on page 166, you can see the source (NIM master) and the destination (NIM client) of the returned IP packet. This IP packet contains the message of an unreachable destination. See lines 8 and 9 in Figure 24 on page 166 and in Figure 25 on page 166. The gateway routed the IP packet into the destination network in which the NIM client resides.

```
1) Packet Number 3
2) ETH: ====( 70 bytes received on interface en1 )==== 16:44:24.057810466
3) ETH:    [ 08:00:5a:f8:d6:a6 -> 02:07:01:1c:57:d2 ]  type 800  (IP)
4) IP:  < SRC =     9.3.187.229 >  (rs1400a.itsc.austin.ibm.com)
5) IP:  < DST =         1.1.2.2 >  (rs1230e3)
6) IP:  ip_v=4, ip_hl=20, ip_tos=0, ip_len=56, ip_id=37917, ip_off=0
7) IP:  ip_ttl=253, ip_sum=61bc, ip_p = 1 (ICMP)
8) ICMP: icmp_type=3 (DEST UNREACH)
9) ICMP: icmp_code=3 (9.3.187.229: UDP PORT 67 unreachable, src=68)
```

Figure 24. IP trace packet 3

As you can see in Figure 25, in line 3 (MAC address of the NIM client), the IP packet that contains the error reached the NIM client. The NIM client recognizes the error, and, after a moment, it will send another IP packet to the NIM master containing a bootp request; so, the same procedure begins again.

```
1) Packet Number 4
2) ETH: ====( 70 bytes transmitted on interface en0 )==== 16:44:24.057843332
3) ETH:    [ 08:00:5a:f8:d6:e0 -> 08:00:5a:f8:d6:d3 ]  type 800  (IP)
4) IP:  < SRC =     9.3.187.229 >  (rs1400a.itsc.austin.ibm.com)
5) IP:  < DST =         1.1.2.2 >  (rs1230e3)
6) IP:  ip_v=4, ip_hl=20, ip_tos=0, ip_len=56, ip_id=37917, ip_off=0
7) IP:  ip_ttl=252, ip_sum=62bc, ip_p = 1 (ICMP)
8) ICMP: icmp_type=3 (DEST UNREACH)
9) ICMP: icmp_code=3 (9.3.187.229: UDP PORT 67 unreachable, src=68)
```

Figure 25. IP trace packet 4

2.12 NIM installation performance considerations

To speed up NIM installations or other NIM operations, it may be helpful to consider using global exports of NIM resources, the multi threaded enabled nimesis daemon, NIM group resources, or a different resource server. We cannot give you special numbers for the speedup of your process, but there might be an improvement of the process.

2.12.1 NIM clients as resource servers

The use of NIM clients as resource servers is a powerful option to decrease the time required for the installation process. Instead of mounting the NIM resources from the NIM master, which can be far away from the NIM client in a different network, it is much better to mount the necessary NIM resources from a NIM resource server that is very close to the NIM client to be installed. The maximum number of eight NIM clients to be installed from a resource server is recommended.

We recommend that you place your resource servers at strategic points in a NIM environment. There may be at least one resource server in each sub network of the NIM environment. Figure 26 illustrates the meaning of strategic placement of the NIM resource servers.

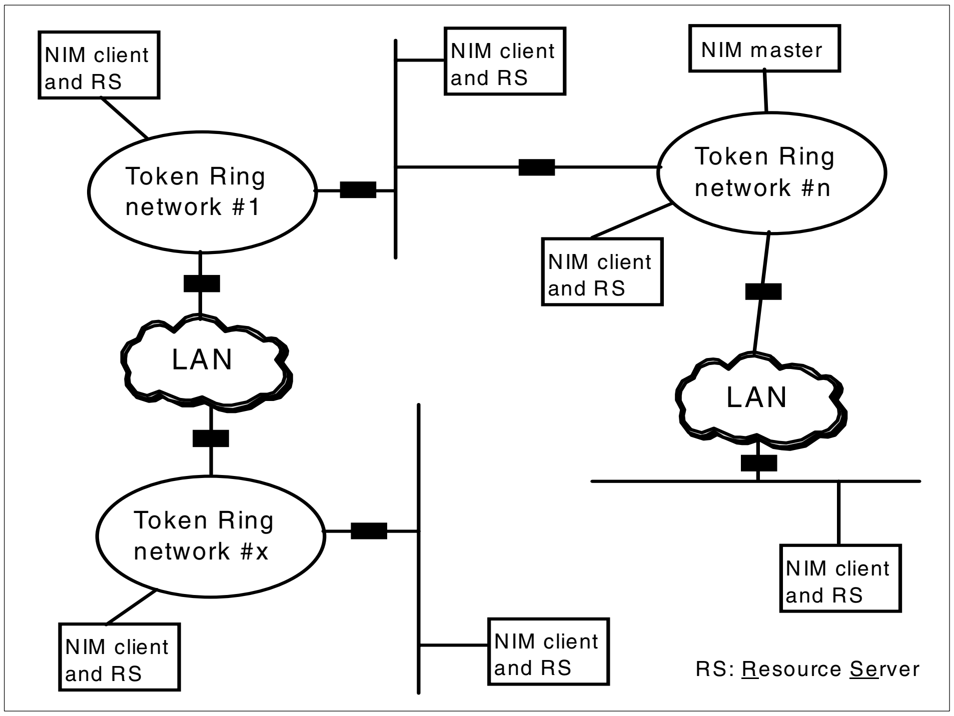


Figure 26. Strategic placement of NIM resource servers

Each NIM client can become a resource server. To create a NIM resource on a NIM client, follow the instructions in the following sections.

2.12.1.1 From the Web-Based System Manager

Create a resource server with the Web-Based System Manager by using these steps:

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Open the Resource container.
4. double-click on **Add New Resource**.
5. A Task Guide window will appear. Follow the instructions of this Task Guide. Be sure to select the NIM client as the server machine.

2.12.1.2 From smitty

You can also create the resource server by using smitty.

1. Open smitty with the `nim_mkres` fastpath.
2. Select the resource type.
3. A dialog field will appear. Supply the correct values to the menu, and be sure to select a NIM client as the server machine.

2.12.1.3 From the command line

To create a resource on a NIM client, be sure to specify the NIM client's name for the server attribute. The command to create the resource that follows is only an example that creates an lpp_source resource named *external_lpp* on the NIM client named *rs1230e2*. The source is the CD drive, /dev/cd0, on the NIM client machine and will be stored in the */export/lpp* directory.

```
# nim -o define -t lpp_source -a server=rs1230e2 \
-a location=/export/lpp -a source=/dev/cd0 external_lpp
```

2.12.2 Multithreaded nimesis daemon (AIX 4.3.3 or later)

NIM can be scaled to support 20 up to 150 NIM client requests simultaneously. This feature should be used in large NIM environments with many NIM clients. With this option, NIM can better handle a huge amount of change requests to change the info field or the state field of NIM clients in the NIM Database. Without using this option in a large environment, the NIM master machine can become overloaded by working on the NIM database. This may cause failures during the installation of a huge number of NIM clients that are performing a NIM operation simultaneously.

It is not necessary to assign each NIM client its own nimesis daemon thread because most NIM client requests can be handled rapidly. Before setting the number of nimesis threads, you should consider the following aspects:

- What is the number of NIM clients that will be installed/operated at the same time?
- How is the processing power of the NIM master? Can the NIM master hardware handle so many requests simultaneously?
- What type of NIM operations are planned with the NIM environment?

In default, the multithreaded function of the nimesis daemon is enabled, and the maximum number of simultaneous requests allowed is set to 20. Since one nimesis daemon thread can support two to four NIM client installation requests, 40 to 80 clients can be handled with these default settings.

Note

The multithreaded nimesis daemon option alone will not increase the performance for simultaneous NIM client processing. This option should be used in conjunction with other performance-related settings, such as global exports and the use of different resource servers in the NIM environment. A network that has a very large volume of throughput should also be used.

2.12.2.1 From Web-Based System Manager

Change the settings for the multithreaded nimesis daemon in the Web-Based System Manager by performing the following steps:

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Go to **NIM** --> **Advanced Configuration** --> **Tune Client Requests**.
4. A dialog will appear in which you can enable or disable the multithreaded option of the nimesis daemon. You can set the value of the maximum number of simultaneous requests allowed from 20 up to 150.

2.12.2.2 From smitty

From smitty, you should follow these steps:

1. Open smitty with the `nim_tune_nimesis` fastpath.
2. A dialog will appear in which you can enable or disable the multithreaded option of the nimesis daemon. You can set the value of the maximum number of simultaneous requests allowed from 20 up to 150.

2.12.2.3 From the command line

To enable the multithreaded function of the nimesis daemon, you must set a *value* from 20 up to 150 to the max_mimesis_threads. Use the following command:

```
# nim -o change -a max_nimesis_threads=value master
```

To disable the multithreated function of the nimesis daemon, you must set the value of the max_nimesis_threads on the NIM master to null. Use the following command:

```
# nim -o change -a max_nimesis_threads=”” master
```

2.12.3 Using the NIM group (AIX 4.2 or later)

A machine group represents a number of NIM clients. The hardware types of these machines does not matter, but all members of a machine group must be the same type of NIM clients, such as diskless, dataless, or stand-alone. When a machine group is defined, a NIM operation can be performed on this group, and NIM will perform this operation on all members of the group automatically. This option will save time required to start the operation but not to perform the operation on the clients; so, there will not really be a great performance enhancement, but it is a better way to administrate the NIM clients. For information about how to create a NIM group, see Section 1.2.4, “Groups” on page 22.

Note

When using machine groups with a large number of NIM clients, the maximum number of simultaneous installs field should be set before performing the operation to the group. We recommend that you not install more than eight NIM clients simultaneously.

2.12.4 Using global exports for NIM resources (AIX 4.3 or later)

Every time a NIM operation starts or stops, file systems from the resource server are exported or unexported for this special NIM client. Each export or unexport causes the NFS subsystem to add or delete entries into the files, /etc/exports and /etc/xtab, and, each time an export or unexport occurs, the file must be locked for writing. These files may become very large if a large number of NIM clients are performing a NIM operation at the same time. This may cause a hit of file size limitations and/or decrease the NIM performance.

To avoid these repeated updates of the files, /etc/exports and /etc/xtab, it may be helpful to do a global export for the NIM resources. But, keep in mind that globally-exported resources are readable for all machines in the network, not only for the NIM clients. This is only an option for environments in which administrators are not concerned about who has access to the NIM resources. Exclusively used resources for dataless and diskless clients may not be globally exported. The global export will be active as long as at least one client has allocated the resource. If the last client deallocates the resource, it will be unexported.

Note

The NIM resources should not be globally exported or unexported during a NIM client has resources allocated. This could cause incorrect permissions for the exports. No NIM operation should run during globally exporting or unexporting resources.

A global export of the resources involves all resource servers in the NIM environment. To do a global export of the NIM resources, do the following on the NIM master.

2.12.4.1 From Web-Based System Manager

Follow these steps to set the global export using Web-Based System Manager.

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Go to **NIM** --> **Advanced Configuration** --> **Export NIM Resources Globally**.
4. A dialog will appear in which you can export or unexport the resources globally.

2.12.4.2 From smitty

Use smitty to set the global exports by performing the following steps:

1. Open smitty with the `nim_global_export` fastpath.
2. A dialog will appear in which you can export or unexport the resources globally.

2.12.4.3 From the command line

To export the resources globally, issue the following command:

```
# nim -o change -a global_export=yes master
```

To unexport the resources globally, issue the following command:

```
# nim -o change -a global_export=no master
```

2.12.5 Network options

In this section, we will discuss some network options you can change in order to increase your network performance. One important parameter is the Maximum Transfer Unit (MTU) parameter. Furthermore, you should watch for buffer overruns in the network environment and, if they occur, increase the buffers.

2.12.5.1 Maximum Transfer Unit and Fragmentation

The Maximum Transfer Unit parameter defines the maximum length of the IP datagram, which can be sent out from the network interface. Any IP datagram longer than the MTU should be split into several smaller IP datagrams before it can be transmitted. This datagram splitting is called fragmentation. The destination of the IP datagram must gather all fragments and rebuild the original IP datagram after receiving. This rebuild job is called assembly. Both fragmentation and the assembly procedure impact network performance.

Since the MTU parameter is a configuration parameter of network interfaces, the value of this parameter may be different between systems. You should determine the most appropriate MTU value for all systems connected to the network. All systems attached to the same IP network must share the same MUT value.

In Table 21, you can see some available MTU interface sizes. As you can see, the default MTU value is not the maximum value. Increasing the MTU value may increase network performance.

Table 21. MTU values of network interfaces

Interface	Default	Minimum	Maximum
Ethernet	1500	60	9000
Token-ring 16M	1492	60	17792
FDDI	4352	1	4352

You can find your actual MTU value with the following command:

```
# netstat -i
```

The output looks like the lines shown below. The MTU value is shown for each network interface in the second column.

```
Name  Mtu   Network     Address          Ipkts   Ierrs Opkts   Oerrs Coll
lo0   16896             link#1           468608  0     468599  0     0
lo0   16896 127         loopback         468608  0     468599  0     0
lo0   16896 ::1                          468608  0     468599  0     0
tr0   1492  link#2      8.0.5a.b9.51.e55108278 0     4001774 0     0
tr0   1492  9.3.187.128 rs1400a          5108278 0     4001774 0     0
```

With the `chdev` command, you can change the MTU value. Here is an example of changing the MTU value of the tr0 interface to the value of 4096:

```
# chdev -l tr0 -a mtu=4096
```

For more information about TCP/IP, look in the IBM Redbook *Learning Practical TCP/IP for AIX V3.2/V4.1 Users: Hints and Tips for Debugging and Tuning*, SG24-4381

2.12.5.2 Network buffers and queues

A buffer or a queue is a place where data is temporarily stored between application and devices driver when data is sent or received. During NIM operations, it may cause such buffer or queues to overrun. The result of a receive buffer overrun in the network interface device driver is that some packets are dropped, and the sender is needed to retransmit those packets again. The result of an input queue overrun in the IP layer is the loss of that packet. These buffer and queue sizes can be increased in order to avoid these overruns. The following are some important parameters you should consider.

The `sb_max` parameter specifies the maximum buffer size allowed for any of these buffers. The default is 65,536 bytes. List the current value by using the `no -o sb_max` command.

The `thewall` parameter specifies the maximum amount of real memory, in kilobytes, that can be used by the communication subsystem. The default value of `thewall` depends on the AIX version you are using. The current value can be viewed by using the `no -o thewall` command.

The `udp_sendspace` parameter sets the limit for the maximum usable memory for one UDP socket to buffer outgoing data. The default value for `udp_sendspace` is 9216, and the current value can be listed with the `no -o udp_sendspace` command.

The `udp_recvspace` parameter sets the limit for the maximum usable memory for one UDP socket to buffer incoming data. The default value for `udp_recvspace` is 41920. Get the default value by using the `no -o udp_recvspace` command.

The `tcp_sendspace` and `tcp_recvspace` parameters are also tunable parameters that you can change with the `no -o` command. The default value of both parameters is 16384. List the current value with the `no -o tcp_sendspace` or `no -o tcp_recvspace` command.

The `xmt_que_size` parameter indicates the number of transmit requests that can be queued for transmission by the device driver. The current value can be displayed with the `lsattr -E -l adapterName` command where, for example, the `adapterName` is `tok0`.

We recommend that you refer to *AIX Performance Monitoring and Tuning Guide*, SC23-2365, before changing some network parameters.

2.13 Case studies

In this section, we would like to describe some case studies that will often occur in the real world. We will show you how to solve these problems and jobs step-by-step.

2.13.1 Implement NIM preload environment

If you are installing many RS/6000 systems with the same environment in a short amount of time, it will become necessary to use NIM and create a preload environment. In order to save time, the procedure to install these machines should be almost automatic. A real job to do for a customer was the following:

Install 100 new RS/6000 model F50 machines with AIX Version 4.3.2 and some applications for a shipment to a customer in one day. The software to be installed is Computer Aided Design (CAD) and Quake 3. There should be no downtime for the customers when receiving the new hardware.

The customer has one F50 with gigabit ethernet and an unused 9.1 GB hard disk already installed with the AIX version and software required for this hardware, and the machine was already running. There was a 128 port adapter and enough hardware to install all 100 machines at once as well. The ethernet cards, which were used to install the machines, must stay at the installation location.

These are the steps required to realize such a NIM preload environment:

1. Create a backup of the running F50 on the extra hard disk. Make sure that this hard disk is not assigned to the rootvg. Install the NIM master filesets using the CD onto this machine. Configure the TCP/IP on the master and add 100 dummy hosts into the /etc/hosts file. Be sure to make the mksysb of the F50 before installing the NIM master; otherwise, you would populate the NIM master onto all the other machines.
2. Configure the NIM master and the 128 port adapter.
3. Create the NIM resources lpp_source and SPOT on the NIM master.
4. Define the recently-created mksysb of the F50 as a NIM resource type mksysb.
5. Create a bosinst_data resource with the filed RECOVER_DEVICES=no.
6. Create a script that adds commands to the /etc/firstboot file. These commands should remove information about the network interface and adapter from the ODM. The /etc/firstboot script is the only way we can safely remove the ODM configuration of the network adapter. If this ODM info is not removed, all installing machines may hang.
7. If there is a need, you can create a script to interact with the firmware of the machine that will cause the network boot. You can, for example, use *expect* to create such a script.

8. Put in the ethernet cards, and hook up the network interface card and the ttys.
9. Execute the bos_inst operation with no_nim_client=yes & boot_client=no.
10. Power on all machines, and perform a network boot manually, or, if you have created a script to interact with the firmware, execute this script in order to start the network boot of the clients.
11. After the installation, power off all machines, take out the gigabit cards, and pack them into the box for shipment to the customer.

2.13.2 Upgrade operating system

The procedure to upgrade a large number of RS/6000 systems from an older AIX version, such as 3.2.5, 4.1, or 4.2, to the newest, 4.3, is a case that often occurs in the real world. The reason for such an upgrade may be the requirement of a new software product that needs a new AIX version. Another example may be the solution of the Y2K problem by upgrading older AIX versions to newer ones. Such a job must often be done in a short amount of time because the RS/6000 systems are usually used for production. A real customer request was the following:

An international company has approximately 500 machines spread across sites all over the world. These machines are currently running AIX levels ranging from 3.2.5 to 4.2.1, and the job is to upgrade them to 4.3.2 including the latest fixes. All machines to be updated have enough free disk space left. The company does not have a system administrator on each site; so, a central point of control is needed to push the installation. The most critical machines are using ATM. All machines running ATM are at Version 4.2.1. At least one machine per site is running AIX 4.3. The solution for that request is described below:

1. Select a machine on the local site, and bring it to AIX Version 4.3.2. Also, install the latest fixes for this AIX version.
2. Install the NIM master filesets, and set up the NIM environment.
3. Create an lpp_source including the latest fixes on the NIM master, and create a SPOT on the NIM master by using that lpp_source.
4. Define all machines that are to be upgraded, and create rsh access for the NIM master on those machines.
5. On each site, define all machines running AIX 4.3 as NIM clients.
6. Make one machine the resource server on each site. Therefore, copy the lpp_source resource from the NIM master to these machines, and create the NIM resource type lpp_source.

7. Update the resource server by using the local newly-defined lpp_source.
8. Create the SPOT resource on each resource server, and create a bosinst_data resource. Set up the bosinst_data file for non-prompted installation and for migrate install.
9. Use NIM to perform an update_all to upgrade all other 4.3 systems, and use the local lpp_source and the local SPOT resource of the resource server.
10. Perform a force_push bos_inst operation on all machines not running AIX 4.3. Use the local resource server as the source of the SPOT and lpp_source. A force_push installation does not require that the machine to be installed is a NIM client. Only rsh permissions for the NIM master must be available at that machine.

2.13.3 NIM in a large environment with multiple networks

In this section, we will describe a way to set up NIM in a large environment with multiple networks. A large NIM environment contains hundreds of NIM clients distributed onto different locations using different networks. The use of the different networks requires the setup of NIM routing and the use of NIM resource servers.

In order to get a working and powerful environment, you should consider a few things before you start to implement the NIM environment. The following steps may be helpful:

1. You should gather information about the network you want to work with. How many subnetworks are there? What kind of networks are these subnetworks? What are the gateways?
2. Get information about the RS/6000 systems that will become NIM clients. What kind of hardware is in the machine? Does it need an IPL ROM device for network boot? What is the current AIX version of these machines? Find out the IP address and the gateway address they are using; perhaps the machine is a gateway itself. How many network adapters do they have, and what kind and what MAC address?
3. Make a diagram with all NIM clients and gateways. An example is shown in Figure 27 on page 178. Also, create a summary table including all information, such as the hostname, hardware type, IP address and MAC address of the client, location, and what kind of network they use. You can find an example of such a table in Table 22 on page 178.
4. Select a NIM master machine. The NIM master should have enough computing power and enough hard disk space. It should be located on a strategic position in the NIM environment. The location of the NIM master

should also be an access-controlled place because of the security aspects of the used $HOME/.rhosts file on al NIM clients.

5. Now, it is time to install the NIM master machine. You should use the latest AIX version for installation. Install the NIM filesets, and set up the NIM master functions.
6. Create the basic NIM resources, such as lpp_source and SPOT. When creating the lpp_source, copy all filesets from the AIX CD into that lpp_source. You can have one bosinst_data file that can do both a non-prompted installation and a migration. You can create different file systems for the different kinds of NIM resources; so, you may be able to maintain the space for these resources in a better way.
7. You should consider the multithreaded function of the nimesis daemon. Make a decision as to how many NIM clients you will install at the same time and how many NIM operations you will perform simultaneously. Depending on those numbers, you will decide whether or not to change the setup of the multithreaded nimesis daemon. If the number is under 50 simultaneously-operating NIM clients, you should make no changes to the nimesis daemon. If the number is higher than 50, you should increase the number of threads allowed for the nimesis daemon as well.
8. You should also consider the use of global exported NIM resources. If the number of simultaneously-active NIM clients is high, it may be better to export the resources globally.
9. Add all NIM client machines into the /etc/hosts file. If you are using another name resolution mechanism, you should check the entries for the clients. If you are using incorrect IP addresses for NIM clients, that also will cause an error in the NIM database. If you are using DNS for name resolution, you should create a resolv_conf NIM resource.
10. Create a NIM network resource for each subnet on the NIM master. Add the necessary network routes in order to get into that subnet. The diagram of your NIM environment may be very helpful when setting the right routing statements.
11. If you need an IPL ROM device to boot at least one NIM client, you should install the devices.base.rte fileset onto the NIM master.
12. Define your NIM clients. Add information that makes sense to the Additional Information field when creating the clients.
13. If you are using different kinds of installations including different kinds of software, users, environments, and so on, you should make an mksysb of each type of machine setup. Create an mksysb NIM resource for each kind of mksysb.

14.Consider a group concept for the clients. Make it NIM Groups, and add the Clients into their group. Working with NIM groups will save administrative time.

15.Select NIM clients as resource servers by using strategic facts. Which NIM client has enough free disk space and computing power? You should have at least one resource server per subnet. A good concept of resource servers will save time during the processing of NIM operations. Do not forget to add the information about the resource server to the diagram of your NIM environment.

16.Create the NIM resources on the resource servers.

17.If a NIM client is a gateway at the same time, do not forget to enable IP forwarding on the gateways.

Figure 27 shows the NIM environment planning diagram.

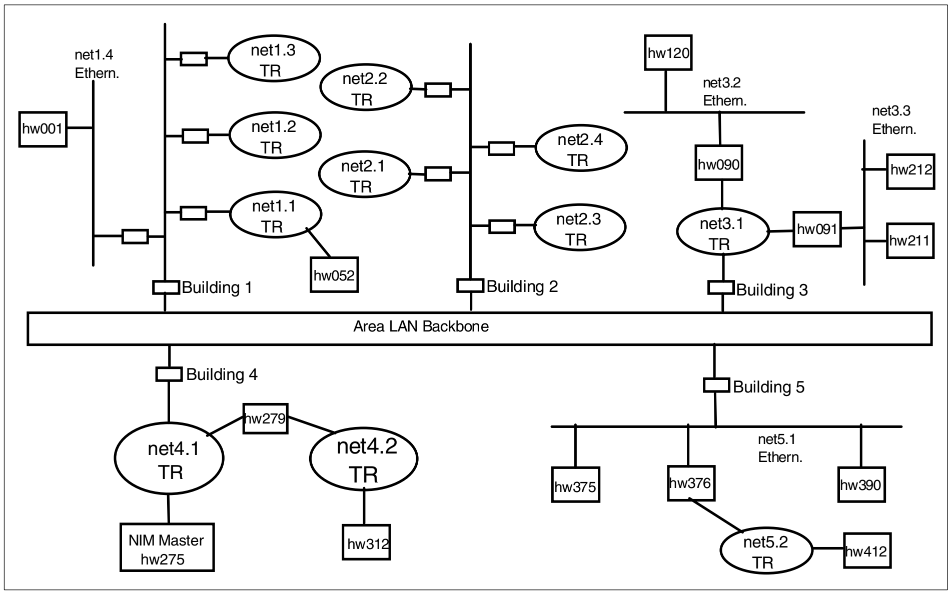


Figure 27. NIM environment planning diagram

Table 22 lists the NIM environment summary data.

Table 22. NIM environment summary table

Host	HW	AIX	if	MAC	IP	net	Info
hw001	140	4.3.3	ent0	08005afcd2e1	9.3.150.5	net1.4	RS
...							

Host	HW	AIX	if	MAC	IP	net	Info
hw052	140	4.3.2	tr0	08006d12a110	9.3.151.2	net1.1	
...							
hw090	F30	4.1.5	tr0 ent0	0800ac569a10 0800ae0020a1	9.3.170.29 9.3.171.1	net3.1 net3.2	GW
hw091	F30	4.1.5	tr0 ent0	0800338a4ce0 08005fa60122	9.3.170.75 9.3.172.1	net3.1 net3.3	GW
...							
hw120	150	4.3.3	ent0	0800fe56710a	9.3.171.7	net3.2	RS
...							
hw211	150	4.3.1	ent0	08005f6e3201	9.3.172.38	net3.3	
hw212	150	4.3.1	ent0	08005f6e3211	9.3.172.39	net3.3	
...							
hw275	F50	4.3.3	tr0	08003226af5e	9.3.180.4	net4.1	Master
...							
hw279	260	4.3.2	tr0 tr1	080064aef5d1 08006573aefd	9.3.180.9 9.3.181.1	net4.1 net4.2	GW + RS
...							
hw312	52H	4.1.5	tr0	080065332441	9.3.181.25	net4.2	IPL-ROM
...							
hw375	150	4.3.2	ent0	0801af5e7c33	9.3.190.22	net5.1	
hw376	150	4.3.2	ent0 tr0	0802cfae3342 0802fce33445	9.3.190.47 9.3.191.2	net5.1 net5.2	GW
...							
hw390	150	4.3.2	ent0	0803757ff722	9.3.190.93	net5.1	RS
...							
hw412	S7A	4.3.2	tr0	08040019aa32	9.3.191.5	net5.2	RS
...							

2.13.4 Software update and maintenance

An example of daily work with NIM might be the installation of new software or fixes on the NIM clients. Removing or customizing software is also a part of the daily work with NIM.

2.13.4.1 Add files and fixes

If you want to install a special fileset or an AIX fix fileset to a NIM client, perform the following steps:

1. Copy the fileset into a lpp_source.

   You should copy AIX level-based files, such as fixes, into a special lpp_source for the specific AIX levels. For special filesets, such as software products not included in the AIX CD, you should create a new lpp_resource. These special lpp_sources may not be used for BOS installation.

2. Perform the NIM `check` command on the lpp_source to which you added the filesets in order to create a new.toc file for that lpp_source. The quickest way to do this is to issue the `nim -o check lpp_source_name` command where `lpp_source_name` is the name of the updated lpp_source.

3. Install the filesets on single NIM clients or on NIM machine groups.

   From Web-Based System Manager

   To install the filesets on single NIM clients or on NIM machine groups from Web-Based System Manager, perform the following steps:

   a. Start the Web-Based System Manager.

   b. Open the NIM container.

   c. Select the NIM client or NIM group.

   d. Go to **Selected -> Install/Update Software -> Install Additional Software (Custom)**.

   e. Select the lpp_source from step 1.

   f. Select **Install specific software from source**.

   g. Click **Browse...** to list the software included in the lpp_source, and select the software you want to install.

   h. Click **Advanced...** to select specific settings for the installation.

   i. Now, you can do a preview for the installation, or you can perform the installation directly.

   From smitty

To install the filesets on single NIM clients or on NIM machine groups from smitty, perform the following steps:

a. Open smitty with the `nim_task_inst` fastpath.
b. Select **Install and Update** from ALL Available Software.
c. Select the NIM client or NIM group.
d. Select the lpp_source from step 1.
e. Select the software you want to install.
f. Change the other values, or accept the default settings.

From the command line

To install the filesets on single NIM clients or on NIM machine groups from the command line, enter the following:

```
# nim -o cust -a filesets=fileset_name -a lpp_source=lpp_source_name
clientname
```

For more information about how to maintain lpp_sources, see Section 2.5.1, “Maintaining software in an lpp_source” on page 127.

2.13.4.2 Work with install_bundle

If you must install a couple of additional software filesets or a couple of AIX fixes, it is very useful to create an installp_bundle NIM resource for those filesets. The advantage of that NIM resource is that you need not select all the single filesets all the time. The only step is to select the installp_bundle before performing the NIM operation on the NIM clients. The following are some guidelines for working with installp_bundles:

1. Create a file named after the software you want to install, and save it to the directory /../installp_bundle/.
2. Write the names of the filesets into that file. The structure of the names is like that shown in Figure 28 on page 182. This is an example installp_bundle file used to install the C++ compiler. It is also possible to specify a special fileset level by writing the level behind the name into the file (separated by a blank space).

```
# more /export/installp_bundles/ibmcxx
ibmcxx.cmp
ibmcxx.html
ibmcxx.include
ibmcxx.ioc
ibmcxx.lib
ibmcxx.memdbg
ibmcxx.msg.en_US
ibmcxx.rescmp
ibmcxx.rte
#
```

Figure 28. Contents of an installp_bundle file

3. Create a NIM resource, installp_bundle. Use the file created in Step 1, and perform the following steps to create a NIM resource installp_bundle.

   From Web-Based System Manager

   To create an installp_bundle with the Web-Based System Manager, perform the following steps:

   a. Start the Web-Based System Manager.

   b. Open the NIM container.

   c. Open the Resources container.

   d. Go to **Resource** --> **New Resource**.

   e. The Task Guide appears. Select **Optional software installation**, **update**, or **customization**, and press **Next**.

   f. Select **installp_bundle - Lists software to install**, and press **Next**.

   g. Type in the name of the installp_bundle resource, the path and name of the file created in step 1, and the server of the resource.

   h. Follow the Task Guide for the next steps.

   From smitty

   Use smitty as described in the following steps to create an installp_bundle:

   a. Open smitty with the `nim_mkres` fastpath.

   b. Select **installp_bundle** from the list.

   c. Type in the requested information. Use the file created in Step 1 as the Location of Resource.

   From the command line

   To create a NIM resource installp_bundle from the command line, enter the following:

```
# nim -o define -t installp_bundle -a server=serverName \
  -a location=file_path/file_name installp_bundleName
```

For example:

```
# nim -o define -t installp_bundle -a server=master \
  -a location=/export/installp_bundles/ibmcxx ibmcxx
```

Where the name of the installp_bundle is ibmcxx, and the location of the file including the fileset names is /export/installp_bundle/ibmcxx.

4. Create a new sub directory in the directory where your lpp_sources reside. You can name it after your file that was created in Step 1.
5. Copy all filesets into that newly-created directory.
6. Create a new lpp_resource, and use the directory created in Step 4 as source.
7. Now, the installp_bundle is ready to use for installing on NIM clients. You can install it in three different ways.

   From Web-based System manager

   To install an installp_bundle, use the Web-Based System Manager as follows:

   a. Start the Web-Based System Manager.
   b. Open the NIM container.
   c. Select a NIM client or a NIM group where you want to install.
   d. Go to **Selected** --> **Install/Update Software** --> **Install Bundles**.
   e. A Task Guide will appear. Select the lpp_source you created in Step 6 as the software source. Select the bundle you created in Step 3.

   From smitty

   You can also use the smitty to install the installp_bundle.

   a. Start with the `nim_inst_bundle` fastpath.
   b. Select a NIM client or a NIM group as target.
   c. Select the lpp_source where the filesets are included.
   d. Select the bundle to use.
   e. A menu will appear. Supply the requested information to the menu, or accept the default values.

   From the command line

   First, you must allocate the bundle you want to install. Therefore, enter:

```
# nim -o allocate -a installp_bundle=bundle1 \
  -a lpp_source=lpp_source1 ClientName
```

Where `bundle1` is the name of your bundle, and `lpp_source1` is the name of your lpp_source where the filesets resides.

Then, you can perform the cust operation to start the installation with the following command:

```
# nim -o cust ClientName
```

2.13.4.3 NIM cust operation

You can use the cust operation to install software or only updates on a NIM client. The target of the installation can be a NIM client, a NIM group, or a spot. The cust operation requires an lpp_source and a list of at least one fileset or an installp_bundle.

Here is one example of using the cust operation: Many NIM clients run the same AIX version with the same maintenance level. Since there is a new maintenance level available, it was installed into lpp_source lpp_maint2 on the NIM master.

Now, all the NIM clients have to be updated from that lpp_source. Consequently, a cust operation was performed on the NIM group, GroupA, including all the NIM clients. The next two commands were needed to do the job.

First, allocate all required resources by issuing the command

```
# nim -o allocate -a lpp_source=lpp_maint2 GroupA
```

Second, perform the cust operation on all clients by issuing the command

```
# nim -o cust -afixes=update_all GroupA
```

2.13.4.4 Removing software and changing the state of software

Sometimes, it becomes necessary to remove software from machines. It is also possible that there will be a need to change the state of the filesets. This kind of operation can be performed on NIM clients, NIM groups, and SPOTs. You have three ways of performing this maintenance job.

From Web-Based System Manager

To perform a maintenance job from Web-Based System Manager, do the following:

1. Start the Web-Based System Manager.
2. Open the NIM container.

3. Select a NIM client, NIM group, or SPOT as target.
4. Go to **Selected** --> **Software Utilities**.
5. Select **Commit Applied Updates**, **Reject Applied Updates**, or **Remove Software**.
6. A Menu will appear. Browse and select the fileset/s you want to deinstall or change.

From smitty

You can start a maintenance job from smitty with the following steps:

1. Start smitty with the `nim_task_maint` fastpath.
2. Select what kind of maintenance you want to do.
3. Select a NIM client machine, a NIM group, or a SPOT as target.
4. Select one or more filesets to be performed with the action you selected in Step 2.
5. Accept the default values of the Force, Preview, and Remove of the Dependent Filesets fields, or make changes.

From the command line

Deinstall software by issuing the following command from the NIM master:

```
# nim -o maint -a installp_flags="u" -a filesets="file.name" clientName
```

You can also deinstall the software from a SPOT or NIM group. Consequently, just type in the name of the SPOT or NIM group instead of the NIM client machine name.

To commit software, you must set the field to installp_flags=c, and, to reject software, set the field to installp_flags=r. If you want to reject all dependent software as well, you must set the field to installp_flags=rg.

2.13.4.5 NIM software querying

One customer had two problems when working with the NIM environment: One problem was that an already-installed NIM client exhibited software problems. The second problem was that a NIM client could not boot from a SPOT without having problems during booting. These are typical problems that may occur when working with NIM.

The reason for these problems can be found with the NIM software querying mechanism. There are different kinds of query software for NIM clients and NIM spots. These NIM mechanisms are similar to the AIX commands, `lslpp`, `lppchk`, and `fix_query`, because NIM provides an interface to the AIX commands and does not have its own functions to rebuild those commands.

The `lslpp` option of the `NIM` command lists the details of the installed software from the NIM client or NIM spot by using the -al flags of the `lslpp` command. The option, lslpp_flags, can be set in order to query different information than with the default setting, `lslpp -al`. If specific software should be listed, you can specify a fileset with the filesets attribute. For example, if a device cannot be configured during a network boot of a client, use the `lslpp` option to find out whether or not the necessary device driver for this device is installed in the SPOT used for the installation.

The lppchk option allows the integrity of the installed software to be checked. You can check the consistency of requisites, checksum, file sizes, and so on. When using the lppchk option, the default flag is the -v flag, which is used to list requisite and version consistency. You can also use other flags by setting the lppchk_flags attribute. For example, you can use this option to check a machine after the NIM client was migrated from one AIX level to another one.

The fix_query operation is provided as an interface to the `instfix` command. With this command, you can check the status of software fixes. When using this command, the default flag, -i, is used to list the installation status of specific or all known fixes, if no APARs are specified.

You can start the commands by using the Web-Based System Manager, with smitty, or from the command line.

Performing the lslpp option

The lslpp option can be performed from the Web-Based System Manager, from smitty, or from the command line.

From the Web-Based System Manager

Start the lslpp option from the Web-Based System Manager with these steps:

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Select a machine, or open the resources container, and select a spot.
4. Go to **Selected** --> **List Installed Software**.
5. From the submenu, select one option of the listed items. These items, except the Fix (APAR) Status item, represent the flags you can use with the `lslpp` command line operation.

6. Depending on which item you have selected in Step 5, a menu appears. Fill in the requested information.

From smitty

When using the smitty, you must select whether you want to perform the `lslpp` option on a machine or on a spot resource.

Performing the `lslpp` operation on a spot:

1. Open smitty with the `nim_res_op` fastpath.
2. Select a spot.
3. Select the lslpp option.
4. Type in the flags you want to use with the `lslpp` command, or accept the default.

When performing the `lslpp` operation on a machine:

1. Open smitty with the `nim_list_installed` fastpath.
2. Select one item from the list. The items of this list represent the different kinds of flags that can be used to perform the `lslpp` operation. The List installed Software item is similar to the default `lslpp` command.
3. Select a NIM client machine.
4. Type in a fileset name(s), or accept the default.

From the command line

Performing the lslpp operation with default settings, use the command:

```
# nim -o lslpp ClientName
```

Or, if you want special filesets with special flags, use

```
# nim -o lslpp -a filesets=”fileset.name” \
-a lslpp_flags=”lslpp flags” ClientMachine
```

You can also use a spot name instead of the client name.

Performing the lppchk option

The lppchk option can be performed from the Web-Based System Manager, the smitty, and also from the command line.

From Web-Based System Manager

Using the Web-Based System Manager will require the following steps:

1. Start the Web-Based System Manager.
2. Open the NIM container.
3. Select a machine or open the resources container, and select a spot.
4. If performing the operation to a machine, go to **Selected** --> **Troubleshooting** --> **Verify Installed Software**. If performing the operation to a spot, go to **Selected** --> **Verify Installed Software**.
5. A menu will appear. Fill in the requested information. You can change the mode of the verify operation by opening the Advanced... menu and selecting another verification type.

From smitty

You can also use the smitty. Perform the following steps:

1. Open smitty with the `nim_task_maint` fastpath.
2. Select **Check Software File Size After Installation** or **Verify Software Installation and Requisites**.
3. Select a spot or a machine.
4. If you have selected Verify Software in Step 2, type in or select one fileset. If you have selected Check Software in Step 2, select one fileset or type it in, and select a verification mode from the menu that appears.

From the command line

To start the operation with the default settings, type:

```
# nim -o lppchk ClientName
```

If you want to check a specific fileset with special flags, type:

```
# nim -o lppchk -a filesets=”fileset.name” \
-a lppchk_flags=”lppchk flags” ClientName
```

Instead of using a NIM client name, you also can use a spot name.

Performing the fix_query option

The fix_query option can be performed from the Web-Based System Manager, from smitty, or from the command line.

From Web-Based System Manager

Follow these steps to use the fix_query option:

1. Start the Web-Based System Manager.

2. Open the NIM container.
3. Select a machine, or open the resources container, and select a spot.
4. Go to **Selected** --> **List Installed Software**.
5. From the submenu, select the Fix (APAR) Status item.
6. A menu will appear; fill in the Fix identifier and select options or not.

From smitty

Use the following steps to perform the fix_query option from smitty:

1. Open smitty with the `nim_show_apar_stat` fastpath.
2. Select a machine or a spot.
3. Type in the fix ID.

From the command line

Using the default settings with the command:

```
# nim -o fix_query ClientName
```

To get information about special APARs, or to use special flags:

```
# nim -o fix_query[-a fixes="FixKeywords"]
[-a fix_bundle=BundleName] \
[-a fix_query=FixQueryFlags] ClientName
```

where FixKeywords are APAR numbers. FixBundlename is the object name of the fix_bundle resource; FixQueryFlags are optional flags to the fix_query operation, and ClientName is the client, group, or SPOT for which to display fix information.

Valid FixQueryFlags are as follows:

-a	Displays symptom text.
-c	Displays output in colon-separated format
-F	Returns failure unless all filesets associated with a fix are installed.
-q	Quiet option; if -q is specified, no heading is displayed.
-v	Verbose option

2.13.4.6 Client-initiated operations

There are a few NIM operations that can be performed from the NIM client as well as from the NIM master. An operation being performed from the NIM client can only target itself. All kind of NIM operations performed from a NIM client can be overwritten by the NIM master when using the force flag to perform an operation. The following commands are only for NIM client use. You can use these command line operations in scrips to administrate your machines in a more automated way.

Allocate NIM resources

To allocate NIM resources from the NIM client, use the following command:

```
# nimclient -o allocate -a attribute=value
```

To allocate the NIM resources spot, spot433, the lpp_source, lpp_433, and the bundle, ibmcxx, do the following:

```
# nimclient -o allocate -a spot=spot433 \
-a lpp_source=lpp_433 -a bundle=ibmcxx
```

List all available resources

To list all available resources for a machine when its NIM name is rs1230c, type:

```
# nimclient -l -L rs1230c
```

It is also possible to only list the information for a single NIM resource. Therefore, use the `nimclient` command with the -t parameter and the resource type you want to list. For example, to list the available spot resources for the NIM client, rs1230c, type:

```
# nimclient -l -L -t spot rs1230c
```

List all allocated resources

To list all allocated NIM resources for a NIM client, type in:

```
# nimclient -l -c resources ClientName
```

Deallocate NIM resources

To deallocate all allocated resources, use the following command:

```
# nimclient -o deallocate -a attribute=value
```

For example, if you want to deallocate the spot resource named spot433 for the NIM client, rs1230c, type in:

```
# nimclient -o deallocate -a spot=spot433
```

The bosinst operation

To start a BOS installation from the client, you must allocate the necessary resources, and then perform the bos_inst operation.

Perform the BOS installation with the following command:

```
# nimclient -o bos_inst
```

The cust operation

Two steps are necessary to perform a cust operation from a NIM client: First, allocate the necessary NIM resources. Second, perform a cust operation. For example, to install all the latest updates from the lpp_source fix_433, do the following:

1. Allocate the resource

```
# nimclient -o allocate -a lpp_source=fix_433
```

2. Perform the cust operation

```
# nimclient -o cust -a fixes=update_all
```

Boot from a diagnostic image

The machine can be enabled to boot a diag image. Two steps are necessary to do this:

1. First enable the diag boot image

```
# nimclient -o diag -a spot=spotName
```

2. Boot the machine over the network manually

Boot into the maintenance mode

To boot a NIM client into the maintenance mode, do the following:

1. Perform the following command:

```
# nimclient -o maint_boot -a spot=spotName.
```

2. Perform a network boot of the client manually.

This function is available for AIX 4.2 or later.

Reset the NIM state

To reset the NIM state of the NIM client, use the following command:

```
# nimclient -o reset
```

If the Mstate of the NIM client prevents that operation, you can force the following command:

```
# nimclient -o reset -F
```

This function is available for AIX Version 4.2 or later.

Display the contents of a NIM resource

You can display the contents of any NIM resource. This function is available for AIX Version 4.2 or later.

From smitty

1. Open smitty with the `nim_c_showres` fastpath.
2. Select a resource from the list.

From the command line

```
# nimclient -o showres ResourceName
```

Set time and date to that from the NIM master

With the `nimclient -d` command, you can set the date and time to that from the NIM master.

Add a machine to the NIM environment

To add a machine to the NIM environment, perform the following steps:

1. Open smitty with the `niminit` fastpath.
2. Type in the machines name in the NIM environment.
3. Select a network devices.
4. Type in the hostname of the NIM master.
5. Change the other values or accept the default values.

Enable or disable push permission for the NIM master

You can use smitty as well as the command line to enable or disable push permission for the NIM master.

From smitty

To enable or disable push permission for the NIM master from smitty, perform the following steps:

1. Open smitty with the `nim_perms` fastpath.
2. Set the field Network Install Master Permissions to **yes** if you want to enable the push permission, or to **no** if you want to disable the push permission.

From the command line

To enable or disable push permission for the NIM master from smitty, perform the following steps:

To enable the push permission for the NIM master, use the following command:

```
# nimclient -p
```

To disable the push permission for the NIM master, type:

```
# nimclient -P
```

Note

The NIM master can overwrite the disabled push permission by using the force flag -F.

Change own definition in NIM database

You can change the information in the NIM database for the client by using the following command:

```
# nimclient -o change -a attribute="value"
```

The following attributes are optional:

- -a if#=<value>
- -a control=<value>
- -a comments=<value>
- -a cpuid=<value>
- -a err_info=<value>
- -a info=<value>
- -a ring_speed=<value>
- -a cable_type=<value>
- -a iplrom_emu=<value>
- -a net_definition=<value>
- -a netboot_kernel=<value>
- -a new_name=<value>
- -a platform=<value>
- -a client_alloc=<value>

An example of changing the CPU-ID of the NIM client from the existing value to the value `01234567890` is the following command:

```
# nimclient -o change -a cpuid="01234567890"
```

2.13.4.7 Using NIM machine group and resource group

A customer who is using a large NIM environment has different kinds of installations on the machines. The installations are different in the way of having additional third-party software installed. A different kind of installation is necessary for each kind of department. Some machines are used from more than one department; other machines are only used from one department. All software products can be installed on the newest AIX Version 4.3; so, the only differences between the machines are different filesets, which are in addition to the operating system.

In order to manage these different kinds of installations in a better way, the customer has created NIM machine groups. A customer has created one group for each additional software. If a machine has installed this special software, it is assigned to that group. Some machines are assigned to more than one group because they have more than one software installed. After that procedure, the customer is able to perform special software updates and software customization operations on those groups.

To install some of these additional software products, it is necessary to create more than one NIM resource including all the different parts of that software. The customer has created NIM resource groups for that kind of software. Each resource group includes the required resources to install the base operating system and the special software required for the different departments; so, the customer was able to install the machine by allocating one resource group.

These two steps were very helpful in order to install many machines with many additional software products.

For more information about how to use NIM groups, see Section 1.2.4, “Groups” on page 22.

2.13.4.8 Defining NIM environment by using the nimdef command

A customer who uses a NIM environment to administrate his or her RS/6000 machines has to add 500 new delivered machines to his or her NIM environment. The machines are two different kinds of hardware: 400 machines are 43P model 150s, and 100 machines are model F50s. The 43P model 150s are using token-ring as a network interface, and the model F50s are using ethernet.

The process to define these machines as NIM clients must be as short as possible.

The use of the `nimdef` command is a very good option to do this job in a short amount of time. Therefore, a NIM client definition stanza file must be created. This file includes all necessary information to define all machines as NIM clients. After the definition file is created, the `nimdef` command can be performed. This command parses the definition file and builds the required commands to add the machines into the NIM environment. This command can also create NIM networks and NIM groups. It is possible to redirect the output of the `nimdef` command into a file. In that case, no NIM operation is performed, but all necessary commands to do this are redirected into a kornshell script file. This file can be executed at a later time.

The structure of the nimdef stanza file for the customer was like the example in Figure 29 on page 196. In that example, the first steps are setting default values for all NIM clients. Then, other default values are set for the 43P model 150 machines. After that, a second group is added for the first 350 43P model 150 machines, and then, the machines (150nr001 up to 150nr350) are added. After that, the machine group is reset, and a new default machine group is set. The next 50 43P model 150s are added (150nr351 up to 150nr400).

The next step is to reset the default settings that are not correct to add the F50 machines. The last step is to add all 100 model F50 machines.

```
# more nimdef.clients
# Stanza file to add 400 new model 150 and 100 new model F50:
# Set default values:
default:
   machine_type    = standalone
   subnet_mask     = 255.255.0.0
   gateway         = gateway1
   platform        = chrp
   machine_group   = all_machines
# Set additional defaults for model 150:
default:
   network_type    = tok
   ring_speed      = 16
# Add all model 150: Names are 150nr001 up to 150nr400
# Take all defaults, and set additional group SWgrpA for first 350 model 150:
default:
   machine_group   = SWgrpA
# now add the machines:
150nr001:
150nr002:
...
150nr349:
150nr350:
# Reset group and set new machine group for the next 50 model 150:
default:
   machine_group   =
   machine_group   = all_machines
   machine_group   = SWgrpB
# now add the last 50 model 150:
150nr351:
...
150nr400:
# Change some defaults for model F50:
default:
   network_type    =
   ring_speed      =
   network_type    = ent
   cable_type      = bnc
   machine_group   =
   machine_group   = all_machines
   machine_group   = SWgrpA
   machine_group   = SWgrpB
# Now, add machines f50nr001 up to f50nr100
f50nr001:
...
f50nr100:
#
```

Figure 29. Example of a NIM definition stanza file

To execute the `nimdef` command, there are four flags possible:

-c Generates commands from a client definition file. This flag processes the definition file and generates the commands to add the definition. The commands are not invoked, but displayed. You can redirect this output into a file and invoke it at a later time.

-d Defines machines from a definition file. This flag processes the definition file and invokes the commands to add the definitions to the NIM environment.

-f You must specify a filename when using the -f flag, which includes the definition stanzas.

-p Displays a preview of the client definition file. This flag processes the definition file but does not add machines to the NIM environment.

Note

We recommend that you specify the -p flag by processing the `nimdef` command to verify that all stanzas are correct before using it for adding machines to the existing NIM environment.

Here are a few examples of how to use the `nimdef` command:

Preview the client definition file `nimdef.clients`:

```
# nimdef -p -f nimdef.clients
```

Add the NIM clients to the NIM environment described in the file `nimdef.clients`:

```
# nimdef -d -f nimdef.clients
```

Create a kornshell script file called `nimdef.ksh` to add the NIM clients described in the client definition file `nimdef.clients`:

```
# nimdef -c -f nimdef.clients > nimdef.ksh
```

2.13.5 Installation multiple level of AIX

A customer has about 450 RS/6000 systems, and all are running AIX 4.1.5. To update this computing environment, the customer ordered new hardware: 5 model S7As, 10 43P model 260s, and 35 43P model 150s. This new hardware must run at least AIX 4.3.0. The customer could not migrate all of the RS/6000 systems to the new AIX level 4.3 because of the software dependencies of their programs. The customer must now also support at least two levels of AIX in his or her NIM environment. They set up their NIM environment to support a multilevel environment of AIX versions. The customer performed the following steps to solve the problem:

1. Since the NIM master is a model F50, there is enough free disk space and computing power on the NIM master available; so, the customer need not change the master.
2. The customer decided to migrate the existing NIM master to the newest available AIX version in order to avoid problems that may occur during the

use of different levels of AIX. The customer did a migrate and not a new and complete overwrite, because of the existing NIM environment.

3. Since the NIM resources for AIX Version 4.1 are still available, the customer created only the NIM resources for the new AIX level 4.3.2.
4. The customer added the new machines to his/her NIM environment by using a client definition stanza file and performing the `nimdef` command.
5. The customer created a new NIM group for the new machines.
6. The customer installed all new NIM clients by using the newly-created NIM resources with AIX Version 4.3.2.
7. After the migration and addition of the new machines, the customer made a NIM backup of the existing NIM database.

2.13.6 Different physical networks on the same logical subnet

Often, there are different kinds of physical networks in the same logical network. NIM provides a function to build a single NIM network including different kinds of networks. These networks use bridges to connect two segments that have different kinds of data link protocols. For example, an ethernet and a token-ring network segment can be used to build a single logical subnet.

Since a single NIM network is used to represent the logical network, the other_net_type attribute is reserved to define the other kind(s) of network interfaces that are used in the existing network. You can add the other_net_type attribute to an existing network definition.

If you define a NIM client with a different kind of network adapter without having set the correct other_net_type attribute, the NIM master will set the wrong network interface information to the NIM clients definition. Then, during the installation of this new client, the process will fail or hang because a wrong boot image has been allocated.

For example, you have only one physical network, a token-ring network including all NIM clients and the NIM master. After a while, it becomes necessary to add an ethernet segment into that subnet by using a bridge. All machines in that second network should become NIM clients managed by the already-existing NIM master. This current situation is shown in Figure 30 on page 199.

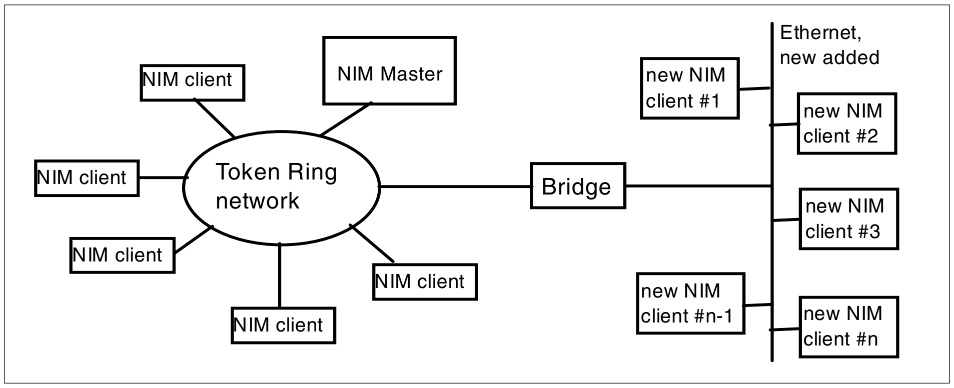


Figure 30. Example of a logical network including two different physical networks

To change the existing NIM environment and add the new NIM clients, the following steps should be performed.

1. Add information about the new physical network into the network definition of the existing network. You can use the Web-Based System Manager, smitty, or the command line to add the information.

   From Web-Based System Manager

   Use the following steps to add the new network (only for AIX 4.3):

   a. Start the Web-Based System Manager.
   b. Open the NIM container.
   c. Open the Networks container.
   d. Select the network to which you want to add another network type.
   e. Go to **Selected** --> **Properties**. The NIM Networks - Properties page will appear.
   f. From Other Network Type, select the new network type to be added.

   From smitty

   You can also use the smitty to add the new network (only for AIX 4.3):

   a. Open smitty with the `nim_chnet` fastpath.
   b. Select the NIM network where you want to add the new network type.
   c. A menu will appear. Specify the new additional network type to be supported from Other Network Type.

   From the command line

   To define additional network types, use the following command:

```
# nim -o change -a other_net_typeSequenceNumber=NewNetworkType NetName
```

Where `SequenceNumber` is the number of additional network types. The first additional network has the other_net_type1 attribute and the next one other_net_type2, and so on.

For example, to add an ethernet and an FDDI network to a token-ring named tok1, use a command similar to the following:

```
# nim -o change -a other_net_type1=ent -a other_net_type2=fddi tok1
```

Note

When you use the command line, it is only possible to add more than one additional network type to a NIM network.

2. Add all clients to the NIM environment by using their own network interface definition.

2.13.7 Using NIM script resource for customization

A customer has all kinds of RS/6000 systems in his or her NIM environment. The range of available main memory in those machines is, depending on the model, from 256 MB (Model 41T) up to 32 GB (Model S7A). Also, the available space of hard disks varies from 1.9 GB up to 28x9.1 GB. All these different machines should be installed from one single mksysb image. The image was created on a fully installed machine with very small file system sizes and a single small paging space in order to be installable on the smallest machines. After the installation, the file system sizes and the paging space has to be fit onto the machine’s environment.

This procedure to fit the new installation or an update onto your NIM client is called customization in the NIM environment. To customize your new installed NIM client, you can use the NIM resource script. After a cust operation, a NIM script can also be processed. A NIM script resource is a user-defined shell script. If a script is allocated, the script resource is always run after software installation is performed with a cust or bos_inst operation. You can allocate more than one script for customization, but the order in which the scripts will be executed is unpredictable.

You need not use a customization script after the installation of NIM clients. You can also perform a normal kornshell script manually, or you can use single commands to fit your new installed machine. But, if you do, it will be very helpful in order to save administration time.

Let us describe a few examples of how the customer uses the customization script in order to set up the newly-installed hardware automatically.

2.13.7.1 Example 1: Script for TCP/IP setup in AIX 4.1 with DNS

Unless no_nim_client=yes is specified, NIM will invoke mktcpip to configure TCP/IP on the newly-installed client. If you are using NIM clients running AIX Version 4.1 and DNS services for name resolution, you should manually customize the DNS setup after the installation. With a script like that shown in Figure 31, you can configure the DNS automatically.

```
# more /export/scripts/cust_AIX41
#!/bin/ksh
# Customizion script to establish nameserver and
# DNS domain name, and configure the routing table.
#
# Set name server and domain name
if [[ -f /etc/resolv.conf ]]
then
   /usr/sbin/namerslv -E '/etc/resolv.conf.sv'
fi
/usr/sbin/namerslv -a -i '9.3.78.11'
/usr/sbin/namerslv -c 'nameresolve.hallo.com'
#
# Flush routing table and add default route
/etc/route -n -f
odmdelete -o CuAt -q "name=inet0 and attribute=route"
chdev -l inet0 -a route=net,,'0','9.3.87.11'
```

Figure 31. Script to configure DNS after AIX 4.1 installation

Note

As described in this section, the DNS setup after installation is only necessary for AIX Version 4.1. For later AIX versions, you can use a special NIM resource, called resolv_conf resource, to set up the DNS function.

2.13.7.2 Example 2: Script for system environment customization

With this script, all NIM clients are customized after the NIM installation. The script does the following steps: It calculates the needed paging space depending on the available free disks and the main memory of the machine. It creates the new paging space with the calculated amount of space and sets the parameters for automatic use at next reboot.

The customizing script also does the following: Since different hardware provides very different computing power, and some machines are SMP machines, it became necessary to increase some local file systems in order to support the processed batch programs in a better way. Consequently, the /tmp and /var file systems were increased, and a new file system for caching very large temporarily files was created. Before increasing and creating the file systems with the following command, the script calculated the available space that was left on the hard disks.

After setting the new paging space and modifying the file systems, a few settings are made according to the system environment. The maximum number of processes allowed per user was increased to 1024. Also, the number of maximum licensed users was, according to the existing licenses, increased to support 64 simultaneous users.

See Figure 32 on page 203 for some example commands that you can execute during system customization.

```
# more /export/script/cust
#!/bin/ksh
# Customization script to fit the installed machine
#
# Set variables needed for the customization
LOG="/tmp/cust.log"
NewDataSize="100"
NewSizeTmp="300000"
NewHardDisk="hdisk2"
NewPP="128"
MaxProc="1024"
LicUser="64"
...
################################################################
# Start the customization:
#
...
#
/usr/bin/echo "Add new logical volume named data" >>$LOG
/usr/sbin/mklv -y data rootvg $(NewDataSize) $(NewHardDisk) >>$LOG 2>>$LOG
/usr/bin/echo "Add new filesystem /data into lv data" >>$LOG
/usr/sbin/crfs -v jfs -d data -m /data -A yes >>$LOG 2>>$LOG
/usr/bin/echo "Mount new filesystem" >>$LOG
/usr/sbin/mount -v jfs /dev/data /data >>$LOG 2>>$LOG
#
/usr/bin/echo "Increase filesystem /tmp" >>$LOG
/usr/sbin/chfs -a size=$(NewSizeTmp) /tmp >>$LOG 2>>$LOG
#
/usr/bin/echo "Add new hard disk to rootvg" >>$LOG
/usr/sbin/extendvg -f rootvg $(NewHardDisk) >>$LOG 2>>$LOG
/usr/bin/echo "Create new paging space on that hard disk" >>$LOG
/usr/sbin/mkps -s $(NewPP) -a'' rootvg $(NewHardDisk) >>$LOG 2>>$LOG
#
/usr/bin/echo "Set max. number of processes allowed per user" >>$LOG
/usr/sbin/chdev -l sys0 -a maxuproc=$(MaxProc) >>$LOG 2>>$LOG
#
/usr/bin/echo "Set autostart after crash to true" >> $LOG
/usr/sbin/chdev -l sys0 -a autorestart='true' >>$LOG 2>>$LOG
#
/usr/bin/echo "Set max. number of licensed user" >>$LOG
/usr/bin/chlicense -u $(LicUser) >> $LOG 2>>$LOG
...
#
```

Figure 32. Parts of a customization script

Chapter 3. NIM in an SP environment

This chapter will look at the way NIM operates in an SP environment. Although the concepts behind NIM remain the same, the way in which we configure and manage it is fundamentally different.

While the components that make up an SP environment are briefly described and discussed, this chapter assumes that the reader already has some familiarity with the operations and management of an SP system.

3.1 General configuration

An SP system is made up of the following components:

- Frames
- Nodes
- Switch (optional)
- Peripheral Devices (optional)
- Control Workstation (CWS)

An SP system can consist of anywhere from two to over one thousand nodes. The nodes are held within frames. A full-height frame contains sixteen slots that can accommodate up to sixteen thin nodes. Different types of nodes can have different hardware architectures and, thus, have different space considerations. While a thin node will take up a single slot, a wide node will take two slots, and a high node will take four slots; these different types of nodes can all be integrated within the one frame.

Although an SP system can be viewed as a single entity and its nodes configured to run in parallel on single large-scale jobs, it is important to remember that each node is basically a stand-alone RS/6000. Every node has, at least, one internal disk, its own processor(s), memory, and a copy of the AIX operating system.

What each node will not have is a console, mouse, keyboard, or any type of front panel display, such as a key switch or LED display. Each SP system will have a stand-alone RS/6000 machine acting as a single point of control, the CWS.

The CWS is linked to the nodes in several ways. The first is via a special RS-232 serial link to each frame, which attaches to a frame supervisor card. The frame supervisor card, in turn, attaches to a node supervisor card in each node. This allows the CWS to become a locally-attached console when needed and also to perform and monitor lower-level hardware tasks, such as turning the keyswitch or checking the status of the LEDs. The second connection is via the administrative ethernet network. This is a dedicated network between the nodes and the CWS and is used for installation and to carry SP-specific network traffic, such as the topology services daemons.

A typical single-frame SP system is shown in Figure 33.

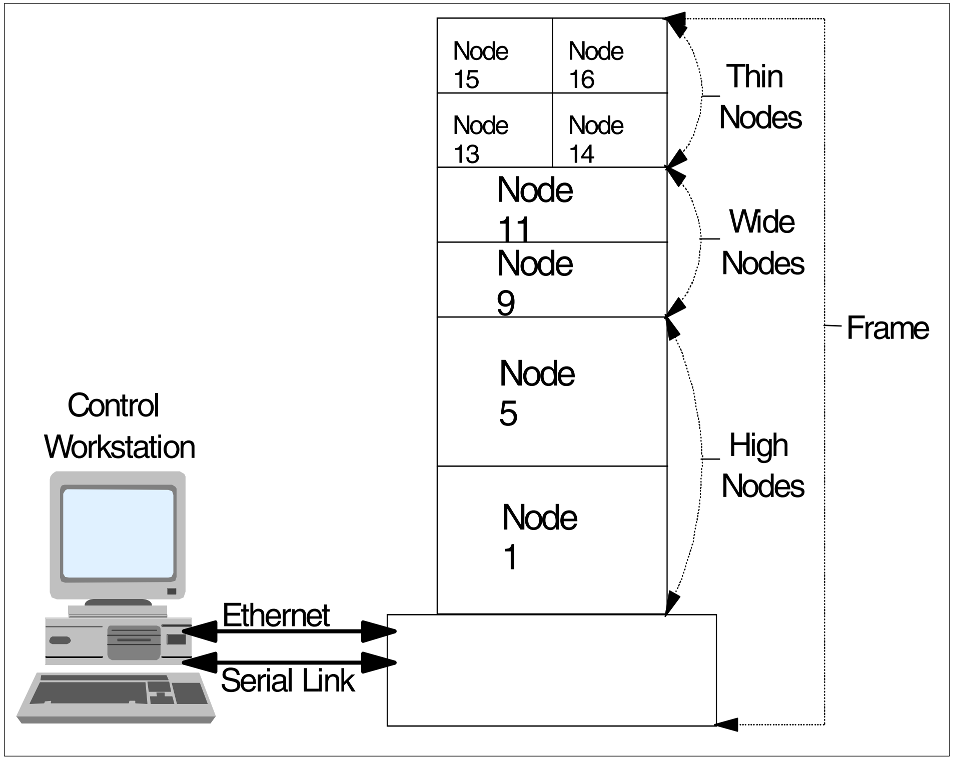


Figure 33. A typical single-frame SP system

Although it is possible to attach a tape or CD-ROM drive to an SP node, generally speaking, it will have no means of installation other than its network interface. This is why NIM is the only supported method of SP installation. However, because of the environment in which the SP runs, there are some key differences in the configuration of NIM and how an installation is performed.

3.1.1 NIM objects

The NIM objects used in an SP environment are the same as in a classic RS/6000 environment; it is, after all, the same version of NIM. The SP, however, has some more rigid rules about what is and what is not supported.

Each node is defined as a stand-alone machine; diskless or dataless nodes are not supported. Nodes must also be installed over the SP's internal ethernet network, although it would be possible to boot a node over, for example, token ring and install it, this is not supported.

The resource objects remain the same, though there are several scripts that are automatically allocated as resources when a node is installed, which would normally be an option on a classic RS/6000 environment.

Finally, the group resource is not used within the SP environment.

3.1.2 Differences with SP

The crucial difference in managing NIM in an SP environment is how the NIM objects are configured and managed. Whereas, in a classic RS/6000 environment, each object is either defined manually via smitty or nim -o define, in an SP environment, this is hidden behind the NIM wrappers, most significantly, a large Perl script called setup_server.

3.1.2.1 System Data Repository

The System Data Repository (SDR) is something unique to the SP. It contains SP-specific information in a central repository that resides on the CWS. It is held in plain ASCII text format under the /spdata/sys1/sdr/ directory structure.

All nodes within the SP system may need to query data or change an attribute of the SDR at some point; so, on the CWS, the sdrd daemon runs to handle requests from the nodes and itself (the CWS) and listens on TCP port 5712.

The activity of this daemon is also written to a log in /var/adm/SPlogs/sdr/sdrlog.*syspar_ip_addr.pid* where *syspar_ip_addr* is the IP address of the system partition, and *pid* is the process identifier (PID) of the SDR daemon.

It is similar in concept to the AIX Object Data Manager (ODM) and is a database containing *Classes*, *Attributes*, and *Objects.*

A Class is a type of device or feature, such as an adapter, frame, node, and so on. *Attributes* are details about a class; for example, some attributes of an adapter would be the adapter type, the network address, netmask, and so on.

Objects are a specific set of attributes; they have no unique identifier, but their combined attributes must be unique.

The reason we mention the SDR at this point is because it is crucial to the installation procedure. In order to install our SP system, we must initialize and populate the SDR with configuration information about the environmental details of the SP, the frame(s), and the nodes. When we then run the setup_server script for the first time, it queries the SDR in order to retrieve the relevant information to make and define the NIM objects necessary to go ahead with the installation.

This is a very broad overview of the SDR. To learn more about its structure and the commands that can be used to manually manipulate it, we recommend reading the following guides:

- *PSSP: Administration Guide*, SA22-7348
- *PSSP: Command and Technical Reference*, SA22-7351

3.1.2.2 Installation process

This is a condensed and very brief overview of the SP installation process concentrating on the steps linked with creating the NIM resources; for detailed information on the complete installation process, refer to the following guides:

- *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*, GA22-7281
- *PSSP: Installation and Migration Guide*, GA22-7347

The first part of the installation procedure deals with setting up the CWS and making sure it has the necessary prerequisites to actually act in this capacity. These initial steps are:

1. Update root's $PATH so that it will pick up SP-specific binaries.
2. Ensure that AIX, bos.net, and perfagent.tools are installed.
3. Connect the frames to the CWS, and configure the RS-232 serial link.
4. Configure and tune the network adapters (the PSSP installation and migration guide has some specific tuning recommendations).
5. Test network connectivity with ping.
6. Check that the necessary daemons are running. That is, check that the System Resource Controller (SRC) is active; check that inetd is running and that the entries for bootps and tftp are uncommented in /etc/inetd.conf.

7. Change the maximum number of runable processes per user to 256.
8. Tune various network options with `no -o`.
9. Define space for /tftpboot and the volume group for /spdata. (see Section 3.1.2.4, “Disk space considerations” on page 216).
10. Create the /spdata directory structure. (see Section 3.1.2.5, “Directory structure” on page 216).
11. Copy over the AIX LPP images.

Of these initial steps, the most significant action, as far as our NIM configuration is concerned, is the last one: Copying over the AIX LPP images.

In performing this step, we are moving the AIX LPP images over to our local disk ready to configure them as an lpp_source NIM resource. One example method of copying over the images would be to have an AIX installation disk in the CWS’ CD-ROM drive and type `smitty bffcreate`. When prompted, use */dev/cd0* as the INPUT device / directory for software, and, in the DIRECTORY for storing software package, type `/spdata/sys1/install/<name>/lppsource`, where `<name>` is the name you have called your lppsource directory. (see Section 3.1.2.5, “Directory structure” on page 216, for details on this naming convention).

The second part of the installation process deals with getting PSSP installed on the CWS. The steps involved in this procedure are as follows:

12. Copy the PSSP installation images over to disk.

    This step has some relevance to NIM because the script, pssp_script, which setup_server defines as a script object, remotely mounts the directory in which the PSSP images reside and installs PSSP on the nodes as part of the customization. Therefore, we must bffcreate the images in the correct directory (see Section 3.1.2.5, “Directory structure” on page 216, for guidelines for this).

13. Copy a basic AIX (mksysb) image to disk.

    Again, this step has particular relevance to NIM because the mksysb image will later be defined as a NIM mksysb object. The SP hardware is shipped with media that contains an *spimg* installp image. This can be installed via `smitty install_latest`, or `installp` commands. Alternatively, you can use your own mksysb image (in which case, you must copy this into the /spdata/sys1/install/images directory).

14. Install PSSP on the CWS. We do this using smitty install or installp using /spdata/sys1/install/pssplpp/PSSP-<x.y> as the input device, where <x.y> is the level of PSSP we are installing. For a list of minimum required PSSP

filesets, refer to Section 3.1.2.7, “Minimum required PSSP filesets” on page 218.

15. Initialize SP Authentication Services. In order to do this, we run the program /usr/lpp/ssp/bin/setup_authent. This initializes the Kerberos database and environment.

16. Run install_cw. This completes the configuration of the CWS, and this final step, among other things, installs the PSSP smitty panels, configures the SDR, and starts the SP daemons.

Now that the installation of the CWS is complete, we can go ahead and enter our configuration details for the frame(s) and nodes into the SDR. There are several ways of entering data: Through the command line, through the Perspectives panel, or via smitty. In this brief overview of the installation process, we will only cover the method using smitty.

17. Enter Site Environment Details

    We enter the site environment details by entering `smitty enter_data` and then choosing **Site Environment Information**. Of the myriad of options we can change, the two that are most likely to need attention are the Default Network Install Image and Control Workstation LPP Source Name.

    The default network install image relates to the default mksysb image to be used and must reside in the /spdata/sys1/install/images directory. If we have installed an image from spimg or used one of our own images, we must put its filename in here.

    The Control Workstation LPP Source Name is the name of the directory the CWS will use in order to install the NIM filesets on itself - At this stage, remember that NIM has not yet been installed). Notice that, like the default network install image option, we do not specify the fully-qualified directory name; this is because the LPP source name must reside in the /spdata/sys1/install directory. An example of the site environment smitty screen is shown in Figure 34 on page 211.

```
                         Site Environment Information

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
  Default Network Install Image                       [bos.obj.ssp.433]
  Remove Install Image after Installals                false               +

  NTP Installation                                     consensus           +
  NTP Server Hostname(s)                              [""]
  NTP Version                                          3                   +

  Automounter Configuration                            true                +

  Print Management Configuration                       false               +
  Print system secure mode login name                 [""]

  User Administration Interface                        true                +
  Password File Server Hostname                       [cws1]
  Password File                                       [/etc/passwd]
  Home Directory Server Hostname                      [cws1]
  Home Directory Path                                 [/home/cws1]

 File Collection Management                            true                +
 File Collection daemon uid                           [102]
 File Collection daemon port                          [8431]                #

  SP Accounting Enabled                                false               +
  SP Accounting Active Node Threshold                 [80]                  #
  SP Exclusive Use Accounting Enabled                  false               +
  Accounting Master                                   [0]

  Control Workstation LPP Source Name                 [aix433]
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 34. Entering the Site Environment Information

18. Enter Frame Information and Reinitialize the SDR.

    From `smitty enter_data`, this time choose **SP Frame Information**, and enter your starting frame, the number of frames you wish to define, and the starting tty port to use (each frame must have a separate tty port into the CWS). Enter `Yes` in the Re-initialize the System Data Repository option.

19. Verify System Monitor Installation by running smitty SP_verify and choosing **System Monitor Configuration**.

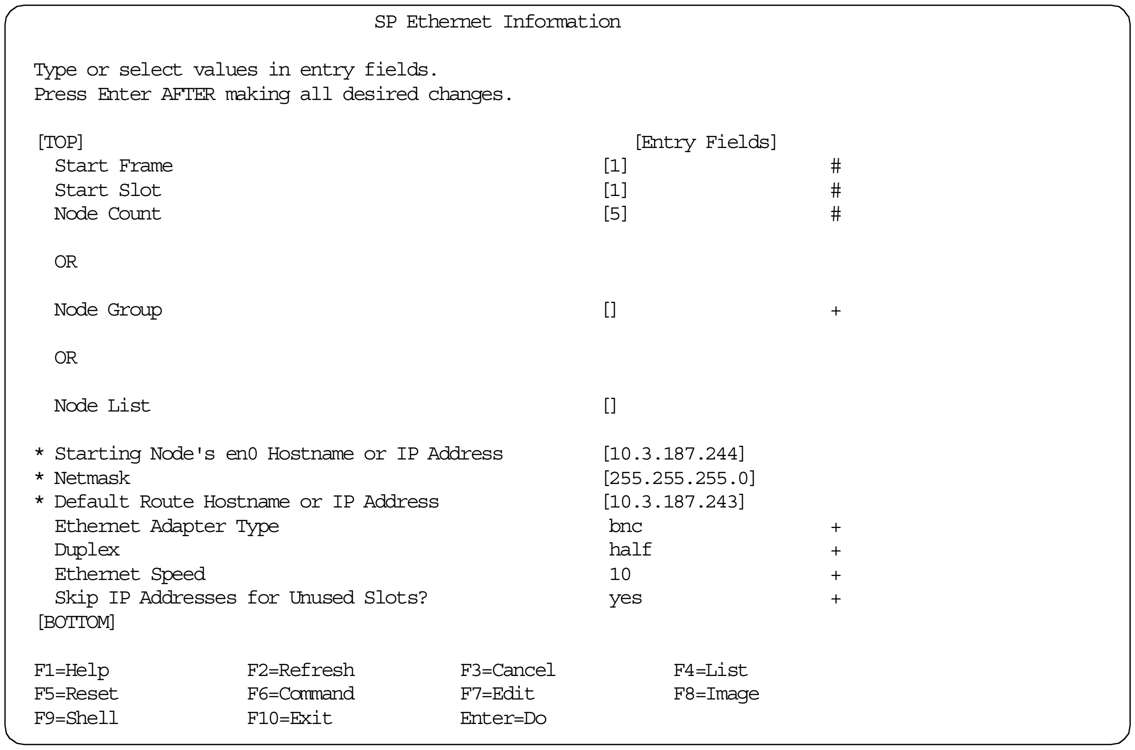
20. Verify Frame Information. We can do this by starting Perspectives and opening up the node pane, which should give us a visual representation of the Frame(s) we have defined along with the nodes within the frame(s) that have been detected.

21. Update the State of the Supervisor Microcode. Enter `smitty supervisor`, and choose **Check For Supervisors That Require Action (Single Message Issued)**. If action is required, you may need to apply PSSP fixes and use the **Update *ALL* Supervisors That Require Action (Use Most Current Level)** option.

22. Enter the required Node Information.

    Enter `smitty node_data` and choose **SP Ethernet Information**. The data we enter here is used to add IP address information to the node objects in the SDR and is used by NIM during the customization phase.

    There are a number of ways of entering the node information: Start Frame, Start Slot, Node Count information, Node Group, or Node List. In the following example, illustrated in Figure 35 on page 213, we have used the Start Frame, Start Slot, and Node Count options to define a single-frame five-node machine.



```
                          SP Ethernet Information

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
  Start Frame                                       [1]                  #
  Start Slot                                        [1]                  #
  Node Count                                        [5]                  #

  OR

  Node Group                                        []                   +

  OR

  Node List                                         []

* Starting Node's en0 Hostname or IP Address        [10.3.187.244]
* Netmask                                           [255.255.255.0]
* Default Route Hostname or IP Address              [10.3.187.243]
  Ethernet Adapter Type                              bnc                 +
  Duplex                                             half                +
  Ethernet Speed                                     10                  +
  Skip IP Addresses for Unused Slots?                yes                 +
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 35. Entering the SP Ethernet information

The other fields of interest we changed are:

- **Starting Node's en0 Hostname or IP Address -** We have entered the first node’s en0 IP address. PSSP will number the other four nodes consecutively.
- **Default Route Hostname or IP Address** - We have given the en0 address of the CWS, which will be used to create the default route on each node at customization.
- **Skip IP Addresses for Unused Slots?** As mentioned earlier, an SP node can take up a single slot, but others can take two or four slots. PSSP numbers these nodes to coincide with their starting slot number; so, for example, if we had two high nodes that took up four slots each and two thin nodes that took up a single slot each, our node numbers would be 1, 5, 9, and 10, even though we have only four nodes. Answering *Yes* to this option can be useful if you are able to link an IP address to the node number.

For example, in the 4-node system we mentioned, if we skip IP addresses for unused slots and give our first node an address of 9.3.187.1, node 5 would take the address 9.3.187.5, node 9 would take 9.3.187.9, and so on. If we chose not to skip IP addresses, node 5 would be assigned 9.3.187.2, node 9 would be assigned 9.3.187.3, and so on.

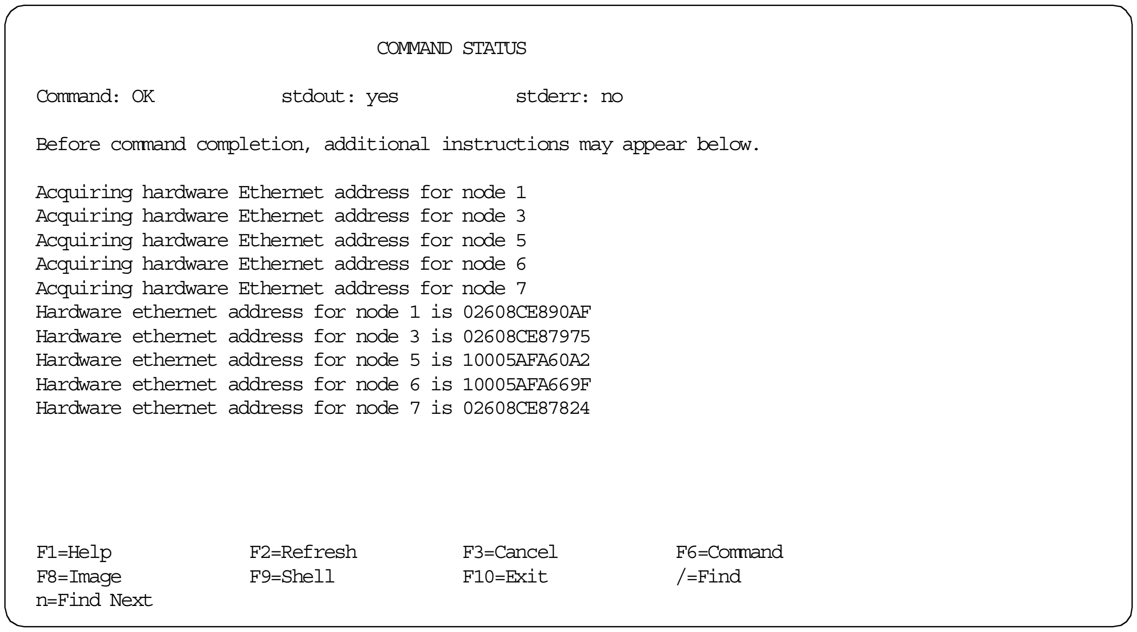
23. Acquire the Hardware Ethernet address

    As previously discussed, part of the definition of a NIM client is the MAC address of its primary adapter. On the SP, we must boot over the administrative ethernet network, but the same rule applies.

    One of the features of the link between the CWS and the frame supervisor cards on the SP frame(s) is that we can acquire these ethernet addresses by running one simple command.

    Enter `smitty node_data` and choose **Get Hardware Ethernet Addresses**. As before, enter a start frame, start slot, node count option, node group, or node list. Depending on the type of nodes you have and on how many nodes you perform this action, you will get the hardware ethernet addresses back within a few minutes. Some typical results from the command are shown in Figure 36 on page 215.

    This should never be performed on a production SP because, in order to get these addresses, the nodes are reset and go partially through a network boot before being powered off.



```
                            COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

Acquiring hardware Ethernet address for node 1
Acquiring hardware Ethernet address for node 3
Acquiring hardware Ethernet address for node 5
Acquiring hardware Ethernet address for node 6
Acquiring hardware Ethernet address for node 7
Hardware ethernet address for node 1 is 02608CE890AF
Hardware ethernet address for node 3 is 02608CE87975
Hardware ethernet address for node 5 is 10005AFA60A2
Hardware ethernet address for node 6 is 10005AFA669F
Hardware ethernet address for node 7 is 02608CE87824





F1=Help             F2=Refresh          F3=Cancel           F6=Command
F8=Image            F9=Shell            F10=Exit            /=Find
n=Find Next
```

Figure 36. Acquiring the nodes ethernet MAC addresses

We have done almost everything necessary in order to start installing the SP system. From this point on, most of the steps are optional and will depend upon your own configuration.

24. Configure additional adapters for nodes. If you have additional adapters on the nodes that you want configured during customization, for example, if you have a switched SP system, perform this action using `smitty node_data`, and choose **Additional Adapter Information**. Entering data in fields is very similar to configuring the SP ethernet information.
25. Configure Initial Host Names for Nodes. By default, the hostname on each node will be set to whatever the en0 address resolves to on the CWS. Use this if you either want a different host name or if you want to use short hostnames (the default is long, that is, fully-qualified, hostnames). To perform this action, enter `smitty node_data`, and choose **Hostname Information**.
26. Select and Enable authentication methods. In this step, we need to decide which authentication methods we are going to use for remote commands, our choices being Kerberos Versions 4 or 5 or standard authorization.
27. Start Partition-sensitive Subsystems. Enter `syspar_ctrl -A.`. This starts daemons, such as Topology services and Host responds.

There are many additional steps that can be followed to further customize the nodes, most significantly setting up the switch (if your SP system has one). However, at this point of the installation, we now have all the node and frame information in the SDR. The nodes will default to the default mksysb and lppsource we entered in the site environment information. They will also default to install on hdisk0 using the version of PSSP we installed on the CWS.

We can change this information with the `spbootins` or `spchvgobj` commands (see Section 3.3.1.12, “spbootins” on page 241, and Section 3.3.1.13, “spchvgobj” on page 242). At this point, we are ready to run setup_server for the first time.

3.1.2.3 Boot/Install server requirements

By default, a single frame SP system will configure the CWS as the only Boot/Install server (BIS) - the NIM Master.

On a multiple frame SP system, by default, the CWS will act as the BIS for the first frame, and then the first node in each additional frame will be configured to act as a BIS for the rest of the nodes in its own frame.

The reason for this is the bandwidth of the administrative ethernet network. Though a limit of thirty hosts per BNC segment exists, when netbooting multiple hosts over the same physical network, it can become very congested and practically unusable. We recommend a maximum netbooting limit of eight nodes simultaneously.

3.1.2.4 Disk space considerations

Space for the boot images requires from 25 MB to 50 MB per lppsource level supported. It depends on the AIX version. We recommend that you either have a lot of space available in / (root) or make /tftpboot a separate file system.

/spdata, which is the top of the directory structure that actually holds all the installation images and SPOTs, should be at least 2 GB in size. Once again, if you intend to support multiple levels of AIX and/or PSSP, you will need to increase this. We recommend making /spdata a separate volume group.

3.1.2.5 Directory structure

As mentioned earlier, as part of configuring the CWS, the administrator will need to create a directory structure under /spdata. The specific directories of interest to NIM lie under the /spdata/sys1/install directory. Figure 37 on page 217 shows a typical /spdata directory structure.

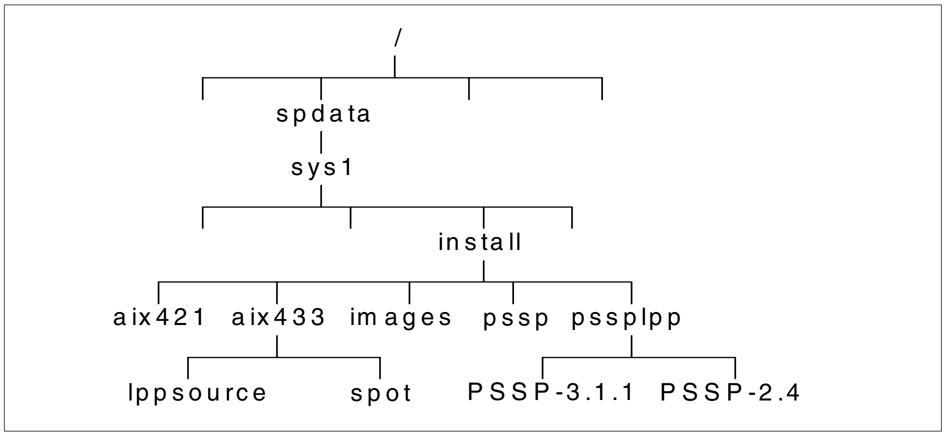


Figure 37. A typical /spdata directory structure

As part of the installation process, we bffcreate the AIX installation images from installation media, such as tape or CD-ROM. Under the /spdata/sys1/install directory, we need to make a directory that corresponds to the level of AIX we are copying over and place the installation images in the lppsource directory under this.

If we do not want to create a specific directory for this, PSSP expects the installation images in /spdata/sys1/install/default/lppsource; but, note that we must still create this directory ourselves. As illustrated in our example directory, it is perfectly allowable to support multiple versions of AIX.

The /spdata/sys1/install/images directory holds the mksysb images. Minimum installation images are shipped on the PSSP media, or you can add your own mksysb images here for installation. mksysb images are of particular importance to the SP because the NIM installation is based upon installing from an mksysb image and then customizing the node with data the CWS holds in the SDR.

The /spdata/sys1/install/pssplpp holds sub-directories that contain bffcreated versions of the PSSP filesets. The naming convention for these subdirectories is PSSP-*<x.y>*, where *<x.y>* is the level of PSSP. In the preceding example, we have both PSSP-3.1.1 and PSSP-2.4. As with lppsource, multiple levels of PSSP are supported.

The /spdata/sys1/install/pssp directory holds some additional NIM scripts, such as pssp_script. It also holds the scripts that make the migrate, prompt, and noprompt bosinst_data NIM objects.

3.1.2.6 Additional required AIX LPPs

The perfagent.server file set is part of the Performance Aide for AIX (PAIDE) feature of the Performance Toolbox for AIX (PTX), a separate licensed program product (LPP). The perfagent.tools fileset is part of base-level AIX 4.3.2 and later.

This product provides the capability to monitor your SP system's performance, collects and displays statistical data for SP hardware and software, and simplifies run-time performance monitoring of a large number of nodes.

The perfagent.server and perfagent.tools file sets must also be copied to all of the lppsource directories on the control workstation of any SP that has one or more nodes at PSSP 2.2 or later. The level of PAIDE copied to each lppsource directory must match the level of AIX in that directory.

3.1.2.7 Minimum required PSSP filesets

The Parallel System Support Programs (PSSP) are a separate LPP, which provides a suite of applications to effectively manage an SP environment.

PSSP is made up of a number of filesets. Some are optional, and some are required. The minimum required PSSP filesets required on the CWS are listed in Table 23.

Table 23. Minimum required PSSP filesets

Fileset	Description
rsct.basic.hacmp	RS/6000 Cluster Technology basic function (HACMP realm)
rsct.basic.rte	Cluster Technology basic function (all realms)
rsct.basic.sp	RS/6000 Cluster Technology basic function (SP realm)
rsct.clients.hacmp	RS/6000 Cluster Technology client function (HACMP realm)
rsct.clients.rte	Cluster Technology client function (all realms)
rsct.clients.sp	RS/6000 Cluster Technology client function (SP realm)
ssp.authent[1]	SP Authentication Server
ssp.basic	SP System Support Package
ssp.clients	SP Authenticated Client Commands
ssp.css[2]	SP Communication Subsystem Package

Fileset	Description
ssp.ha_topsvcs.compat	Compatibility for ssp.ha and ssp.topsvcs clients
ssp.perlpkg	SP PERL Distribution Package
ssp.sysctl	SP Sysctl Package
ssp.sysman	Optional System Management programs
ssp.top[2]	SP Communication Subsystem Topology Package
[1]Only necessary if the CWS will act as the Kerberos authentication server. [2]Only necessary if the SP system has a switch.	

3.1.2.8 SP Perspectives

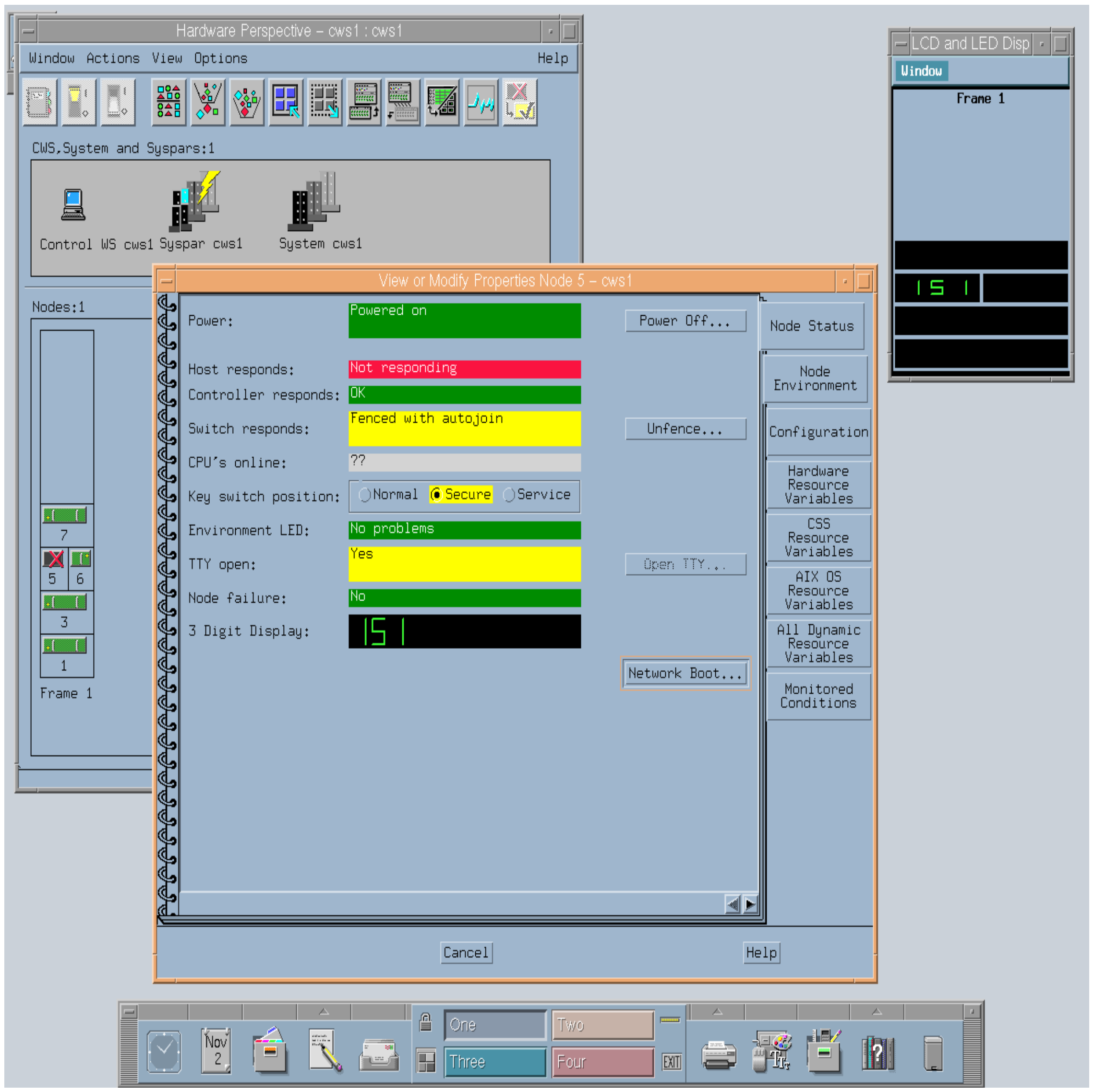
PSSP offers two programs that assist in monitoring and administrating the SP system. The first of these is called Perspectives.

Perspectives uses a Graphical User Interface (GUI) in order to provide a simple way of performing some complex monitoring and administration tasks. Among the things perspectives can do are:

- Monitor and Control Hardware
- Create and Monitor System Events
- Define and manage Virtual Shared Disks (VSDs)
- Generate and save system partition configurations
- Set up performance monitoring hierarchies and archiving

Because we are concentrating on the configuration and management of NIM, the part of perspectives that is of most use to us is called SP Hardware Perspective.

Using Hardware Perspectives, we can view a graphical representation of our full SP system, the LED displays of the nodes if they are powered on, the state of the key switch, and so on, thus, making Perspectives a valuable problem determination tool. Figure 38 on page 220 shows an example Perspectives GUI.



Power: Powered on
Host responds: Not responding
Controller responds: OK
Switch responds: Fenced with autojoin
CPU's online: ??
Key switch position: Normal [✓ Secure] Service
Environment LED: No problems
TTY open: Yes
Node failure: No
3 Digit Display: 15 1

Figure 38. Example of the Hardware Perspectives GUI

3.1.2.9 spmon

spmon is the second of the two programs that PSSP provides for monitoring and administering your SP system. In previous versions of PSSP, spmon could run as a GUI in a similar fashion to perspectives. In PSSP release 3.1 and later, it is only available via the command line.

It can perform most of the functions of the perspectives GUI and is useful for quickly checking the status of the system, powering a node on or off, or changing the keyswitch.

The screen shot shown in Figure 39 shows the output from an `spmon -d` command, which checks some diagnostics, and is useful for providing an overall picture of the system.

```
# spmon -d
1.  Checking server process
    Process 44164 has accumulated 14 minutes and 51 seconds.
    Check ok

2.  Opening connection to server
    Connection opened
    Check ok

3.  Querying frame(s)
    1 frame(s)
    Check ok

4.  Checking frames

    This step was skipped because the -G flag was omitted.

5.  Checking nodes
--------------------------------- Frame 1 -------------------------------------
Frame  Node    Node         Host/Switch  Key     Env   Front Panel   LCD/LED is
Slot   Number  Type  Power   Responds    Switch  Fail    LCD/LED      Flashing
-------------------------------------------------------------------------------
                                                            _   _
  1       1    wide    on     no  no      normal   no     | |_   _|     yes
                                                          |  _| |_
  3       3    wide    on    yes  yes     normal   no   LEDs are blank   no
  5       5    thin    on    yes  yes     normal   no   LEDs are blank   no
  6       6    thin    on    yes  yes     normal   no   LEDs are blank   no
  7       7    wide    on    yes  no      normal   no   LEDs are blank   no
#
```

Figure 39. Example spmon -d output

3.1.2.10 Netbooting

As we have already mentioned, one of the differences between SP nodes and stand-alone RS/6000s is their physical attributes.

Normally, to netboot an RS/6000, we would turn a keyswitch and go into SMS or use a ROM IPL disk, depending on the model.

On the SP, this is all handled for us, thanks to our connection with the frame supervisor card. In order to netboot a node, we simply need to click on netboot on the perspectives panel or, from the command line, use the nodecond command. It is possible to manually netboot the node if you wish to use the spmon command to change the power or keyswitch setting and a writable s1term in order to enter the boot address or, in the case of a PCI node, enter SMS mode.

Using the nodecond command or perspectives to netboot a node also provides a log file to check for errors, although you can also watch the progress of a netboot by using a read-only s1term. Do not use the *open a console* option from the perspectives panel if you want to simply watch the netboot progress of a node because this will open a writable console and stop automatic node conditioning.

The example output in Figure 40 shows an example of a node conditioning logfile.

```
# cat /var/adm/SPlogs/spmon/nc/nc.1.1
Nodecond Status: invoking /usr/lpp/ssp/bin/nodecond_mca
Nodecond Status: start frame 1, slot 1
Nodecond Status: get bootp response type from SDR
Nodecond Status: bootp response type is disk
Nodecond Status: get default boot device from SDR
Nodecond Status: default boot device is en0
Nodecond Status: get Ethernet type from SDR
Nodecond Status: Ethernet type is bnc
Nodecond Status: get nodes card type
Nodecond Status: nodes card type is 81
Nodecond Status: get node type (thin/wide/high)
Nodecond Status: node type is wide
Nodecond Status: power off the node
Nodecond Status: open S1 port
Nodecond Status: change key to Secure
Nodecond Status: start monitoring the LEDs
Nodecond Status: turn on the node and wait for 200
Nodecond Status: got 200, change key to Service
Nodecond Status: reset the node
Nodecond Status: wait for 260 or 262
Nodecond Status: got 260
Nodecond Status: in main menu, get adapter address
Nodecond Status: selected adapter matched ==> 3. Ethernet:  Slot 0/1, BNC connector (1-pin)
Nodecond Status: checking IP addresses
Nodecond Status: all IP addresses are zero; continuing
Nodecond Status: go back to main menu
Nodecond Status: in main menu, start network boot
Nodecond Status: change key to Normal
Nodecond Status: start network boot
Nodecond Status: waiting for "Booting . . .  Please wait." menu.
Nodecond Status: sent XON to S1
Nodecond Status: holding the s1 port for 4 minutes 0 seconds
Nodecond Status: network boot proceeding, nodecond is exiting
#
```

Figure 40. Example of node conditioning logfile

The logs are written in the directory /var/adm/SPlogs/spmon/nc on the CWS, the filenames follow the convention nc.<f>.<n> where <f> is the frame number, and <n> is the node number.

3.1.2.11 Node customization

The customization of the SP nodes is performed by a Korn shell script called pssp_script. This is used by NIM after a migration or installation of a node. setup_server defines this as a NIM resource of the type script; therefore, it is run on the node before NIM reboots it.

pssp_script is also run on bootup if a node is set to customize in the SDR. In order to detect this, the script, /etc/rc.sp, which is run from inittab, checks the bootp response of the node in the SDR, and, if it is set to customize, spawns off the pssp_script process.

During the customization phase, pssp_script configures the node's environment based on the data in the two files in the /tftpboot directory: <node>.config_info and <node>.install_info where <node> is the hostname of the node in question.

These two files are created by the setup_server wrappers, mkconfig and mkinstall. See Section 3.3.1.10, "mkconfig" on page 240, and Section 3.3.1.11, "mkinstall" on page 241, for more information on these two modules.

Although the main function of pssp_script is to install the PSSP software and configure it. The script has several more notable functions. Among its extra tasks are configuring a separate dump logical volume, updating the /etc/hosts file, and starting or stopping volume group mirroring.

The final part of node customization is performed after the node is rebooted. As part of an installation or migration operation, pssp_script places an entry for a script called spfbcheck in /etc/inittab. pssp_script copies this script, along with another called psspfb_script from the directory /usr/lpp/ssp/install/bin on the BIS to the local /tftpboot directory on the node.

On reboot, the /tftpboot/spfbcheck script is run, which renames the /tftpboot/psspfb_script so that it is not run again accidently, and executes it. It then removes its own entry from /etc/inittab to stop itself from being run on subsequent boots.

The main job of psspfb_script is to configure the network adapters that have previously been defined in the SDR. Once this has been done, the final stage is for the script to set the bootp response field of the node back to disk in the SDR.

Both pssp_script and psspfb_script create log files that can be used for problem determination, and both change the nodes' LED status as they run; so, from this, we can determine what stage of customization a node is in or whether it may have hung or not. Problem determination during the customization phase is discussed further in Section 3.5.4, "SP LEDs" on page 284.

3.2 setup_server

This section attempts to explain how the script, setup_server, works. It is a crucial function to understand since it is called on every node and BIS on bootup and is central to configuring and maintaining the SP NIM environment.

We will first examine setup_server as a whole and give an overview of what each of the modules does. The modules that are more NIM-biased will be discussed in more detail in Section 3.3, "NIM commands in SP" on page 228.

3.2.1 Structure

The setup_server Perl source code configures and controls a huge portion of the SPs environment but is smaller than expected with less than 1000 lines of code.

This is because setup_server (since PSSP 2.2) is modular. It can be broken down into stand-alone pieces of code (wrappers), such as delnimmast, allnimres, mkconfig, and so forth. Each one of these wrappers is called as part of setup_server but can also be called manually from the command line.

This can make is much easier to pinpoint where an error may have occurred and gives us the ability to rerun a specific wrapper rather than rerunning setup_server in its entirety.

Exactly how setup_server runs these wrappers depends upon how the environment is currently configured and the changes that have to be made, such as allocating some resources or building an extra boot install server.

setup_server only has a single allowable flag with which it can be called: `-h`.. However, all this does is display the help text.

As discussed earlier, setup_server is run on every boot of a node, boot/install server, or the CWS (called from /etc/rc.sp). However, setup_server will also need to run manually whenever we are changing an element of a node install. For example, if the node is set to install and we set it back to disk, we need to run setup_server so that it will deallocate the NIM resources, remove the nodes entry in /etc/tftpboot, unexport the exported file systems, and so on.

The flowchart in Figure 41 shows the logic of the script.

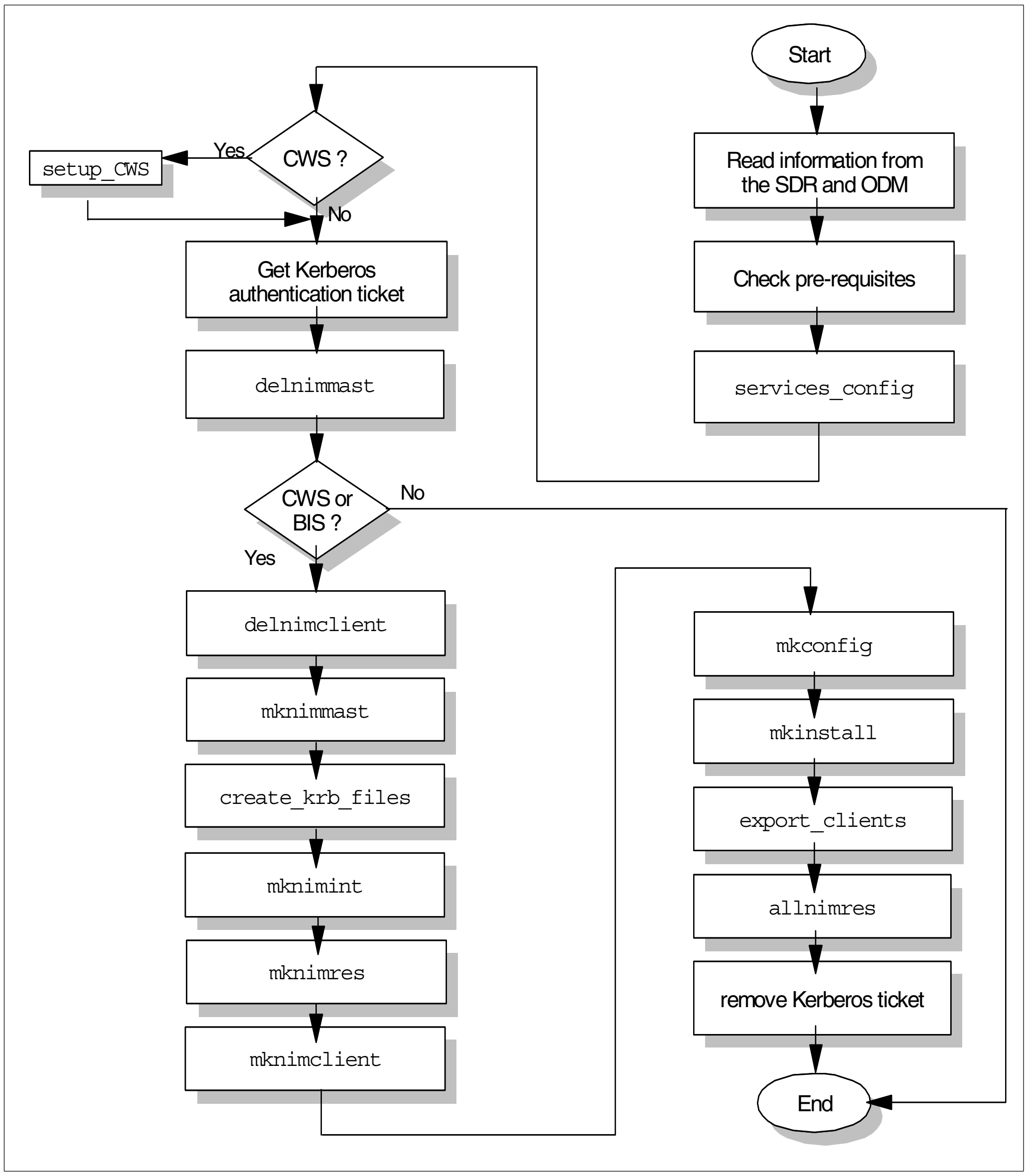


Figure 41. setup_server flowchart

The first part of setup_server, *Read information from the SDR and ODM*, is just an information-gathering exercise. setup_server builds global arrays to hold information on every node in the SP storing its hostname and boot/install server.

In the step, Check pre-requisites, setup_server performs a few *sanity* checks, most significantly: Is the system setup_server running on a valid node, and does a Kerberos configuration file exist for it? If the answer to either of these questions is no, setup_server will exit at this point.

Next, setup_server runs its first wrapper, services_config. This module can be called manually from the command line, but this would rarely be done; so, it checks whether the correct elements of PSSP are installed, depending on the configuration of the node or CWS. If needed, services_config will NFS mount the appropriate PSSP images directory and install the necessary filesets.

Next, if setup_server is run on the CWS, that is, if the node number attribute of the ODM is set to 0, the setup_CWS module is run. Basically, setup_CWS updates Kerberos files to reflect CWS and node network interface names. If you are using AFS or remote Kerberos servers, srvtab files for the nodes are created in the /tftpboot directory.

In the step, Get Kerberos authentication ticket, we get an rcmd ticket.

Now the delnimmast module is run, although this is dependent on a few factors. If the node is not a NIM master, or if this node is a NIM master, but also a boot/install server, we can skip this step. If, however, this node is a NIM master but should not be, that is, the NIM software is installed, but the node is not defined as a boot/install server, we run delnimmast on it. This will unconfigure the node as a NIM master and remove the NIM filesets.

At this point, if setup_server is running on a normal node, that is, one that is neither the CWS nor a BIS, the program terminates.

For nodes defined in the SDR as a BIS or for the CWS, the next module that is run is delnimclient. This checks to see whether the nodes defined as NIM clients for this NIM master are actually set in the SDR to use this NIM Master as a boot/install server. If the clients are *dead*, that is, if there is a NIM object for them, but they are set to use a different boot/install server, the client object is deleted from the NIM master.

The mknimmast module is now run. Basically, this will install the NIM filesets on the node in question and configure it as a NIM master if it is not already. It does this by exporting and mounting the lppsource from the CWS, if necessary, to install the filesets and then uses the `nimconfig` command to make the NIM master. In order to get the primary network interface, mknimmast finds the first ethernet interface that is active and uses it in the configuration.

The create_krb_files module is now run. This wrapper looks for nodes whose bootp response is set to install, customize, or migrate and creates a Kerberos definition for it. It creates a srvtab file, which it places in /tftpboot if the node boots from the CWS, or, if the node boots from another BIS, it will be copied over to the /tftpboot directory of the BIS and removed from the /tftpboot directory of the CWS. create_krb_files also updates or creates the /etc/tftpaccess.ctl file to allow the node access to the appropriate directories when they boot.

The mknimint module defines the NIM network objects. It first checks the ethernet networks known to NIM and then compares them against the active ethernet interfaces from the `netstat -in` command where it defines any *new* interfaces to NIM.

If setup_server is running on a BIS that is not the CWS, all the ethernet and token ring networks on the CWS are defined as network objects on the BIS with a route to get to the CWS' interface. The CWS is then added as a NIM client to the BIS; so, the CWS can still act as a NIM resource server for the BIS.

The mknimres module is now run. It checks the value of the bootp response to see what resources need to be created (if they have not already been created). Before creating the resources, it uses lsnim -l to take a look at the Rstate attribute of all the existing resources. The Rstate attribute is an ASCII text field that describes whether the resource is ready for use. mknimres looks for any whose Rstate is not *ready for use*; It deems these resources to be damaged; so, it deletes them. The resources that can be created from these modules are the script resource, pssp_script, the bosinst_data resources, prompt, noprompt, and migrate, the lpp_source, and the SPOT.

The next module is mknimclient. This piece of code finds out what NIM clients will need to be defined on the NIM master, whether it is the CWS or a BIS, and, if they do not already exist, define them. If a client is defined on a BIS, one extra step that takes part is that a NIM route is added to enable the BIS client to reach the CWS.

The **mkconfig** module is now run. This module creates the /tftpboot/<hostname>.config_info file for each node whose bootp response is not set to disk. This file is used during the network installation of the mksysb image and, among other things, contains node-specific adapter and switch information.

The next module, mkinstall, is similar to mkconfig. It creates the /tftpboot/<hostname>.install_info file for each node whose bootp response is not set to desk. This file contains environmental variables, such as the hostname and IP address of the CWS, the name of the lpp_source, and the hostname and IP address of the Kerberos authentication server.

The last module to be run by setup_server is allnimres. allnimres is responsible for actually allocating the resources to the clients and performing the required NIM operation. For example, if we had set a node to install, allnimres would deallocate any resources that were already allocated to the client, reset its state, and then allocate the appropriate lpp_source, SPOT, mksysb, bosinst_data, and scripts to it before performing a NIM bos_inst operation.

In the final part of setup_server, remove Kerberos ticket, it deletes the rcmd ticket it previously created.

In summary, the job of setup_server is to completely control the SP's NIM environment. It will build, allocate, and deallocate the resources it needs depending on what you set the nodes to do in the SDR. The only piece of manual intervention needed then is to netboot the nodes in question.

3.3 NIM commands in SP

The SP runs the same version of NIM as is used in the classic RS/6000 environment, and, therefore, the normal NIM commands are available to manipulate the objects and resources.

In this section, we will seek to explore this and expand on the NIM wrappers that were briefly discussed in the previous section.

3.3.1 Managing NIM objects

The commands we use with NIM fall into two distinct categories. The first is managing the NIM objects. These are commands in which it is possible to change the state of the NIM object in some way.

3.3.1.1 The nim command

We can use the `nim` command to perform all the normal operations we previously described in this book, such as allocating a resource or performing a bos_inst operation. However, manually-manipulating NIM objects or performing NIM operations is not recommended in an SP environment.

This is because the SP is reliant on the SDR to look at the status of the environment and on setup_server to carry out any changes. For example, if we were to allocate resources to a node with nim -o allocate and then allow the node to install by performing the nim -o bos_inst operation, from a NIM perspective, everything would be fine, but, if we then ran setup_server, the node would still be set to disk in the SDR, and, so, the resources would be deallocated and the NIM object would be reset. Even if we went ahead and netbooted the node without running `setup_server`, we would not have the NFS exports setup without more manual intervention, and, so, the installation would most probably fail.

There are, of course, always exceptions to these rules, and the first is problem determination, in which we may have to manually alter the state of an object using the `nim` command.

Later, we will describe how we use the `nim` command to maintain lpp_source resources, install fixes, and put a SPOT into debug mode.

3.3.1.2 delnimclient

delnimclient is called as part of the setup_server script and is used to undefine a node as a NIM client. It can also be called on its own using delnimclient -l <node_list> | -s <server_list>. Where node_list is the list of nodes on which to perform the operation and server_list is the list of NIM masters from which to delete the client definitions.

The procedure is quite straightforward if we run delnimclient -l. The script determines the clients boot/install server and then deletes the NIM client object from the masters database.

However, if delnimclient -s is used, a little more prerequisite checking is done. The script will check what clients the boot/install server has in its database and then check in the SDR to see where it is the clients are set to boot from. If the clients are still set to boot from the boot/install server in question, they will not be removed from the database. If they are set to boot from somewhere else and, thus, considered *dead* to the boot/install server, they will be removed.

Figure 42 on page 230 show the logic flow of the script.

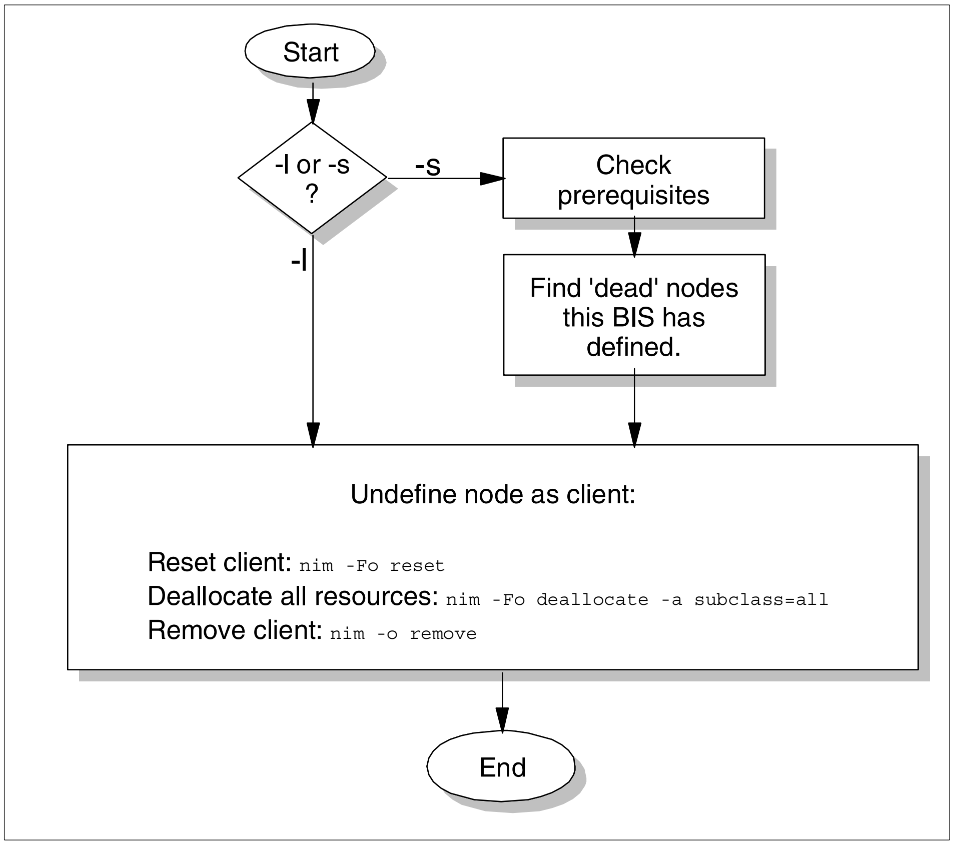

Start
-l or -s ?
-s
-l
Check prerequisites
Find 'dead' nodes this BIS has defined.
Undefine node as client:
Reset client: nim -Fo reset
Deallocate all resources: nim -Fo deallocate -a subclass=all
Remove client: nim -o remove
End

Figure 42. delnimclient flowchart

3.3.1.3 mknimclient

This script is called from setup_server, but can be run manually with the syntax `mknimclient -l <node_number_list>`, where `<node_number_list>` is a list of node numbers to define as NIM clients.

There are a number of prerequisites that must be met before a NIM client is created: The script checks that the node and the node's BIS are valid SP nodes, that we have dsh access to the BIS (the BIS is actually configured as a NIM master and has the appropriate filesets installed), and that the client and BIS are on the same ethernet subnet. If any one of these conditions is not met, the node will be ignored, and processing will continue on the next one in the list.

If the BIS is not the CWS, a route is added in to the BIS spnet NIM object; so, it is able to reach the CWS. This is so the BIS can reach the NIM resources available on the CWS.

Figure 43 shows the logic flow of the script.

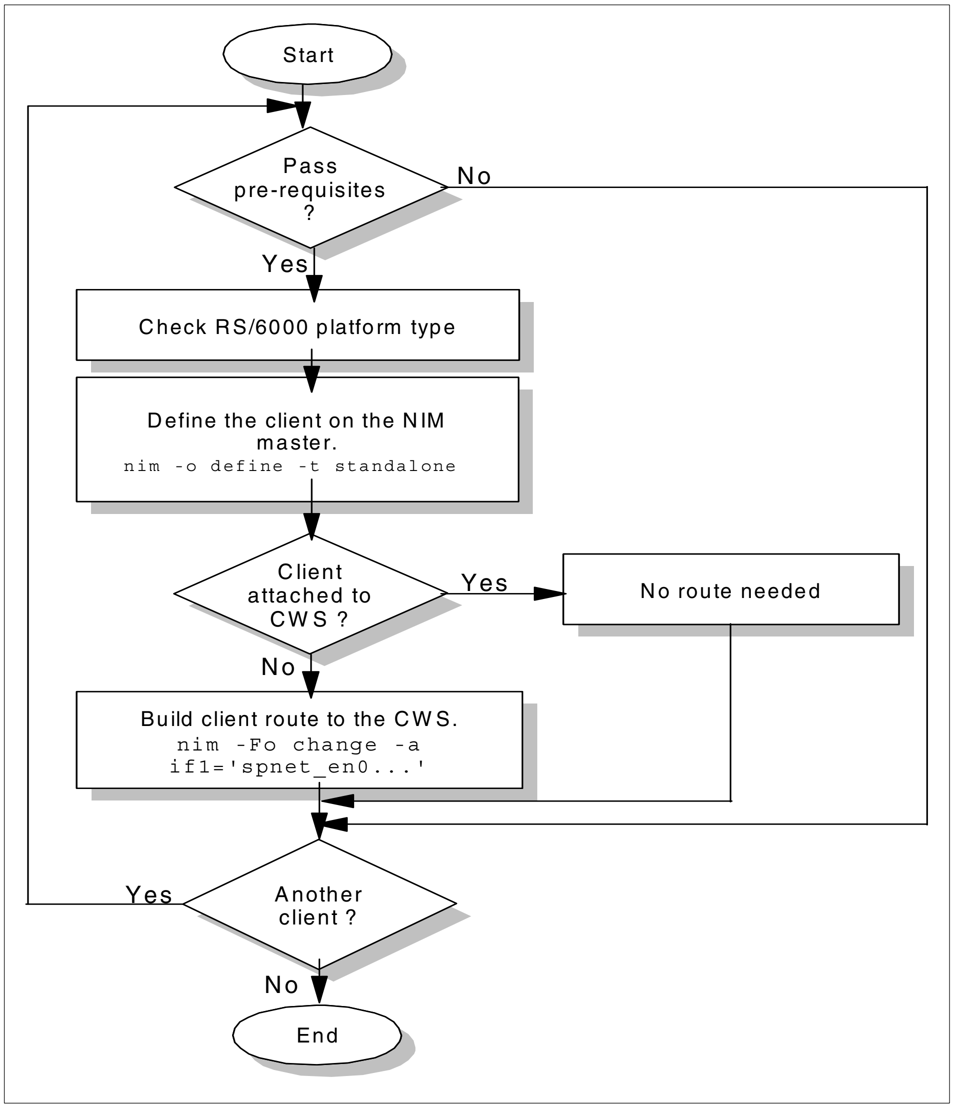

Start
Pass pre-requisites ?
No
Yes
Check RS/6000 platform type
Define the client on the NIM master.
nim -o define -t standalone
Client attached to CWS ?
Yes
No route needed
No
Build client route to the CWS.
nim -Fo change -a if1='spnet_en0...'
Yes
Another client ?
No
End

Figure 43. mknimclient flowchart

3.3.1.4 mknimres

mknimres is called from setup_server, but also can be run on its own with the syntax `mknimres -l <node_list>`. Where `node_list` signifies the list of node numbers to define as NIM masters. Node number 0 signifies the CWS.

It is responsible for creating the NIM resources, but what it creates depends on the current status of NIM and on what the bootp response field is set to in the SDR. The flowchart in Figure 44 shows the logic flow.

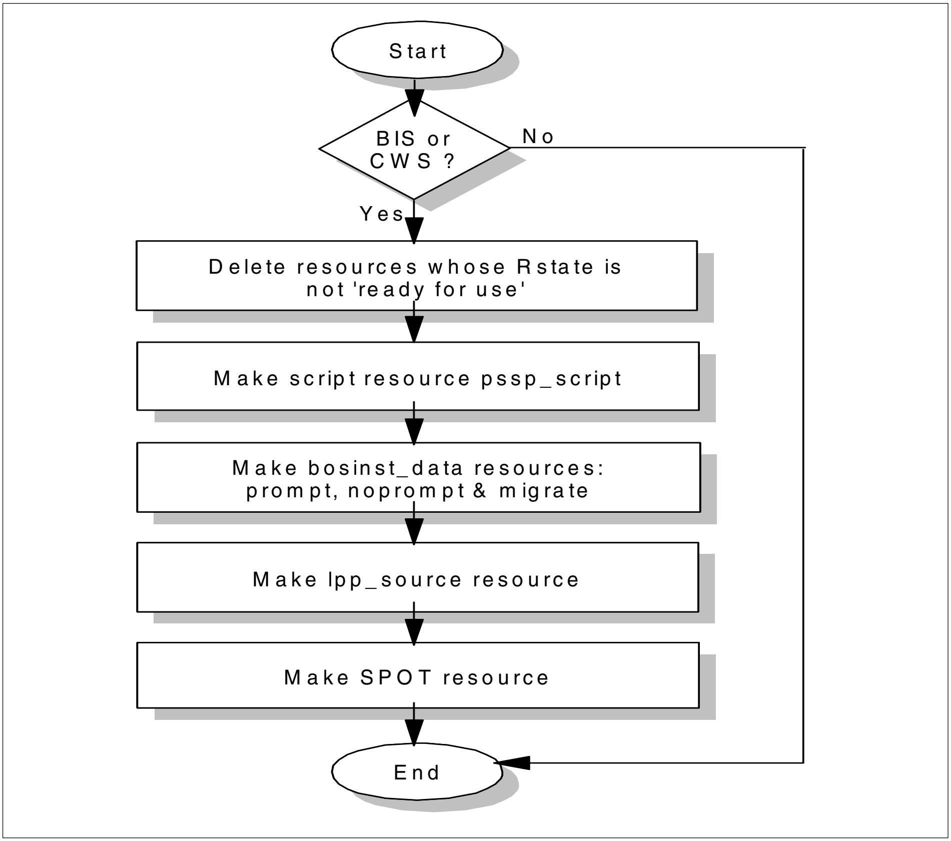

Start
BIS or CWS ?
No
Yes
Delete resources whose Rstate is not 'ready for use'
Make script resource pssp_script
Make bosinst_data resources: prompt, noprompt & migrate
Make lpp_source resource
Make SPOT resource
End

Figure 44. mknimres flowchart

The script first checks to see whether the node_number is the CWS or a boot install server; if it is neither, it exits. It then checks what resources exist and what their state is by running an lsnim -l. If it find resources whose Rstate attributes are not *ready for use*, it will perform a nim -o remove operation on them.

mknimres will then create NIM resources using the normal `nim -o define` command and, depending on what is already defined, will define the script resource, pssp_script, the bosinst_data resources, prompt, noprompt, and migrate, as well as the lpp_source, spot, and mksysb resources.

Although the mknimres structure looks simple, the actions it performs are very important to the NIM environment, and so we need to make sure that the wrapper completes successfully. Perhaps its most crucial function is the creation of the SPOT. If the SPOT is not created properly, mknimres will delete the object because of its Rstate, which means we have to rely on log files in order to perform problem determination. We cover these types of problems in more detail in Section 3.4, “NIM problems in SP” on page 246.

3.3.1.5 allnimres

allnimres is called from `setup_server`, but can be called from the command line with the syntax `allnimres -l <node_list>`, where `<node_list>` is a list of comma-separated node numbers that you want the script to work on.

allnimres really does a lot of work for us; all we need to do is set the bootp response in the SDR via spchvgobj, spbootins, or smit, and allnimres will go ahead and have the correct NIM resources allocated and perform the correct NIM operation ready for us to netboot the node if necessary.

If our bootp response field is set to disk, this means that the node will just boot from its internal disk, and no NIM operation will be performed.

A customization operation does not get any resources allocated to the node as it is not really a NIM operation. The node itself queries the SDR on bootup and will then handle the customization locally by running pssp_script.

The source for a NIM operation is dependant upon the type of operation we performed, and certain types of NIM installations are not possible. For example, when we set a node to install, the source object for the installation is always an mksysb resource where, as with an RS/6000 classic NIM environment, the default source for installation is the rte image. In an SP environment, the rte image, that is, the lppsource, is only used when a migration operation is being performed.

Figure 45 on page 234 shows the logic flow of the script with the specific resources that are allocated and the NIM operation performed depending on the bootp response.

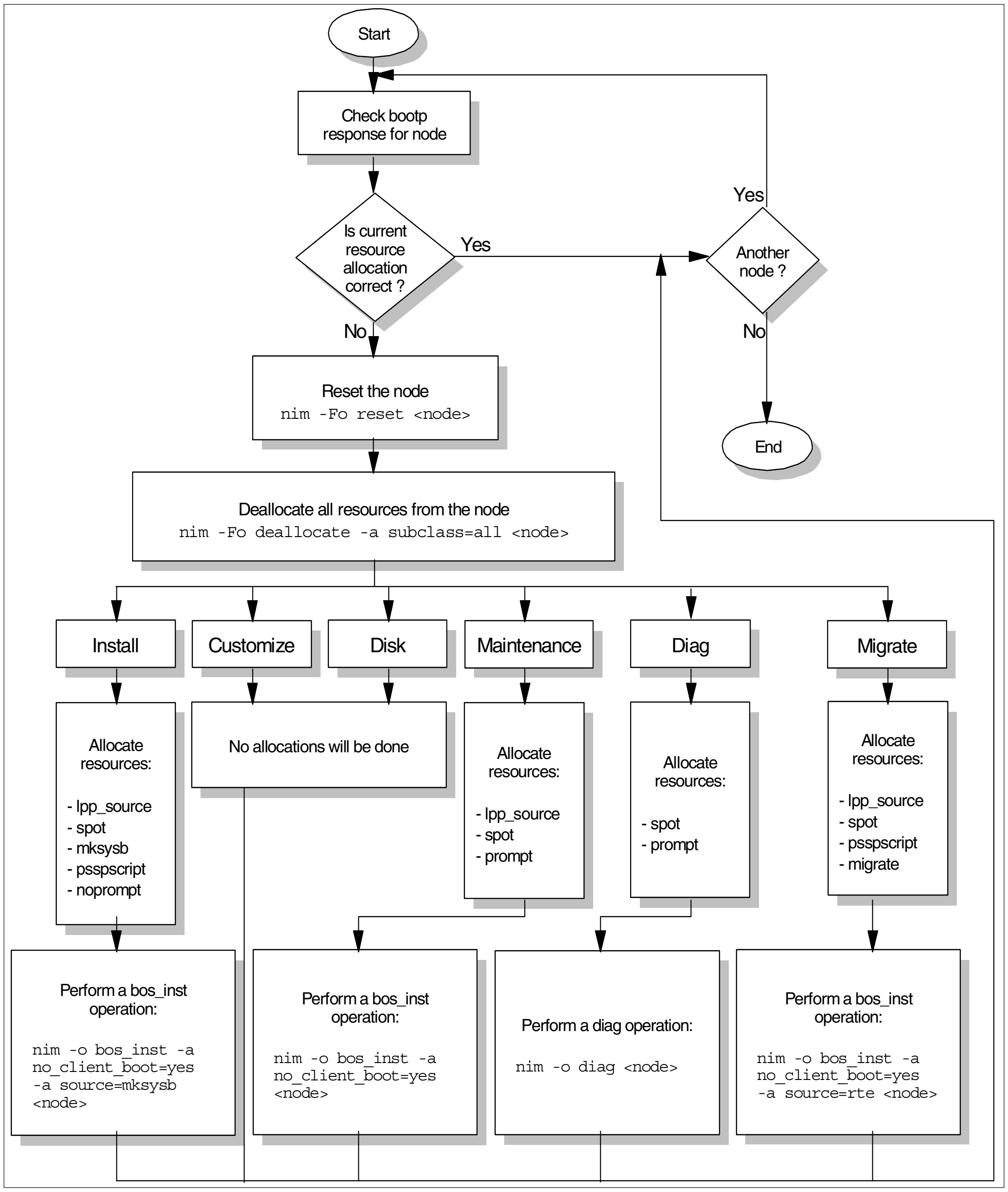

Start
Check bootp response for node
Is current resource allocation correct ?
Yes
No
Another node ?
Yes
No
End
Reset the node
nim -Fo reset <node>
Deallocate all resources from the node
nim -Fo deallocate -a subclass=all <node>
Install
Customize
Disk
Maintenance
Diag
Migrate
Allocate resources: - lpp_source - spot - mksysb - pssp script - noprompt
No allocations will be done
Allocate resources: - lpp_source - spot - prompt
Allocate resources: - spot - prompt
Allocate resources: - lpp_source - spot - psspscript - migrate
Perform a bos_inst operation: nim -o bos_inst -a no_client_boot=yes -a source=mksysb <node>
Perform a bos_inst operation: nim -o bos_inst -a no_client_boot=yes <node>
Perform a diag operation: nim -o diag <node>
Perform a bos_inst operation: nim -o bos_inst -a no_client_boot=yes -a source=rte <node>

Figure 45. allnimres flowchart

3.3.1.6 unallnimres

This module is one of the few that setup_server does not call; so, it is only called via the command line with the syntax, `unallnimres -l <node_list>`, where `<node_list>` is a list of comma-separated node numbers.

The script just provides a useful method of deallocating NIM resources to multiple nodes and resetting their state.

There are a few prerequisites that must be met before the script will continue; the client and the client's BIS must be valid SP nodes; we must have dsh access to the BIS, and it must be configured as a NIM master with the node defined as a NIM client.

The script will then check to see if any resources are actually allocated to the node by running lsnim -c resources <node>. If nothing is allocated, no further action needs to be taken; so, it skips to the next node.

If the node has NIM resources allocated, the script first resets the node's state with nim -Fo reset <node> before finally deallocating all the resources with nim -Fo deallocate -a subclass=all <node>.

There is no real difference in running this script in order to deallocate resources from a node or actually running the `nim -Fo reset` and `nim -Fo deallocate` commands, both accomplish the same basic goal.

However, if you need to deallocate the resources of, for example, one hundred nodes, clearly, using the unallnimres wrapper is the easiest solution. Another alternative may be to run spbootins -r disk for all the nodes you want and let setup_server run, but this would take much more time to complete and would, of course, change the state of nodes in the SDR.

Figure 46 on page 236 shows the logic flow of the script.

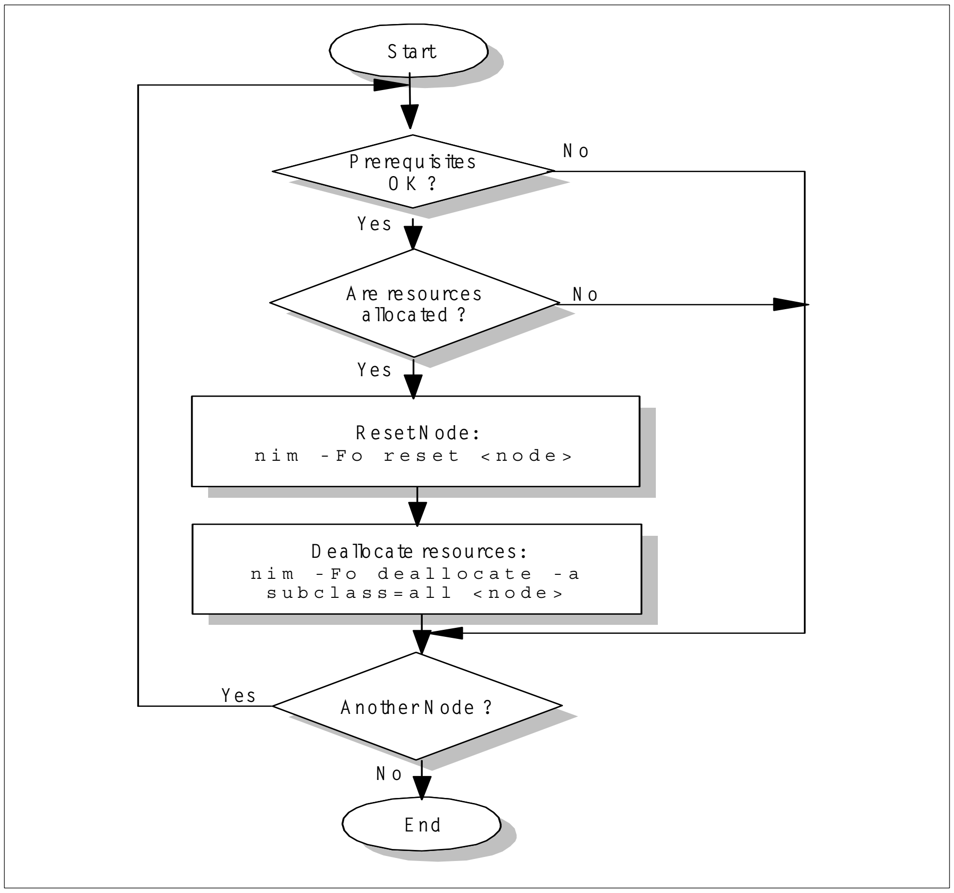


Figure 46. unallnimres flowchart

3.3.1.7 mknimmast

This script is called from setup_server, but can be run manually with the syntax `mknimmast -l <node_list>`, where `<node_list>` is a comma-separated list of nodes to configure as NIM masters.

Although there should be only one NIM master per environment, the SP is somewhat of a special case. As far as NIM is concerned, on the CWS, it only knows about the clients in its own database and, thus, in a traditional NIM sense, has no knowledge or control over the nodes under control of the other NIM masters - the boot/install servers.

However, the SP has full knowledge of its environment and its node from the data in the SDR, and, so, will simply customize NIM the way the SDR dictates

- including configuring or unconfiguring other NIM masters. In this way, the CWS still has full control over all the NIM clients, without necessarily acting as the NIM master to all of them. Figure 47 shows the mknimmast flowchart.

There are some prerequisites that must be met before the script will configure the node as a NIM master:

- The node may *not* already be a NIM master.
- The node must be contactable via dsh.
- The node must be either the CWS or a BIS.
- The lppsource directory on the CWS must exist.

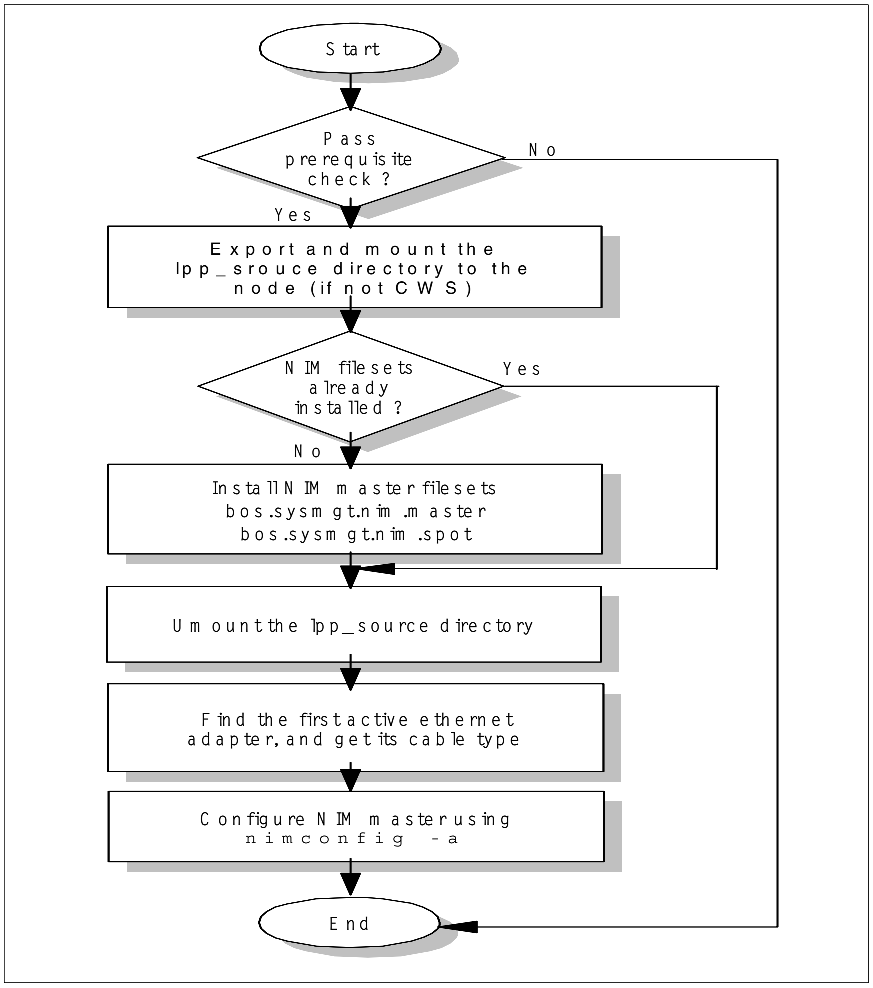

Start
Pass prerequisite check ?
No
Yes
Export and mount the lpp_srouce directory to the node (if not CWS)
NIM filesets already installed ?
Yes
No
Install NIM master filesets bos.sysmgt.nim.master bos.sysmgt.nim.spot
Unmount the lpp_source directory
Find the first active ethernet adapter, and get its cable type
Configure NIM master using nimconfig -a
End

Figure 47. mknimmast flowchart

One of those prerequisite checks may sound a little strange: *The node must either be the CWS or a BIS.* If the node is a BIS, is it not surely a NIM master already?

As far as the SDR is concerned, the node is a BIS if another node is set to boot from it; so, it need not be configured as a NIM master. If you need to run this script manually to set a node as another NIM master, another node can easily be set to boot from it using the `spchvgobj` command.

If the prerequisites are passed, the script will go ahead and mount the lppsource directory from the CWS, if necessary, and install the NIM master filesets. It will then configure the master using `nimconfig`. This will configure the master machine, the boot and nim_script resources, and the network object. The network object is configured by using the first active ethernet interface detected on the node.

Figure 47 on page 237 shows the logic flow of the script.

3.3.1.8 delnimmast

`delnimmast` unconfigures a NIM master definition from a node of a CWS. It is called from `setup_server` but can be used as a command with the syntax `delnimmast -l <node_list>` where `<node_list>` is a list of comma-separated node numbers.

delnimmast does some initial prerequisite checking to make sure the node is a valid node number and a BIS before going ahead and unconfiguring NIM and deleting the filesets.

The SPOTs are removed with nim -o remove before the master is unconfigured with nim -o unconfig master, and the NIM master filesets are removed with installp -u. The script also clears up some files and directories left over from the NIM master being configured on it.

The script may not seem that useful; after all, how often do you have to unconfigure a NIM master? If you change the node to act as a BIS, you will need it, although it is easier to let setup_server take care of it rather than running an individual script. Occasionally, when things have gone really wrong, one of the common problem resolution steps to try is to unconfigure your CWS as a NIM master and rebuild it. Although, in a normal NIM environment, this could potentially be a huge task, on the SP, it is simply a case of unconfiguring the CWS as a NIM master and running setup_server to rebuild the entire environment once again.

The flow chart in Figure 48 on page 239 shows the logic flow.

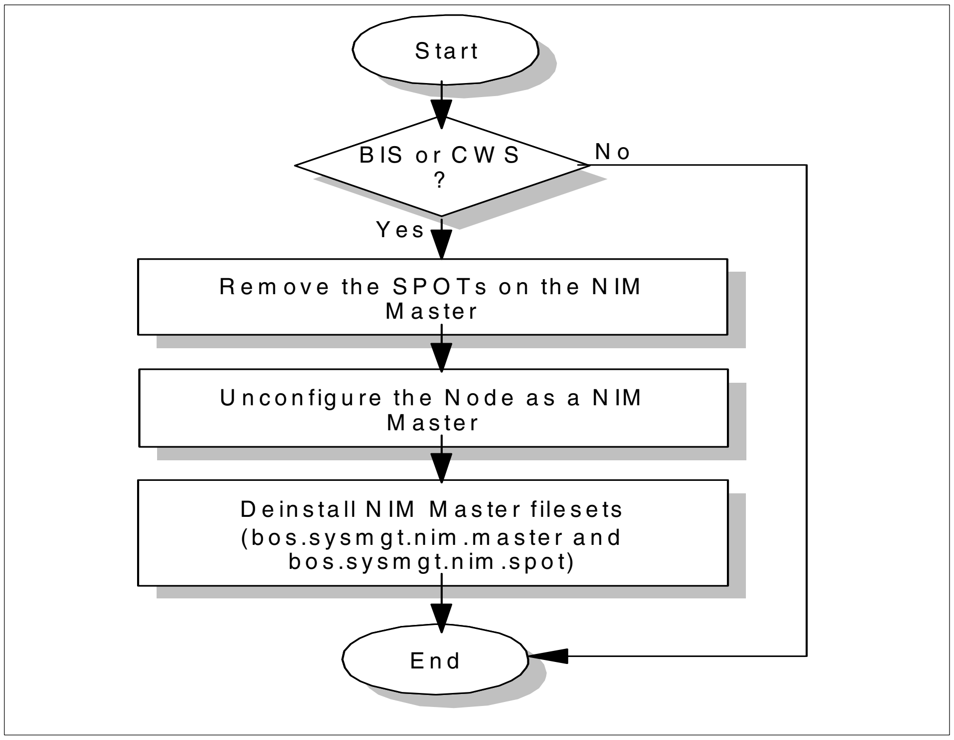

Start
BIS or CWS ?
No
Yes
Remove the SPOTs on the NIM Master
Unconfigure the Node as a NIM Master
Deinstall NIM Master filesets (bos.sysmgt.nim.master and bos.sysmgt.nim.spot)
End

Figure 48. delnimmast flowchart

3.3.1.9 mknimint

This module is called from setup_server, but it can be run manually with the syntax, mknimint -l <node_list>, where <node_list> is a comma-separated list of node numbers.

This script creates the necessary NIM networks, checks what network objects are already defined with an lsnim -Z -a net_addr, and compares this with the active ethernet adapters it finds from the output of a `netstat -in` command. For any new ethernet adapters discovered, a NIM network object is defined.

As far as the CWS is concerned, that marks the end of the script, but, for a boot/install server, there is some more work to do: The BIS needs to make sure there is always a route back to the CWS; so, on the CWS, the script reruns the `netstat -in` command looking for active Token ring or ethernet interfaces. If these interfaces are on a different subnet than the ethernet adapter on the BIS, the networks are defined as NIM objects on the BIS.

Using one of the networks we have defined for the CWS, the script now defines the CWS as a NIM client of its own; so, it can act as a resource server.

Figure 49 shows the logic flow of the script.

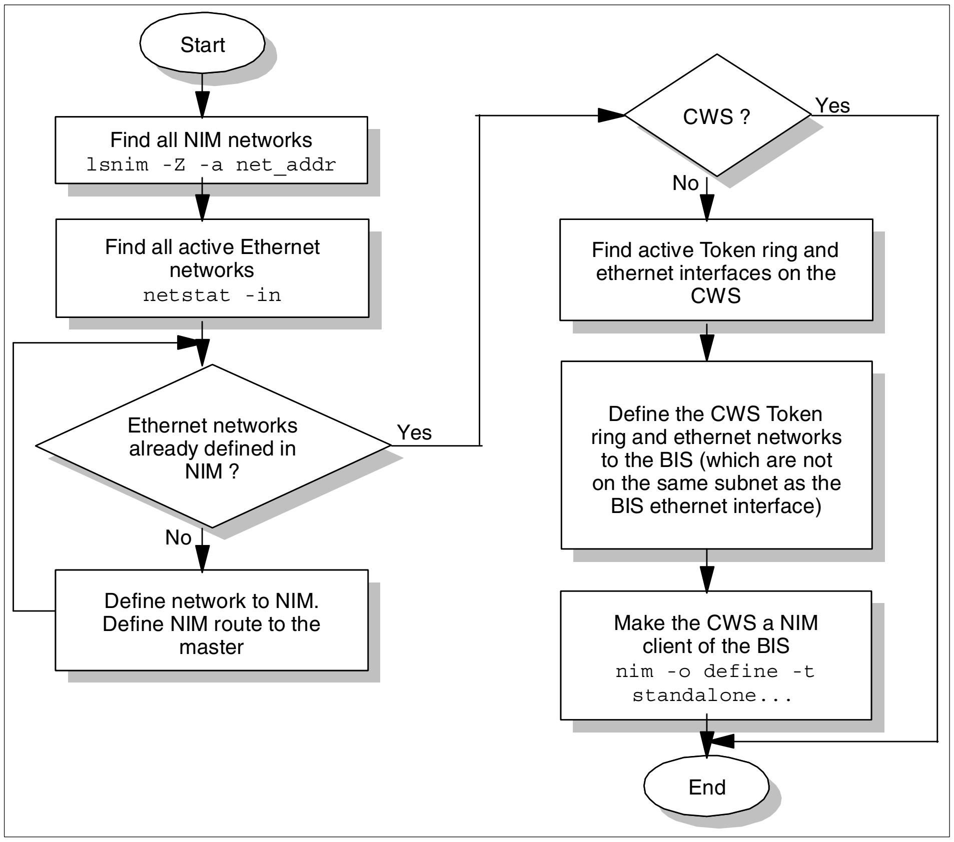

Start
Find all NIM networks
lsnim -Z -a net_addr
Find all active Ethernet networks
netstat -in
Ethernet networks already defined in NIM ?
Yes
No
Define network to NIM. Define NIM route to the master
CWS ?
Yes
No
Find active Token ring and ethernet interfaces on the CWS
Define the CWS Token ring and ethernet networks to the BIS (which are not on the same subnet as the BIS ethernet interface)
Make the CWS a NIM client of the BIS
nim -o define -t standalone...
End

Figure 49. mknimint flowchart

3.3.1.10 mkconfig

This module is run from setup_server, but can be run manually - It has no input parameters. mkconfig doesn't actually perform any NIM commands or operations, but it is still important to the NIM customization phase because the file it creates is used by the NIM script resource, pssp_script.

This script creates a file, /tftpboot/<hostname>.config_info file, where <hostname> is the hostname of the node for every node in the SDR whose bootp response is not set to disk.

The file is made by querying the SDR to get information on the nodes network interfaces, switch details (if it has one), hostname, default route, and so on. If this file is not created or if it contains incorrect information, it can cause serious problems in the customization phase.

3.3.1.11 mkinstall

This module, like mkconfig, creates a file within the /tftpboot directory that pssp_script utilizes during customization. It is called from setup_server, but can also be run manually - it has no input parameters.

This script creates the file, /tftpboot/<hostname>.install_info for each node whose bootp response is not set to disk, where <hostname> is the nodes hostname.

This file contains environmental variables, such as the name of the lpp_source, its server and the IP address of the server, the realm, and the primary Kerberos server. Once again, if this file is missing or contains incorrect information, it will cause problems in the customization phase of the install.

3.3.1.12 spbootins

The `spbootins` command is used to enter configuration or booting data for a node or nodes into the SDR and, so, is normally used in conjunction with spchvgobj. There are a few flags to this command as shown in Table 24.

Table 24. Flags available in spbootins

Flag	Description
-r	Specify the boot response for the node; this can be set to disk, install, customize, migrate, maintenance, or diag.
-s	Specify yes or no to indicate whether setup_server should be run after the completion of the `spbootins` command; the default is yes.
-c	Specify the volume group on which to perform the operation. This defaults to what is currently set in the SDR.
-l	Specify the comma-separated node list on which to perform the operations. Alternatively, you can also not specify a flag and use the startframe, startslot, nodecount format, or -N to specify a node group.

If, for example, we wanted to set all 16 nodes in our first frame to install and then let setup_server go ahead and allocate the necessary resources, we would use the following command:

```
# spbootins -r install 1 1 16
```

3.3.1.13 spchvgobj

This is an important command that is used in conjunction with spbootins in order to configure some other booting options within the SDR, which will again be used by setup_server when allocating and/or building the resources for the node.

Using this command, we can set which volume group to install to, what disks to install to, the number of mirrors, whether quorum should be on, the SPOT and lpp_source to use, and what PSSP version to install. It is very useful to be aware of what this command is capable of because it could well save you a lot of manual node customization later on. The flags we have available in this command are shown in Table 25.

Table 25. Flags available in spchvobj

Flag	Description
-r	The volume group to work on. This is a required switch.
-h	The physical disk(s) to use. You can either specify them as logical disks (that is, hdisk0, hdisk1, and so on) or physical locations (that is, 00-00-00, 0). We strongly recommend using the physical locations because the logical names can change depending on the hardware or other failures.
-i	The install image to use. This duplicates the `spbootins` command.
-p	The PSSP code version to use. The version you choose must match the directory name under /spdata/sys1/install/pssplpp.
-v	The lpp_source to use. This version must match the directory name under /spdata/sys1/install/.
-n	This identifies the boot/install server to use for the nodes installation. By default, a single frame SP system uses node 0 (the CWS); multiple frame SPs use the CWS as the boot/install server for the first frame and the first node of each additional frame thereafter.
-c	This sets the number of mirrors to create for the volume group. The default, 1, means no mirroring. This can be set to either 2 or 3, but each copy must have at least one physical disk to itself.
-q	Can specify true or false. This determines whether quorum should be used in the volume group; it is set to true by default.
-l	Specify a comma-separated list of node numbers on which to perform this operation. If you do not use the -l flag, you can, instead, specify a startframe startslot nodecount parameter at the end of the command.

So, we could set nodes 1 and 2 to install rootvg and mirror between two disks with the following command:

```
# spchvgobj -r rootvg -c2 -h hdisk0,hdisk1 -l 1,2
```

At this point, we are still just configuring the data in the SDR; so, we will still need to set the nodes to install, run setup_server, and netboot them before they will actually go ahead with the installation.

3.3.2 Reviewing NIM information

Unlike the NIM object management commands, the following commands are only used to list and review information; so, they cannot be used to change an objects state and, therefore, cannot cause any harm to the configuration.

3.3.2.1 lsnim

lsnim is still a very useful diagnostic tool within the SP environment and is, perhaps, the very first command to use when looking at an SP NIM problem. Use this command to check what objects have been defined, what resources have been allocated to machines, if there is a problem with the Rstate, and so on. Figure 50 shows some example output of the `lsnim -l` command.

```
# lsnim -l f01n01
f01n01:
   class          = machines
   type           = standalone
   platform       = rs6k
   netboot_kernel = up
   if1            = spnet_en0 f01n01 02608CE890AF ent
   cable_type1    = bnc
   Cstate         = BOS installation has been enabled
   prev_state     = ready for a NIM operation
   Mstate         = not running
   boot           = boot
   bosinst_data   = prompt
   lpp_source     = lppsource_aix433
   nim_script     = nim_script
   spot           = spot_aix433
   cpuid          = 000185927000
   control        = master
```

Figure 50. Example lsnim output

3.3.2.2 splstdata

The `splstdata` command is an SP-specific command, which gives us the state of the environment by interrogating the SDR; the command itself has many different switches available to it that can list almost any detail of the SP. However, the switch that will prove most useful to us when looking at NIM is the `splstdata -b` command, which will produce an output similar to the example shown in Figure 51.

```
# splstdata -b
                  List Node Boot/Install Information

node#         hostname  hdw_enet_addr srvr     response           install_disk
      last_install_image   last_install_time  next_install_image lppsource_name
              pssp_ver        selected_vg
-------------------------------------------------------------------------------
    1 f01n01             02608CE890AF    0        disk               hdisk0
          node7-pssp311 Wed_Oct_20_17:06:17       node7-pssp311        aix432
              PSSP-3.1.1              rootvg
    3 f01n03             02608CE87975    0        disk               hdisk1
          node7-pssp311 Wed_Oct_13_14:18:14       node7-pssp311        aix432
              PSSP-3.1.1              rootvg
    5 f01n05             10005AFA60A2    0     install               hdisk0
        bos.obj.ssp.432 Thu_Oct_21_14:59:48     bos.obj.ssp.432        aix432
              PSSP-3.1.1              rootvg
    6 f01n06             10005AFA669F    0        disk               hdisk0
        bos.obj.ssp.432 Thu_Oct_21_16:03:38     bos.obj.ssp.432        aix432
              PSSP-3.1.1              rootvg
    7 f01n07             02608CE87824    0        disk               hdisk0
        bos.obj.ssp.432 Tue_Oct_12_16:39:06     bos.obj.ssp.432        aix432
              PSSP-3.1.1              rootvg
#
```

Figure 51. Example output from the splstdata -b command

It is important to know how to interpret the output from this command since setup_server uses the same SDR information to build and allocate NIM resources to the clients as necessary. See Table 26 for further explanation.

Table 26. Understanding the output from splstdata -b

Field	Example	Description
node#	5	The node number.
hostname	f01n05	The nodes hostname (will truncate if too long).
hdw_enet_addr	100005AFA669F	The nodes ethernet MAC address.
srvr	0	The nodes Boot/Install server (0 indicates the CWS).
response	install	What the node will do at next boot. That is, the disk will boot from its internal disk, while install will network boot and perform a NIM install. The other possible examples are migrate, customize, diag, and maintenance.

Field	Example	Description
install_disk	hdisk0	hdisk to which to install.
last_install_image	bos.obj.ssp.432	mksysb resource that was previously installed.
last_install_time	Thu_Oct_21_14:59:48	Date and time of the last installation.
next_install_image	bos.obj.ssp.432	mksysb image to install at next netboot.
lppsource_name	aix432	The lppsource to use with the install.
pssp_ver	PSSP-3.1	The pssp version currently installed or the one to migrate over to at the next netboot.
selected_vg	rootvg	The volume group to which to install.

From the perspective of our NIM configuration, the -b flag is, by far, the most useful, but splstdata can tell us much more about the SP environment, both from querying the SDR and by running processes on the nodes themselves.

Table 27 describes what other flags are available in the `splstdata` command and gives a brief outline of their function.

Table 27. Flags available in splstdata

Flag	Description
-A	Displays the node’s SDR accounting data.
-n	Displays information on the nodes from the SDR. Information includes the node number, frame number, hostname, processor type, description, and so on.
-s	Displays the switch information from the SDR in three lists. The first list shows switch chip information on a node-by-node basis; the second list shows switch information on a frame-based level, and the third list shows topology information.
-b	Shows boot/install information as discussed previously.
-a	Displays the SDR adapter information for the nodes, for each adapter this will show, the node number, adapter type, network address, netmask, the hostname, and, if applicable, the type of physical connection and the speed.
-u	Displays the SDR /usr data for each of the nodes.

Flag	Description
-v	Displays the SDR volume group information for the nodes; the information shown includes the node number, volume group name, number of copies (for mirroring), the BIS to use, and so on.
-h	Displays hardware data for each node by running the `lscfg` command remotely.
-i	Displays network adapter data for each node by running the `netstat -n` command remotely.
-d	Displays file system data for each node by running the `df` command remotely.
-e	Displays SP object attributes and their values from the SDR.
-f	Displays frame information, such as the frame number, the type of frame, and the tty port it is attached to on the CWS.
-p	Displays information about different SP partitions or just a single list if your SP is not partitioned.

3.4 NIM problems in SP

Within an SP environment, we are presented with more things that can potentially go wrong with our NIM configuration. In addition to NIM itself having a problem, we may also have to consider problems with setup_server. In this section, we list some specific problems and the way in which we investigated and solved them, although the same basic techniques can be applied to many other situations.

For some advice on general problem determination, review Section 3.5, “NIM debugging on the SP” on page 272.

3.4.1 SPOT creation problem

Before we can go ahead and install our nodes, we need the resources available to do so. As we have described during the installation process, all we do is copy installation images into the appropriate directories and then run setup_server.

setup_server will create several resources. In some of these, all we need to give NIM is a directory location while some, such as the SPOT, may need to be built from another resource: The lppsource.

Let us consider a common failure in building the SPOT. In this example, we got as far as running setup_server for the first time in the installation process; setup_server terminates with output similar to that shown in Figure 52.

```
mknimres: Creating the spot resource spot_aix433.
mknimres: 0016-391 Failed to define the spot named spot_aix433 at location
 /spdata/sys1/install/aix433/spot on cws1 with return code 1.
mknimres: 0016-407 Refer to /tmp/spot.out.45370 for more debug information.
setup_server: 0016-279 Failure of internally called command: /usr/lpp/ssp/bin/mk
nimres; rc= 2.
```

Figure 52. Spot creation error

This output tells us the name of the SPOT that failed to create and its location, but it also refers us to a log file we can look at to get a better understanding of the failure, in this case, /tmp/spot.out.45370.

We need to examine this log file in order to look for any error messages that could point towards resolving the SPOT creation problem. Our example in Figure 53 shows a portion of the log file /tmp/spot.out.45370.

```
get_attr: id=941821566; value=; seqno=0; pdattr=143;
         find_attr: obj=lppsource_aix433; pattern=; seqno=0; pdattr=34;
         disable_err_sig
         nim_malloc: size = 101
         nim_log: str=0042-162 m_mkspot: an lpp_source which has the "simages"
        attribute is required for this operation;
         get_glock: retry=0; delay=0;
         query_glock: lf_name=/var/adm/nim/errlock
         disable_err_sig
         enable_err_sig
         rm_glock: force=0; lock_file=/var/adm/nim/errlock; fd=5
         query_glock: lf_name=/var/adm/nim/errlock
rc=162
0042-162 m_mkspot: an lpp_source which has the "simages"
        attribute is required for this operation
         mstr_exit
         nim_exit
```

Figure 53. Checking the spot.out file

A spot.out file contains a great deal of logging information, not all of which is of use to us. If we take a look at this portion of the log, we can see the same error mentioned twice:

```
0042-162 m_mkspot: an lpp_source which has the "simages" attribute is
required for this operation
```

Again, if we refer back to this log, we can see that the error is associated with the lppsource, lppsource_aix433. We should take a closer look at this resource. We can do this with the `lsnim -l` command as shown in our example in Figure 54.

```
# lsnim -l lppsource_aix433
lppsource_aix433:
   class       = resources
   type        = lpp_source
   Rstate      = ready for use
   prev_state  = verification is being performed
   location    = /spdata/sys1/install/aix433/lppsource
   alloc_count = 0
   server      = master
#
```

Figure 54. Examining the lppsource object with lsnim -l

At first glance, this would appear to look fine. The Rstate is ready for use, but notice we don't have the simages attribute that the log mentions. The simages attribute is set to yes on an lppsource if it can be used to perform an installation. The fact that our lppsource hasn't got this means that we must be missing one of the required filesets.

We should be able to check the lppsource to tell us what is missing using the `nim -o check` command.

```
# nim -o check lppsource_aix433
warning: 0042-267 c_mk_lpp_source: The defined lpp_source does not have the
        "simages" attribute because one or more of the following
        packages are missing:
                bos.net
#
```

Figure 55. Example nim -o check command

From this point on, problem resolution is relatively simple: Copy the required filesets into the lppsource directory, and rerun the `nim -o check lppsource_aix433` command.

Once we have taken this corrective measure, we can take another look at our lppsource object to confirm that we have the simages attribute set to yes. Figure 56 on page 249 shows our output.

```
# lsnim -l lppsource_aix433
lppsource_aix433:
   class       = resources
   type        = lpp_source
   Rstate      = ready for use
   prev_state  = verification is being performed
   location    = /spdata/sys1/install/aix433/lppsource
   simages     = yes
   alloc_count = 0
   server      = master
```

Figure 56. Checking the lppsource for the simages attribute

setup_server should now complete the mknimres section without a problem. Note that one step we must not forget is the `nim -o check` command. Simply copying the required filesets into the directory is not enough; we must run this command to update the table of contents file (.toc) and, thus, update the status of the NIM resource.

So, how does NIM know exactly what filesets must be available in the lppsource in order to create a SPOT? It uses a file called c_sh_lib, which is located in the /usr/lpp/bos.sysmgt/nim/methods directory. If we look at this file, we will see the environment variable, REQUIRED_SIMAGES, which contains the necessary filesets needed in an lppsource in order to set the simages attribute. Figure 57 on page 250 shows this variable in the c_sh_lib file.

```
REQUIRED_SIMAGES="\
        bos \
        bos.64bit \
        bos.up \
        bos.mp \
        bos.net \
        bos.diag \
        bos.sysmgt \
        bos.terminfo \
        bos.terminfo.all.data \
        devices.base.all \
        devices.buc.all \
        devices.common.all \
        devices.graphics.all \
        devices.mca.all \
        devices.rs6ksmp.base \
        devices.scsi.all \
        devices.sio.all \
        devices.sys.all \
        devices.tty.all \
        xlC.rte"
```

Figure 57. Looking at the filesets required for the Semites attribute

3.4.2 Corruption of an lppsource

This can be a confounding error to track down since, on the face of it, everything will appear to look OK, it's linked with adding fixes into your lppsource.

Let us take an example of an initial SPOT build. In this example, we have created an lppsource directory from our AIX 4.3.3.0 base CD-ROM media, and we have also downloaded some fixes for bos.mp and bos.up, which have been copied into the lppsource directory.

We run setup_server for the first time to build our resources and see output similar to that shown in Figure 58.

```
mknimres: Creating the spot resource spot_aix433.
mknimres: 0016-400 Creation of the spot_aix433 resource apparently failed, since
 the Rstate
 is not 'ready for use'.  Check previous messages for possible cause.
mknimres: 0016-407 Refer to /tmp/spot.out.39264 for more debug information.
setup_server: 0016-279 Failure of internally called command: /usr/lpp/ssp/bin/mk
nimres; rc= 2.
setup_server: Processing incomplete (rc= 2).
```

Figure 58. mknimres failure

At this point, there is no indication what the problem is. If we do an lsnim -l lppsource_aix433, we have the simages attribute as yes; so, let us look at the /tmp/spot.out.39264 file.

This file will be quite large if it's run through an attempt to build a SPOT from scratch; so, we need to look out for something unusual. A portion of this file is shown in Figure 59.

This output looks quite strange initially. We have our base level 4.3.3.0 filesets asking for a perquisites of 4.3.0.0. If we look for the filesets within the lppsource directory, we can see the 4.3.3.0 and 4.3.3.1 versions but nothing else; so, how can a base level 4.3.3.0 fileset be asking for a prerequisite 4.3.0.0 fileset?

```
 Be sure to check the output from the SPOT installation
 to verify that all the expected software was successfully
 installed.  You can use the NIM "showlog" operation to
 view the installation log file for the SPOT.

+-----------------------------------------------------------------------------+
                    Pre-installation Verification...
+-----------------------------------------------------------------------------+
Verifying selections...done
Verifying requisites...done
Results...

FAILURES
--------
  Filesets listed in this section failed pre-installation verification
  and will not be installed.

  Requisite Failures
  ------------------
  SELECTED FILESETS:  The following is a list of filesets that you asked to
  install.  They cannot be installed until all of their requisite filesets
  are also installed.  See subsequent lists for details of requisites.

    bos.mp 4.3.3.0                            # Base Operating System Multip...
    bos.mp 4.3.3.1                            # Base Operating System Multip...
    bos.up 4.3.3.0                            # Base Operating System Unipro...
    bos.up 4.3.3.1                            # Base Operating System Unipro...

  MISSING REQUISITES:  The following filesets are required by one or more
  of the selected filesets listed above.  They are not currently installed
  and could not be found on the installation media.

    bos.mp 4.3.0.0                            # Base Level Fileset
    bos.up 4.3.0.0                            # Base Level Fileset

  << End of Failure Section >>
```

Figure 59. Requisite failures in the spot.out file

The problem is difficult to solve without knowing where to look, but the next place to check is the .toc file within the lppsource directory, since this tells us all the packages available in the directory as well as their prerequisite requirements.

We need to look at this file and find the records for the bos.mp.4.3.3.0 and bos.up.4.3.3.0 filesets. Figure 60 shows a small portion of our .toc file for the bos.up record.

```
bos.up.4.3.3.0.bff 4 R S bos.up {
bos.up 04.03.0003.0000 1 b U en_US Base Operating System Uniprocessor Runtime
[
*ifreq bos.64bit (4.3.0.0) 4.3.3.0
*ifreq bos.acct (4.3.0.0) 4.3.3.0
*ifreq bos.adt.debug (4.3.0.0) 4.3.3.0
*ifreq bos.adt.include (4.3.0.0) 4.3.3.0
*ifreq bos.adt.lib (4.3.0.0) 4.3.3.0
*ifreq bos.adt.prof (4.3.0.0) 4.3.3.0
*ifreq bos.adt.samples (4.3.0.0) 4.3.3.0
*ifreq bos.adt.syscalls (4.3.0.0) 4.3.3.0
...
...
```

Figure 60. Examining the .toc file

The line we are most concerned with is the first one, and looking further into the file, bos.mp, shows a similar line. Table 28 explains what the fields in this first line mean.

Table 28. Understanding the .toc file

Field Name	In the example	Explanation
lpp_name	bos.up.4.3.3.0.bff	The filename containing the installable image.
Format	4	The release level of Installp from which the package was built.
Platform	R	The platform for which this image was built. (R is the only allowable value).
Package type	S	Indicates whether this is an installation of update package. S represents *Single Update.*
Package Name	bos.up	The fileset the lpp_name contains.

The point of concern we have about this line is the Package type field. It is set to S meaning that it is an update fileset. But how can a base level fileset (that is 4.3.3.0) be an update fileset?

In this case, the problem lies with the fixes that were previously downloaded and placed in the lppsource directory. There are actually two versions of a base level fileset: One that comes on the installation media, the install image, and one that can be downloaded, a single update image. When we downloaded the 4.3.3.1 versions of bos.up and bos.mp, we specified our existing level of AIX as 4.3, and, so, we got single update versions of bos.up and bos.mp 4.3.3.0 downloaded as prerequisites.

A single update fileset is intended to bring a previous version of the fileset up to that level; so, in this case, if we had been running AIX 4.3.2, the 4.3.3.0 versions of the code could have been applied to the existing code in order to bring our base level up to 4.3.3.0; so, we could then apply our 4.3.3.1 version of code.

There are two ways that a single update fileset can hurt our lppsource in this example, and, therefore, there are two possible ways to fix it. The first is that, when we copy in the single update base fileset, it has the same filename as our existing one and, therefore, simply overwrites it. This is simple enough to fix as soon as we know which filesets have been affected. We can simply delete the single update versions and copy the installation filesets back into the directory from our installation media using bffcreate.

The second way in which this can affect us is if the filename of the single update fileset is not the same as our installation fileset; in that case, we get both versions of the same fileset as shown in the following example.

```
# ls -l /spdata/sys1/install/aix433/lppsource/bos.up.4.3.3.0.*
-rw-r--r--   1 root     sys      5529600 Oct 22 09:06 bos.up.4.3.3.0.I
-rw-r--r--   1 root     sys      5623808 Nov 10 13:47 bos.up.4.3.3.0.bff
```

Depending on the version of installp, the difference is the order in which they may appear in the .toc file.

Even without problems, having, installing, and updating a single fileset in the same directory is not a good idea. In our example, we just need to delete the single update filesets and rebuild our .toc file with the `nim -o check lppsource_aix433` command.

The reason this problem can be so confounding is that it's quite possible not to notice any problem when updating your SPOT or even doing installations. Depending on the circumstances, this could go unchecked for a long time.

In our previous example, we talked about creating the SPOT from scratch. This time, let us imagine that we have created the SPOT successfully from the base AIX media and have steadily been downloading PTFs, placing them into the lppsource directory, rebuilding the .toc file, and customizing the SPOT with an update_all operation. As long as our SPOT has the prerequisites installed, it will not try and install the base level filesets from the lppsource; so, even if we have some single-update filesets in the lppsource, we will not see a problem.

There are three different situations that could occur that *would* cause a problem. The first one would occur while performing customization on the SPOT. For example, if we copy some new fixes into the lppsource and perform an update_all operation on the SPOT, one of the fixes may prereq another fileset not currently installed in the SPOT. If this happens, it will need to install the base level fileset from the lppsource. If we have a single-update fileset as a base level fileset, we will have a failure.

The other two scenarios are more common and have a greater impact on your system because they deal with the installation of nodes. We can get a failure both on an installation and a migration.

When we perform an installation on an SP, we use an mksysb resource. Our mksysb will at least contain the basic filesets to bring the node up, but any extra device support needed will be installed from the lppsource.

The SP minimal images will generally contain some of the more common device driver filesets; so, the problem is by no means certain to occur. It is only going to happen if a specific set of criteria are met, that is, if the device driver support is not included in the mksysb, and a single-update fileset exists in the lppsource directory.

Even if these criteria are met, the installation will not necessarily fail. It will depend on the device(s) that will not be able to be configured; so, a manual fix may be possible after the node has installed.

Our second potential problem is with a migration. When we install/overwrite a node, our source image is an mksysb, and, therefore, the lppsource is only utilized for device support not already present in the mksysb. But, when we migrate a node, our source image becomes the lppsource, and, thus, if our lppsource has multiple single update base-level images, most likely, our migration will fail because of prerequisite failures.

In order to avoid getting any of these types of problems, we need to spend a little extra time examining exactly what filesets we are placing into our lppsource. If we have downloaded filesets that have pulled in prerequisite base-level filesets, we must ensure these single-update base-level filesets do not go into our lppsource.

3.4.3 LED 231 problem

If we have a node that appears to be hanging on LED 231, it means it is attempting to send a bootp request to the boot/install server but not getting a response. If a node is attempting to send a bootp but gets no response, it just keeps trying. We can use an s1term to check this as shown in our example in Figure 61.

```
STARTING SYSTEM (BOOT)



Booting . . .  Please wait.




Ethernet:  Slot 0/1, BNC connector (1-pin)
Hardware address ....................................... 02608CE890AF

              Packets Sent       Packets Received

BOOTP             00006               00000
```

Figure 61. A failing bootp request

There could be quite a few reasons why the node is not getting a response. One of them is physical connectivity. For example, has the nodecond selected the correct ethernet adapter, and is it using the correct cable type? A simple way to test the connectivity is to manually node condition the node and perform a *ping test*.

How we manually condition a node depends on the node type. For Microchannel (MCA) based nodes, in order to start a ping test, we perform the following actions (wherever we use *<#>*, replace it with your node number):

1. Use perspectives to power off the node you want to condition, or use the command `spmon -power off node`*<#>*.
2. Use perspectives to set the key position of the node you want to condition to Secure, or use the command `spmon -key secure node`*<#>.*
3. When the LED for the node reaches 200, use perspectives to set the key position of the node to Service, or use the command `spmon -key service node`*<#>.*
4. Use perspectives to reset the node, or use the command `spmon -reset node`*<#>.*

5. Start a read/write `s1term` with the command `s1term -w <frame> <node>`, where `<frame>` is the number of the frame the node is in, and `<node>` is the node number.
6. After a few minutes, a menu will appear in the `s1term`.
7. In order to start the ping test, select the first option, **Select BOOT (Startup) Device**, and, from this menu, select the correct ethernet device, which should be the first BNC selection, or the integrated adapter.
8. Fill in the Client address and the Bootp server address fields, then type `99` to return to the main menu.
9. Select option **3**, **Send Test Transmission (PING)** to start the ping test.
10. You will be prompted again to change any of the client, bootp server, or gateway addresses. Take option **4** to start the test.

If you have a Peripheral Component Interconnect (PCI) node, the procedure is slightly different because the node has no concept of a keyswitch. To start the ping test on a PCI node, use the following procedure:

1. Use perspectives to power off the node, or use the command `spmon -power off node<#>`.
2. Use perspectives to power the node back on, or use the command `spmon -power on node<#>`.
3. Open a read/write `s1term` with the command `s1term -w <frame> <node>`, where `<frame>` is the number of the frame the node is in, and `<node>` is the node number.
4. When the RS/6000 banner appears, press **1** to enter the SMS menu.
5. From the SMS menu, select **3** to enter the Utilities menu.
6. From the Utilities menu, select **4** to take the **Remote Initial Program Load** (RIPL) option.
7. From the RIPL menu, select option **1**, IP Parameters, and fill in the relevant details, namely the client IP address, the Bootp server address, and the subnet mask.
8. Press **x** to return to the RIPL setup menu, and then press **3** to enter the ping test.
9. From the ping test menu, choose the Integrated ethernet adapter option.

In our example, Figure 62 shows the IP information we entered using our client address and the CWS address as the Bootp server.

```
SET OR CHANGE NETWORK ADDRESSES

Select an address to change

Currently selected BOOT (startup) device is:
Ethernet:  Slot 0/1, BNC connector (1-pin)
Hardware address ................................... 02608CE890AF

1. Client address                                    010.003.187.244
     (address of this machine)
2. BOOTP server address                              010.003.187.243
     (address of the remote machine you boot from)
3. Gateway address                                   000.000.000.000
     (Optional, required if gateway used)

97. Return to Select BOOT (Startup) Device Menu (SAVES addresses)
99. Return to Main Menu (SAVES addresses)
```

Figure 62. Entering the IP information for the ping test

As you can see in Figure 63, the ping test was successful, therefore, we do have communications between the node and the CWS.

```
TEST TRANSMISSION (PING) RESULTS

SUCCESSFUL TEST.  Transmission sent and received.









97. Return to Send Test Transmission screen.
99. Return to Main Menu

Type the number for your selection, then press "ENTER"
(Use the "Backspace" key to correct errors)
```

Figure 63. Checking the ping test results

It is also useful to check whether the bootp daemon is running. This runs under the control of the inetd daemon, therefore, it can be checked with the `lssrc` command.

```
# lssrc -ls inetd | grep bootp
 bootps       /usr/sbin/bootpd         bootpd /etc/bootptab     active
```

Next, we need to check that bootp has the correct information to let our client boot. To do this, we need to look in the /etc/bootptab file. The information on clients is always appended to the end of the file; so, we can just use the `tail` command to check it. We show this in Figure 64.

```
# tail /etc/bootptab

#       hn   -- name switch
#       bs   -- boot image size
#       dt   -- old style boot switch
#       T170 -- (xstation only) -- server port number
#       T175 -- (xstation only) -- primary / secondary boot host indicator
#       T176 -- (xstation only) -- enable tablet
#       T177 -- (xstation only) -- xstation 130 hard file usage
#       T178 -- (xstation only) -- enable XDMCP
#       T179 -- (xstation only) -- XDMCP host
#       T180 -- (xstation only) -- enable virtual screen
```

Figure 64. Checking the /etc/bootptab file

We have no client records in the file; so, what has gone wrong? setup_server completed without error but has not placed the correct information in the file. Our next step is to check the boot information for our node. In our example, we use node 7.

When we check the boot information of node 7 with the `splstdata` command, it shows us that node 7 is set to boot from an alternate boot/install server (node 1), not the CWS.

This node is currently down, which explains the problems. Remember that, in a multi-frame environment, the nodes in your additional frames will boot by default off the first node in the frame. In the example shown in Figure 65 on page 259, this is incorrect; so, we set the boot/install server back to node 0, the CWS.

```
# splstdata -b -l 7
                       List Node Boot/Install Information

node#         hostname  hdw_enet_addr srvr     response            install_disk
      last_install_image   last_install_time  next_install_image lppsource_name
                pssp_ver         selected_vg
-------------------------------------------------------------------------------
    7 f01n07            02608CE87824    1      install                 hdisk1
         bos.obj.ssp.433 Fri_Nov_12_17:55:22     bos.obj.ssp.433         aix433
              PSSP-3.1.1             rootvg
# spchvgobj -r rootvg -n 0 -l 7
spchvgobj: Successfully changed the Node and Volume_Group objects for node number 7,
volume group rootvg.
spchvgobj: The total number of changes successfully completed is 1.
spchvgobj: The total number of changes which were not successfully completed is 0.
#
```

Figure 65. Correcting the boot/install server information

All we need to do now is run `setup_server`, and the correct resources will be allocated on the CWS and the correct information appended to the /etc/bootptab file. We should then be able to successfully netboot the node.

3.4.4 LED c48 hang during migration

A hanging LED of c48 can occur during an installation or a migration; it tells us that it’s prompting for some input at the console. This section documents one of the most common LED c48 migration hangs.

In this example, we are migrating one of our nodes from AIX 4.2.1 to AIX 4.3.3, and our migration appears to have hung with an LED of c48; so, our initial reaction is to perform an lsnim -l on our node NIM machine object as shown in Figure 66 on page 260.

We can see in the info field that we need to enter some data at the console, and yet, setup_server has allocated the bosinst_data resource 7_migrate, so this should be a noprompt install. Perhaps we need to examine this bosinst_data script.

```
# lsnim -l f01n07
f01n07:
   class          = machines
   type           = standalone
   platform       = rs6k
   netboot_kernel = up
   if1            = spnet_en0 f01n07 02608CE87824 ent
   cable_type1    = bnc
   Cstate         = Base Operating System installation is being performed
   prev_state     = ready for a NIM operation
   Mstate         = in the process of booting
   info           = prompting_for_data_at_console
   boot           = boot
   bosinst_data   = 7_migrate
   lpp_source     = lppsource_aix433
   nim_script     = nim_script
   script         = psspscript
   spot           = spot_aix433
   cpuid          = 000047067900
   control        = master
   Cstate_result  = success
#
```

Figure 66. lsnim -l of migrating node

We can find out the location of the 7_migrate script by using `lsnim -l 7_migrate` command as shown in Figure 67.

```
# lsnim -l 7_migrate
7_migrate:
   class       = resources
   type        = bosinst_data
   Rstate      = ready for use
   prev_state  = unavailable for use
   location    = /spdata/sys1/install/pssp/7.migrate
   alloc_count = 0
   server      = master
#
```

Figure 67. Locating the bos_inst migration script

Now that we have the location of the script, we can examine this more closely. We are interested in the control_flow: section, which tells us what type of install it is and whether it should prompt at the console for input. Figure 68 on page 261 shows us this portion of our script.

```
control_flow:
    CONSOLE = /dev/tty0
    INSTALL_METHOD = migrate
    PROMPT = no
    EXISTING_SYSTEM_OVERWRITE = yes
    INSTALL_X_IF_ADAPTER = no
    RUN_STARTUP = no
    RM_INST_ROOTS = no
    ERROR_EXIT =
    CUSTOMIZATION_FILE =
    TCB = no
    INSTALL_TYPE = full
    BUNDLES =
```

Figure 68. Examining the migration script

The bos_inst script looks fine; it is correctly set to migrate, and prompt is set to no; so, it must be something else. Our next task is to open a writable `s1term` and see what input is required by entering the `s1term -w 1 7` command.

Figure 69 shows our console display from running the `s1term` command. At this point, entering `1` to continue would ultimately end in a failed migration.

Although the console is complaining about a missing image.template file, at this point, it is trying to get the image.data file from bos image in the lppsource directory.

```
       Error Warning

Missing image.template file. The network install server is not set up correctly.




To reboot in normal mode, turn key to normal (if necessary) and press reset.

>>> 1 Continue with Install
```

Figure 69. Checking the console in a LED c48 hang

We know that the bos image must exist within the lppsource because it is one of the images required to get the simages attributes (see Section 3.4.1, “SPOT creation problem” on page 246, for more information on simages). We now need to check if the image.data file exists within the image.

The bos image is a backup/restore format file; so, we can use the `restore` command, as shown in Figure 70 on page 262, to check what is in the image.

```
# restore -Tvqf /spdata/sys1/install/aix433/lppsource/bos | more
Files are backed up by name.
            0 ./
         8091 ./bosinst.data
        12071 ./image.data
         3766 ./lpp_name
            0 ./usr
            0 ./usr/lpp
            0 ./usr/lpp/bos
       431510 ./usr/lpp/bos/liblpp.a
```

Figure 70. First portion of files archived in the bos image

The file is in the image and the image is in the correct directory, and yet it still cannot be read; so, we check the permissions of the bos file with `ls -l`. We get the permissions back as:

```
-r--r-----   1 root     sys      31878144 Oct 22 09:06 bos
```

These permissions are incorrect. An lppsource directory is exported to the world with read-only permission, but we only have read permissions set for owner and group. To correct this, all we need to do is set the permissions correctly with `chmod 644 bos` and netboot the node once again.

3.4.5 Resource maintenance, creation, allocation, and deallocation

In this section we will look at how we maintain the resources we already have on our SP as well as how to create new ones and then allocate and deallocate them.

3.4.5.1 Resource creation

Although the SP environment is more restrictive than a NIM master in a classic RS/6000 environment, it does allow us to create new SPOT, lppsource, and mksysb resources very easily.

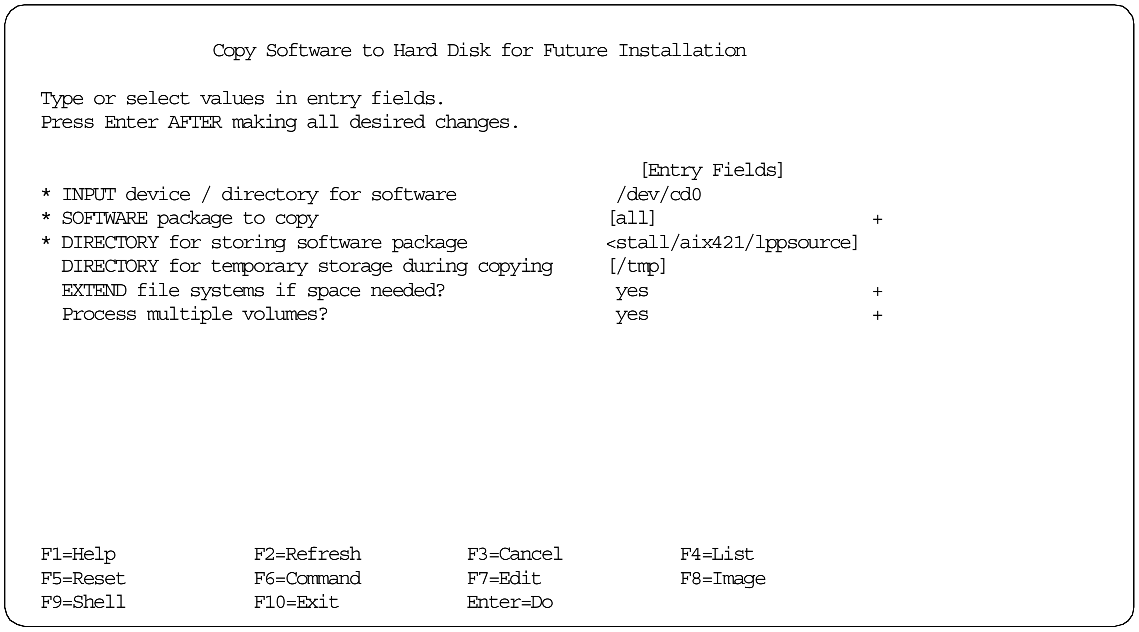
Let us look at a scenario on our SP system. We currently have PSSP 3.1.1 and AIX 4.3.3 installed on all our nodes and our CWS, but in order to test an application, we need to run one of our nodes at 4.2.1, and, so, we have to create the necessary resources to do this.

First, we need to `bffcreate` the AIX 4.2.1 filesets from our CD-ROM media. As we have already discussed, our lppsource directory structure must reside within /spdata/sys1/install; so, we need to make a new directory here called aix421 and another directory within that called lppsource:

```
# mkdir /spdata/sys1/install/aix421
# mkdir /spdata/sys1/install/aix421/lppsource
```

Next, we use `smitty bffcreate` to copy the images over from CD-ROM into our newly-created directory as shown in Figure 71. In this screen shot, note that we have entered the full path of /spdata/sys1/install/aix421/lppsource in the field, DIRECTORY for storing software package, but part of this has scrolled off the displayable part of the input area.

Now that our lppsource directory has been populated, we need to copy over our mksysb image. Once again, the SP environment restricts us in where we place the mksysb file, and, so, the image must reside within /spdata/sys1/install/images. We can either use a minimal install image from the media supplied with the SP hardware or use an existing mksysb image from an AIX 4.2.1 node. In our example, we have already installed the spimg installp image, and we have a minimum image file named bos.obj.ssp.421.



```
               Copy Software to Hard Disk for Future Installation

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* INPUT device / directory for software             /dev/cd0
* SOFTWARE package to copy                         [all]                  +
* DIRECTORY for storing software package           <stall/aix421/lppsource]
  DIRECTORY for temporary storage during copying   [/tmp]
  EXTEND file systems if space needed?              yes                   +
  Process multiple volumes?                         yes                   +









F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 71. Using bffcreate to copy over AIX 4.2.1 images

If you have your own mksysb image, you may name it what you like, but you must ensure that the permissions on the file are set to `rw-r--r--` (644).

In this example, we also need to make the directory, /spdata/sys1/install/pssplpp/PSSP-2.4, and then `bffcreate` our required filesets from the PSSP install media into this directory. This is because PSSP 3.1.1 only supports AIX 4.3.2 and above.

We can do this in the same way that we created the AIX 4.2.1 install images, but we can press **F4** on the SOFTWARE package to copy option in order to pick only the images we want.

We now have everything we need to create our NIM resources, and, so, we can make `setup_server` do this for us. All we need to do is allocate them to a node in the SDR using spchvgobj and then set the node to install with spbootins. Figure 72 shows the process of performing these operations.

```
# splstdata -b -l 7
                  List Node Boot/Install Information

node#         hostname  hdw_enet_addr srvr     response           install_disk
      last_install_image   last_install_time  next_install_image lppsource_name
                pssp_ver         selected_vg
-------------------------------------------------------------------------------
    7 f01n07            02608CE87824    0         disk                 hdisk1
         bos.obj.ssp.433 Mon_Nov_10_14:34:33     bos.obj.ssp.433        aix433
                PSSP-3.1.1              rootvg
# spchvgobj -r rootvg -i bos.obj.ssp.421 -v aix421 -p PSSP-2.4 -l 7
spchvgobj: Successfully changed the Node and Volume_Group objects for node numbe
r 7, volume group rootvg.
spchvgobj: Successfully updated the Syspar object for partition cws1.
spchvgobj: There is no custom file to update for partition cws1.
spchvgobj: The total number of changes successfully completed is 1.
spchvgobj: The total number of changes which were not successfully completed is 0.
# spbootins -r install -s no -l 7
# splstdata -b -l 7
                  List Node Boot/Install Information

node#         hostname  hdw_enet_addr srvr     response           install_disk
      last_install_image   last_install_time  next_install_image lppsource_name
                pssp_ver         selected_vg
-------------------------------------------------------------------------------
    7 f01n07            02608CE87824    0      install                 hdisk1
         bos.obj.ssp.433 Wed_Nov_10_14:34:56     bos.obj.ssp.421        aix421
                PSSP-2.4                rootvg
#
```

Figure 72. Changing the volume group SDR objects

Notice that we use `splstdata` to check the status of the node before and after our changes. We also run `spbootins` with the `-s no` flag; so, `setup_server` is not run automatically. In this way, we can double-check that everything looks alright before we go ahead and run it.

Now, all we have to do is run `setup_server`. Then, this creates the three new NIM objects: lppsource_aix421, spot_aix421, and mksysb_2. Notice that the

SP naming convention adds lppsource_ and spot_ before your own lppsource and spot names. It also numerically names the mksysb resources. In our example, we only had one other mksysb resource, which was an AIX 4.3.3 minimal image given the NIM name of mksysb_1.

3.4.5.2 Maintaining lppsource and SPOT resource

We have already discussed this topic in some detail in Section 1.5, “Administration” on page 87, and, in an SP environment, the rules are the same.

Our aim is to keep our SPOT at the latest level of code and install the device support we require into it. When placing fixes in lppsource, we must run a nim -o check <lppsource> where <lppsource> is our lppsource name and then run an update_all operation on the SPOT with nim -Fo cust -a lpp_source=<lppsource> -a fixes=update_all <spot>, where <lppsource> is the name of our lppsource and <spot> is the name of our SPOT.

In a similar way, if we install specific fixes to a node, we should also install these fixes to the lppsource and update the SPOT in case we then want to use an mksysb image from the node to install others.

3.4.5.3 Using an mksysb image to clone nodes

When we perform an SP installation, we are effectively doing a clone every time since we are using an mksysb image and then letting AIX install any extra filesets it requires from the lppsource.

One very popular method of installing a new SP system is to install your first node with the basic image and then manually add your required application software and configure/customize anything else you may require.

You can then simply mksysb the node to a file either on the node itself or by exporting an NFS directory from the CWS to the node and performing the mksysb in this directory.

In our example, we have created a directory on the CWS called /mksysb, which we have NFS exported to the node from which we want to take the mksysb. We then mount this and take the mksysb as shown in Figure 73 on page 266.

```
# mount cws1:/mksysb /tmp/mksysb
# mksysb -X -i /tmp/mksysb/mksysb.node5

Creating information file (/image.data) for rootvg....
0512-039 mksysb: WARNING: /tmp/mksysb/mksysb.node5 does not appear to be a tape device and
        will NOT have a bootable image.

Creating list of files to back up.
Backing up 12679 files.............................
10084 of 12679 files (79%)......
0512-038 mksysb: Backup Completed Successfully.
0512-040 mksysb: WARNING: /tmp/wayne/mksysb.node5 does not appear to be a tape device and
        does NOT have a bootable image.
#
```

Figure 73. Taking an mksysb of a node

You will notice that mksysb warns us, at the beginning and at the end of the backup, that it is not a bootable image. This is a normal warning when backing up to a file, and it is nothing to worry about.

Now that the backup has completed, we can unmount the NFS directory from the node with umount /tmp/mksysb and move the image into our /spdata/sys1/install/images directory with mv /mksysb/mksysb.node5 /spdata/sys1/install/images.

If we have installed any PTFs to AIX on the node, we must ensure that our lppsource and SPOT contain the same level of code (see Section 3.4.5.2, “Maintaining lppsource and SPOT resource” on page 265, for more details) before we utilize this mksysb image to install other nodes.

In order to use the mksysb image to install our other nodes, we simply allocate the mksysb image to the relevant nodes in the SDR and set the nodes to install. In our example, we want to use the image on nodes 3 and 7.

This example is shown in Figure 74 on page 267. After setting the installation image and bootp response in the SDR, we check to see that everything looks correct with splstdata. We can now go ahead and run setup_server, which will define the mksysb as a NIM resource, and then netboot the nodes to install them.

```
# spchvgobj -r rootvg -i mksysb.node5 -l 3,7
spchvgobj: Successfully changed the Node and Volume_Group objects for node number 3,
volume group rootvg.
spchvgobj: Successfully changed the Node and Volume_Group objects for node number 7,
volume group rootvg.
spchvgobj: The total number of changes successfully completed is 2.
spchvgobj: The total number of changes which were not successfully completed is 0.
# spbootins -r install -s no -l 3,7
# splstdata -b -l 3,7
                  List Node Boot/Install Information

node#        hostname  hdw_enet_addr srvr     response           install_disk
      last_install_image   last_install_time  next_install_image lppsource_name
                pssp_ver        selected_vg
-------------------------------------------------------------------------------
    3 f01n03            02608CE87975    0      install                hdisk1
        bos.obj.ssp.433 Thu_Oct_28_12:25:48       mksysb.node5        aix433
             PSSP-3.1.1              rootvg
    7 f01n07            02608CE87824    0      install                hdisk1
        bos.obj.ssp.433 Wed_Nov_10_14:34:56       mksysb.node5        aix433
             PSSP-3.1.1              rootvg
#
```

Figure 74. Setting nodes to install from another mksysb resource

3.4.5.4 Installing PTFs on CWS and nodes

In a classic RS/6000 environment, NIM can make installing fixes on multiple machines very simple with the use of machine groups: Simply define a machine group containing all the stand-alone machines on which you want to install the fixes, and perform an update_all operation on it using your lppsource as the source.

On an SP, the environment is very different. The reason why machine groups are not supported in an SP environment is becausethe CWS, in its capacity as a NIM master, does not necessarily have control over all the nodes within the system.

This is because, if we have a multiple frame system, by default, each first node within each additional frame becomes a boot/install server and, thus, the NIM master for the other nodes within the frame. On the CWS, it retains the new boot/install server as a NIM client, but the nodes the boot/install server controls are deleted from the NIM database on the CWS.

However, while we do not have the benefit of NIM machine groups, we do have access to some useful parallel commands, such as `pcp` and `dsh`.

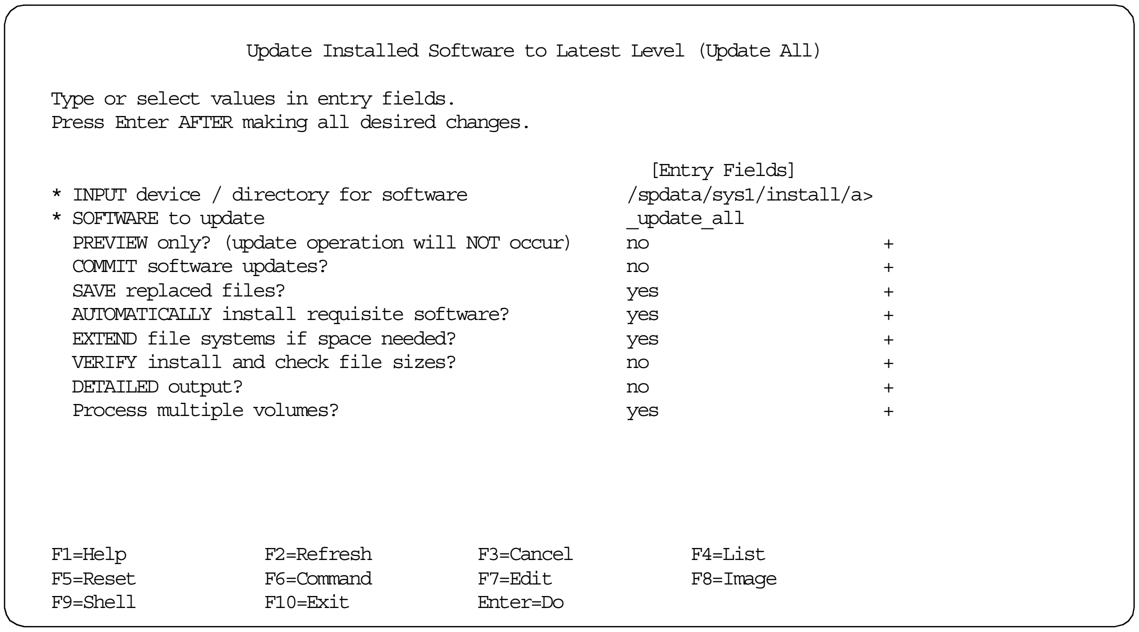
In our example scenario, we are running a base level version of AIX 4.3.3 on all our nodes and our CWS, but we want to install higher than the recommended maintenance level (RML) AIX 433-01.

Our first task is to move our newly-downloaded or bffcreated files into our lppsource (paying attention to any base level filesets that may have been included - see Section 3.4.2, “Corruption of an lppsource” on page 250, for more details).

Next, we use nim -o check on our lppsource and then nim -o cust on our SPOT to update them with the new filesets, as we discussed in Section 3.4.5.2, “Maintaining lppsource and SPOT resource” on page 265.

Our next step is to update the CWS with the fixes, which we can do using smitty update_all. From the initial input prompt, INPUT device / directory for software, we need to give our fully-qualified lppsource directory, which, in our case, is /spdata/sys1/install/aix433/lppsource.

From the next smitty screen, one option we may consider is PREVIEW only? (update operation will NOT occur). If we set this to yes, the installp program will perform all the prerequisite checking to see if the fixes can be applied without actually doing so.



```
                 Update Installed Software to Latest Level (Update All)

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                       [Entry Fields]
* INPUT device / directory for software              /spdata/sys1/install/a>
* SOFTWARE to update                                 _update_all
  PREVIEW only? (update operation will NOT occur)    no                    +
  COMMIT software updates?                           no                    +
  SAVE replaced files?                               yes                   +
  AUTOMATICALLY install requisite software?          yes                   +
  EXTEND file systems if space needed?               yes                   +
  VERIFY install and check file sizes?               no                    +
  DETAILED output?                                   no                    +
  Process multiple volumes?                          yes                   +




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 75. Choosing the update_all option on the CWS

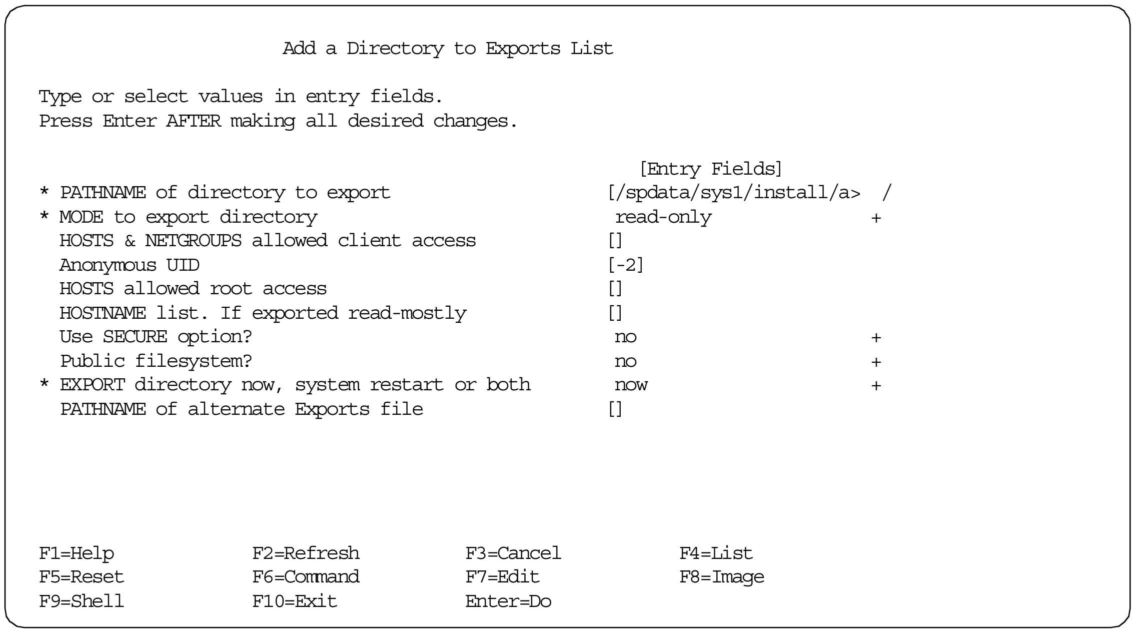
It is a good idea to set the COMMIT software updates? option to **no** (the default if yes) because the software will only be applied, and it can simply be rejected and the older version of the software restored if you encounter any problems. If you set this option to yes, you must also set the SAVE replaced files? option to yes. Figure 75 on page 268 shows our example smitty screen.

Depending on the filesets that are updated, the CWS may need to be rebooted before the changes in code take effect.

Before we install the fixes on all the nodes, it’s a good idea perform the update on a single node first to check that the new versions of the filesets are stable and don’t create any problems.

In order to do this, we simply NFS export the lppsource directory, mount it on our test node and then perform the update, we can utilize the `dsh` command so we can perform all the operations from the CWS.

We can make the NFS export by using the `mknfsexp` command or by using the smitty fastpath `smitty mknfsexp`.



```
                       Add a Directory to Exports List

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
* PATHNAME of directory to export                     [/spdata/sys1/install/a>  /
* MODE to export directory                             read-only               +
  HOSTS & NETGROUPS allowed client access             []
  Anonymous UID                                       [-2]
  HOSTS allowed root access                           []
  HOSTNAME list. If exported read-mostly              []
  Use SECURE option?                                   no                      +
  Public filesystem?                                   no                      +
* EXPORT directory now, system restart or both         now                     +
  PATHNAME of alternate Exports file                  []




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 76. Exporting the lppsource directory

In the example shown in Figure 76, we use smitty to export the lppsource directory; in the field, PATHNAME of directory to export, we use /spdata/sys1/install/aix433/lppsource. We also change the MODE to export directory to *read-only*, and we change EXPORT directory now, system restart or both to *now*. We choose read-only to prevent our lppsource from being accidently written to and *now* so the directory is not reexported after a reboot of the CWS.

In order to dsh the needed `installp` command to the nodes, we use the command line from the /smit.script file. There are two reasons to do this: First, so that we run exactly the same command as we used on the CWS and get the same installation options, and, second, because the `installp` command has many options and flags available, and, therefore, it is easy to make a mistake unless you are very familiar with it.

We only performed the update_all option a few stages ago in smitty; so, we can just `tail` the /smit.script file to file out the command we need to use as shown in Figure 77.

```
# tail /smit.script
#
#     [Nov 13 1999, 11:57:41]
#
/usr/lib/instl/sm_inst installp_cmd -a -d '/spdata/sys1/install/aix433/lppsource'
-f '_update_all'    '-g' '-X'
#
#     [Nov 13 1999, 11:58:05]
#
/usr/sbin/mknfsexp -d '/spdata/sys1/install/aix433/lppsource' -t 'ro' '-N'
```

Figure 77. Examining the /smit.script

It would also be possible for us to create the commands in this /smit.script file without actually performing the installation. If we use smitty -x update_all with the same options, smitty will just write the command it would have used into the /smit.script file without actually carrying out the actions. We can use the -x flag in smitty in order to create script files from all the smitty screens, not just for installation.

Because we will be mounting this directory over the /mnt mount point, the only change we need to make to the command line is the directory given from the -d flag. Our example is shown in Figure 78 on page 271.

Note that, for the sake of simplicity, we cut the 433-01 RML level down to a single fileset for bos.rte.console; however, the process we follow is exactly the same.

```
# dsh -w f01n01 mount cws1:/spdata/sys1/install/aix433/lppsource /mnt
# dsh -w f01n01 /usr/lib/instl/sm_inst installp_cmd -a -d '/mnt' -f '_update_all' '-g' '-X'
f01n01: installp -agqwX -d /mnt -f File 2>&1
f01n01:
f01n01: File:
f01n01:     bos.rte.console               4.3.3.1
f01n01:
f01n01: +-----------------------------------------------------------------------------+
f01n01:                    Pre-installation Verification...
f01n01: +-----------------------------------------------------------------------------+
f01n01: Verifying selections...done
f01n01: Verifying requisites...done
f01n01: Results...
f01n01: SUCCESSES
f01n01: ---------
f01n01:   Filesets listed in this section passed pre-installation verification
f01n01:   and will be installed.
f01n01:
f01n01:   Selected Filesets
f01n01:   -----------------
f01n01:   bos.rte.console 4.3.3.1                     # Console
f01n01:
f01n01:   << End of Success Section >>
f01n01:
f01n01: FILESET STATISTICS
f01n01: ------------------
f01n01:     1  Selected to be installed, of which:
f01n01:         1  Passed pre-installation verification
f01n01:   ----
f01n01:     1  Total to be installed
f01n01:
f01n01: +-----------------------------------------------------------------------------+
f01n01:                         Installing Software...
f01n01: +-----------------------------------------------------------------------------+
f01n01:
f01n01: installp:  APPLYING software for:
f01n01:         bos.rte.console 4.3.3.1
f01n01:
f01n01: +-----------------------------------------------------------------------------+
f01n01:                                Summaries:
f01n01: +-----------------------------------------------------------------------------+
f01n01:
f01n01: Installation Summary
f01n01: --------------------
f01n01: Name                        Level           Part        Event       Result
f01n01: -------------------------------------------------------------------------------
f01n01: bos.rte.console             4.3.3.1         USR         APPLY       SUCCESS
...
...
```

Figure 78. Installing PTFs on a node using dsh

Even from our simple example of installing a single fix, a great deal of output is generated. In fact, our example was edited to remove the copyright notices so it would fit on a single page. Based on the fact that our installation was successful, we can now go ahead and install it on all our other nodes, but because of the amount of output generated, we recommend redirecting this to a file and examining it later for any errors. This is especially important if you will be applying multiple PTFs over a large number of nodes.

Using dsh, we can specify a list of nodes we want to use, set up a working collective, or use the -a flag to execute the command on all the nodes in the current partition. In our example in Figure 79, we install the PTFs to all our nodes.

```
# dsh -a mount cws1:/spdata/sys1/install/aix433/lppsource /mnt
# dsh -a /usr/lib/instl/sm_inst installp_cmd -a -d '/mnt' -f '_update_all' '-g' '-X' >
/tmp/install.log
# dsh -a umount /mnt
#
```

Figure 79. Using dsh to install PTFs on all the nodes in the SP

All that remains to be done now is to examine the /tmp/install.log to check for any errors that might have occurred during the PTF installation.

3.5 NIM debugging on the SP

In this section, we cover the broader aspects of problem determination and devote the majority of it to documenting how to perform an install with a debug version of a SPOT in order to view debug information as the installation happens.

3.5.1 Basic problem determination steps

Performing a NIM debug installation can be a time-consuming task; so, before we go down this route, we need to ensure that all our basic checks have been done and ask ourselves some of the following questions.

First, if we are not even at the point of installation failure and we have a setup_server problem:

- Has our environment been correctly set up, and does the `splstdata` command show us the correct information?
- If we have a problem creating or changing the status of a NIM resource, does lsnim -l give us any clue towards solving the problem?
- Are there any appropriate log files to look at, such as the spot.out file?
- Is there space to create a resource within the file system?

- Name resolution problems can cause multiple problems in any environment. We should check that we are able to resolve hostnames to the correct IP addresses and the IP addresses back to hostname using the `host`, or `nslookup` command (if a name server is involved).
- Are the correct interfaces associated with the correct IP address and name? Is there a name resolution order to consider in the form of a /etc/netsvc.conf file or NSORDER environment variable?
- Is there any inconsistency between short and long hostnames being used, that is, do we use node1 in one instance and node1.domain.com in another?

A common solution for solving an unexplainable setup_server problem is to remove the NIM environment and then rebuild it. This can be time-consuming because of the resources that have to be rebuilt but can ultimately cure some other problems that could, in turn, be even more time-consuming.

Perform the following steps to rebuild the NIM environment on the CWS:

1. Set the nodes bootp response to disk with the `spbootins` command, for example: `spbootins-r disk -l 1`.
2. Remove all files except the *.cust files from the /tftpboot directory.
3. Remove the NFS exports related to the SP, that is, those within the /spdata directory. Use the `rmnfsexp` command or the `smitty rmnfsexp` fastpath.
4. Remove the NIM Master configuration with the `nim -o unconfig master` command.
5. Remove the NIM filesets with the `installp -ug bos.sysmgt.nim.master` and `installp -ug bos.sysmgt.nim.client` commands.
6. Set one of the nodes to install with `spbootins`, for example, `spbootins -r install -l 1`.

When the `spbootins` command is run, it will, in turn, run `setup_server`, which will reinstall the NIM filesets from the lppsource, reconfigure the NIM environment, and define all the resources.

Do not forget that, although the SPOTs are rebuilt, they are rebuilt from whatever the lppsource directory contains; so, if you have problems with files within the lppsource directories, this procedure will not help unless the files within the directories are removed and recreated from scratch.

If we have a problem installing or just attempting to install, the initial checks are still valid, but we go a little further. Are the correct directories exported? For an installation, the pssplpp directory and lppsources should be exported as read-only to all the nodes while the bosinst script, pssp_script, nim script, mksysb, and SPOT should only be exported to the node(s) on which we are installing, and the node should also have root access. To check this, we can use the `exportfs` command as shown in Figure 80.

```
# exportfs
/spdata/sys1/install/pssplpp                     -ro,root=f01n01:f01n03:f01n05:f
01n06:f01n07,access=f01n01:f01n03:f01n05:f01n06:f01n07
/spdata/sys1/install/aix421/lppsource            -ro
/spdata/sys1/install/aix433/lppsource            -ro
/spdata/sys1/install/pssp/1.noprompt             -ro,root=f01n01:,access=f01n01:
/spdata/sys1/install/pssp/pssp_script            -ro,root=f01n01:,access=f01n01:
/spdata/sys1/install/images/bos.obj.ssp.433      -ro,root=f01n01:,access=f01n01:
/export/nim/scripts/f01n01.script                -ro,root=f01n01:,access=f01n01:
/spdata/sys1/install/aix433/spot/spot_aix433/usr -ro,root=f01n01:,access=f01n01:
#
```

Figure 80. exportfs showing the correct exports for an install on f01n01

Another thing to check if an installation is causing some problems is to make sure the correct files within /tftpboot are present. In order to boot, install, and configure successfully, for each node, we should have a -new-srvtab file that is used by Kerberos, .info, .config_info and install_info files, which are used during the installation, and a file that is a symbolic link to the boot image. In the example shown in Figure 81, we show that the correct files exist in order to install the node f01n01.

```
# ls -l /tftpboot
total 8088
lrwxrwxrwx   1 root     system        33 Nov 15 18:40 f01n01 -> /tftpboot/spot_a
ix433.rs6k.up.ent
-r--------   1 nobody   system        80 Nov 15 18:17 f01n01-new-srvtab
-rw-r--r--   1 root     system       173 Nov 15 18:39 f01n01.config_info
-rw-r--r--   1 root     system      1139 Nov 15 18:40 f01n01.info
-rw-r--r--   1 root     system       704 Nov 15 18:39 f01n01.install_info
-rw-r--r--   1 root     system   4118460 Nov 15 18:39 spot_aix433.rs6k.up.ent
-rw-r--r--   1 root     system      3003 Oct 26 18:07 tuning.cust
#
```

Figure 81. Checking the files in /tftpboot

In addition to ensuring that the files exist, we also need to ensure that .info, .config_info and .install_info contain the correct information and that the symbolic link to the bootfile links to the correct bootfile for the node's hardware type.

3.5.2 Setting NIM to debug mode

This follows a procedure that is very similar to the classic RS/6000 version, the big difference being that the node has no console; so, we need to take some extra steps.

Before we even start, we should check that the node supervisor card on the node we want to debug is at the required level. If you have microcode prior to level 1294, the debug installation may loop while issuing the message:

```
032-001 You entered a command that is not valid
```

To determine this microcode level, we can use the spmon command in the format:

```
# spmon -G -q -l frame<fnumber>/node<nnumber>/codeVersion/value
```

where *<fnumber>* is the frame number that the node is in, and *<nnumber>* is the node, for example to check the microcode level on node 7 on our single frame system, we use the following:

```
# spmon -G -q -l frame1/node7/codeVersion/value
/SP/frame/frame1/node7/codeVersion/value/1557
```

Our node supervisor microcode is at the requisite level (1557); so, we can go ahead and create the debug SPOT. Even without the requisite level, the hang is by no means a certainty; so, if the microcode cannot be updated easily, it is well worth following the procedure through on the old microcode.

Before we can create a debug SPOT, we need to make sure that the resource is free, that is, that the SPOT should not be allocated to any of the nodes. The node we want to debug should be set back to disk, which we can do with the `spbootins` command; in our example, we are going to debug node 7; so, our command line would be:

```
# spbootins -r disk -l 7
```

In order to check if the SPOT is allocated to any other machine we need to check the alloc_count attribute of the SPOT. If the alloc_count is free, we should get a 0 returned. In our example, we use our spot called spot_aix433. If you are not sure of your spot's name, the best way of listing all the spots is the "`lsnim -t spot`" command:

```
# lsnim -a alloc_count spot_aix433
spot_aix433:
   alloc_count = 2
```

We need to find out which nodes have the SPOT allocated and deallocate it from them.

We can see which nodes have the SPOT allocated by using the `lsnim -a spot` command and then deallocate in one of two ways: The first way is to set the nodes back to disk and then let setup_server run with the `spbootins` command. The second is a more manual method but may save some time because setup_server will not have to run.

In our example shown in Figure 82, we use the manual method by first resetting the client and then deallocating the specific resource. We can then check the alloc_count attribute to ensure that the SPOT is free.

In this way, we are putting the NIM database and the SDR out of sync; so, it will remain like this until setup_server is run once again.

```
# lsnim -a spot
f01n05:
   spot = spot_aix433
f01n06:
   spot = spot_aix433
# nim -Fo reset f01n05
# nim -Fo reset f01n06
# nim -o deallocate -a spot=spot_aix433 f01n05
# nim -o deallocate -a spot_spot_aix433 f01n06
# lsnim -a alloc_count spot_aix433
spot_aix433:
   alloc_count = 0
#
```

Figure 82. Freeing up the SPOT resource

Next, we can go ahead and produce our debug version of the SPOT with the `"nim -Fo check -a debug=yes ..."` command. Right after we do this, we need to examine the SPOT with lsnim -l to check the enter_dbg attributes. This is illustrated in Figure 83 on page 277.

```
# nim -Fo check -a debug=yes spot_aix433
# lsnim -l spot_aix433
spot_aix433:
   class         = resources
   type          = spot
   enter_dbg     = "rs6k.up.ent 0x001fa694"
   Rstate        = ready for use
   prev_state    = verification is being performed
   location      = /spdata/sys1/install/aix433/spot/spot_aix433/usr
   version       = 4
   release       = 3
   mod           = 3
   alloc_count   = 0
   server        = master
   if_supported  = rs6k.up ent
   Rstate_result = success
   plat_defined  = chrp
   plat_defined  = rs6k
   plat_defined  = rspc
```

Figure 83. Checking the enter_dbg attribute

Within a debug SPOT, it is possible to have multiple enter_dbg attributes, that is, one attribute for each type of node. We need to take note of the correct `enter_dbg` line, depending on the node we want to debug; so, for example, if we have an SMP PCI node, then we would use the chrp.mp line, for Uni processor PCI, we would use chrp.up, and, if we had an MCA SMP node, we would use the line, rs6k.mp. In our case, we have only one choice and only one type of node: Uniprocessor MCA; so, we would use the `rs6k.up` entry, specifically, the line:

```
enter_dbg     = "rs6k.up.ent 0x001fa694"
```

Before we start the debug, we need to take a note of the hex number, 0x001fa694. The first two characters of the line, 0x, just indicate that the number that will follow is hexadecimal; so, we can ignore these. We can also ignore the leading zeros in the hex number, therefore, the actual number we need to take note of is 1fa694.

The hexadecimal number is important to note down because it is a specific memory address that we have to change a value of. Storing a certain number into this memory address allows us to actually turn on the debug output.

Our next step is to reallocate the SPOT to the node we want to debug, in our example, node7. We can do this once again with `spbootins`, and then let setup_server run.

```
# spbootins -r install -l 7
```

Once `setup_server` has completed, we need to netboot our node using perspectives or the `nodecond` command and open a read-only `s1term`. In our example, we perform this manually with the following commands:

```
# nodecond 1 7 &
# s1term 1 7
```

We then wait for the line beginning `Trap instruction interrupt` to appear on the screen as shown in Figure 84.

```
XER 0000000C  SRR0 0007616C  SRR1 000210B0  DSISR 40000000  DAR  00000000

IAR 0007616C  (ORG+0007616C)  ORG=00000000 Mode: VIRTUAL
00076160   67697374 65723020 80001024 7C810808   |gister0 ...$|...|
                                       |      teq   r1,r1
00076170   4E800020 60000000 2C030010 38A00001   |N.. `...,...8...|

                                       |
00076160   67697374 65723020 80001024 7C810808   |gister0 ...$|...|
00076170   4E800020 60000000 2C030010 38A00001   |N.. `...,...8...|
00076180   41800008 38A00000 0C850000 38A00001   |A...8.......8...|
00076190   1C630150 7CA903A6 80C21F48 7CA61A14   |.c.P|......H|...|
000761A0   38C5FF5C 3860FFFF 84E600A8 48000018   |8..\8`......H...|
000761B0   41820010 80060094 7C002040 41820074   |A.......|. @A..t|
000761C0   84E600A8 7C071840 4200FFE8 41820010   |....|..@B...A...|

Trap instruction interrupt.
>
```

Figure 84. Waiting for the Trap Instruction interrupt line

Now that the boot has stopped at this breakpoint, we need to exit our read-only s1term by issuing a <CTRL-C> and then open a read/write terminal using s1term or spmon. It is possible for the `nodecond` command to keep the writable terminal open, which may stop you from opening your own writable terminal. If this happens, you can find the command in the ps -ef listing and kill the PID.

Once we have our read/write terminal open, we store the number 2 into the hex address we jotted down from the enter_dbg attribute of the SPOT. In order to do this, we use the `st` command using the following format:

`st` *<address> <value>*

Where *<address>* is the address to write into, and *<value>* is a word value to write into the address. Writing the 2 to the enter_dbg address makes the SPOT write debug messages to the console as it runs. Our last task is to manually get the process running again by using the `g` command, which means go. This process is illustrated in Figure 85.

Note that this screen only shows the start of the NIM debug output, not a full debug output. In order to capture the output, you should log the output from your terminal to a file.

```
# spmon -o node7
Connecting to frame 1, node 7.

Press <Enter> to proceed.  Press <ctrl>-x to close connection.
> st 1fa694 2
> g
AIX Version 4.3
+ [ 1 -ne 1 ]
+ PHASE=1
+ + bootinfo -p
PLATFORM=rs6k
+ [ ! -x /usr/lib/boot/bin/bootinfo_rs6k ]
+ [ 1 -eq 1 ]
+ 1> /usr/lib/libc.a
+ init -c unlink /usr/lib/boot/bin/!(*_rs6k)
+ chramfs -t
+ init -c unlink /usr/sbin/chramfs
+ 1> /dev/null
+ + bootinfo -t
BOOTYPE=5
...
...
```

Figure 85. Start of the NIM debug output

In some instances, the node installation may hang on LED c46. If this happens, it is not indicative of a problem, but it does mean that we have to manually restart the process by issuing a <CTRL-q> into the s1term window.

It would be impossible to cover every situation in which a node fails to install, but, as a guide, we should first look at the LEDs to see if the installation is hanging on a value and check out the configuration depending on what the LED means.

If we cannot see a reason for an LED hang, it is time to try a NIM debug installation. Generally speaking, we are interested in where the debug messages stop or hang for any reason. We can check if there is an associated error message and, thus, investigate further.

There is an example of how we use NIM debug to solve a problem in Section 3.5.3, “An example of using NIM debug in an LED 611 hang” on page 280.

When you have completed your debug installation, you need to set your SPOT back to normal. This entails using the `nim -Fo check` command again but, this time, without using the -a debug option. In our example, we would use the command:

```
# nim -Fo check spot_aix433
```

3.5.3 An example of using NIM debug in an LED 611 hang

Let us look at a common problem as a practical example. We have set one of our nodes to install, but it is hanging on an LED of 611.

A hanging LED of 611 is a fairly simple problem to determine; so, it probably would not normally be investigated by running NIM in debug, but it does illustrate a typical problem quite well.

We have gone ahead and made our debug SPOT, netbooted the client, and started the debug installation as described in Section 3.5.2, “Setting NIM to debug mode” on page 275.

As we watch the NIM debug information scroll past the screen, the LED reaches 611, and the debug output stops, leaving us with the screen shown in Figure 86.

```
+ OIFS=

+ IFS=:
+ set -- 10.3.187.243 cwst1
+ IFS=

+ echo 10.3.187.243 cwst1
+ 1>> /etc/hosts
+ [ -n  ]
+ 1> /etc/filesystems
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED 610: mount -r
cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr /SPOT/usr
+ /usr/lib/methods/showled 0x610
 showled + mount -r cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr /SPOT
/usr
mount: access denied for cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr
mount: giving up on:
        cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr
Permission denied
+ [[ 1 -ne 0 ]]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED 611: failure:
mount -r cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr /SPOT/usr
+ loopled 0x611
 showled
```

Figure 86. NIM debug console output

All we need to do is glance up from the last line of output to see that the client attempted to execute the command:

```
mount -r cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr /SPOT/usr
```

Next, we see an error message returned from the CWS:

```
mount: access denied for
cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr
mount: giving up on:
        cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr
Permission denied
```

NIM itself helps us in our debugging by getting the client to update its own info field. This is seen in the following command.

```
nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED 611:
failure: mount -r cwst1:/spdata/sys1/install/aix433/spot/spot_aix433/usr
/SPOT/usr
```

It shows us that another way of determining a lot of NIM-related problems is to examine the client itself to see if it provides any useful information as shown in Figure 87.

```
# lsnim -l f01n07
f01n07:
   class          = machines
   type           = standalone
   platform       = rs6k
   netboot_kernel = up
   if1            = spnet_en0 f01n07 02608CE87824 ent
   cable_type1    = bnc
   Cstate         = BOS installation has been enabled
   prev_state     = ready for a NIM operation
   Mstate         = currently running
   info           = LED 611: failure: mount -r cwst1:/spdata/sys1/install/aix433
/spot/spot_aix433/usr /SPOT/usr
   boot           = boot
   bosinst_data   = 7_noprompt
   lpp_source     = lppsource_aix433
   mksysb         = mksysb_2
   nim_script     = nim_script
   script         = psspscript
   spot           = spot_aix433
   cpuid          = 000047067900
   control        = master
#
```

Figure 87. Using lsnim -l to check for errors

Therefore, the problem must be that the /spdata/sys1/install/aix433/spot/spot_aix433/usr directory is not exported to the node. We can confirm this with the `exportfs` command as shown in Figure 88.

```
# exportfs
/spdata/sys1/install/pssplpp                     -ro,root=f01n01:f01n03:f01n05:f
01n06:f01n07,access=f01n01:f01n03:f01n05:f01n06:f01n07
/spdata/sys1/install/aix421/lppsource            -ro
/spdata/sys1/install/aix433/lppsource            -ro
/spdata/sys1/install/pssp/7.noprompt             -ro,root=f01n07:,access=f01n07:
/spdata/sys1/install/pssp/pssp_script            -ro,root=f01n07:,access=f01n07:
/spdata/sys1/install/images/bos.obj.ssp.433      -ro,root=f01n07:,access=f01n07:
/export/nim/scripts/f01n07.script                -ro,root=f01n07:,access=f01n07:
/spdata/sys1/install/aix433/spot/spot_aix433/usr -ro,root=f01n07:,access=f01n07:
#
```

Figure 88. Checking the NFS exports

In fact, the directory is exported, and our node, f01n07, has root access to it; so, perhaps, something else is going wrong.

From a purely IP problem determination point of view, our next step might be to further investigate the NFS mount request using the `iptrace` or `tcpdump` commands. This is a perfectly valid exercise to perform, but we should still see what else we can learn from the log file.

Therefore, our next task is to scan further back up the log file we produced to look for any other possible problems. We show another section of our output in Figure 89 on page 283.

```
Method error (/usr/lib/methods/defssar):
        0514-068 Cause not known.
sh: /usr/lib/methods/defssar:  not found

----------------
Time: 18        LEDS: 0x538
Invoking top level program -- "/usr/lib/methods/deftmssar"
 cfgmgr  cfgmgr Time: 19        LEDS: 0x539
Return code = 127
*** no stdout ****
***** stderr *****
sh: /usr/lib/methods/deftmssar:  not found

Method error (/usr/lib/methods/deftmssar):
        0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found

 cfgmgr Configuration time: 21 seconds
+ export NSORDER=local
+ bootinfo -b
+ [ ent0 = atm0 ]
+ native_netboot_cfg
+ bootinfo -c
+ set -- 10.3.187.250 10.3.187.243 10.3.187.243 68 0 0 /tftpboot/f01n07 99.130.8
3.99.1.4.255.255.255.0.255.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0. 7
+ CLIENT_IPADDR=10.3.187.250
+ BOOT_SERV_IP=10.3.187.243
+ BOOT_GATE_IP=10.3.187.243
+ E802=0
+ BOOTFILE=/tftpboot/f01n07
+ VEND=99.130.83.99.1.4.255.255.255.0.255.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
+ [ -n 255.255.255.0 ]
+ SUBMASK=netmask 255.255.255.0
+ [ 10.3.187.243 = 0 -o 10.3.187.243 = 0.0.0.0 -o 10.3.187.243 = 10.3.187.243 ]
+ unset BOOT_GATE_IP
+ + bootinfo -b
PHY_BOOT_DEV=ent0
+ pdev_to_ldev
+ /usr/lib/methods/showled 0x606
 showled + ifconfig lo0 inet 127.0.0.1 up
+ ifconfig en0 inet 10.3.187.250 up netmask 255.255.255.0
+ [ 0 -ne 0 ]
+ [  ]
+ [ -n  ]
+ CLIENT_INFO_FILE=/tftpboot/f01n07.info
+ [ 0 -ne 0 ]
+ /usr/lib/methods/showled 0x608
 showled + tftp -go /SPOT/niminfo 10.3.187.243 /tftpboot/f01n07.info image
Received 1139 Bytes in 0.0 Seconds
```

Figure 89. Examining further NIM debug information

This portion of our debug output illustrates several useful things. First, near the top of the screen, we can see method errors returned by `cfgmgr`. It is quite possible to see numerous errors of this type without any of them indicating a problem because, at the moment `cfgmgr` is running from the SPOT, there may be items that cannot be configured.

The second portion of interest to us is the fact that there's no problems in performing the tftp transfers which we can see from the last few lines.

```
tftp -go /SPOT/niminfo 10.3.187.243 /tftpboot/f01n07.info image
Received 1139 Bytes in 0.0 Seconds
```

This shows is that it believes the CWS interface to use is 10.3.187.243, but also that in order for a tftp transfer to work then the ethernet interface must have been configured correctly, we can this from the line:

```
ifconfig en0 inet 10.3.187.250 up netmask 255.255.255.0
```

So this means that when the NFS mount request comes in to the CWS the source IP address will be 10.3.187.250. The CWS has exported the directory to f01n07; so, does the IP address and the hostname match? We can find out with the `host` command on the CWS.

```
# host f01n07
f01n07 is 10.3.187.251
```

Therefore, this is our problem: The /etc/bootptab file and the config files within /tftpboot all give the node's client IP address as 10.3.187.250, but the nodes hostname resolves to the IP address 10.3.187.251. This is what is stopping our NFS mount and, consequently, the install.

In our example, the IP address within our /etc/hosts file was incorrect; so, our action was to change this entry and rerun the installation procedure.

3.5.4 SP LEDs

In Section 2.10.1, "NIM relevant LED codes" on page 140, we looked at the NIM-specific LED values that are displayed during an installation or migration.

In this section, we will look at the SP-specific LED that you may see during an installation, migration, or customization. These codes will appear when running the pssp_script and psspfb_script scripts; so, in the case of an installation or migration, either will appear after the normal NIM LEDs. However, in the case of a customization, they will either be the only codes we see (if pssp_script is run manually), or, in the case of rebooting the node to perform the customization, they will appear right after the node boots up.

Table 29 shows the chronological order in which the LEDs should appear when the node is customizing. Note that, depending on the circumstances, not all the LEDs will appear. For example, u52 or u51 will appear depending on whether the client is a uniprocessor or multiprocessor; therefore, both cannot appear.

Table 29. pssp_script LED/LCD codes

LED/LCD Value	Description
u20	Create log directory (enter function create_directories)
u21	Establish working environment (enter function setup_environment)
u03	Get the node.install_info file from the master
u04	Expand node.install_info file
u22	Configure node (enter function configure_node)
u57	Get the node.config_info file from the master
u59	Get the cuat.sp template from the master
u23	Create/update /etc/ssp files (enter function create_files)
u60	Create/update /etc/ssp files
u24	Update /etc/hosts file (enter function update_etchosts)
u25	Get configuration files (enter function get_files)
u61	Get /etc/SDR_dest_info from boot/install server
u79	Get script/cust from boot/install server
u50	Get tuning.cust from boot/install server
u54	Get spfbcheck from boot/install server
u56	Get psspfb_script from boot/install server
u58	Get psspfb_script from control workstation
u26	Get authentication files (enter function authent_stuff)
u67	Get /etc/krb.conf from boot/install server
u68	Get /etc/krb.realms from boot/install server
u69	Get krb-srvtab from boot/install server
u27	Update /etc/inittab file (enter function update_etcinittab)

LED/LCD Value	Description
u28	Perform SMP-specific functions (enter function upmp_work)
u52	Processor is “MP”
u51	Processor is “UP”
u29	Install prerequisite filesets (enter function install_prereqs)
u55	Fatal error in bosboot
u30	Install ssp.clients (enter function install_ssp_clients)
u80	Mount lppsource and install ssp.clients
u31	Install ssp.basic (enter function install_ssp_basic)
u81	Install ssp.basic
u32	Install ssp.ha (enter function install_ssp_ha)
u53	Install ssp.ha
u33	Install ssp.sysctl (enter function install_ssp_sysctl)
u82	Install ssp.sysctl
u34	Install ssp.pman (enter function install_ssp_pman)
u41	Configure switch (enter function config_switch)
u35	Install ssp.ccs (enter function install_ssp_css)
u84	Install ssp.ccs
u36	Install ssp.jm (enter function install_ssp_jm)
u85	Install ssp.jm
u37	Delete master .rhosts entry (enter function delete_master_rhosts)
u38	Create new dump logical volume (enter function create_dump_lv)
u86	Create new dump logical volume
u45	Start silver node surveillance (enter function start_silv_surv)
u43	Start mirroring/unmirroring (enter function start_mirroring)
u40	Run script.cust (enter function run_script_cust)
u87	Run script.cust file
u42	Run psspfb_script (enter function run_psspfb_script)

In Table 30, we show the chronological order of LED, and we show when the pssp first boot script (psspfb_script) runs. This will run after the initial reboot if we are installing or migrating a node or if we are specifically customizing a node. It is run from the pssp_script.

Table 30. psspfb_script LED/LCD codes

LED/LCD Value	Description
u90	Set up working environment (enter function setup_environement)
u91	Unconfigure adapters (enter function unconfig_adapters)
u92	Configure adapters (enter function config_adapters)
u93	Configure inet0 (enter function config_inet0)
u94	Run cfgmgr (enter function run_cfgmgr)
u95	Run complete_node on boot/install server (enter function complete_node)
u78	Set the KRBTKFILE variable and get an rcmd ticket
u96	Run the firstboot.cust script (enter function run_firstboot_cust)

A customization problem can be handled slightly differently than a pure installation problem because, in many cases, the node will be installed successfully with AIX. If a node is installed with AIX, it means we can generally log into the node and determine what the error is without going through the time and effort of debugging installations.

Our first step in a customization problem is to check for an LED on which the process is hanging and check to see what should be happening according to the table. In some cases, it may be easier to refer to the script itself (using the copy on the CWS) to see what action is being attempted in more detail.

Both the pssp_script and the psspfb_script log to files on the local node stored in /var/adm/SPlogs/sysman. The pssp_script writes to the file *<node>*.config.log.*<PID>*, and the psspfb_script writes to *<node>*.configfb.log.*<PID>*, where *<node>* is the hostname of the node, and *<PID>* was the process identifier of the script while it was running.

If we cannot work out the problem purely from the LED, these files will show us more information on the problem. It may be that, although the node is installed, the network is not configured; so, we cannot use rsh or telnet. If this is the case, we can use a read/write s1term in order to log in. If, however, the node will not boot up into normal mode, we can put the node into maintenance mode using spbootins -r maintenance and then use the *Access* *a root volume group and start a limited function maintenance shell* option from the maintenance menu to access the node and examine the file.

Once we discover the problem, a manual fix may be simple, but it is still a good idea to make sure the scripts can run cleanly. In order to run these manually, we can start them off in one of two ways: First, we need to set the node to customize with the `spbootins` command; so, if we are doing this for node 1, we use the following command:

```
# spbootins -r customize -l 1
```

When a node is set to customize in the SDR, we do not netboot the node. Instead, we need to run the /etc/rc.sp script on the node either manually or by rebooting the node in normal mode. The /etc/rc.sp script is run from the /etc/inittab and will check the SDR bootp response for the node. If it is set to customize, it will go ahead and spawn off the pssp_script process.

Appendix A. Key NIM scripts used in the network boot process

This appendix is a summary of some important NIM scripts.

NIM_BOSINST_RECOVER
This is not really a script. It is an environment variable that NIM adds to a <client>.info file indicating what script to run to recover NIM information from the RAM file system during a BOS install. This is also used to recover some information that may have been lost during the installation.

It is created by the variable, $NIM_BOSINST_RECOVER, which is defined in the /tftpboot/<clienthostname>.info file. It generally points at the /SPOT/usr/lpp/bos.systmgt/NIM/method/c_bosinst_env script.

/SPOT/usr/lpp/bos.systmgt/NIM/method/c_bosinst_env script
This script will do the following:

- Copy the .info file to /etc/niminfo.
- Populate /etc/hosts file with values from $NIM_HOSTS variable in the niminfo file.
- Invoke the `hostname` command to set the host's name.

NIM customization scripts
The customization scripts are defined by the $NIM_CUSTOM variable in the /tftpboot/<clienthostname>.info file. This variable references the /SPOT/usr/lpp/bos.sytmgt/nim/methods/c_script.

/SPOT/usr/lpp/bos.sytmgt/nim/methods/c_script
It does not get created during a NIM bos_inst or cust operation. However, this script gets called because of one of those two operations. The NIM_CUSTOM environment variable gets added to a <client>.info file during a bos_inst operation. At that point, it will reference the c_script, which will trigger the nim_script resource for that client. During a *cust* operation though, the NIM_CUSTOM variable is not involved. Instead, the c_script script gets called directly, and that, in turn, calls the nim_script resource for that client. When this script is run, it NFS mounts and executes the /export/nim/scripts/<clienthostname>.script file.

/export/nim/scripts/<clienthostname>.script
This file will, in turn, start the following:

/SPOT/usr/lpp/bos.sysmgt/nim/methods/c_mk_nimclient
This is used if no_nim_client=no or is not specified. If it is set to yes, this script does not get called. It performs the following tasks:

- It will install bos.sysmgt.nim.client and bos.net*.client.
- It will deinstall the bos.sysmgt.nim.master if it is present.
- Invoke mktcpip to configure the network on the installed machine.
- Populate the new /etc/hosts file with values from the $NIM_HOSTS from /etc/niminfo (created by $NIM_BOSINST-RECOVER in the previous bi_main step).
- Add routes as specified in $ROUTES from /etc/niminfo.

/SPOT/usr/lpp/bos.sysmgt/nim/methods/c_installp
This file executes and mounts all the installp_bundle files that are allocated.

If there are additional nim_script resources, the /SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script is run for every script resource allocated.

Appendix B. Cloning script

This cloning script will change the boot kernel from the one located in the mksysb to the one necessary for these machines. This is only necessary if you clone one hardware type to another, for example, rs6k to rspc.

```
#!/usr/bin/ksh

set -x
installp -C
RV=$(bootinfo -z)

if [ "$RV" -eq 1 ]
then
    installp -abcgXd/../SPOT/usr/sys/inst.images bos.rte.mp
    ln -fs /usr/lib/boot/unix_mp /usr/lib/boot/unix
fi

if [ "$RV" -eq 0 ]
then
     installp -abcgXd/../SPOT/usr/sys/inst.images bos.rte.up
     ln -fs /usr/lib/boot/unix_up /usr/lib/boot/unix
fi

devinstall -b -d /../SPOT/usr/sys/inst.images -f /../tmp/device.pkgs
cfgmgr -v -i /../SPOT/usr/sys/inst.images

BLVDISK=$(lslv -l hd5 | grep hdisk | head -1 | cut -d' ' -f1)
ln -f /dev/r$BLVDISK /dev/ipldevice

bosboot -a -d /dev/ipldevice
bootlist -m normal $BLVDISK

rm -f /etc/firstboot

sync
sync
sync

exit 0
```

Appendix C. Special notices

This publication is intended to help professionals who need to plan for and implement NIM. The information in this publication is not intended as the specification of any programming interfaces that are provided by NIM or the AIX operating system. See the PUBLICATIONS section of the IBM Programming Announcement for AIX operating system for more information about what publications are considered product documentation.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only the IBM product, program, or service may be used. Any functionally-equivalent program that does not infringe any IBM intellectual property rights may be used instead of the IBM product, program, or service.

Information in this book was developed in conjunction with the use of the equipment specified and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently-created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including, in some cases, the payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor, and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into his or her operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee

AIX	AIX/6000
AIXwindows	AS/400
HACMP/6000	Home Director
IBM	Micro Channel
Netfinity	NetView
PowerPC	POWERparallel
POWER2 Architecture	RISC System/6000
RS/6000	Scalable POWERparallel Systems
SP	System/390
XT	

Appendix D. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 IBM Redbooks publications

For information on ordering these publications see “How to get IBM Redbooks” on page 301.

- *AIX Version 4.3 Differences Guide*, SG24-2014
- *Learning Practical TCP/IP for AIX V3.2/V4.1 Users: Hints and Tips for Debugging and Tuning*, SG24-4381
- *RS/6000 SP: Problem Determination Guide*, SG24-4778
- *AIX Version 4.2 Differences Guide*, SG24-4807
- *IBM Certification Study Guide - AIX V4.3 System Administration*, SG24-5129
- *IBM Certification Study Guide - AIX V4.3 System Support*, SG24-5139
- *RS/6000 SP Software Maintenance*, SG24-5160
- *PSSP 3 Survival Guide*, SG24-5344
- *IBM Certification Study Guide - RS/6000 SP*, SG24-5348
- *The RS/6000 SP Inside Out*, SG24-5374
- *RS/6000 SP System Management: Power Recipes for PSSP 3.1*, SG24-5628

D.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates, and formats.

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849

CD-ROM Title	**Collection Kit Number**
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.3 Other resources

These publications are also relevant as further information sources:

- *AIX Commands Reference V4.3*, SBOF-1877 (six volumes)
- *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127
- *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126
- *AIX Version 4.3 Problem Solving Guide and Reference*, SC23-4123
- *AIX Version 4.3 Network Installation Management Guide and Reference*, SC23-4113
- *AIX Version 4.3 Installation Guide*, SC23-4112
- *AIX Version 4.2 Installation Guide*, SC23-1924
- *AIX Version 4.1 for RISC System/6000 Installation Guide*, SC23-2550
- *AIX Version 3.2 Installation Guide*, SC23-2341
- *AIX and Related Products Documentation Overview*, SC23-2456
- *AIX Performance Monitoring and Tuning Guide*, SC23-2365
- *RS/6000: Planning Volume 2*, GA22-7281
- *PSSP: Installation and Migration Guide*, GA22-7347
- *PSSP: Administration Guide*, SA22-7348
- *PSSP: Command and Technical Reference*, SA22-7351
- *What's New in AIX v4.3.3 Presentation Material*
- *SP Problem Determination Course Material in ILS*
- *NIM Course Material in ILS*
- *News Group* - ibm.ibmunix.nim
- *Forum* - NIM FORUM on IBMUNIX

D.4 Referenced Web sites

The following Web site is also relevant as a further information source:

`http://www.ibm.com/servers/aix/`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access `MyNews` at `http://w3.ibm.com/` for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

AIX	Advanced Interactive Executive
APAR	Authorized Program Analysis Report
ASCII	American National Standards Code for Information Interchange
ATM	Asynchronous Transfer Mode
BIS	Boot/Install Server
BOOTP	Boot Protocol
BOS	Base Operating System
CAD	Computer Aided Design
chrp	Common Hardware Reference Platform
CD	Compact Disc
CPU	Central Processing Unit
CWS	Control Workstation
GB	Giga Byte
DNS	Domain Name Services
DWM	Diskless Workstation Management
EPROM	Erasable Programmable Read-Only Memory
GUI	Graphical User Interface
GW	Gateway
IBM	International Business Machines
ID	Identifier
IP	Internet Protocol
IPL-ROM	Initial Program Load - Read-Only Memory
ITSO	International Technical Support Organization
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LPP	Licensed Program Product
LUM	Licence Use Management
MCA	Micro Channel Architecture
MB	Mega Byte
MP	Multi Processor
MTU	Maximum Transfer Unit
NFS	Network File System
NIM	Network Installation Management
NIS	Network Information Services
NVRAM	Non Volatile Random Access Memory
ODM	Object Data Manager
PCI	Peripheral Component Interconnect
PID	Process Identifier
PSSP	Parallel System Support Programs
RIPL	Remote Initial Program Load
RISC	Reduced Instruction Set Computing
ROM	Read-Only Memory
ROS	Read-Only Storage
RS	NIM Resource Server
rs6k	Micro Channel based RISC System/6000

rspc	IBM Power PC computer
rte	Run Time Environment
SDR	System Data Repository
SMP	Symmetric Multi Processor
SMS	System Management Service
smitty	System Management Interface Tool
SP	Scalable Power parallel
SPOT	Shared Product Object Tree
TCP/IP	Transmission Control Protocol/Internet Protocol
tftp	Trivial File Transfer Protocol
TR	Token Ring
UDP	User Datagram Protocol
UP	Uni Processor
URL	Universal Resource Locator
VSD	Virtual Shared Disk
WSM	Web-based System Manager

Index

Symbols
$NIM_BOSINST_RECOVER 35, 289
$NIM_CUSTOM 36
$NIM_HOSTS 289
$RC_CONFIG 34
$ROUTES 290

A
Administration 87
allnimres 228
    usage of 233
alt_disk_install log 151
alternate disk installation 117
    cloning the rootvg 118
    command 121
    debug messages log 119
    error messages log 119
    Installing a new mksysb 118
    log 119
    prerequisites 118
    progress messages log 119
    steps 119
APAR 186
ATM 130
    boot failed 132
    boot over 130
    change generic into ATM network 132
    filesets 131
    prerequisites 131

B
bffcreate 209, 253, 263
Bi_main script
    flow diagram 38
    procedure 35
bi_main script 35
BIND 140
boot image 132, 142
    create 132
    obtaining 142
    recover 132
    running 142
    transfer to client 143
boot log 151
Boot resource 14, 69
Boot/Install server 216
bootinfo command 26
bootlist 37, 122, 124
    Set 122
bootp 33, 142, 255
bootpd 144
    debug mode 144
bos file
    permissions of 262
BOS installation 71
    mksysb 71
    rte 71
    SPOT 71
BOS restore 113
bosboot command 36
bosinst log 151
bosinst_data 18
    CUSTOMIZATION_FILE 37
    debugging purposes 161
    enhancements in AIX 4.3 105
    example 19
bridge 198
buffer overrun 171, 172

C
c_installp 290
c_mk_client 290
c_script 36, 289
c_sh_lib file 249
cases studies 173
cfgmgr 34, 284
Client
    C_state 35
    defining 58
    properties when configured 62
client 4
Client control 93
    states 94
Client definition file 62
    keywords 63
    rules 63
    sample 66
    stanza errors 65
Client initiated 189
    allocate resources 190
    change client definition info 193
    cust operation 191

diag boot 191
display contents 191
list resources 190
maintenance boot 191
push permission 192
reset NIM state 191
set date and time 192
start BOS installation 190
Cloning 113
command 117
graPHIGS 114
Licence Use Management 114
mksysb 113
OpenGL 114
prerequisites 113
procedure 113
restoring the devices 114
script for AIX 4.1 117
steps 114
command
nimclient 190, 191, 192, 193
nimdef 194, 197
showlog 150
Control Workstation
description of 205
create_krb_files 227
Customization 87, 200
installing software in a SPOT or machine 89
problems with 287
rules 89, 90
update level of AIX on machine 87

D
data link protocols 198
Dataless machines 5
defining resources 68
resources 100
Dataless/diskless client 136
add to NIM environment 136
boot 137
boot image 136
command to add 137
command to boot 138
command to uninitialize 139
initialize 137
paging space 136
prerequisites to add 137
resource server 136
uninitialize 139
DDNS 135
debugging 154
log the debug output 158, 160
producing debug output from bootp 155
producing debug output from BOS install 160
producing debug output from IP-Trace 162
producing debug output from network boot 158
default route 53
establishing 53
definition stanza file 194
delnimclient 226
usage of 229
delnimmast 226
usage of 238
devinst log 151
DHCP 133
dhcpsd 134
diagram of large environment 176
Differences between AIX versions 102
commands and operations 107
groups 106
Kerberos 4 supported 108
lock granularity 107
machines 102
multi-threaded nimesis daemon 108
networks 103
resources 103
smitty and web-based system manager 106
directory
/tftpboot 274
Diskless machines 5
defining resources 68
resources 100
DNS 140
dsh 267, 272
working collective 272

E
Easy startup menu 85
errors 146
common errors 146
LED 605 146
LED 608 146
LED 611 147
LED 613 150
Exclude_files resource 17
definition 17

exportfs 274

F
files
  .rhosts 177
  .toc file 252
  /etc/bootptab 135, 143, 156, 257
  /etc/dhcpsd.cnf 134, 135
  /etc/export 170
  /etc/exports 149
  /etc/firstboot 174
  /etc/hosts 140, 177, 223
  /etc/inetd.conf 133, 144, 155
  /etc/inittab 223
  /etc/netsvc.conf 140, 273
  /etc/niminfo 47, 48
  /etc/rc.sp 223
  /etc/syslog.conf 155
  /etc/tftpaccess.ctl 80, 147, 227
  /etc/xtab 149, 170
  /tftpboot/(ClientHostname).info file 49
  /tftpboot/(ClientName).info file 79
  /usr/sbin/bootptodhcp 135
  rc.boot 145
  rc.bos_inst 145
firmware 122
  change the bootlist 124
  enter the firmware 122
  get the newest 125
  ping test 123
  set up RIPL 123
fix_bundle resource 21
fix_query 186
fixes 180
frame supervisor card 205

G
gateway 53
generic network 132
Globally exported resources 104
Groups 22
  machine group 22
  resource groups 25
  strategy 22

H
Host responds 215

I
ifconfig 34
image.template 261
Image_data resource 18
  definition 18
inetd 143, 257
Initial Program Load 32
install_cw 210
Installation 5, 40
  force_push 85
  migration 40
  overwrite 40
  preservation 40
  pull 5, 85
  push 5, 85
installp 253
installp_bundle 20
  create a installp_bundle 182
  definition 20
  install a installp_bundle 183
IP datagram 171
ipforwarding 56
IPL-ROM 32, 38, 122
  creation 39
  function 32
  requirement 38
IP-Trace 162
  generate report 163
  start 162, 163
  stop 162, 163

K
Kerberos 226, 274
keyswitch 206
Korn shell 223

L
large environment 176
LED 140, 206
  codes 140
  meaning of the codes 140
log 150
  alt_disk_install 151
  boot 151
  bosinst 151
  command to view logs 152
  devinst 151
  lppchk 151

nimerr 151
niminst 151
script 151
view logs 151
Lpp_source resource 14
definition 15, 67
lppchk 186
lppchk log 151
lppsource
corruption of 250
maintenance 265
lslpp 185
lsnim
in reviewing config information 243
the info field 259
lssrc 257

M
MAC address 214
Machine group 22, 193
add members 23
asynchronous operations 25
definition 22
display status 25
include or exclude members 24
remove members 24
Machines 4
IPL-ROM required 38
Maintenance 91
rules 91, 92
Master
setting up 42
strategy 7
master 4
method errors 284
microcode 125, 275
get the newest 125
migrate 197
mirroring 223, 243
mkconfig 227
usage of 240
mkinstall 228
usage of 241
mknfsexp 269
mknimclient 227
usage of 230
mknimint 227
usage of 239
mknimmast 226
usage of 236
mknimres 227
usage of 232
mksysb 16
cloning with 265
definition 16
limit size 17
location of 217
multiple level of AIX 197
multiple networks 176

N
name resolution 10, 140, 273
avoid problems 140
mix up 140
NSORDER 140
order 140
setup different order 140
Net definition 7
netbooting
maximum limits 216
network boot 26
debugging 142
how to 26
problem 142
process 33
rs6k multiprocessor machine 28
rs6k uniprocessor machine 27
rspc machine 29
network device 57
Network interfaces 49
create additional interfaces 51
primary 49
Networks 8
define 8, 49
routing 9, 53
strategy 10
NFS 170, 284
exporting a directory 269
issues 41
unmounting a directory 266
NIM environment
rebuilding 273
NIM Master 267
removing 273
NIM objects
in an SP environment 207

NIM Resource
  creation 262
  deallocating 235
NIM wrappers 207
nim_script resource 14
  enhancements in AIX 4.3 105
nimclient command 96
  context 96
  flags 98
nimconfig command 46, 227
nimdef command 62, 66, 194
nimerr log 151
nimesis daemon 168
  multithreaded function 168
niminit command 95
  syntax 95
niminst log 151
NIS 140
Node conditioning
  on a PCI node 256
  on an MCA node 255
node supervisor 275
nodecond
  log file 222
  netbooting 222
Non-prompted install 18
NSORDER 273
NVRAM 32

O

Objects 4
  groups 22
  machines 4
  networks 8
  resources 10
ODM database 4, 174
  initialization 46
  machine objects 5
  network 9
  resources 11
Operations 5
  bos_inst 99
  change 98
  check 88, 98
  cust 88
  deallocate 99
  define 98
  diag 99
  dkls_init 99
  dtls_init 99
  fix_query 98
  lppchk 99
  lslpp 99
  maint 91
  maint_boot 99
  remove 98
  reset 99
  showlog 99
  showres 99
  sync_roots 99
  unconfig 99

P

pcp 267
performance 166
  global exports 170
  machine group 169
  MTU 171
  network option 171
  placement of resource server 167
  sb_max 173
  simultaneously client handling 168
  tcp_recvspace 173
  tcp_sendspace 173
  thewall 173
  udp_recvspace 173
  udp_sendspace 173
  xmt_que_size 173
Performance Toolbox 218
Perl 224
Perspectives
  description of 219
ping 122, 208, 257
Planning 40
Platform 5
  chrp 5
  rs6k 5
  rspc 5
Ports 47, 49
preload environment 174
PSSP
  minimum required filesets 218
pssp_script 223, 284
  location of 217
psspfb_script 223, 284

Q
quorum 242

R
rc.boot 34
Remote Initial Program Load 122
  Common Hardware Reference Platform 124
  Micro channel multiprocessor 124
  Micro channel uniprocessor 123
  Power PC 124
  Set up 122
REQUIRED_SIMAGES 249
resolv_conf resource 21
  definition 21
  history 104
  sample 21
resource
  mksysb 113
Resource allocation 69
  checks performed 70
Resource group 25, 193
  allocation 26
  definition 26
resource server 166
Resources 10
  minimum required 12, 69
  states 67
  strategy 21
root volume group 118
  backup 118
ROS 37
routing 53
RS-232 serial link 205
Rstate 227, 232, 243, 248

S
script 200
script log 151
Script resource 20
  definition 20
second network adapter 57
security
  globally exported resources 170
services_config 226
setup_CWS 226
setup_server 243
  description 224
  pre-requisite checks 226
  problems with 246
  running from spbootins 241
simages 15, 68, 248
single update fileset 253
software
  cust operation 184
  customizing 180
  installp_bundle 181
  maintenance 180
  queering 185
  removing 180
  update 180
software dependencies 197
Software management
  dataless and diskless clients 100
  list fixes installed 126
  list installed software 125
  updating network boot images 108, 125
software problems 185
SP Authentication Services
  initializing 210
sparse files 114
spbootins 216
  usage of 241
spchvgobj 216
spfbcheck 223
splstdata
  usage of 243
spmon
  description of 220
SPOT 12
  /usr 12, 68
  alloc_count attribute 275
  boot images 12, 33
  creating the debug version 276
  debug version 272
  definition 13
  enter_dbg attribute 276
  log file 246
  maintenance 265
  non-/usr 12, 68
  requisite failures 251
Stand-alone machines 5
static route 54
  defining 54
switch 205, 216
syspar_ctrl 215
System Data Repository 207
system environment customization 201

System Management Services
  diskette 122
  Enter 122
  graphic based 122
  text based 122
System Resource Controller 208

T
TCP/IP setup in AIX 4.1 201
tftp 284
topology services 206
traceroute 56
Trivial File Transfer Protocol 79
Troubleshooting 140
  boot image transfer to client 143
  communication between client and server 142
  LED display 140
  running the boot image 146
  viewing NIM logs 150
tty 161
  interface a RISC as tty 161

U
unallnimres
  usage of 235
upgrade os 175

V
Virtual Shared Disks 219

IBM Redbooks evaluation

NIM: From A to Z in AIX 4.3
SG24-5524-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction __________

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

____

____

____

____

What other Redbooks would you like to see published?

____

____

____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

____

____

____

____

SG24-5524-00
Printed in the U.S.A.

NIM: From A to Z in AIX 4.3

SG24-5524-00