

# NIM from A to Z in AIX 5L

Configuring NIM illustrated

Sample best practices

NIM case scenarios  
included



Dino Quintero  
Jean-Michel Berail  
Hassan Elsetohy  
Chris Gibson  
Markus Lang  
Sven Meissner  
Pablo Pereira  
Bjorn Roden  
Kelvin Tan





International Technical Support Organization

**NIM from A to Z in AIX 5L**

May 2006

**Note:** Before using this information and the product it supports, read the information in “Notices” on page ix.

**First Edition (May 2006)**

This edition applies to AIX 5L V5.3 Technology Level 5, Cluster Systems Management (CSM) V1.5.1, and IBM Director V5.1.

This document created or updated on March 13, 2007.

© Copyright International Business Machines Corporation 2006. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	ix
Trademarks .....	x
<b>Preface</b> .....	xi
The team that wrote this redbook .....	xi
Become a published author .....	xv
Comments welcome .....	xv
<b>Chapter 1. Introduction</b> .....	1
1.1 Why Network Installation Manager (NIM)? .....	2
1.1.1 Benefits and features of NIM .....	2
1.2 Redbook overview .....	4
1.2.1 The content of this book .....	4
1.2.2 Test environment .....	4
<b>Chapter 2. Network Installation Manager (NIM) definitions and concepts</b> .....	7
2.1 Basic NIM configuration .....	7
2.1.1 NIM overview .....	8
2.1.2 How does a NIM environment work? .....	16
2.1.3 Before working with a NIM environment .....	25
2.1.4 NIM master selection .....	26
2.1.5 Easy steps to start with NIM .....	28
2.1.6 Easy steps to maintain a NIM environment .....	36
2.2 What is new in NIM for AIX 5L .....	38
2.2.1 NIM improvements with AIX 5L V5.1 .....	39
2.2.2 NIM improvements with AIX 5L V5.2 .....	39
2.2.3 NIM improvements with AIX 5L V5.3 .....	42
2.3 Planning for your NIM environment .....	44
2.3.1 Choosing your NIM master .....	45
2.3.2 Defining a consistent naming convention .....	46
2.3.3 Sizing your NIM master .....	47
2.3.4 Designing for availability .....	48
<b>Chapter 3. Basic configuration of a Network Installation Manager (NIM) environment</b> .....	49
3.1 Setting up a basic NIM environment .....	50
3.1.1 Planning the NIM environment .....	50
3.1.2 Setting up the NIM environment .....	56
3.2 Web-Based System Manager (WebSM) .....	96

3.2.1	Configuring a NIM master through WebSM	96
3.2.2	Configuring NIM clients using Web-Based System Management	105
3.2.3	Installing a NIM client using Web-Based System Management	109
3.3	EZ NIM	114
3.3.1	Configuring a NIM master using <code>nim_master_setup</code>	114
3.3.2	Configuring NIM clients using <code>nim_clients_setup</code>	119
<b>Chapter 4. Network Installation Manager (NIM) scenarios</b>		<b>123</b>
4.1	High Availability NIM (HA NIM)	124
4.1.1	Configuring the alternate (backup) NIM master	125
4.1.2	Initialize the backup server as the alternate NIM master server	128
4.1.3	Perform NIM synchronization	130
4.1.4	NIM master takeover	134
4.1.5	NIM master fallback	137
4.2	Using the <code>OS_install</code> command	138
4.2.1	<code>OS_install</code> basics	139
4.2.2	<code>OS_install</code> resource definition	140
4.2.3	<code>OS_install</code> controlling host definition	141
4.2.4	<code>OS_install</code> client definition	141
4.2.5	<code>OS_install</code> resource allocation and installation	142
4.2.6	<code>OS_install</code> XML repository	144
4.3	NIM environment with additional resource servers	145
4.3.1	Configuring a resource server	145
4.3.2	Performing a BOS installation	151
4.4	Using NIM to perform AIX migrations of the NIM master and clients	153
4.4.1	Migrating a NIM master to AIX 5L V5.3	153
4.4.2	Migrating a NIM client to AIX 5L V5.3	164
4.4.3	NIM client AIX migration example	168
4.4.4	Debugging a NIM AIX migration	198
4.5	NIM <code>mksysb</code> migration and <code>nim_move_up</code> POWER5 tools	206
4.5.1	<code>mksysb</code> migration	206
4.5.2	Performing a <code>mksysb</code> migration with NIM	209
4.5.3	Migrating a NIM client to a POWER5 logical partition using <code>nim_move_up</code>	217
4.5.4	Example <code>nim_move_up</code> operation	221
4.5.5	POWER5 to POWER5 <code>nim_move_up</code>	261
4.6	NIM alternate disk migration	261
4.6.1	Local disk caching versus NFS	263
4.6.2	The 12 phases of <code>nimadm</code> migration	264
4.6.3	<code>nimadm</code> migration example	266
4.6.4	Cleaning up after a failed <code>nimadm</code> operation	283
4.7	Network Installation Manager (NIM) and Linux distributions	284
4.7.1	Configuring DHCP	285

4.7.2 Intel PXE configuration . . . . .	297
4.8 Common OS image management (COSI). . . . .	299
4.8.1 Thin server, diskless, dataless, and common OS image . . . . .	300
4.8.2 Creating a common OS image (COSI) . . . . .	302
4.8.3 Creating a thin server . . . . .	304
4.8.4 Remove a thin server . . . . .	307
4.9 Using HACMP with HANIM . . . . .	308
4.10 NIM and Service Update Management Assistant . . . . .	321
4.10.1 How to use SUMA. . . . .	324
4.10.2 Settings for SUMA. . . . .	325
4.10.3 SUMA tasks . . . . .	328
4.10.4 Maintenance models . . . . .	334
4.10.5 Removing duplicate filesets from a NIM lpp_source directory . . . . .	340
4.10.6 Check if a system has all available updates installed . . . . .	342
4.10.7 IBM AIX 5L service strategy . . . . .	349
4.10.8 IBM support fix central Web site . . . . .	352
4.11 Backing up clients with NIM . . . . .	353
4.11.1 Space requirements . . . . .	355
4.11.2 NFS export during mkysyb image file creation . . . . .	356
4.11.3 Exclude file for mkysyb image creation. . . . .	357
4.12 How to create bundles, BFF, and RPM packages . . . . .	359
4.12.1 BFF (native installp packages) . . . . .	360
4.12.2 RPM . . . . .	369
4.13 NIM and Virtual I/O (VIO) . . . . .	377
4.13.1 Virtual SCSI. . . . .	378
4.13.2 Virtual shared ethernet . . . . .	378
4.13.3 Consideration when using a VIO server . . . . .	379
4.13.4 Setting up the VIO server and configuring the NIM master in the VIO client . . . . .	379
4.14 Backup and restore of the VIO Server using NIM . . . . .	391
4.14.1 Backup of VIO server . . . . .	391
4.14.2 Restoring the VIO server. . . . .	393
4.15 Using RSCT PeerDomain with HANIM . . . . .	397
<b>Chapter 5. Network Installation Manager (NIM) best practices . . . . .</b>	<b>417</b>
5.1 Updating the NIM environment (lpp_source, SPOT) . . . . .	418
5.2 Secondary adapter . . . . .	423
5.2.1 Secondary adapter support. . . . .	423
5.2.2 Working with secondary adapter file rules . . . . .	424
5.2.3 Secondary adapter files . . . . .	425
5.2.4 Configuring a secondary adapter . . . . .	429
5.3 NIMSH, OpenSSL and firewall considerations . . . . .	433
5.3.1 Authentication process . . . . .	434

5.3.2	Setting up NIMSH . . . . .	435
5.3.3	Verifying NIMSH startup . . . . .	436
5.3.4	Enabling cryptographic authentication . . . . .	436
5.3.5	Enabling a secondary port for NIMSH communication . . . . .	443
5.3.6	Disabling push operations using NIMSH. . . . .	444
5.3.7	Disabling pull operations . . . . .	445
5.3.8	NIM communication within a firewall environment . . . . .	445
5.3.9	NFS reserved ports . . . . .	450
5.3.10	Firewall considerations . . . . .	459
5.4	“Cloning” NIM clients using mksysb . . . . .	460
5.4.1	Using a mksysb image to clone NIM Client. . . . .	460
5.4.2	Prerequisites . . . . .	461
5.4.3	mksysb image installation on a NIM Client . . . . .	462
5.4.4	Mksysb resource creation . . . . .	462
5.4.5	Correlating mksysb and SPOT resources . . . . .	464
5.4.6	SPOT creation from an existing mksysb . . . . .	465
5.4.7	Mksysb installation . . . . .	466
5.4.8	Monitoring the mksysb installation . . . . .	468
5.5	Using NIM to migrate systems to new hardware . . . . .	470
5.5.1	Migrating the image of an LPAR to another system using NIM . . .	470
5.5.2	Relocating the image of an LPAR to a remote location . . . . .	473
5.5.3	Migrating to new hardware utilizing custom bosint.data and image.data files . . . . .	474
5.6	Using SAN zoning for “cloning” LPARs . . . . .	485
5.7	System maintenance for NIM clients . . . . .	488
5.7.1	Software maintenance approach . . . . .	488
5.7.2	Booting in maint_boot and diag modes. . . . .	506
5.8	Automatic scripts . . . . .	514
5.8.1	Install time customization . . . . .	514
5.8.2	Customization for BOS installations . . . . .	514
5.9	Implementing a standard operating environment for AIX 5L V5.3 systems with NIM . . . . .	529
5.9.1	Do you need an SOE for your AIX systems? . . . . .	529
5.9.2	What is an SOE for AIX systems? . . . . .	529
5.10	How to backup and re-install a NIM master . . . . .	547
5.10.1	Scenario 1: Backup and restore a NIM master on the same machine using a tape . . . . .	547
5.10.2	Scenario 2: Backup and restore a NIM master on a different machine using a tape . . . . .	548
5.10.3	Scenario 3: Backup and restore a NIM master using another NIM master. . . . .	551
5.10.4	How to change the NIM master hostname . . . . .	554
5.10.5	How to change the NIM master IP address . . . . .	558



<b>Chapter 6. Network Installation Manager (NIM) and CSM</b> .....	569
6.1 NIM and CSM .....	570
6.1.1 General overview of CSM .....	570
6.1.2 The management server (MS) .....	571
6.1.3 NIM and CSM .....	572
<b>Chapter 7. Basic NIM problem determination and tuning</b> .....	575
7.1 Troubleshooting methodologies .....	576
7.1.1 Troubleshooting a network boot problem .....	576
7.1.2 Debugging NIM BOS installations .....	586
7.1.3 Re-creating the SPOT from an existing directory .....	586
7.2 Leds .....	587
7.2.1 LED values during NIM operations .....	587
7.2.2 Hang on LED 611 -- NFS mount problem .....	590
7.2.3 NIM installation hangs on LED 613 .....	590
7.3 NFS .....	591
7.3.1 Exporting NIM resources globally .....	591
7.4 Booting .....	594
7.4.1 Booting .....	594
7.5 Log files .....	597
7.5.1 NIM related files .....	597
7.5.2 RSH related files .....	599
7.5.3 BOOTP related files .....	600
7.5.4 TFTP related files .....	601
7.5.5 NFS related files .....	602
7.6 Case study: NIM server performance .....	603
7.6.1 Setting up the environment .....	603
7.6.2 Monitoring NIM master using topas .....	605
7.6.3 Upgrading NIM environment to Gbit Ethernet .....	609
7.6.4 Upgrading the disk storage .....	611
7.6.5 Real workload with spread file system .....	619
7.6.6 Summary .....	621
7.6.7 Multi-threaded option for the NIM nimesis daemon .....	622
<b>Appendix A. General LPAR sizing and creation with mig2p5 tools</b> . . . .	625
The mig2p5 tools .....	626
/usr/lpp/bos.sysmgt/nim/methods/getSrcUtil .....	627
/usr/lpp/bos.sysmgt/nim/methods/getSrcCfg .....	627
/usr/lpp/bos.sysmgt/nim/methods/getTgtRsrc .....	627
/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg .....	627
/usr/lpp/bos.sysmgt/nim/methods/createTgtLPAR .....	627
Standalone mig2p5 tool .....	628
Get the source LPARs hardware configuration .....	629

Get the source LPARs CPU and memory utilization . . . . .	630
Flags for getSrcCfg command . . . . .	630
Flags for the getSrcUtil command . . . . .	631
Get the target POWER5 machine resources via its HMC . . . . .	631
Flags for the getTgtRsrc command . . . . .	632
Flags for the genTgtCfg command . . . . .	633
Generate the target LPARs profile information . . . . .	634
Generated Target LPAR configuration information . . . . .	635
Create the target LPAR definition on the POWER5 machine . . . . .	636
Flags for the createTgtLPAR command . . . . .	638
<b>Appendix B. Automatic provisioning NIM and Tivoli Provisioning Manager (TPM) . . . . .</b>	<b>641</b>
TPM Overview . . . . .	642
Why to use TPM? . . . . .	642
Architecture picture . . . . .	642
TPM terminology . . . . .	643
Workflow to define a NIM client. . . . .	646
Automated provisioning scenario . . . . .	648
Ways to automate (interfaces, cmd line, web services) . . . . .	650
<b>Appendix C. NIM and System i OS installation for an AIX 5L LPAR . . . . .</b>	<b>651</b>
IBM System i operating system installation for an AIX 5L LPAR . . . . .	652
<b>Appendix D. Additional material . . . . .</b>	<b>655</b>
Locating the Web material . . . . .	655
Using the Web material . . . . .	655
System requirements for downloading the Web material . . . . .	656
How to use the Web material . . . . .	656
<b>Abbreviations and acronyms . . . . .</b>	<b>657</b>
<b>Related publications . . . . .</b>	<b>661</b>
IBM Redbooks . . . . .	661
Other publications . . . . .	661
Online resources . . . . .	661
How to get IBM Redbooks . . . . .	662
Help from IBM . . . . .	662
<b>Index . . . . .</b>	<b>663</b>

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:  
*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

## Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo)  ™	DB2®	Redbooks™
Eserver®	HACMP™	Requisite®
^®	IBM®	RS/6000®
eServer™	Micro-Partitioning™	System i™
pSeries®	POWER™	System p™
xSeries®	POWER Hypervisor™	System p5™
AIX 5L™	POWER3™	Tivoli®
AIX®	POWER4™	Virtualization Engine™
BladeCenter®	POWER5™	WebSphere®

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates.

i386, Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

This redbook is aimed at the AIX® 5L™ technical community in particular at systems administrators who are well versed in the concepts and terminology of the AIX operating system but would like to know why they should consider implementing a Network Installation Manager (NIM) environment in their data centre.

The concept of Cluster (at least, two IBM System p machines connected through a network) exists now for several years and brings the need of how to install, maintain, update, backup up the different participants of the cluster. NIM does these tasks and becomes now a crucial feature of the AIX 5L operating system and an important part of any System p server farm. For sites that are looking for the most efficient way of installing and/or upgrading their AIX 5L servers, NIM provides an excellent method for performing these activities and more.

This redbook does not aim to replace IBM® technical AIX 5L manuals. This redbook is a compliment of the manuals, and it can be used as a desk reference guide.

**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 IBM System p machines and LPARs with the same or different images at the same time.**

## The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the Product and Solutions Support Center PSSC in Montpellier, France.

**Dino Quintero** is a Senior Certified Consulting IT Specialist at ITSO in Poughkeepsie, New York. Before joining ITSO, he worked as a Performance Analyst for the Systems and Technology Group and as a Disaster Recovery Architect for IBM Global Services. His areas of expertise include disaster recovery and pSeries clustering solutions. He is certified on pSeries system administration and pSeries clustering technologies. He is also an IBM Senior Certified Professional on pSeries technologies. Currently, he leads teams delivering IBM Redbook solutions on pSeries clustering technologies and delivering technical workshops worldwide.

**Jean-Michel Berail** is an AIX Specialist at the Product and Solution Support Center (PSSC) in IBM Montpellier, France. He is working for IBM for 28 years. He has 12 years of experience in AIX, mainly regarding the installation, backup and migration of System p™ clusters. He developed several cluster material for Education and Workshops on Parallel System Support Programs (PSSP), Cluster System Management (CSM) and Network Installation Management (NIM). He teaches CSM and NIM courses in IBM Education Centers or customer areas. He has written or reviewed several redbooks whose subjects are PSSP, CSM and NIM. He is also a Professor of Computer Science at a Montpellier University. Jean-Michel is a graduate of the E.N.S.E.I.R.B. (Bordeaux, France) engineering school.

**Hassan Elsetohy** works for IBM in Integrated Technology Delivery dept. in Sydney, Australia. He joined IBM in 1994 as a Systems Engineer after attaining his Bachelor of Engineering degree. He also holds a coarse-work Masters degree in VLSI design. Hassan is a multi-certified IBM Professional. He is an IBM Certified IT Architect, an IBM Certified Specialist, and an IBM Certified Advanced Technical Expert. He has also recently attained his SNIA and Linux® LPI certifications. Hassan has been performing the Lead IT Architect role, as well as the Method Exponent role in major engagements. In most of the engagements, he is also performing the SME role as well as undertaking the development of detailed end-to-end designs in his areas of specialty, which include IT Optimization, AIX, System p, SAN Storage, Systems Performance & Capacity Management, and Disaster Recovery Planning. Hassan has a sound experience in utilizing field-proven methodologies in undertaking his engagements. Methodologies include: The IBM GS Method, The IBT Consultancy Methodology, and the Zodiac/ALIGN Server consolidation methodologies.

**Chris Gibson** is a Senior AIX Systems Administrator for Insurance Australia Group (IAG), Australasia's leading general insurance provider. Chris has been working with UNIX® systems for more than 12 years. Since 1999 he has been involved with RS/6000®, SP and pSeries systems running AIX. His areas of expertise include AIX 5L and NIM. He has also worked with HACMP™, CSM and PSSP. He is an IBM Certified Advanced Technical Expert - pSeries and AIX 5L. This is his first redbook. He would like to thank his wife Jacqui and his daughter Jade for all their support during the residency.

**Markus Lang** is an IT-specialist at an IBM premier business partner in Sweden, Pulsen Systems. He has worked with AIX/UNIX for about 10 years and is a RedHat Certified Engineer, AIX Advanced Technical Expert and holds various other IBM/AIX certifications. Markus has been working with SP, HACMP, TSM and WebSphere® as well as AIX/DB performance tuning, disaster recovery, kernel dump analysis, benchmarking and programming.

**Sven Meissner** is an IT specialist at Bayer Business Services GmbH in Germany. He has more than 10 years of experience in AIX. His areas of expertise include planning, installation and managing of AIX environments including HMCs, CSM and HACMP. Sven is an IBM Certified Technical Expert - pSeries Administration and Support for AIX 5L v 5.2.

**Pablo Pereira** is an Advisory IT Specialist working at the Service Delivery Center (SDC) in IBM Uruguay. He has been working for IBM since 1997. He has been working with RS/6000, SP, pSeries and storage servers since then. His areas of expertise include AIX, PSSP, HACMP, LPAR, NIM and TCP/IP. He is an IBM Certified Advanced Technical Expert - pSeries and AIX 5L. He has been teaching AIX 5L Administration, AIX 5L Performance, HACMP Administration, LPAR and Virtualization in several countries including Uruguay, Paraguay, Brasil and Argentina. He is also co-author of the previous redbook: An Introduction to the New IBM eServer™ pSeries High Performance Switch, SG24-6978-00.

**Bjorn Roden** is an AIX Expert from Sweden. He has experience with AIX since version 2.2.1 and 3.1, starting from his work at IBM in the late 1980s, early 1990s, and today Pulsen Systems. He has worked with most parts of AIX and SP2/PSSP, HACMP, ADSM/TSM, LPAR, and various Open Source Software. He has been continuously IBM AIX certified since 1994, accumulating more than 30 certificates, including Advanced Technical Expert, HACMP, Enterprise and Virtualization Technical Support, and Tivoli® Storage Manager, to mention the most recent. He is also a co-author of four previous redbooks: RS/6000 SP System Performance Tuning Update, SG24-5340-01, AIX 5L Performance Tools Handbook, SG24-6039-00, Tivoli Storage Manager Version 5.1 Technical Guide, SG24-6554-00, and Linux Applications on pSeries, SG24-6033-01. And reviewer of the IBM Certification Study Guide - pSeries HACMP for AIX, SG24-6187-00 and the IBM System p 24x7 and RAS Redbook, SG24-7196-00.

**Kelvin Tan** is an Advisory I/T Specialist working in ITS, IBM Singapore. He has been working in IBM since year 2000. He has been supporting on AIX, RS/6000-SP, HACMP, Micro-Partitioning™ and Virtualization. He is an IBM @server® Certified Advanced Technical Expert - pSeries and AIX 5L. He holds a Bachelor of Computing Engineering from Nanyang Technological University. He is also co-author of redbook on Transition from PSSP to Cluster Systems Management (CSM) redbook, SG24-6967-00.





Team members starting from the left: Kelvin Tan, Dino Quintero (project leader), Pablo Pereira, Bjorn Roden, Chris Gibson, Jean-Michel Berail.

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

Octavian Lascu  
International Technical Support Organization, Poughkeepsie Center

Paul Finley, Julie Craft, Eddie Reyes, Roji John  
IBM Austin

Brian Crosswell, Ning-Wu Wang, Paul Swaitocha, Linda Mellor, Margaret Moran  
IBM Poughkeepsie

Guillermo Diez  
IBM Uruguay

Sebastien Llaurency, Olivier Merlet  
IBM France

Joefon Jann  
IBM T. J. Watson Research Center



## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review redbook form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbook@us.ibm.com](mailto:redbook@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400





# Introduction

This chapter outlines various reasons why you would use Network Installation Manager (NIM) in your AIX 5L environment. The information in this section is aimed at AIX 5L systems administrators who are well versed in the concepts and terminology of the AIX operating system but would like to know why they should consider implementing a NIM environment in their data centre.

The concept of a cluster (i.e. at least, two IBM System p machines connected through a network) has existed for several years now and brings with it the need for installing, maintaining, updating, and backing up the different participants of the cluster. NIM performs all of these tasks and is now a crucial feature of the AIX 5L operating system, and an important part of any IBM System p server farm. For sites that are looking for the most efficient way of installing and/or upgrading their AIX 5L servers, NIM provides an excellent method for performing these activities and more.

## 1.1 Why Network Installation Manager (NIM)?

So, why use NIM? Before describing the basic concepts of NIM, first take a look at what factors may encourage you to consider NIM for your site. For example, what is NIM and what can it do for you?

NIM provides an efficient and easy method of performing various installation and software maintenance tasks over several network types (for example, Ethernet). When you have more than two IBM System p servers in your AIX 5L environment, you are working with a Cluster and you are most likely looking for a method of installing and upgrading your IBM System p systems without the need for multiple CD-ROMs and tapes. When you need a way of managing AIX 5L software installations and upgrades remotely without actual physical access to the hardware. Or when you need a standard method of maintaining several AIX 5L operating systems images across many IBM System p systems, NIM is the tool of choice.

Another place to use NIM could be a large IBM System p machine, such as a p590 or p595 partitioned in several LPARs, using dedicated or virtual I/O resources: even though a CD/DVD drive may be available to the CEC, installing and maintaining several partitions by passing along the CD/DVD drive from one LPAR to another may be time consuming and prone to human error (software selection for example). By using one LPAR as a NIM server and installing the over a virtual network, you save time and is less likely to install wrong or incomplete software.

### 1.1.1 Benefits and features of NIM

This section describes the benefits and features of NIM:

- ▶ It's free! The NIM package is included in the AIX 5L CDs. There is no additional cost required for this package.
- ▶ NIM allows system administrators to install, upgrade, backup and maintain AIX 5L systems remotely. No need for CDs or tapes. Physical access to the systems is no longer required except maybe during first time installation. In this circumstances, Cluster Systems Management (CSM) can help us in conjunction with NIM for the installation of remote systems. See "NIM and CSM" on page 570.
- ▶ Multiple machines can be installed or maintained at once. For example, you may choose to install a specific group of systems at once while applying the latest fixes to another specific group of machines.
- ▶ Multiple AIX 5L versions, maintenance levels (ML) or technology levels (TL) can be used in the same NIM environment. This is called coexistence.

- ▶ Disaster recovery - Local systems can be recovered quickly with NIM. In many cases, a system can be recovered in as little as 15-30 minutes. At disaster recovery sites, NIM can be deployed to quickly recover system images to disaster recovery systems without the need for AIX CDs or mksysb backup tapes.
- ▶ Systems can be installed with standard and consistent AIX 5L operating system images. By defining a standard operating environment (SOE) for your AIX 5L system images, systems can be customized to meet your specific needs and maintain a level of consistency across all AIX 5L servers. This makes administering multiple servers much easier as the administrator deals with a similar environment, for example same LPPs, tools, man pages, file system/logical volume naming, utilities, etc., across all systems. Using this SOE image also means new system builds can be accomplished quickly with very little customization or manual labor required. In some cases, new builds can be completed in under 30 minutes.
- ▶ Recovering a system backup (mksysb) from tape can be difficult when booting from CD media. You must boot off media that is at the same level as the backup, for example AIX 5L base install CDs. If the CD media is not at an equal to or higher technology level than the level of the backup image being installed from tape, unpredictable results might occur. As a result, every time a new technology level is released and applied to your machine, it is also necessary to immediately obtain the latest install media for that level of AIX 5L at the corresponding technology level. However, if you use NIM to install, backup and recover your systems there is no need to rely on CDs or tapes to recover your system. Machines can be recovered easily using a network boot image at the correct level for that system. It is possible to create bootable CD media from the source system, but this can be a time consuming task involving CD/DVD media and may require additional manual tasks for handling the media, for example sending it off site. Avoid this challenge and take advantage of the flexibility of NIM.
- ▶ “Cloning” systems is another excellent feature of NIM which can assist you in moving from one server to another or even from one site to another site. By taking a backup of your operating system image you can use NIM to reinstall this image onto another machine, potentially with different hardware, for example from a IBM System p p650 to a IBM System p p595, which may even reside at a different physical location. You may want to do this for several reasons, for example moving to new hardware, cloning a system to test an upgrade or moving a server to a new data centre. There are also several new cloning features with NIM on AIX 5L V5.3 and POWER5™ that can assist you even further when migrating from earlier versions of AIX (4.3.3/AIX 5L V5.1) to AIX 5L V5.3 and POWER5 hardware.

## 1.2 Redbook overview

The following section describes in brief the contents of this redbook and the test environment we have used for the scenarios and examples throughout the book.

### 1.2.1 The content of this book

Now you must be convinced that NIM is a powerful tool to install and maintain machines in a Cluster environment. Thus, before going in-depth with NIM, the basic concepts and configuration must be described. Chapter 2, “Network Installation Manager (NIM) definitions and concepts” on page 7 and Chapter 3, “Basic configuration of a Network Installation Manager (NIM) environment” on page 49 give the related details. Chapter 4, “Network Installation Manager (NIM) scenarios” on page 123 and Chapter 5, “Network Installation Manager (NIM) best practices” on page 417 give information on important improvements introduced with AIX 5L V5.3 using simple scenarios or guidelines for best practices.

As previously described, NIM is designed to install and maintain machines in a Cluster. However, in addition to NIM, to fully automate, control and manage your environments, a Single Point Of Control (SPOC) comes in handy, and this is Cluster Systems Management (CSM). CSM can be associated with NIM to perform tasks that NIM can't handle by itself, such as hardware control. Chapter 6, “Network Installation Manager (NIM) and CSM” on page 569 describes the relationship between NIM and CSM and gives the basic knowledge you can use to configure one of the cluster machines as CSM Management Server and NIM master.

When using a NIM environment, issues might arise, so, the immediate question is: how to solve the problem? Chapter 7, “Basic NIM problem determination and tuning” on page 575, presents a brief troubleshooting methodology and gives tips on the most usual problem areas. Also, the appendices list general guidelines on how to backup/restore a NIM master, and information about NIM usage in other environments, such as Automatic provisioning (Tivoli Provisioning manager) and System i™.

### 1.2.2 Test environment

One of the peculiarities of a Redbook is the collection of practical cases or scenarios. To run these scenarios, the ITSO team has used different IBM System p® machines as shown in Figure 1-1 on page 6.

Our environment consisted of:

- ▶ A partitionable IBM System p p670 (POWER4) with its own HMC used for testing basic NIM operations such as:
  - Backup and restore of the NIM master
  - High availability NIM
  - AIX migrations
  - Secure NIM
  - and so forth
- ▶ A partitionable IBM System p p521 and IBM System p p520 with their respective HMCs used for testing the VIO server features:
  - Micro-partitioning and Virtual I/O resources
  - Backup and restore of the VIO server
- ▶ An IBM System p p615 used as a NIM master and server.

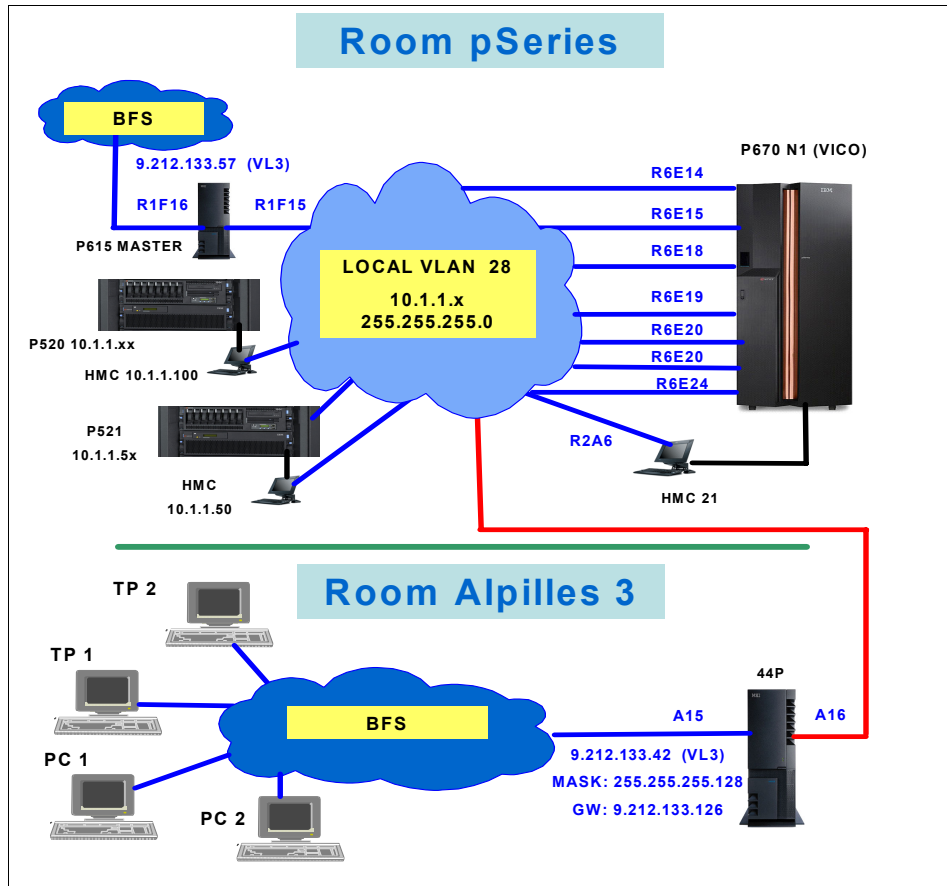


Figure 1-1 ITSO test environment diagram





# Network Installation Manager (NIM) definitions and concepts

In this chapter, we present NIM basic terminology and definitions, followed by the basic concepts of a Network Installation Manager (NIM) environment.

Although these topics are also covered in the manual AIX 5L V5.3 Installation and Migration, SC23-4887-03, we consider that understanding the NIM terminology and concepts is the key in deploying and using a NIM environment to its full potential, thus we provide detailed explanations of these topics.

## 2.1 Basic NIM configuration

In the following sections, we describe the basic NIM configuration. After a general overview of what is behind NIM, we give definitions of the key words used in a NIM environment, and we finish by presenting a basic setup of a NIM environment.

## What is a NIM environment?

NIM is a client/server application which uses object oriented technology. It provides an environment to install and manage AIX filesets (base operating system, maintenance levels, individual fixes etc.) on machines over the network. Thus, we need to keep in mind the general picture of what NIM needs to work properly, and how NIM uses its different components.

This section gives answers to the first question (what is NIM?) while the next section answers the second question (how does it work?).

### 2.1.1 NIM overview

A basic NIM environment consists of several IBM System p machines (at least two) connected via a TCP/IP network. A physical network can be shared by several NIM environments, therefore the machines attached to the same physical network can be part of several NIM environments.

**Note:** A NIM environment is a logical group of machines and multiple NIM environments can share the same TCP/IP network, however, one machine can be part of only one NIM environment a time, and there can be only one active NIM master per environment.

The machines (or clients) can be standalone systems, LPARs, diskless, and dataless clients (also known as thin clients). The machines can be controlled by one or more Hardware Service Consoles (better known as Hardware Management Consoles - HMC) or be stand-alone machines without a HMC.

The first task is to determine which machine or LPAR will be the NIM master. This machine will need some disk space, a good network connection and spare CPU cycles, more details later in this chapter. The other machines or LPARs are then considered potential NIM clients.

The next question is, which machines will serve the NIM resources? In complex environments (lots of clients, complex network), “stuffing” all resources on the same machine (NIM master) may result in the NIM master becoming a bottleneck, or, even worse, a security exposure. This can be avoided by distributing NIM resources on several other standalone machines or LPARs (see “Resources class” on page 15 for details).

Finally, there are three basic machine roles in a NIM environment: the master, the client (NIM “machine”), and the resource server. The resource server can be either the NIM master or a standalone machine (a system that has its own local copy of the AIX OS).

**Note:** Whereas the master and client (machines) can be explicitly defined via SMIT menus, the resource server is actually tied to the concept of NIM resource, and it cannot be explicitly defined as an entity through SMIT.

Figure 2-1 on page 9 gives a general picture of two NIM environments sharing the same physical network. We can easily recognize the three machine categories (master, client and resource server) and two NIM environments. Also, two machines are not yet part of a NIM environment.

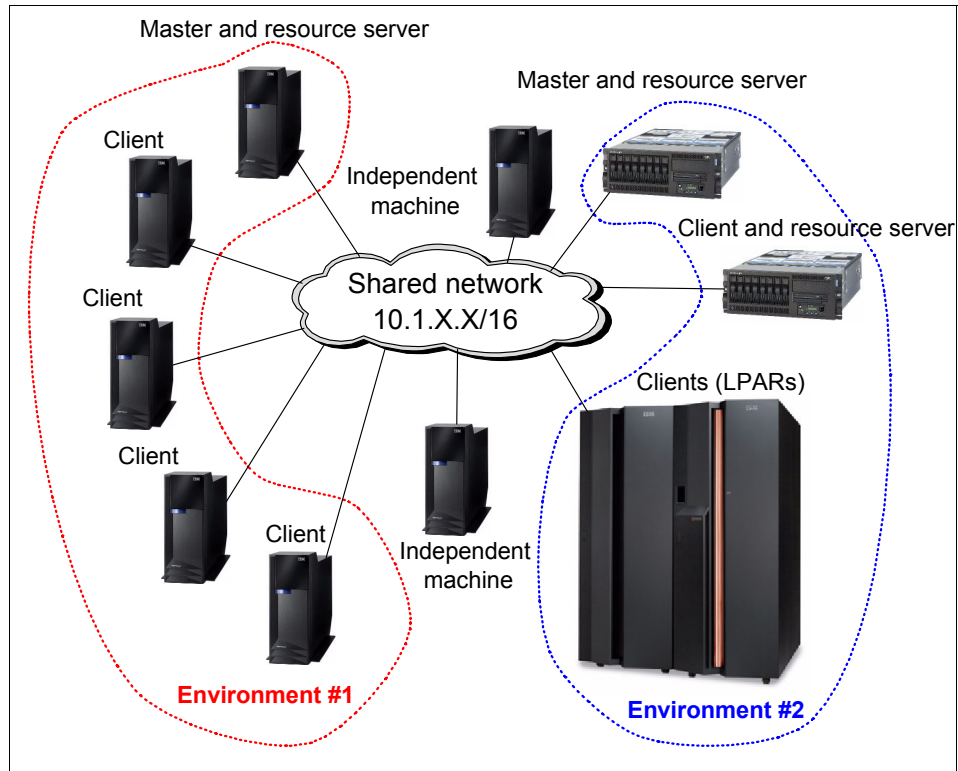


Figure 2-1 NIM environment overview

The starting point is therefore to decide which machine is the master. General information to help you choosing this system are explained in 2.1.4, “NIM master selection” on page 26.

## NIM definitions

The following section provides NIM definitions to help us understand naming convention throughout the rest of the redbook:

### **Master**

This is the machine where you set up and maintain your NIM environment. You can also initiate installations from here (PUSH mode). It is the key piece of the NIM environment. Refer to section “NIM master selection” on page 26 which describes how to choose a machine to become the NIM master.

### **Client**

The NIM client can be the target for NIM master initiated operations: installation, updates, and so forth (push mode). Also, a client can initiate its own installation or update (pull mode). A NIM client automatically becomes a resource server when it holds NIM resources.

### **Resource server**

Any machine, the master or a standalone client, can be configured by the master as a server for a particular software resource. In most environments, the master is also resource server.

**Note:** In case others machines are already reporting to the master and they are installed (AIX), you can choose one of them to act as a resource server, relieving the NIM master of the heavy I/O load (disk and network). In this example, the NIM master is only used to run administrative tasks.

**Tip:** If your environment has lots of nodes or consists of a complex network environment, you may want to configure standalone clients to act as resource servers for performance and security reasons.

### **Push and pull modes**

The push mode operation is initiated from the master whereas the pull mode operation is initiated from the client. The very first time a client is installed, only the pull mode can be used. For the push mode to be successful, the client must have a minimum AIX image already installed and TCP/IP configured. To use the pull mode, you need access to the clients' SMS menu. For this you either need a console attached to the machine, or, for HMC managed systems, you need access to the HMC.

Figure 2-2 on page 11 shows the push and pull modes for NIM operations.

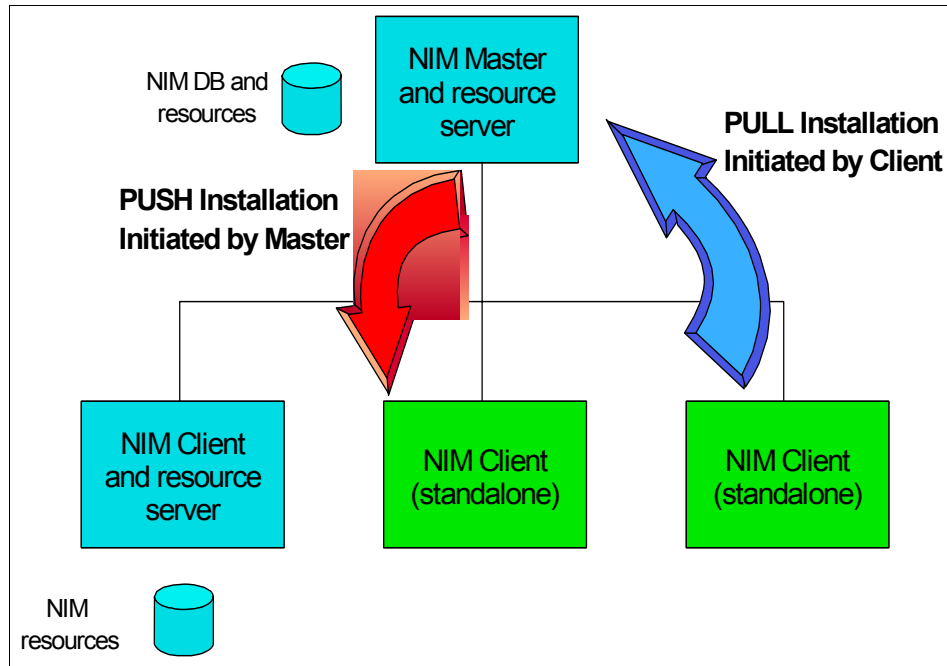


Figure 2-2 Push and pull modes

### **NIM database**

The NIM database is stored in the AIX Object Data Management (ODM) repository on the NIM master and is divided into four classes: machines, networks, resources and groups (see Table 2-1).

Table 2-1 The four NIM classes and their main used types

<b>Machines</b>	<b>Network</b>	<b>Resources</b>	<b>Groups</b>
master	ent	lpp_source	mac_group
standalone	tok	spot	res_group
diskless	fddi	mksysb	
dataless	atm	bosinst_data	
alternate_master	generic	script	
		image_data	
		installp_bundle	
		.....	

To illustrate the contents of this NIM database, we use the `lsnim` command to extract this content (on a NIM master that is already set up and working), as shown in Example 2-1:

*Example 2-1 Output of the `lsnim` command*

---

```
{nimmast}:/ # lsnim
master          machines      master
boot            resources    boot
nim_script      resources    nim_script
NET_EN1         networks     ent
LPP_53_ML4     resources    lpp_source
SPOT_53_ML4    resources    spot
BID_NP_HD0     resources    bosinst_data
LPAR1           machines     standalone
LPAR2           machines     standalone
LPAR3           machines     standalone
LPAR4           machines     standalone
LPAR5           machines     standalone
LPAR6           machines     standalone
LPAR123456     groups       mac_group
nimgrp          groups       mac_group
spotaix5104    resources    spot
lpp_sourceaix5204 resources    lpp_source
spotaix5204    resources    spot
AllDevicesKernels resources    installp_bundle
```

---

Example 2-2 and Example 2-3 on page 13 provide details on each of the four classes represented in the NIM database. We start with the Machines class followed by the Networks class and Resources class, and ending with the Groups class.

### ***Machines class***

Example 2-2 gives an example of the output of the `lsnim -c machines` command.

*Example 2-2 Example of `lsnim -c machines` command*

---

```
{nimmast}:/ # lsnim -c machines
master          machines      master
LPAR1           machines     standalone
LPAR2           machines     standalone
LPAR3           machines     standalone
LPAR4           machines     standalone
LPAR5           machines     standalone
LPAR6           machines     standalone
```

```

VI01      machines      standalone
VLPAR1    machines      standalone
VLPAR2    machines      standalone
VLPAR3    machines      standalone
VLPAR4    machines      standalone
LPAR4_p5  machines      standalone
v\par1    machines      alternate_master
VLPAR6    machines      standalone
{nimmast}:/ #

```

---

The first column lists the NIM names, while the second column lists the class name (only machines in this example), and the third column lists the type names related to this class name.

To see detailed client information, run the `lsnim -l <client_name>` command. Example 2-3 shows the output of the command `lsnim -l LPAR3`.

*Example 2-3 Output of the `lsnim -l <client_name>` command*

```

{nimmast}:/ # lsnim -l LPAR6
LPAR6:
class      = machines
type       = standalone
connect    = shell
platform   = chrp
netboot_kernel = mp
if1        = NET_EN1 lpar6 0
cable_type1 = tp
Cstate     = ready for a NIM operation
prev_state = in the process of booting
Mstate     = not running
cpuid      = 001A85D24C00
Cstate_result = success
current_master = nimmast
sync_required = yes
{nimmast}:/ #

```

---

The Cstate field determines whether the machine object is in use. If the Cstate value is not “*ready for a NIM operation*”, we can’t perform any operation on this object.

**Attention:** The Mstate field may not be accurate, as it depends on whether NIM handled the latest shutdown/reboot.

## Networks class

The network is what allows the machines in a NIM environment to communicate with each other. If the network is a simple local area network (LAN), the definition of the network object is simplified. The purpose of the network object is to depict the network topology used by the NIM environment. Each modification made on the physical network must be reflected to the NIM database.

Example 2-4 gives an example of the output of the `lsnim -c networks` command.

*Example 2-4 Example of `lsnim -c networks` command*

---

```
{nimmast}:/ # lsnim -c networks
NET_EN1     networks      ent
{nimmast}:/ #
```

---

**Note:** NIM supports multiple network types, such as: Standard Ethernet, IEEE 802.3 Ethernet, Token-Ring, FDDI, ATM, and Generic. The network type determines how certain NIM operations are handled. For example, network boot is only supported on Standard Ethernet, Token-Ring and FDDI.

In our environment we use only one network type, Standard Ethernet.

To see the details of a particular NIM network object, run the `lsnim -l <network_name>` command, as shown in Example 2-5.

*Example 2-5 Output of the `lsnim -l <network_name>` command*

---

```
{nimmast}:/ #
{nimmast}:/ # lsnim -l NET_EN1
NET_EN1:
  class      = networks
  type       = ent
  Nstate     = ready for use
  prev_state = information is missing from this object's definition
  net_addr   = 10.1.1.0
  snm        = 255.255.255.0
{nimmast}:/ #
```

---

The Nstate field determines whether the network object is in use. If the Nstate value is different from “*ready for a NIM operation*”, we can’t perform any operation on this object.



**Resources class**

A NIM resource is in fact a pointer to a file or a directory located on a Resource Server - the NIM Master is the default Resource Server. For example, the `bosinst_data` resource points to a file that contains information used to perform the the installation process without requiring any manual intervention. The basic characteristics of a resource are the NIM name, the location (the location of the file or directory) and the resource server hosting the resource. Example 2-6 shows a list of resources, while Example 2-7 gives details of a resource. In particular, you can see the location and server attributes which determine where is the resource located.

*Example 2-6 `lsnim -c resources` command output*


---

```
{nimmast}:/ # lsnim -c resources
boot                resources          boot
nim_script          resources          nim_script
LPP_53_ML4          resources          lpp_source
SPOT_53_ML4         resources          spot
BID_NP_HDO          resources          bosinst_data
spotaix5104         resources          spot
lpp_sourceaix5204   resources          lpp_source
spotaix5204         resources          spot
spot5304            resources          spot
lpp_sourceaix5104   resources          lpp_source
lpar1nim-mksysb     resources          mksysb
exclude_lpar5       resources          exclude_files
MK_LPAR6_AIX530403  resources          mksysb
MKSYSB_VIO12_53_ML3 resources          mksysb
MK_AIX5104          resources          mksysb
mksysb_lpar5c       resources          mksysb
mksysb_lpar5d       resources          mksysb
AIXTFLINUX          resources          lpp_source
BID_NP_HDO_MKMIG    resources          bosinst_data
nim_move_up_bid      resources          bosinst_data
nim_move_up_exclude resources          exclude_files
LPAR4_phys_mksysb   resources          mksysb
LPAR4_p5_mksysb     resources          mksysb
{nimmast}:/ #
```

---

Example 2-7 shows details of a resource retrieved using the `lsnim -l <resource_name>` command.

*Example 2-7 Resource details*


---

```
{nimmast}:/ # lsnim -l BID_NP_HDO
BID_NP_HDO:
```

```

class      = resources
type       = bosinst_data
Rstate    = ready for use
prev_state = unavailable for use
location = /other_res/bid.np.hd0
alloc_count = 0
server   = master
{nimmast}:/ #

```

---

The location and the server are well identified. In this case, the server is the master.

**Note:** Another important attribute is Rstate. Rstate is for resource state. The shown state determines whether the object is currently in use or not. If the Rstate is “not ready”, it cannot be used and a warning message will appear on the console.

### **Groups class**

A group is a collection of either machines or resources. Only two types are available:

- res\_group for grouping a set of resources
- mac\_group for grouping a set of machines

Example 2-8 shows the output of the command `lsnim -c groups`.

*Example 2-8 Listing NIM groups*

---

```

{nimmast}:/ # lsnim -c groups
LPAR123456   groups      mac_group
nimgrp       groups      mac_group
res_53_m13   groups      res_group

```

---

## **2.1.2 How does a NIM environment work?**

Now that we know *what* NIM consists of, we need to understand *how* NIM works, in particular, how NIM uses different components of the operating system to fulfill its duties. This section describes the most commonly used resources and their utilization. A basic diagram of how NIM installation works is presented in Figure 2-3 on page 17.

To start a machine installation, only two resources are mandatory: the LPP source (lpp\_source) and the shared product object tree (SPOT). However, to avoid manual intervention during the installation process, additional resources

providing information for the AIX\_install process can be used (i.e. bosinst\_data, image\_data, and script).

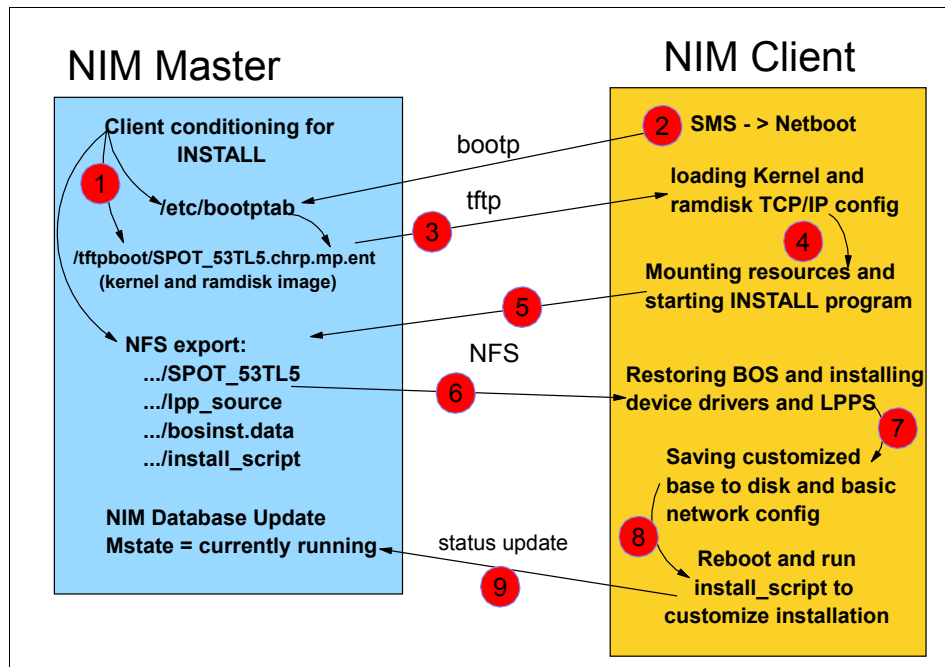


Figure 2-3 How NIM installation works

The following list presents the basic resources needed for installing machines in your environment:

► **SPOT**

This stands for Shared Product Object Tree. It is a directory of code (installed filesets) that is used during client booting procedure. It is equivalent in content to the code that resides in /usr file system on a system running AIX 5L (binary objects - executables and libraries, header files and shell scripts). This resource (directory) replaces the content of the basic ramdisk image available on installation CDs.

**Remember:** The installation is performed over a network, and there are no AIX CDs available. Device drivers, the BOS install program and other necessary code needed to perform a base OS install are found inside the SPOT. During the installation, the client machine NFS mounts this resource in order to access the code needed for the installation process.

The SPOT also contains the code needed to generate the boot images (kernels, that will be stored in the /tftpboot directory) that the client uses until it can manage to NFS mount the SPOT directory.

**Note:** Multiple boot images can exist in the /tftpboot directory. Maintaining multiple smaller boot images helps saving time when transferring the files via TFTP to the client.

Also, as the number of kernels (boot images) in the /tftpboot directory may increase with adding new OS levels, we recommend you to create a separate file system for the /tftpboot directory.

A general structure of a non /usr SPOT directory is presented in Figure 2-4.

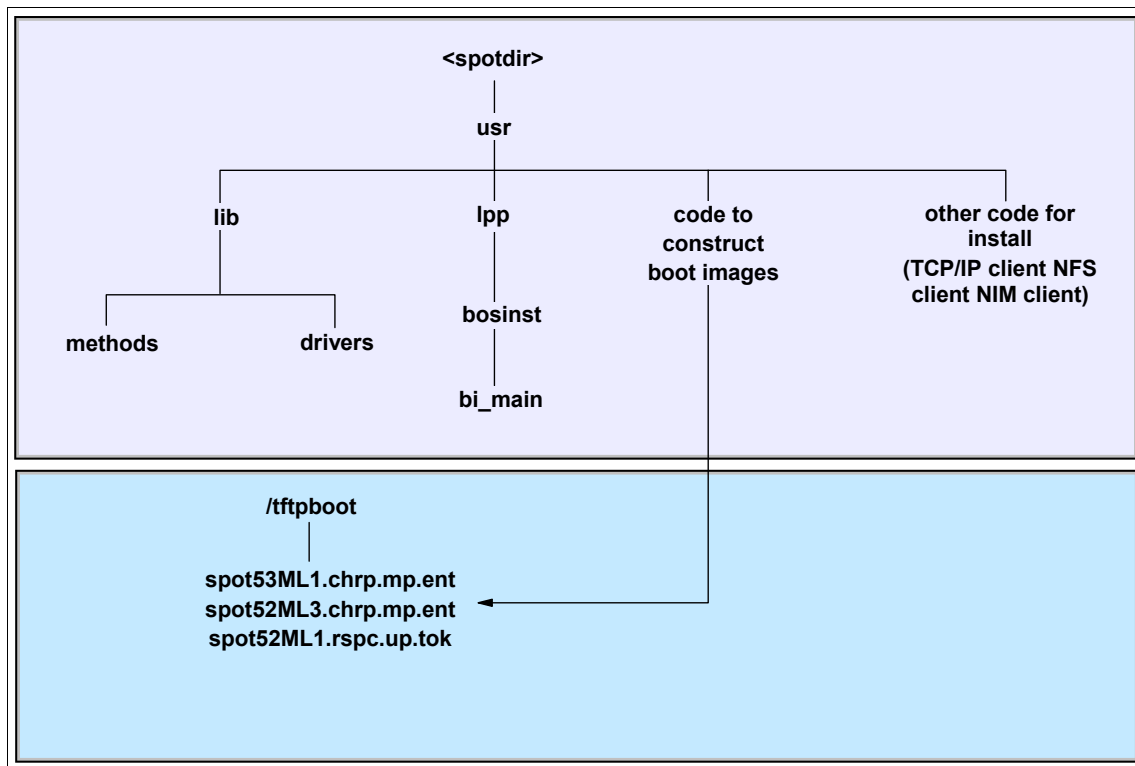


Figure 2-4 General non /usr SPOT structure

We have already seen that the NIM database consists of several classes. Classes also consist of types. For example, the resources class consists of several resource types, like: spot, lpp\_source etc.

By running the `lsnim -t spot` command, we have the list of the different available SPOT resources. The output of the `lsnim -t spot` command is shown in Example 2-9.

*Example 2-9 Example of `lsnim -t spot` command*

---

```
{nimmast}:/ #
{nimmast}:/ # lsnim -t spot
SPOT_53_ML4      resources      spot
spotaix5104     resources      spot
spotaix5204     resources      spot
spot5304        resources      spot
{nimmast}:/ #
```

---

Example 2-10 gives the detail of a particular SPOT resource.

*Example 2-10 Details of a particular spot resource*

---

```
{nimmast}:/ #
{nimmast}:/ # lsnim -l SPOT_53_ML4
SPOT_53_ML4:
  class      = resources
  type       = spot
  plat_defined = chrp
  arch       = power
  bos_license = yes
  Rstate     = ready for use
  prev_state = verification is being performed
  location   = /AIX53ML4/SPOT_53_ML4/usr
  version    = 5
  release    = 3
  mod        = 0
  oslevel_r = 5300-04
  alloc_count = 1
  server     = master
  if_supported = chrp.mp ent
  Rstate_result = success
{nimmast}:/ #
```

---

**Tip:** One of the details returned by the `oslevel -r` command by `lsnim -l <spot_resource_name>` is the AIX level. In our example this is 5300-04. It is important to know the OS level when you want to install a client, as the levels of the SPOT resource and mksysb resource should match.

**Note:** In case SPOT and mksysb are not at the same level, installation will only work if the SPOT is at a higher level than the mksysb, but after the mksysb installation, the level of the system just installed will be updated to match the level of the SPOT using the `lpp_source` object which also needs to be allocated for the installation.

► **lpp\_source**

An `lpp_source` is a directory similar to AIX install CDs. It contains AIX LPPs (Licensed Program Products) in BFF (Backup File Format) format and RPM (RPM Package Manager - see <http://www.rpm.org/>) filesets that you can install. A general structure of the `lpp_source` directory is shown in Figure 2-5.

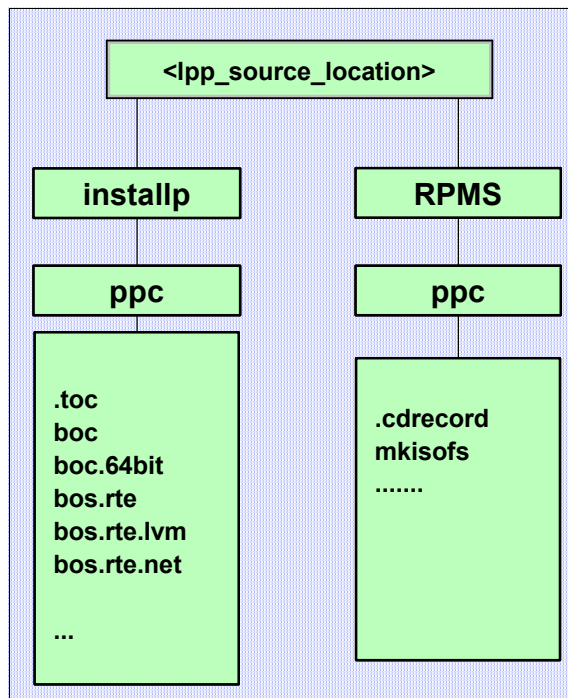


Figure 2-5 `lpp_source` structure

**Note:** The AIX `installp` command can handle both AIX installable packages (BFF - Backup File Format) and RPMs (RPM Package Manager) files.

**Warning:** The `.toc` file (only found in directories containing BFF packages) must be kept up to date to reflect the latest filesets in the `lpp_source`.

Example 2-11 on page 21 shows the several `lpp_source` type resources (using the `lsnim -t lpp_source` command).

*Example 2-11 Example of `lsnim -t lpp_source` command*

---

```
{nimmast}:/ #
{nimmast}:/ # lsnim -t lpp_source
LPP_53_ML4          resources      lpp_source
lpp_sourceaix5204  resources      lpp_source
lpp_sourceaix5104  resources      lpp_source
AIXTFLINUX         resources      lpp_source
{nimmast}:/ #
```

---

Example 2-12 shows the attributes of one of the `lpp_source` resource.

*Example 2-12 Details of a particular `lpp_source` resource*

---

```
{nimmast}:/ # lsnim -l LPP_53_ML4
LPP_53_ML4:
  class      = resources
  type       = lpp_source
  arch       = power
  Rstate     = ready for use
  prev_state = ready for use
  location   = /AIX53ML4/LPP_53_ML4
  images    = yes
  alloc_count = 1
  server     = master
{nimmast}:/ #
```

---

**Important:** The *simages* attribute, when set to *yes* means that the *lpp\_source* resource contains the necessary filesets to support a base operating system type “rte” (Run-Time Environment) installation. The *simages* attribute is set to *yes* for an *lpp\_source* resources that contains a minimum set of install images (LPPs) necessary to support the creation of a SPOT resource or to support the installation of the AIX operating system over the network. For more information for the minimum list, refer to the following script:

```
/usr/lpp/bos.sysmgt/nim/methods/c_sh_lib
```

► **mksysb**

This resource is a file containing the image of the root volume group (generated with the AIX **mksysb** command) of a machine. It is used to restore a machine, or to install it from scratch (also known as “cloning” a client). An **mksysb** resource can be defined by first creating the **mksysb** image with SMIT or the **mksysb** command, then defining the resource that references the backup image file. A **mksysb** resource can also be defined using CLI:

```
nim -o define -t mksysb -a souce=<client_to_backup> -a server=master
-a location=<location to store file> RESOURCE_NAME
```

Example 2-13 on page 22 shows the different **mksysb** resources which are part of our NIM environment.

*Example 2-13 Example of `lsnim -t mksysb` command*

---

```
{nimmast}:/ # lsnim -t mksysb
lpar1nim-mksysb      resources      mksysb
MK_LPAR6_AIX530403  resources      mksysb
MKSYSB_VIO12_53_ML3  resources      mksysb
MK_AIX5104           resources      mksysb
```

---

Example 2-14 shows the detail of one particular **mksysb** resource.

*Example 2-14 Details of a particular **mksysb** resource*

---

```
{nimmast}:/ # lsnim -l MK_LPAR6_AIX530403
MK_LPAR6_AIX530403:
  class      = resources
  type       = mksysb
  arch       = power
  Rstate     = ready for use
  prev_state = unavailable for use
  location   = /export/images/mk_lpar6_aix530403
  version    = 5
  release    = 3
  mod        = 0
```



```

oslevel_r    = 5300-04
alloc_count  = 0
server       = master
{nimmast}:/ #

```

---

**Attention:** One of the attributes shown by the `lsnim -l <mksysb_resource_name>` command is the AIX 5L level. This value is actually the one returned by the `oslevel -r` command on the client from which the image has been retrieved (5300-04 in our case). It is important to know this value when you want to install a client because the levels of the SPOT resource and mksysb resource should match.

► **bosinst\_data**

The `bosinst_data` resource is a flat ASCII file similar to `bosinst.data` file used for restoring system backup images from tape or CD/DVD. When you start the installation of a IBM System p machine, you will be prompted to select the console, language, installation method, the target disk(s) for rootvg etc. This resource comes in handy for both push or pull installation of multiple machines at the same time, as it will automate the installation process without requiring interactive answers for the installation parameters.

Example 2-15 on page 23, Example 2-16 on page 23, and Example 2-17 on page 24 provide the different elements related to a `bosinst_data` resource.

*Example 2-15 bosinst\_data resources*

```

{nimmast}:/ # lsrim -t bosinst_data
BID_NP_HD0          resources      bosinst_data
BID_NP_HD0_MKMIG   resources      bosinst_data
nim_move_up_bid     resources      bosinst_data
{nimmast}:/ #

```

---

*Example 2-16 Details of a bosinst\_data resource*

```

{nimmast}:/ # lsrim -l BID_NP_HD0
BID_NP_HD0:
  class      = resources
  type       = bosinst_data
  Rstate    = ready for use
  prev_state = unavailable for use
  location = /other_res/bid.np.hd0
  alloc_count = 0
  server   = master
{nimmast}:/ #

```

---

*Example 2-17 Example of the contain a bosinst.data file*

---

```
control_flow:
CONSOLE = /dev/lft0
INSTALL_METHOD = overwrite
PROMPT = no
EXISTING_SYSTEM_OVERWRITE = yes
INSTALL_X_IF_ADAPTER = yes
RUN_STARTUP = yes
RM_INST_ROOTS = no
ERROR_EXIT =
CUSTOMIZATION_FILE =
TCB = no
INSTALL_TYPE =
BUNDLES =
SWITCH_TO_PRODUCT_TAPE =
RECOVER_DEVICES = Default
BOSINST_DEBUG = no
ACCEPT_LICENSES = yes
DESKTOP = NONE
INSTALL_DEVICES_AND_UPDATES = yes
IMPORT_USER_VGS =
ENABLE_64BIT_KERNEL = yes
CREATE_JFS2_FS = yes
ALL_DEVICES_KERNELS = yes
GRAPHICS_BUNDLE = yes
MOZILLA_BUNDLE = no
KERBEROS_5_BUNDLE = no
SERVER_BUNDLE = no
ALT_DISK_INSTALL_BUNDLE = no
REMOVE_JAVA_118 = no
HARDWARE_DUMP = yes
ADD_CDE = no
ADD_GNOME = no
ADD_KDE = no
ERASE_ITERATIONS = 0
ERASE_PATTERNS =
MKSYSB_MIGRATION_DEVICE =
```

```
locale:
BOSINST_LANG = en_US
CULTURAL_CONVENTION = en_US
MESSAGES = en_US
KEYBOARD = en_US
```

```
target_disk_data:  
  PVID =  
  PHYSICAL_LOCATION =  
  CONNECTION =  
  LOCATION =  
  SIZE_MB =  
  HDISKNAME = hdisk0
```

---

**Note:** The highlighted parameters in Example 2-17 are the minimum mandatory parameters necessary to perform an installation without a manual entry.

► **script**

The script resource is a file. Once the BOS installation is finished on your client, you can perform customization such as file system resizing, additional user creation, and so forth. The script resource points to a file (usually a shell script) that contains all the commands needed to perform customization.

### 2.1.3 Before working with a NIM environment

We have now the knowledge on the different components of a NIM environment. However, in order to help you build your NIM cluster, you need additional information, further explained in this section.

For simplicity, we can say that NIM has two basic elements: AIX and the network. If one of them is not working properly, NIM does not function and its behavior can be erratic. Although Chapter 7, “Basic NIM problem determination and tuning” on page 575 describes some basic problem determination scenarios, we find useful to introduce here a few common troubleshooting examples, even if they are not directly related to NIM.

**Note:** Before you start with NIM you must familiarize yourself with the environment and have the basic AIX system administration and problem determination skills in the following two areas:

***AIX 5L related issues:***

- A file system becomes full when you are creating a new lpp\_source resource.
- A file system becomes full when you are creating a new mkysb resource.

- ▶ File limit is set to 2GB while the actual backup file is larger (either on the NIM master or on the NIM clients to be backed up).
- ▶ A file part of, or configured as a resource does not have the right permissions.
- ▶ An lpp\_source's table of contents file (.toc) file has not been updated.

***Network related issues:***

- ▶ Physical network connection not working properly.
- ▶ Network parameters related to a specific adapter (speed, duplex) are not consistent across systems and network elements.
- ▶ Name resolution not working properly (duplicated addresses in the /etc/hosts file, a mix of long names and short names etc.).

The following section presents a method to help you building a NIM environment.

## 2.1.4 NIM master selection

Choosing the machine that will become the NIM master is perhaps the most important step in building a reliable NIM environment.

Before you start working with NIM, ensure your network is up and running. Then, among the available machines, select the one that will become the NIM master. To help you on this, check the following criteria:

- ▶ The machine that will become the NIM master must have AIX installed.

**Important:** The NIM master should always have the highest version of AIX installed across the entire NIM environment. Same AIX level for master and client is acceptable.

- ▶ The NIM Master machine must NOT be a production workload machine because you will have to regularly shutdown down the NIM Master machine to perform AIX upgrades.
- ▶ Most people have the NIM Master as a separate dedicated purpose machine or LPAR, so the maintenance of the images and its AIX version does not affect other workloads.
- ▶ A Micro-partition (POWER5 or above machine using a shared processor LPAR) is ideal as the NIM Master does not use a lot of CPU cycles. However, other prefer to use a separate (perhaps a machine from previous hardware generation that is available after an upgrade) as the NIM Master machine.

- ▶ Some customers use the NIM Master for other system administration tasks like performance monitoring, an operations webserver and NFS source of useful tools.
- ▶ Regardless of the network complexity, the number of network adapters in your NIM environment must be planned.

**Note:** If you have a simple network with all machines on a single segment and a single IP address range then only one network adapter is required.

- ▶ The disk space must be estimated and compared to any existing resources. There are no official guidelines because NIM environments may vary. Table 2-2 shows some considerations in planning disk space for your NIM master. The two largest resources to consider are the lpp\_source and the mksysb backups. Keep in mind that a client can become a NIM resource server. Therefore, all disk space information related to this resource must be taken into account for the disk space planning of this machine. See Table 2-2 for average figures for the main files or directories needed by NIM.

Table 2-2 Average recommended values for disk space estimation

Item	Recommended Size	Location	Comments
lpp_source	3GB per AIX Version or TL	Set by the administrator	The size of an lpp_source for base install is less than 1 GB per AIX version or TL. More if there are optional lpp_source filesets.
spot	1GB per AIX Version or TL	Set by the administrator	N/A
mksysb	4GB	Set by the administrator	This figure is an average. Generally, the size can be between 1 GB (for a basic rte install) to 6GB or 7GB for a running machine.
other	1GB	Set by the administrator	This place is used to put other resources as bosins_data, scripts and so forth. The basic space is quite negligible

**Note:**

1. Volume groups backups for non rootvg volume groups are not shown in Table 2-2 on page 27. If you are using NIM to backup also user volume groups, add their space requirements.
2. Consider additional space for future growth.

For example, two releases of AIX (5.2 and 5.3) with three maintenance levels each, 10 mksysb images from production machines and three generations of these require the following disk space:

$$3 \times 6 + 1 \times 6 + 4 \times 10 \times 3 + 1 = 145 \text{ GB as a minimum.}$$

Having selected your NIM Master machine or LPAR, the following section details the actions to setup a basic NIM environment.

## 2.1.5 Easy steps to start with NIM

In this section, we list the steps to start a NIM environment.

**Note:** At this point, no NIM definition or object has been yet created, and the objective is to install a system over the network. But first, this system has to become a client to a NIM environment. We only know that one machine will become the NIM master, the others, will be clients.

### ***Step 1: Select a machine to be the Master***

Section 2.1.4, “NIM master selection” provides information for this decision.

### ***Step 2: Install AIX for the NIM master***

Install a completely fresh copy of the latest AIX release and level you have to avoid inheriting problems. It is recommended you start with the highest possible AIX level to avoid having to update it later on.

### ***Step 3: Check the software requirements***

The following steps are used to satisfy the minimum requirements for your NIM master:

- ▶ The level of AIX 5L that is installed on the NIM master:  
`oslevel -r` or `oslevel -s`
- ▶ Available disk space  
`lspv` and `lsattr -El hdisk*`

- ▶ Network adapters
  - `lscfg` and `netstat -in`

#### **Step 4: Install the following two NIM filesets**

The following NIM filesets must be installed:

- `bos.sysmgmt.nim.master`
- `bos.sysmgmt.nim.spot`

**Note:** The fileset `bos.sysmgmt.nim.spot` is required when you want to allow a client server to serve SPOT resources. This fileset is optional, and should be installed if you are planning to work with thin servers (diskless or dataless clients).

- ▶ SMIT panels are shown in Example 2-18, Example 2-19 on page 30, Example 2-20 on page 31, Example 2-21 on page 31 and Example 2-22 on page 32.

In Example 2-18 we use the AIX CD for TL4 to install the NIM LPPs. We start using the `smitty install_latest` fastpath and select `/dev/cd0` as install source, then select the NIM packages from the list:

#### *Example 2-18 SMIT panel to install NIM filesets*

```

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

                                [Entry Fields]
* INPUT device / directory for software      /dev/cd0
* SOFTWARE to install                        [@ 5.3.0.40 Network I> +
PREVIEW only? (install operation will NOT occur)  no +
COMMIT software updates?                      yes +
SAVE replaced files?                          no +
AUTOMATICALLY install requisite software?      yes +
EXTEND file systems if space needed?           yes +
OVERWRITE same or newer versions?             no +
VERIFY install and check file sizes?          no +
Include corresponding LANGUAGE filesets?      yes +
DETAILED output?                              no +
Process multiple volumes?                     yes +
ACCEPT new license agreements?                no +
Preview new LICENSE agreements?               no +

F1=Help          F2=Refresh          F3=Cancel          F4=List
F5=Reset         F6=Command          F7=Edit           F8=Image
  
```

F9=Shell

F10=Exit

Enter=Do

**Note:** This is a straight forward step to perform. It is a standard AIX fileset installation. The *highlighted* values must be entered according to your environment.

### **Step 5: Configure the selected machine as a NIM master**

In our configuration we use **smitty nimconfig** fastpath as shown in Example 2-19. Other tools, such as EZ NIM are detailed in section 3.3, “EZ NIM” on page 114.

#### *Example 2-19 SMIT panel to configure an AIX machine as a NIM master*

Configure Network Installation Management Master Fileset

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

	[Entry Fields]	
* Network Name	[NET_ENO]	
* Primary Network Install Interface	[en0]	+
Allow Machines to Register Themselves as Clients?	[yes]	+
Alternate Port Numbers for Network Communications (reserved values will be used if left blank)		
Client Registration	[ ]	#
Client Communications	[ ]	#
F1=Help	F2=Refresh	F3=Cancel
F5=Reset	F6=Command	F7>Edit
F9=Shell	F10=Exit	Enter=Do
	F4=List	F8=Image

**Note:** The highlighted values must be entered according to your environment.

### **Step 6: Creating the file systems for NIM**

As explained in 2.1.2, “How does a NIM environment work?” on page 16, the lpp\_source and the SPOT resources are directories and the related file systems must be created. For their estimate sizes, refer to Table 2-2 on page 27.

### **Step 7: Defining basic resources**

We define first the lpp\_source resource followed by the SPOT. The lpp\_source resource is a directory containing installable files. Use the **smitty nim\_mkres** fastpath, select lpp\_source and enter the required values in the input screen shown in shown in Example 2-20.



**Tip:** You can also consider creating the basic resources using the following steps: **smitty nim** --> Configure the NIM Environment --> configure a Basic NIM Environment (Easy Startup) to create the `lpp_source` and SPOT together instead of creating individual resources.

**Note:** We suggest to create the `lpp_source` first from CD, then use it to create the SPOT. In this way, you use the CD only once, thus reducing the time for creating the SPOT.

*Example 2-20 SMIT panel to define a `lpp_source` resource*

---

Define a Resource

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

```
[Entry Fields]
* Resource Name                [LPP_53_ML4]
* Resource Type                lpp_source
* Server of Resource           [master]                +
* Location of Resource         [ /nim/aix53m14/lppsour>  /
Architecture of Resource      []                        +
Source of Install Images       [cd0]                   +/
Names of Option Packages       []
Show Progress                  [yes]                    +
Comments                        []
```

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

---

**Note:** The highlighted values must be entered according to your environment. Remember to check the “`simages`” attribute of the newly created `lpp_source` resource by running the `lsnim -l <lpp_source_name>` command.

Once the `lpp_source` resource is created, we define the SPOT resource. Use the **smitty nim\_mkres** and select `spot`, then enter the information in the smit panel as shown in Example 2-21.

*Example 2-21 SMIT panel to define a SPOT resource*

---

Define a Resource

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

	[Entry Fields]	
* Resource Name	[SPOT_53_ML4]	
* Resource Type	spot	
* Server of Resource	[master]	+
* Source of Install Images	[LPP_53_ML4]	+
* Location of Resource	[/nim/aix53m14/]	/
Expand file systems if space needed?	yes	+
Comments	[]	
installp Flags		
COMMIT software updates?	no	+
SAVE replaced files?	yes	+
AUTOMATICALLY install requisite software?	yes	+
OVERWRITE same or newer versions?	no	+

**Note:** The highlighted values must be entered according to your environment. Remember to check the status of the resource by running the `lsnim -l <spot_name>` command.

### Step 8: Defining the client

Once the two basic resources have been defined, we add the first NIM client. Use the `smitty nim_mkmac` fastpath, enter the name of the machine and fill in the required fields in the SMIT panel is shown in Example 2-22.

**Important:** The NIM machine name must be a resolvable IP label (usually the same as the short hostname of the machine to be added).

Example 2-22 SMIT panel to define a NIM client

#### Define a Machine

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

	[Entry Fields]	
[TOP]		
* NIM Machine Name	[LPAR3]	
* Machine Type	[standalone]	+
* Hardware Platform Type	[chrp]	+
Kernel to use for Network Boot	[mp]	+
Communication Protocol used by client	[]	+
Primary Network Install Interface		
* Cable Type	tp	+
Network Speed Setting	[]	+

```

    Network Duplex Setting          []          +
*   NIM Network                    NET_EN1
*   Host Name                       lpar3
    Network Adapter Hardware Address [0]
    Network Adapter Logical Device Name []
[MORE...4]

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

```

---

Remember that NIM programs must be able to resolve the IP labels using the /etc/hosts file or a DNS server. See section “Setting up IP name resolution” on page 62 for additional information.

### **Step 9: Start the client installation**

At this time, as there is no mksysb resource available yet, the only way to perform a BOS install on a client is by using a Run Time Environment (rte). Use the **smitty nim\_task\_inst** fastpath, then select “Install the Base Operating System on Standalone Clients”, the client to be installed (LPAR3), “rte” as installation method, and the resources (SPOT\_53\_ML4 and LPP\_53\_ML4), as shown in Example 2-23.

#### *Example 2-23 Starting base OS installation on a client*

---

#### Install the Base Operating System on Standalone Clients

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

```

[TOP]                                [Entry Fields]
* Installation Target                 LPAR3
* Installation TYPE                   rte
* SPOT
  SPOT_53_ML4
  LPP_SOURCE                          [LPP_53_ML4]      +
  MKSYSB

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

```

```

FORCE PUSH the installation?                [no]          +
Initiate reboot and installation now?      [no]          +
-OR-
Set bootlist for installation at the      [no]          +
next reboot?
[MORE...22]

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

```

---

**Remember:** You need to accept the licence agreements and to change the value of Initiate reboot and installation now from **yes** to **no**. Also, since this is the very first time we install this machine, we need to manually initiate client network boot (to allow client to “pull” resources from server).

### Step 10: Additional checks

Perform the following checks before starting the client installation.

We want to verify that the **bos\_inst operation** is correctly set up. There are three things you should always check before you actually perform the installation itself:

- Check that the correct entry has been added in the `/etc/bootptab` file.

Example 2-24 shows, in highlighted text, LPAR3 client entry.

#### Example 2-24 Content of the `/etc/bootptab` file

```

{nimmast}:/ # tail /etc/bootptab
.....
#      T179 -- (xstation only) -- XDMCP host
#      T180 -- (xstation only) -- enable virtual screen
lpar3:bf=/tftpboot/lpar3:ip=10.1.1.13:ht=ethernet:sa=10.1.1.1:sm=255.255.255.0:
{nimmast}:/ #

```

---

- Check that the boot files have been created in the `/tftpboot` directory. The `/tftpboot` directory must contain a boot image with the name of the client. This file is in fact a symbolic link to the boot image (kernel) created from the SPOT resource. This directory also must contain a file `<client_name>.info` use during the installation to set environment variables which are used by the boot image file and the install scripts.

Example 2-25 should list the files and resources in `/tftpboot`.

#### Example 2-25 Content of the `/tftpboot` directory

```

{nimmast}:/ # ls -l /tftpboot
total 151248

```

```

-rw-r--r--  1 root    system    12211712 Jun 20 15:57
AIX53_ML04_debug.chrp.mp.ent
-rw-r--r--  1 root    system    11688960 Jun 12 11:13 SPOT-VIO.chrp.mp.ent
-rw-r--r--  1 root    system    12211712 Jun 19 16:24
SPOT_53_ML4.chrp.mp.ent
-rw-r--r--  1 root    system    12211712 Jun 22 17:43
SPOT_PRUEBA.chrp.mp.ent
-rw-r--r--  1 root    system    12211712 Jun 22 18:01 SPOT_TEST.chrp.mp.ent
lrwxrwxrwx  1 root    system    33 Jun 26 09:11 lpar3 ->
/tftpboot/SPOT_53_ML4.chrp.mp.ent
-rw-r--r--  1 root    system    1109 Jun 26 09:11 lpar3.info
-rw-r--r--  1 root    system    8162755 Jun 02 10:37
spotaix5104.chrp.mp.ent
-rw-r--r--  1 root    system    8720910 Jun 02 11:49
spotaix5204.chrp.mp.ent
{nimmast}:/ #

```

- Check that the required resources are allocated to the client and that its Cstate parameter has the right value. Refer to Example 2-26 for details related to the LPAR3 client. There is information in highlighted text that must be checked before starting the pull mode.

*Example 2-26 Output of the `lsnim -l LPAR3` command*

```

{nimmast}:/ # lsnim -l LPAR3
LPAR3:
  class      = machines
  type       = standalone
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = NET_EN1 lpar3 0
  cable_type1 = tp
  Cstate     = BOS installation has been enabled
  prev_state = ready for a NIM operation
  Mstate     = not running
  boot       = boot
  lpp_source  = LPP_53_ML4
  nim_script = nim_script
  spot       = SPOT_53_ML4
  cpuid      = 001A85D24C00
  control    = master
  current_master = nimmast
  sync_required = yes
{nimmast}:/ #

```

**Step 11: Start the client installation**

To start the client installation, the only way is to power on the client and go into the SMS menus, using the console, press the “1” key during the boot sequence. For details on how to get to SMS menu, see “SMS and console flow during NIM client installation” on page 83.

**Step 12: Redo the checks in Step 10: Additional checks**

The /etc/bootptab file does not contain anymore a specific line related to the machine, neither no specific link into the /tftpboot directory and the Cstate of the client must be back to Ready for a NIM operation.

**2.1.6 Easy steps to maintain a NIM environment**

In this section, a real and simple example is used to present the different steps to maintain a NIM environment. The starting point is a running NIM environment which consists of a master and its clients. One of the clients needs to be updated by either a new fileset, or a PTF (fix) or an entire Technology Level (TL, formerly ML). This section contains the steps that must be followed:

- ▶ Find a source to install the missing fileset on a client.  
Remember that we are working in a NIM environment, thus the installation packages’ source is an lpp\_source. Because NIM is able to work with multiple AIX versions and technology/maintenance levels, we must know the appropriate lpp\_source that is used as the source of the filesets.
- ▶ Check the AIX level of the client by using the commands:
 

```
oslevel -r
and
oslevel -s
```
- ▶ Check if the required fileset is part of the current lpp\_source by performing the **lslpp operation** on a lpp\_source. You can use: **smitty nim --> Perform NIM Administrative Tasks --> Manage Resources**, as shown in Example 2-27.

*Example 2-27 Example of lslpp operation on an lpp\_source resource*

---

Manage Resources

Move cursor to desired item and press Enter.

```
List All Network Install Resources
Define a Resource
Change/Show Characteristics of a Resource
Show the Contents of a Resource
Remove a Resource
+-----+
|                                         Network Install Operation to Perform                                         |
+-----+
```

```

Move cursor to desired item and press Enter.
showres = show contents of a resource
lslpp = list LPP information about an object
check = check the status of a NIM object
lppmgr = eliminate unnecessary software images in an lpp_source
update = add or remove software to or from an lpp_source

F1=Help          F2=Refresh      F3=Cancel
F8=Image         F10=Exit       Enter=Do
F1| /Find        n=Find Next
F9+-----+

```

- ▶ Optionally, update the current lpp\_source if the required fileset is not present by performing the update operation on the related lpp\_source resource, as shown in Example 2-28. We use: **smitty nim --> Perform NIM Software Installation and Maintenance Tasks --> Software Maintenance and Utilities --> Add Software to an lpp\_source --> Select Target lpp\_source --> Select Software Source.**

*Example 2-28 Example of update operation on an lpp\_source resource*

```

Add Software to an lpp_source

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

                                [Entry Fields]
TARGET lpp_source                LPP_53_ML4
SOURCE of Software to Add       cd0
SOFTWARE Packages to Add         [] +
-OR-
INSTALLP BUNDLE containing packages to add  [] +

gencopy Flags
  DIRECTORY for temporary storage during copying  [/tmp]
  EXTEND filesystems if space needed?             yes +
  Process multiple volumes?                       no  +

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

```

- ▶ Perform a “push” operation from your master to the client by using the following SMIT fast path:  
**smitty nim\_task\_inst**  
and choosing *Install Software*. Example 2-29 shows the related SMIT panel.

*Example 2-29 SMIT panel to do a fileset installation on a client*

---

Install and Update Software

Move cursor to desired item and press Enter.

```

Install the Base Operating System on Standalone Clients
Install Software
Update Installed Software to Latest Level (Update All)
Install Software Bundle
Update Software by Fix (APAR)
Install and Update from ALL Available Software

```

```

F1=Help      F2=Refresh   F3=Cancel    F8=Image
F9=Shell     F10=Exit    Enter=Do

```

---

- Check the result of the installation by using the following SMIT fast path (see Example 2-30):

```
smitty "nim_list_installed_sw"
```

*Example 2-30 SMIT panel to list the installed software on a client*

---

List Installed Software

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

```

                                     [Entry Fields]
* Target                             LPAR3
* Software Names                       [all]          +

Show SUPERSEDED levels?                [yes]          +

```

```

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

```

---

## 2.2 What is new in NIM for AIX 5L

NIM was introduced with AIX V 4.1. NIM improvements have been added with each release of the AIX operating system. In covering the differences between



versions (NIM objects, NIM application, maintenance, administration and the tools to support migration from one level to the next), we start with AIX 5L V5.1 and trace its progress up to AIX 5L V5.3. For AIX versions older than AIX 5L V5.1 refer to the redbook “NIM: From A to Z in AIX 4.3”, SG24-5524.

## 2.2.1 NIM improvements with AIX 5L V5.1

One of the main improvement coming with AIX 5L V5.1 is the capability to use AIX tools to manage RPMS filesets. We recall that the basic command `installp` has been replaced by `geninstall` and also the `gencopy` command has been created. Thus, it has been fairly easy to administrate RPM packages.

## 2.2.2 NIM improvements with AIX 5L V5.2

This section lists the main improvements coming with AIX 5L V5.2:

- ▶ `lpp_source` management tools  
Enhancements to the management of `lpp_source` resources including the following:
  - The `lpp_source` is no longer required for `mksysb` installations.
  - The NIM update operation, which allows you to update an `lpp_source` resource by adding or removing packages. Previously, you could copy or remove packages into/from an `lpp_source` directory and then run `nim -o check` command to update the `lpp_source`. SMIT allows you to add packages to the `lpp_source` directory through the `smitty nim_bffcreate` fast path. However, this SMIT function does not check if the `lpp_source` is allocated (or locked), nor does it update the “`simages`” attribute when finished. The update operation has been created to address this situation.
  - The `lppmgr` operation is available to help you manage your `lpp_source` resource. The `lppmgr` facility can be used to clean an `lpp_source` that contains superfluous filesets. For example, you might have loaded all eight AIX install CDs into the `lpp_source` and yet really do not want international language support other than your own language. Or, if continuing to download ML updates into your `lpp_source`, you could have superseded fixes that are no longer needed. Again, a cleanup would help to save some space and some time when this resource is used. If desired, you can choose to save off your cleaned up filesets by setting the `save removed files` field to `yes` as shown by Example 2-31. For example, `smitty nim --> Perform NIM Software Installation and Maintenance Tasks --> Software Maintenance and Utilities --> Eliminate Unnecessary Software Images in an lpp_source --> Select Target lpp_source.`

*Example 2-31 Example of SMIT panel for an lppmgr action*

Eliminate Unnecessary Software Images in an lpp\_source

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

	[Entry Fields]	
TARGET lpp_source	LPP_53_ML4	
PREVIEW only?	yes	+
REMOVE DUPLICATE software	yes	+
REMOVE SUPERSEDED updates	yes	+
REMOVE LANGUAGE software	yes	+
PRESERVE language	[en_US]	
REMOVE NON-SIMAGES software	no	+
SAVE removed files	no	+
DIRECTORY for storing saved files	[]	
EXTEND filesystems if space needed?	yes	+

► **Creating resources simultaneously**

Previously, when NIM ran a process that calculated and allocated file system space, such as creating a SPOT, lpp\_source and mksysb resource, no other similar operation could take place at the same time. Because these operations calculate free space and enlarge the size of a file system, NIM master was limited to one of these operation at a time. Starting in AIX 5L V5.2, you can use NIM to simultaneously create multiple lpp-source and mksysb resources in separate file systems on the same server.

**Note:** You cannot simultaneously create multiple SPOT resources (creating a SPOT requires an installp process, and there can be only one such instance at a time), but you can simultaneously create a SPOT, lpp\_source and mksysb resources.

The locking mechanism is set for each file system instead of each server. However, if you know that you have enough space in a file system to create multiple resources simultaneously, you can use the force option (-F flag) to prevent the locking mechanism from being used.

► **Resource groups**

This allows you to create a group of resources and specify clients (or a group of clients) as default. Previously, every time a NIM operation occurred, the NIM resources had to be specified to be allocated manually. A default resource group can be created containing the required resources, so when a NIM operation is performed, the resources are already associated to the client (or the group of clients) by the default resource group.

- ▶ **Alternate disk migration installation**  
This allows the user to create a copy of rootvg to a free disk and simultaneously migrate it through NIM to a new release level. Using alternate disk migration installation over a conventional migration provides several advantages:
  - Less down time
  - Quick recovery in case of migration failure
  - High degree of flexibility and customization
  - Easy Configuration Utilities
- ▶ **EZNIM**  
The SMIT EZNIM menu helps the system administrator by organizing the commonly used NIM operations and simplifies frequently used advanced NIM operations
- ▶ **nim\_master\_setup**  
The **nim\_master\_setup** command does the basic steps which are further detailed in 3.1.2, “Setting up the NIM environment” on page 56. It installs the bos.sysmgmt.nim.master fileset, configures the NIM master and creates the resources for installation (SPOT, lpp\_source), optionally including a system backup resource (mksysb). The **nim\_master\_setup** command uses the rootvg volume group and creates an /export/nim file system by default. You can change these defaults using the **volume\_group** and **file\_system** options.  
  
Example 2-32 shows the options available with the **nim\_master\_setup** command.

---

*Example 2-32 Example of nim\_master\_setup*

---

```
# nim_master_setup
Usage nim_master_setup: Setup and configure NIM master.
nim_master_setup [-a mk_resource={yes|no}]
[-a file_system=fs_name]
[-a volume_group=vg_name]
[-a disk=disk_name]
[-a device=device]
[-B] [-v]
-B Do not create mksysb resource.
-v Enable debug output.
Default values:
mk_resource = yes
file_system = /export/nim
volume_group = rootvg
device = /dev/cd0
```

---

- ▶ **nim\_clients\_setup**

The `nim_clients_setup` command as shown in Example 2-33, is used to define your NIM clients, allocate the installation resources, and initiate a NIM BOS installation on the clients (see details in section 4.3.2, “Performing a BOS installation” on page 151).

*Example 2-33 Example of nim\_clients\_setup*

---

Usage `nim_clients_setup`: Setup and Initialize BOS install for NIM clients.

```
nim_clients_setup [-m mksysb_resource]
[-c] [-r] [-v] client_objects
```

-m specify mksysb resource object name -OR- absolute file path.

-c define client objects from `client.defs` file.

-r reboot client objects for BOS install.

-v Enables debug output.

---

**Note:** If no client object names are given, then all clients in the NIM environment are enabled for BOS installation. If the (-c) flag is used the newly created LPARs are initialized for install, otherwise specified clients or all existing LPARs are initialized for install.

## 2.2.3 NIM improvements with AIX 5L V5.3

This section gives general information on the improvements coming with AIX 5L V5.3. Among the improvements, *nimsh* and *alternate NIM master* are of particular importance:

- ▶ `nimsh`

By default, the NIM environment uses `rsh` to communicate with your client when performing push operations, which means using the `~/.rhosts` file which may not be acceptable (as it is considered not to be secure) in certain environments.

NIM in AIX 5L V5.3 allows an alternate communication method using `nimsh`. Out of the box, `nimsh` is also considered semi-secure, as it requires root privileges and uses the NIM master and client `cpu_id` to authenticate. If you require stronger security, you have the option to implement SSL with `nimsh`. See 5.3, “NIMSH, OpenSSL and firewall considerations” on page 433 for further details.

- ▶ Alternate NIM master

NIM in AIX 5L V5.3 introduces a facility to do have a backup NIM master to take-over if your primary NIM master suffers a disaster. Section “High Availability NIM (HA NIM)” on page 124 gives details on how to setup an alternate NIM master.

- ▶ Detailed output when creating a NIM lpp\_source resource  
Before AIX 5L V5.3, the NIM lpp\_source creation did not give any information about the progress of the procedure. AIX 5L V5.3 extends the NIM lpp\_source creation adding verbose output. There is a new attribute (show\_progress) which has been introduced to the **nim command**. The NIM administrator can modify it to turn the verbose attribute on or off. Example 2-34 shows how to turn on this attribute.

*Example 2-34 Example to turn on the show\_progress attribute*

---

```
nim -o define -t lpp_source ... -a show_progress=yes
```

---

- ▶ Creating a SPOT resource from a mksysb  
Before AIX 5L V5.3, the NIM SPOT resource could be created from an existing lpp\_source resource or from the installation media. The disadvantage of such generic SPOT is the amount of disk space it consumes and the long creation time.

**Note:** A SPOT created from a mksysb can only be used for that mksysb.

The SPOT is created from the mksysb using the /usr/lib/bootpkg/bootpkg\_list file that lists only the necessary files for the SPOT. The size of the SPOT created using this method is approximately 50 MB while the size of the SPOT created from the lpp\_source is about 400 MB.

- ▶ **nim\_move\_up** command and operation  
The **nim\_move\_up** command enables users of existing AIX environments to take advantage of the capabilities available on new hardware (namely POWER5 servers or later). The command provides an interface that migrates an existing AIX system onto an LPAR residing on a POWER5 (or later) server. The level of AIX on the original machine is raised to a level that supports operation on POWER5 hardware. The original system's hardware resources are closely replicated on the equivalent POWER5 hardware. By the end of the migration, the same system is fully running on a POWER5 LPAR. In addition, nim\_move\_up can use the Virtual I/O capabilities of POWER5 servers by optionally migrating a client onto virtualized hardware, such as virtual disks and virtual Ethernet.
- ▶ mksysb migration using NIM  
mksysb migration allows you to restore the mksysb from an old system to a system that supports AIX® 5.3 and then migrate the mksysb. Traditional migration moves the operating system of a supported hardware configuration to a newer level. A mksysb migration installation is the recommended method of installation to move unsupported hardware configurations running AIX 4.3 and later to new supported hardware running AIX 5.3.

A mksysb migration is not intended for systems that you can migrate with a traditional migration. This method allows you to bypass the hardware limitation by restoring the mksysb on the new hardware configuration and migrate it without running AIX 4.3. The resulting system will be running the new level of AIX.

- ▶ Restore the SPOT copy function  
In AIX 5L V5.3, the restore SPOT copy function was reintroduced. The administrator can select the SPOT residing on the NIM server to install the BOS on the clients.
- ▶ NIM and Service Update Management Assistant (SUMA); for details, see 4.10, “NIM and Service Update Management Assistant” on page 321.
- ▶ NIM ability to restore volume group other than the rootvg to clients using the `nim -o restvg` operation.
- ▶ EZNIM enhancements:
  - Creates an `lpp_source` resource when setting up the master.
  - Adds `nimsh` and cryptographic authentication option when setting up the master and reinstalling the clients.
  - Adds an `rte` option to be selected for client reinstall.
  - Adds install options of overwrite, preservation, and migration for client reinstall.
  - Adds new options for viewing the NIM environment.

## 2.3 Planning for your NIM environment

Successfully implementing a reliable and efficient NIM environment highly depends on the first step of this endeavour, which is planning. Performing adequate planning for your NIM environment will most likely save you from needing to rebuilding your NIM environment in case of unsatisfactory results.

Planning of your NIM environment involves the following tasks:

- ▶ Choosing your NIM master
- ▶ Defining standard and consistent naming topologies
- ▶ Sizing your NIM master
- ▶ Designing for availability

## 2.3.1 Choosing your NIM master

The first step in the planning process is choosing the NIM master. A correct decision when choosing your NIM master highly contributes to increasing the resiliency of your entire environment. Why?

The answer to this question is directly related to the answer of “What is the purpose of implementing NIM in your environment, and what is the main role of the NIM master?”. From your experience with NIM, perhaps you found that you spent more time with the NIM master performing upgrades and system backups, than the initial install of the servers. The NIM master is usually used in servers’ bare-metal-restore operations in case of major failures or disaster recovery (DR) scenarios.

The following tips would help you making the optimal choice of the NIM master:

- ▶ In environments where NIM is needed, usually there exists a network backup server, such as IBM Tivoli Storage Manager (ITSM) server. Consolidating the backup and recovery functionality on one server has its obvious benefits. Therefore, we recommend to install the NIM master on the same machine or LPAR as the TSM server.
- ▶ The NIM master needs to be installed with the highest AIX version in your environment, and must be maintained at the latest AIX level. This may represent a constraint to installing the NIM master on the same machine/LPAR as the TSM server.
- ▶ The NIM master must not be installed on a production machine/LPAR, a LPAR cannot be a NIM master and a NIM client at the same time because of AIX level constraints.
- ▶ We recommend that the NIM master to be installed on a separate physical machine from your production environment. The same recommendation would apply to the TSM server.
- ▶ The NIM master needs to have reliable IP connectivity to all NIM clients.
- ▶ The NIM master needs to be implemented in a highly secure zone because, most likely, it will contain the system backup images of your mission critical business application servers.
- ▶ We recommend to use the NIM master to also store backup images of the NIM clients’ volume groups. These images need to be created with appropriate exclude files. Files excluded from backup may include temporary or unnecessary files.
- ▶ The NIM master need to have access to adequate disk storage with adequate scalability.

## 2.3.2 Defining a consistent naming convention

Defining and adhering to a consistent naming convention contributes to increasing the efficiency of the NIM environment management.

We highly recommend that you plan for and define a standard naming topology for:

- ▶ The NIM directory structure
- ▶ The NIM resources

We recommend to define short names, yet they should be indicative.

For the NIM directory structure, five main NIM-specific file systems are needed, which are to contain NIM resources directories. Two suggestions are proposed for the NIM directory and file system structure.

The first one, shown in Figure 2-6 on page 46, has separate file systems for each different AIX maintenance level or technology level which include the related `lpp_source` and `SPOT` directories. Beside those file systems, there is one to store the `mksysb` images, another one for additional software packages and the last one for resources related to small files like `bosinst_data`, `script`, `image_data` and so forth. Refer to Table 2-2 on page 27 for an estimate size.

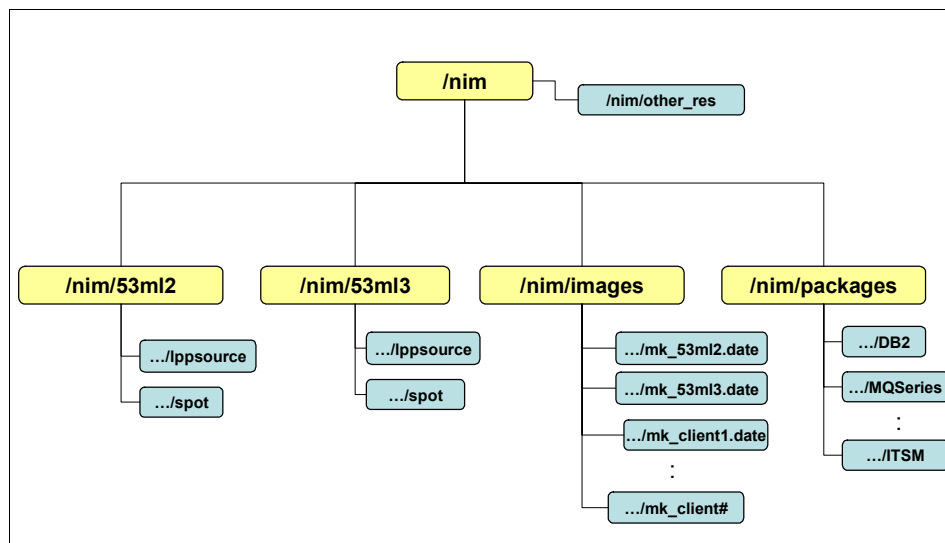


Figure 2-6 NIM directory structure based on PSSP

The second one, shown in Figure 2-7 on page 47, has a different structure: there is one file system for the `lpp_source` resources and one for the `SPOT` resources,



each of them containing the different AIX maintenance levels or technology levels. For the rest, same organization: one file system to store the mksysb images, another one for additional software packages and the last one for resources related to small files like bosinst\_data, script, image\_data and so forth. Refer to Table 2-2 on page 27 for an estimate size.

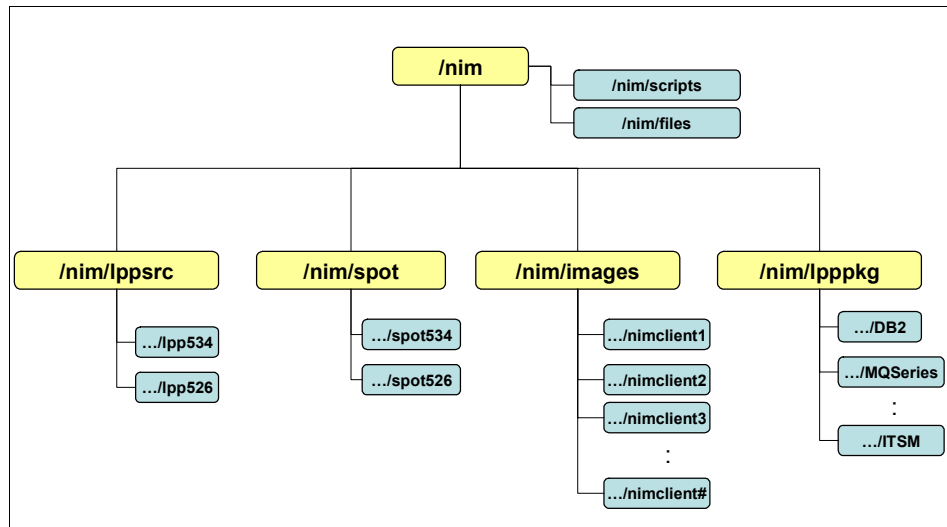


Figure 2-7 Other NIM directory structure.

**Tip:** For naming convention, the following general rule can be applied: all the NIM object names are in upper case and all the AIX names (host names, files, or directories names related to resources) are in lower case. This helps you to quickly identify if something has to be changed in either AIX (in case of lower case names) or NIM itself (in case of upper case).

Of course, there's also a third option: create a large file system (/nim) on a separate volume group dedicated entirely to NIM.

### 2.3.3 Sizing your NIM master

The NIM master operations do not require extensive CPU and memory resources. This is one of the reasons why we recommend to consider consolidating it with your TSM network backup server on the same server (LPAR).

As for the I/O sizing (disk, network), the NIM master would require an adequate I/O bandwidth for performing regular system backups of the NIM clients and

various other administration operations. From our experience, we found that nearly all fibre channel host bus adapters (HBAs) are under utilized. Therefore, most likely you do not need to worry about the HBAs.

The network interfaces situation could be different. Dual Gigabit Ethernet interfaces configured in an Etherchannel would be a good start for your combined TSM and NIM servers.

The disk space must be calculated upfront. Although NIM environments vary so much, Table 2-2 on page 27 provides some guidelines for planning disk space for your NIM master.

### 2.3.4 Designing for availability

Due to the nature of their operation, NIM master and/or resource servers are generally not considered a single point of failure for the critical business operations. In most cases, the business can continue to run while the NIM master and/or resource servers are being recovered, provided that they can be recovered before the next system backup cycle or so.

For TSM though, our experience is different. We have implemented high available (HA) clustered TSM servers using IBM HACMP product for good business reasons. Combining the NIM master and the TSM server on the same system could make a stronger business case to implement a HA NIM master and TSM server.

A reasonable degree of high availability can be implemented within a single NIM master instance. This can be achieved by implementing the following:

- ▶ Multiple (two or more) Ethernet interfaces configured in an Etherchannel. This will not only provide a seamless recovery from a network interface failure, but also will multiply the bandwidth available.
- ▶ Multiple fibre channel HBAs (even number of HBAs recommended).
- ▶ Implement a RAID5 configuration at the disk storage level or use AIX mirroring.
- ▶ Extra care needs to be taken if your NIM Master is using Virtual SCSI disks. In the case of a disaster, the Virtual I/O Server may need to be recovered before you can recover your NIM master but you might want to recover the Virtual I/O Server using the NIM Server. A classic "catch 22" dependency problem.

## 3

# Basic configuration of a Network Installation Manager (NIM) environment

This chapter contains the following topics:

- ▶ “Setting up a basic NIM environment”, covering the following:
  - name resolution order
  - bootp file format
  - NIM environment variables
  - NIM server options
  - NIM database
  - NIM operational log files
- ▶ Using SMIT step by step to setup a NIM master and install a client
- ▶ Using Web-Based System Manager to setup a NIM master and install a client
- ▶ Using EZNIM scripts to setup a NIM master and install a client
- ▶ Using OS\_install command (wrapper) to install a client

For more information on configuring NIM, please refer to the manual *AIX 5L Version 5.3 Installation and migration*, SC23-4887-03.

## 3.1 Setting up a basic NIM environment

Setting up a NIM master and bringing it to use is very simple and easy. We recommend following these steps:

- ▶ Plan (the NIM environment)
- ▶ Setup (implementing the plan)
- ▶ Maintain (the installed environment)

We start explaining how to set up a basic NIM environment using the command line interface (CLI). Next we show the same steps through more user friendly interfaces, such as `nim_master_setup` command, the EZNIM and the Web-Based System Management (WSM).

Also, if you just want to install both AIX and Linux clients, you can use the `OS_install` command introduced with AIX 5L V5.3 technology level 5 (TL5).

### 3.1.1 Planning the NIM environment

In 2.3, “Planning for your NIM environment” on page 44 you can find detailed information on planning the NIM environment, and in Chapter 4, “Network Installation Manager (NIM) scenarios” on page 123 details on how to use NIM to maintain and manage your AIX/Linux software environment.

In this section we will only describe a simple design for a basic NIM environment, with the NIM master serving also as resource server for all resources.

#### Purpose

The purpose of the basic NIM environment is to be able to install the operating system on System p logical partitions (LPARs) and physical machines using one of the following install methods:

- fileset based installation (Run-Time Environment - `bos rte install`)
- system backup based installation, by restoring a system image created using the AIX backup procedure (`mksysb`).

#### System resources

The first step to set up a new NIM environment is to obtain the appropriate computer resources needed. And first there will have to be a NIM server (master) and the network environment (physical connectivity, TCP/IP configuration).

The NIM master in a LPAR environment can be a small LPAR (using dedicated or shared resources, depending on requirements and the available System p

architecture); network and disk resources can be made available to the NIM server via a Virtual I/O Server.

The same NIM master can serve NIM clients residing outside the physical machine (System p) where it is allocated using a VIO server virtual network interface, as well as NIM clients within the same physical machine using a hypervisor virtual LAN (VLAN).

The NIM server (as described in Chapter 2., “Network Installation Manager (NIM) definitions and concepts” on page 7) needs at least one network interface and, depending on the planned storage repository, plenty of storage space. The amount of storage space, even for a basic NIM environment, depends on how many versions of base AIX filesets you need to copy from AIX CDs/DVDs, and the number of downloaded Technology Levels (TL), Service Packs (SP), Concluding Service Packs (CSP) and additional Program Temporary Fixes (PTF).

**Note:** In the basic NIM environment described here we do not take into account additional storage space for other installable application software (such as DB2®, Websphere Application Server, MQ etc.), as well as storage space that would be needed for NIM client system backups using NIM mksysb operation.

### File system hierarchy

For our basic NIM environment, the storage utilization will benefit from a structured file system hierarchy. It is up to you to decide which structure you want. In 2.3.2, “Defining a consistent naming convention” on page 46 and throughout the book, we present several ways to name and structure different parts of the NIM environment.

**Note:** There is no “golden rule” for naming your directory structure and resources, but the purpose of the file structure is the same: to make it easier to find, use, manage, and minimize storage redundancy in the file system storage used for the NIM environment.

Also, keep in mind that certain directory names should be kept as default (/tftpboot, for example). However, we recommend that you use a naming convention relevant and easy to understand in your environment.

For the basic NIM environment you can use the following file system hierarchy:

/tftpboot	Is used for storing NIM master boot images (kernels) and info files for the NIM clients.
-----------	--

<code>/export/lpp_source</code>	Used for storing versions of AIX base level filesets in specific directories; a NIM <code>lpp_source</code> resource represents a directory in which software installation images are stored (in BFF or RPM format). In our basic NIM environment a minimum of one base level and one base level with applied TL and CSP for the “Maximum Stability” maintenance model (see 4.10.4, “Maintenance models” on page 334).
<code>/export/spot</code>	Used for storing non- <code>/usr</code> Shared Product Object Three (SPOT) in individual directories (one per SPOT version). A SPOT is required to install or initialize all types of machine configurations, as well as for generating a specific kernel image used for network booting a client system. Everything that a client machine requires in a <code>/usr</code> file system, such as the AIX kernel, executable commands, libraries, and applications should be included in the SPOT.
<code>/export/images</code>	Used for storing system backup images. These images can be created using NIM <code>mksysb</code> operation (“pulling” a <code>mksysb</code> image from a NIM client), or copying a system backup as a file from a previously created backup file, tape, or DVD. To create subdirectories during NIM <code>mksysb</code> creation, you must export the <code>NIM_MKSYSB_SUBDIRS=yes</code> variables before using the <code>nim -o define -t mksysb</code> command. See 4.11, “Backing up clients with NIM” on page 353.

Plan for using relevant *naming convention* to keep track of all NIM resources. In our example in this section we use a straight forward short coded UNIX style variant, whereas in 2.3.2, “Defining a consistent naming convention” on page 46 and in “Directory structure setup” on page 293 we present other naming convention examples.

**Note:** The `/export/nim/scripts` directory is used for the `nim_script` resource that is managed by NIM, and should not be used to store other files or scripts.

### ***Basic naming convention for the NIM `lpp_source` objects***

The NIM `lpp_source` objects can be named in the same way for both the directory and the NIM `lpp_source` object.

If an `lpp_source` only contains update filesets, the directory name is prefixed to reflect the type of update. If the `lpp_source` directory contains all the filesets needed to perform basic operating system (`rte`) installation (the `lpp_source`’s

s image attribute is set to “yes”), then the directory name is prefixed with “lpp” (Licensed Program Products). The following list presents the naming convention method we used in our test:

lpp<OS#>00	General Availability (GA) level for AIX version <OS#>
lpp<OS#><TL#>	GA base <OS#> with Technology Level <TL#>
lpp<OS#><TL#><SP#>	GA base <OS#> with Technology Level <TL#> and Service Pack <SP#>
lpp<OS#><TL#>CSP	GA base <OS#> with Technology Level <TL#> and its Concluding Service Pack
TL<OS#><TL#>	Technology Level <TL#> for <OS#>
SP<OS#><SP#>	Service Pack <SP#> for <OS#>
CSP<OS#><CSP#>	Concluding Service Pack <CSP#> for <OS#>

**Note:** The AIX level is shown by the `oslevel` command as four digits, for example AIX 5.3 GA base level are shown as “5300”. The two trailing digits are always zero (0) and can be omitted. The two leading digits represent the <OS#> in this naming scheme. However, to make the GA base level name stand out, you can keep all four digits, you could suffix the name with two zeroes (“00”), thus `lpp53` and `spot53` would then show as `lpp5300`, and `spot5300` respectively.

For each AIX version you must create one directory in `/export/lpp_source`, for example `/export/lpp_source/lpp53` or `/export/lpp_source/lpp5300` for the base level of AIX 5.3.

### ***Basic naming convention for the NIM spot objects***

The NIM spot objects can be named in a similar fashion, and based on the numbering for the corresponding NIM `lpp_source`, for both the directory and the NIM spot object.

spot<OS#>00	General Availability (GA) level for AIX version <OS#>
spot<OS#><TL#>	GA base <OS#> with Technology Level <TL#>
spot<OS#><TL#><SP#>	GA base <OS#> with Technology Level <TL#> and Service Pack <SP#>
spot<OS#><TL#>CSP	GA base <OS#> with Technology Level <TL#> and its Concluding Service Pack

For each AIX version plan to create one directory in `/export/spot`, such as `/export/spot/spot53` or `/export/spot/spot5300` for the base level of AIX 5.3, which is created from the corresponding NIM `lpp_source`, in this case the `/export/lpp_source/lpp53` or `/export/lpp_source/lpp5300`.

**Note:** We recommend that you pair the `lpp_source` and the SPOT versions and levels. For installing a new system, or for upgrading/migrating, you can also use SPOT resources at a lower level than the `lpp_source`.

However, you should check always the software levels, and *never* use an `lpp_source` at a lower level than the SPOT.

### **Basic naming convention for the NIM mksysb objects**

For *generic* NIM mksysb images, with general pre-customization, the NIM mksysb objects and image filenames can be named in the same way as the NIM `lpp_source` and NIM spot previously described:

```
mksysb<OS#>00      General Availability (GA) level for AIX version <OS#>
mksysb<OS#><TL#>   GA base <OS#> with Technology Level <TL#>
mksysb<OS#><TL#><SP#> GA base <OS#> with Technology Level <TL#> and
                    Service Pack <SP#>
mksysb<OS#><TL#>CSP GA base <OS#> with Technology Level <TL#> and its
                    Concluding Service Pack
```

For *particular* LPARs or physical machines, `mksysb image` files can be named in a similar way (when you plan to use these images only on the same system they were created from), using the hostname and the date when the image are created either as a prefix/suffix. The corresponding NIM object name could consist of the hostname simply suffixed or prefixed with the word “mksysb” or “mk”.

If the specific NIM mksysb objects are created only for certain maintenance tasks (when needed, and then removed), it is not necessary to include time stamp information, such as the current date, but this can be helpful, as shown in the following are examples:

```
mk5305_lpar55_060618 => /export/images/lpar55/mk5305_lpar55_060618
mksysb_lpar55 =>      /export/images/lpar55/lpar55_5305_18JUN06
```

In this case, as shown in Example 3-1, the environment variable `NIM_MKSYSB_SUBDIRS` is used on the NIM master for storing system backups. When this variable is set to “yes”, subdirectories are used to separate NIM mksysb image files created from client machines using the `nim command`. The subdirectories are transparent to the user, but they provide separate locations for NFS exporting. The subdirectories are named after the NIM `machines` object for which they are created (the NIM name used for the source attribute).



*Example 3-1 Using the NIM\_MKSYSB\_SUBDIRS environment variable*

```

root@master:/: export NIM_MKSYSB_SUBDIRS=yes
root@master:/: nim -o define -t mksysb -a server=master -a source=LPAR55
-a mk_image=yes -a location=/export/images/lpar55_5305_18JUN06 mksysb_lpar55

```

The **nim** command will create the mksysb image file lpar55\_5305\_18JUN06 in the /export/images/LPAR55 directory and not directly in the /export/images directory.

**Note:** For additional information on mksysb image creation and usage, see 4.11, “Backing up clients with NIM” on page 353.

### Basic naming convention for the NIM network objects

In order to perform certain NIM operations, the NIM master must be able to supply information necessary to configure client network interfaces. The NIM master must also be able to verify that client machines can access all the resources required to support operations. To avoid the overhead of repeatedly specifying network information for each individual client, NIM networks are used to represent the networks in a NIM environment.

The NIM network objects are named by default as net#, where # is a number starting from 1. This might be sufficient for most installations which only utilize one IP network. But when one NIM master is used to support several networks, the NIM network objects can be named so that the IP network is included in the name, to easier identify different networks in the environment:

```

net_<IP#NET>      Network with the IP network address included in the name
                  in decimal form with underscore (_) replacing dots (.)
                  since dots are not allowed to use in NIM object names.

```

If the network has a 24-bit mask (FF.FF.FF.0, C-class network), the IP network address would be the three first octets, such as 10.1.1, the NIM networks name could then be net\_10\_1\_1, or net\_10\_1\_1\_0. The last 0 to indicate the lowest number on the IP network, which can be useful if the NIM master will serve several IP networks. Table 3-1 shows how this could be achieved by using the lowest number on the IP network.

Table 3-1 NIM networks names for IP networks using different netmasks

Netmask (decimal dot notation)	NIM networks name
255.255.255.0	net_10_1_1_0
255.255.255.128	net_10_1_1_0, net_10_1_1_128

Netmask (decimal dot notation)	NIM networks name
255.255.255.192	net_10_1_1_0, net_10_1_1_64, net_10_1_1_128, . . .
255.255.255.224	net_10_1_1_0, net_10_1_1_32, net_10_1_1_64, . . .
255.255.255.240	net_10_1_1_0, net_10_1_1_16, net_10_1_1_32, . . .
255.255.255.248	net_10_1_1_0, net_10_1_1_8, net_10_1_1_16, . . .
255.255.255.252	net_10_1_1_0, net_10_1_1_4, net_10_1_1_8, . . .

To set up a NIM network object, the essential information is the IP network address (`net_addr`) and the mask (`snm`). Example 3-2 shows a NIM network object.

*Example 3-2 Using `lsnim` to display NIM network objects*

---

```

root@master:/: lsnim -l -c networks
net_10_1_1:
  class      = networks
  type       = ent
  Nstate     = ready for use
  prev_state = information is missing from this object's definition
  net_addr   = 10.1.1.0
  snm        = 255.255.255.0

```

---

The `prev_state` (Previous State) message can be ignored since this is a newly created definition.

### 3.1.2 Setting up the NIM environment

The setup of a NIM environment consists of two major steps:

- ▶ Setup of the NIM master itself and basic master customization
- ▶ Setting up the resources and object definitions for at least one `lpp_source` and one spot resource.

## Basic setup of the NIM master

Install and configure the designated system for the NIM master (LPAR or physical machine), with the latest AIX version and Technology Level (TL) with appropriate Service Packs (SP) (see 4.10.7, “IBM AIX 5L service strategy” on page 349).

**Important:** The NIM master should always have the highest AIX version of the of all machines defined (NIM clients).

### *Install the bos.sysmgt.nim.master fileset*

After initial BOS installation, install the NIM master software from the `bos.sysmgt.nim.master` fileset.

### *Create the file systems used as repository for the NIM master*

Create the file systems for the NIM master as Enhanced Journaled File System (JFS2). The sizes (see Example 3-3) have to be calculated depending on your own NIM environment requirements.

#### *Example 3-3 Creating basic NIM environment master file systems*

---

```
root@master:/: crfs -v jfs2 -g rootvg -a size=64M -m /tftpboot -A yes -p rw -a logname=INLINE
root@master:/: crfs -v jfs2 -g rootvg -a size=4G -m /export/lpp_source -A yes -p rw -a logname=INLINE
root@master:/: crfs -v jfs2 -g rootvg -a size=1G -m /export/spot -A yes -p rw -a logname=INLINE
root@master:/: crfs -v jfs2 -g rootvg -a size=6G -m /export/images -A yes -p rw -a logname=INLINE
```

---

After creating the file systems, they should be mounted and appropriate permissions set, such as 600 (only the owner have read and write permission).

### *Initialize the NIM master*

Now it is time to initialize the NIM master. Note the naming of the `netname` attribute. Here we will use `nimconfig` command, but you can also do this step with the SMIT `nimconfig` fast path.

### *Configuring the NIM master using the nimconfig command*

This creates the master, boot, the first network object, `nim_script` with the `/export/nim` directory, and the master (`nimesis`) daemon is started. In Example 3-4 we do not allow NIM clients to register themselves with the NIM master. The `netname` attribute value is selected using the IP network address, see “Basic naming convention for the NIM `lpp_source` objects” on page 52.

#### *Example 3-4 NIM master initialization*

---

```
root@master:/: nimconfig -a netname=net_10_1_1 -a pif_name=en0 -a netboot_kernel=mp
-a cable_type=tp -a client_reg=no
```

0513-071 The `nimesis` Subsystem has been added.

0513-071 The nimd Subsystem has been added.

0513-059 The nimesis Subsystem has been started. Subsystem PID is 491636.

If you plan to configure a high availability NIM environment (multiple NIM servers), use “master\_net” as *netname*. This is automatically set by the **niminit** command when the attribute *is\_alternate* is set to “yes”, see 4.1, “High Availability NIM (HA NIM)” on page 124.

Figure 3-1 shows the SMIT *nimconfig* fast path panel.

```

Configure Network Installation Management Master Fileset

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

                                     [Entry Fields]
* Network Name                       []
* Primary Network Install Interface   []          +

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

Esc+1=Help      Esc+2=Refresh      Esc+3=Cancel      Esc+4=List
Esc+5=Reset     Esc+6=Command      Esc+7=Edit       Esc+8=Image
Esc+9=Shell     Esc+0=Exit          Enter=Do

```

Figure 3-1 SMIT *nimconfig* fast path

Figure 3-2 presents the SMIT panel shown in Figure 3-1, after entering the values.

Configure Network Installation Management Master Fileset			
Type or select values in entry fields. Press Enter AFTER making all desired changes.			
	[Entry Fields]		
* Network Name	[net_10_1_1]		
* Primary Network Install Interface	[en0]	+	
Allow Machines to Register Themselves as Clients?	[no]	+	
Alternate Port Numbers for Network Communications (reserved values will be used if left blank)			
Client Registration	<input type="checkbox"/>	#	
Client Communications	<input type="checkbox"/>	#	
Esc+1=Help	Esc+2=Refresh	Esc+3=Cancel	Esc+4=List
Esc+5=Reset	Esc+6=Command	Esc+7=Edit	Esc+8=Image
Esc+9=Shell	Esc+0=Exit	Enter=Do	

Figure 3-2 SMIT nimconfig fast path with input

After the initial setup is performed, you can use the `lsnim` command to view the newly created NIM objects, as shown in Example 3-5.

Example 3-5 The basic NIM master objects

---

```
root@master:/: lsnim
master      machines      master
boot       resources    boot
nim_script  resources     nim_script
net_10_1_1 networks     ent
```

---

The first object of the machine type is the “master” itself. The boot resource points to the `/tftpboot` directory and the `nim_script` resource points to the `/export/nim/scripts` directory, as shown in Example 3-6. The networks resource contains the IP information network interface on the NIM master, as shown in Example 3-2 on page 56.

Example 3-6 The basic NIM master objects

---

```
root@master:/: lsnim -l boot
boot:
  class      = resources
  type       = boot
  comments   = represents the network boot resource
  Rstate     = ready for use
  location   = /tftpboot
  alloc_count = 0
  server     = master
```

```

    reserved    = yes
root@master:/: lsnim -l nim_script
nim_script:
  class       = resources
  type        = nim_script
  comments    = directory for customization scripts created by NIM
  Rstate      = ready for use
  location    = /export/nim/scripts
  alloc_count = 0
  server      = master
  reserved    = yes

```

---

The NIM master configuration is stored in Object Data Manager (ODM) classes in `/etc/objrepos`, the `nim_attr`, `nim_patrr` and `nim_object` files.

When a machine is added to a NIM environment (both masters and client are machines), the `/etc/niminfo` file is created. Example 3-7 shows the contents on the NIM master.

*Example 3-7 A NIM masters `/etc/niminfo` file*

---

```

root@master:/: cat /etc/niminfo
export NIM_NAME=master
export NIM_CONFIGURATION=master
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_MASTER_HOSTNAME=master

```

---

The value of the `NIM_CONFIGURATION` variable shows that this is a NIM master. For standard clients the value will be `standalone`. The value of the `NIM_MASTER_PORT` variable represents the TCP port on which the `nimesis` daemon listens. The `NIM_REGISTRATION_PORT` (TCP port) will be used for NIM client registration, if this is enabled.

Example 3-8 you can observe the difference between a NIM master's and a NIM client's `/etc/niminfo` file. The information in the file is primarily used during NIM BOS installation. The main fields that are used for `nimclient` calls are `NIM_MASTER_HOSTNAME`, `NIM_MASTER_PORT`, and `NIM_REGISTRATION_PORT`. The client's NIM name in this example is `LPAR55` and its hostname is `lpar55` (IP address `10.1.1.55`). The NIM master's hostname is "master" and its IP address is `10.1.1.2`.

*Example 3-8 NIM client `/etc/niminfo` file*

---

```

#----- Network Install Manager -----
# warning - this file contains NIM configuration information

```

```
#      and should only be updated by NIM
export NIM_NAME=LPAR55
export NIM_HOSTNAME=lpar55
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=master
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_LICENSE_ACCEPT=yes
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_RECOVER="/../SPOT/usr/lpp/bos.sysmgmt/nim/methods/c_bosinst_env -a
hostname=lpar55"
export SPOT=master:/export/lpp_source/spot504/usr
export NIM_CUSTOM="/../SPOT/usr/lpp/bos.sysmgmt/nim/methods/c_script -a
location=master:/export/nim/scripts/LPAR55.script"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.55:lpar55 10.1.1.2:master "
export NIM_MOUNTS=" master:/export/lpp_source/spot504:/SPOT/usr/sys/inst.images:dir "
```

---

### Rebuilding and recovering a corrupt /etc/niminfo file

To rebuild the NIM master /etc/niminfo file use the **nimconfig** command on the master with the **-r** flag:

```
nimconfig -r
```

To rebuild and recover a NIM client /etc/niminfo file use the **niminit** command on the client:

```
niminit -a master=<MASTER_HOSTNAME> -a name=<CLIENT_NIM_NAME>
```

The NIM masters hostname or IP address should be specified and the NIM clients NIM name, *not* the NIM clients hostname or IP address. In Example 3-9, the NIM master is “master” and the NIM client is “lpar6”.

#### *Example 3-9 Initializing the NIM master*

---

```
root@lpar6:/: niminit -v -a master=master -a name=lpar6
```

```
niminit
Testing if we can access the Master @ master
Establishing contact with NIM master
Writing request to master
Building niminfo file
Granting master rhost permission
giving master rhost permissions
```

Adding routes from ninfo file  
 Adding entry to inittab

---

**Tip:** To convert an old NIM master into a client, you should:

- Create a client definition on the NIM master
- Remove the `/etc/ninfo` file (this will remove also the `nim` command from the system).
- Uninstall the `bos.sysmgt.nim.master` fileset
- Check if the `bos.sysmgt.nim.client` fileset is installed
- Use the `niminit` command to register with the new NIM master (client registration must be enabled for this procedure to work).

In Example 3-10, the NIM master's IP address is specified correctly, but the NIM clients name is spelled incorrect from what is stored in the NIM database.

*Example 3-10 Error message when recreating the NIM client /etc/ninfo file*

---

```
root@lpar55:/: niminit -a master=10.1.1.2 -a name=LpAr5500
0042-008 NIMkid: Request denied -
    The specified NIM master is not currently configured to
    define machines on your subnet. The machine must be defined
    from the master OR a network corresponding to your subnet must
    be defined on the master before you can add yourself as a client
    to this NIM environment.
```

---

**Tip:** The `/etc/NIM.level` file only contains the version of the `bos.sysmgt.nim.master` fileset, and can be recreated using the `lslpp` command:

```
lslpp -L bos.sysmgt.nim.master | awk '/master/{print $2}'
```

### **Setting up IP name resolution**

Host names (IP labels) used to identify the NIM clients are symbolic names and need to be resolved into IP addresses. This is done by the NIM master using a resolver subroutine (`gethostbyname()`). The method used by the set of resolver subroutines to resolve names depends on the local host configuration. In addition, the organization of the network determines how a resolver subroutine communicates with remote name server hosts (the hosts that resolve names for other hosts).

When creating a NIM machine object, the NIM object name are associated with a hostname. If the NIM master uses Domain Name Server (DNS) to resolve the host name to IP address mapping, the host name will be a "Fully Qualified



Domain Name” (FQDN). Other name resolution methods include flat files (/etc/hosts) and NIS (Network Information Services).

**Tip:** We recommend using local name resolution (/etc/hosts), as this will avoid NIM operation to be affected by DNS or NIS service availability.

To make sure that the NIM masters /etc/hosts file is used before any DNS, if the /etc/resolv.conf file exists, you can use several methods. First you can include the NSORDER variable in the /etc/environment file, or you can include a specification line in either the /etc/irs.conf or /etc/netsvc.conf files.

**Note:** The settings in the /etc/netsvc.conf configuration file override the settings in the /etc/irs.conf file. The NSORDER environment variable overrides the settings in the /etc/irs.conf and the /etc/netsvc.conf files.

In Figure 3-3, Figure 3-4, and Figure 3-5 the resolver subroutine (gethostbyname()) cannot find the host name in the /etc/hosts file and continues to search for the host name in DNS.

Add the following NSORDER variable declaration to /etc/environment to be effective after the next system reboot. All system software that starts at boot time, Initial Program Load (IPL), will have the /etc/environment variables set at that time. See Figure 3-3.

```
NSORDER=local,bind
```

*Figure 3-3 Example of NSORDER configuration*

The /etc/netsvc.conf and /etc/irs.conf files are used by the resolver routines as soon as the files exist. Refer to Figure 3-4.

```
hosts=local,bind
```

*Figure 3-4 Example of /etc/netsvc.conf configuration*

Notice the syntactical difference between the /etc/netsvc.conf and /etc/irs.conf files. See Figure 3-5

```
hosts local continue
hosts dns
```

Figure 3-5 Example of /etc/irs.conf configuration

### Verify auxiliary services (inetd, bootpd and tftpd)

The NIM master uses the **bootpd** (BOOT Protocol) and **tftpd** (Trivial File Transfer Protocol) daemons (services). Both services are normally started by the **inetd** daemon (InterNET master daemon).

**Note:** For additional information on troubleshooting, refer to Chapter 7, “Basic NIM problem determination and tuning” on page 575.

Check if the **inetd** daemon is running by using the **lssrc** command, as shown in Example 3-11, adding the **-ls** flags to the **lssrc** command also shows the services controlled by **inetd** and their status (in our case we are looking for **bootpd** and **tftpd**).

Example 3-11 Checking the inetd daemon with the lssrc command

```
root@msater:/: lssrc -ls inetd
```

Subsystem	Group	PID	Status
inetd	tcPIP	176238	active

Debug	Inactive
-------	----------

Signal	Purpose
SIGALRM	Establishes socket connections for failed services
SIGHUP	Rereads configuration database and reconfigures services
SIGCHLD	Restarts service in case the service dies abnormally

Service	Command	Arguments	Status
<b>tftpd</b>	<b>/usr/sbin/tftpd</b>	<b>tftpd -n</b>	<b>active</b>
<b>bootps</b>	<b>/usr/sbin/bootpd</b>	<b>bootpd /etc/bootptab</b>	<b>active</b>
xmquery	/usr/bin/xmtopas	xmtopas -p3	active
time	internal		active
daytime	internal		active
time	internal		active
daytime	internal		active
ntalk	/usr/sbin/talkd	talkd	active
exec	/usr/sbin/rexecd	rexecd	active
login	/usr/sbin/rlogind	rlogind	active

shell	/usr/sbin/rshd	rshd	active
telnet	/usr/sbin/telnetd	telnetd -a	active
ftp	/usr/sbin/ftpd	ftpd	active

---

### ***The bootpd daemon***

Check that the `/etc/inetd.conf` file contains the line shown in Example 3-12 for the **bootpd** daemon.

*Example 3-12 Example of /etc/inetd.conf configuration for bootpd*

---

```
bootps dgram udp wait root /usr/sbin/bootpd bootpd /etc/bootptab
```

---

The corresponding line in the `/etc/services` file is for the service name to IP port mapping as shown in Example 3-13.

*Example 3-13 Example of /etc/services configuration for bootpd*

---

```
bootps 67/tcp # Bootstrap Protocol Server
```

---

The **bootpd** daemon will also use the `/etc/bootptab` file when a NIM client is configured to be booted from the NIM master as shown in Example 3-14.

*Example 3-14 /etc/bootptab configured for network boot of a NIM client*

---

```
lpar55:bf=/tftpboot/lpar55:ip=10.1.1.55:ht=ethernet:sa=10.1.1.2:sm=255.255.255.0:
```

---

The fields in the `/etc/bootptab` file are separated by a colon (`:`) and the attributes are separated from the assigned value by an equal sign (`=`). In the example above the attributes used by NIM are:

the first field	Hostname or full domain name (for the NIM client)
bf	Boot file name (in the NIM boot resource directory)
ip	Host IP address (for the NIM client)
ht	Hardware type
sa	TFTP server IP address for the boot file
sm	Subnet mask (for the NIM client)

Additional fields that can be used for NIM clients are:

gw	Gateway IP address (for the NIM client)
ha	Hardware address (MAC for the NIM client)

When the **inetd** daemon receives BOOTP requests, it will start the **bootpd** daemon. The **bootpd** daemon reads the `/etc/bootptab` configuration file, and, if the request matches one of the entries, passes the boot file information to the

client. The **bootpd** daemon will stay alive for a while, in case other BOOTP requests arrive, by default 15 minutes.

Each request will be logged to the SYSLOG subsystem, depending on the verbosity/debug level set with one or more `-d` flag(s) in the `/etc/inetd.conf` for the `bootps` services. By default, only a line indicating that the **bootpd daemon** has started will be logged. To activate “info” level SYSLOG messages, append `-d` to the `bootpd` service entry in `/etc/inetd.conf` and refresh **inetd**:

```
refresh -s inetd
```

*Example 3-15 Using the “-d” verbosity level for bootpd*

---

```
bootpd: reading "/etc/bootptab"
bootpd: read 1 entries from "/etc/bootptab"
bootpd: dumped 1 entries to "/etc/bootpd.dump".
bootpd: bootptab mtime is Fri Jun 16 11:38:47 2006
bootpd: Received boot request.
bootpd: request from IP addr 10.1.1.55
bootpd: found 10.1.1.55 lpar55
bootpd: bootfile = /tftpboot/lpar55
bootpd: vendor magic field is 99.130.83.99
bootpd: RFC1048 vendor data ( bp_vend[64] )
bootpd: sending RFC1048-style reply
bootpd: The following addresses are included in the bootp reply
bootpd: Client IP address (bp->bp_ciaddr) = 10.1.1.55
bootpd: Server IP address (bp->bp_siaddr) = 10.1.1.2
bootpd: Gateway IP address (bp->bp_giaddr) = 10.1.1.2
bootpd: Finished processing boot request.
```

---

In Example 3-15, you can see the request line, followed by a matching NIM created boot file (kernel). The reply to the client is sent including the Server and Gateway IP addresses. Sending the Client IP address might seem unnecessary, but it is also possible to only use the MAC address, without the client knowing about its IP address the first time it is activated for installation.

### ***The tftpd daemon***

Check that the `/etc/inetd.conf` file contains the `tftp` line, as shown in Example 3-16.

*Example 3-16 tftpd configuration in /etc/inetd.conf*

---

```
tftp dgram udp6 SRC nobody /usr/sbin/tftpd tftpd -n
```

---

Also, check the `/etc/services` file for the service name and the UDP port, as shown in Example 3-17.

*Example 3-17 tftpd definition in /etc/services*


---

```
tftp 69/udp # Trivial File Transfer
```

---

The **tftpd** daemon uses the `/etc/tftpaccess.ctl` file to determine which directory hierarchy is allowed to share.

Since the TFTP does not specify any authentication, it is mandatory to restrict the **tftpd** daemons file system access to the `/tftpboot` directory for NIM usage. The `/etc/tftpaccess.ctl` file should only contain the following line if you only use **tftpd** for NIM purposes as shown in Example 3-18.

*Example 3-18 Sample /etc/tftpaccess.ctl*


---

```
# NIM access for network boot
allow:/tftpboot
```

---

When the **inetd** daemon receives TFTP requests, it will start the **tftpd** daemon to service it, and start the transfer of the boot image file from the `/tftpboot` directory.

To check that the **tftpd** daemon is working, you can use the `tftp` command on the NIM master. Example 3-19 shows how to check the status of the **tftpd** daemon.

*Example 3-19 Checking tftpd connection*


---

```
root@master:/: echo "status"|tftp 0
tftp> Connected to 0.
Mode: netascii Verbose: off Tracing: off
Max-timeout: 25 seconds
tftp>root@master:/:
```

---

Example 3-20 shows how to display the contents of the `/tftpboot` directory.

*Example 3-20 Displaying the contents of the /tftpboot directory*


---

```
root@master:/: tftp -r - 0 /tftpboot
./../lost+found/spot5305.chrp./LPAR55.infoLPAR55Received 96 Bytes in
0.0 Seconds
```

---

Example 3-21 shows how to display the contents of the `/tftpboot/LPAR55.info` file.

*Example 3-21 Displaying the contents of a file using tftp*


---

```
root@master:/: tftp -o - 0 /tftpboot/LPAR55.info
#----- Network Install Manager -----
```

---

```
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=LPAR55
. . .
Received 1096 Bytes in 0.0 Seconds
```

---

If the `-v` flag is set in the `/etc/inetd.conf` for the `tftpd` service, for each request **tftpd** will send “info” level SYSLOG messages. However, the default behavior is NOT to send messages.

The `-v` flag brings detailed information, such as IP addresses, and file transfer information. Example 3-22 shows the messages logged when the “-vi” flags are used for the `tftp` service entry in `/etc/inetd.conf`.

*Example 3-22 syslogd output, using “-vi” verbosity level for tftpd*

---

```
tftpd: 10.1.1.55 RRQ <file=Received, mode=/tftpboot/lpar55, recognized options: octet>
tftpd: Status Read request for 10.1.1.55: /tftpboot/lpar55
tftpd: Status Transaction completed successfully
```

---

In Example 3-22, you can see the request line (RRQ), followed by a match for the request with the NIM created boot file. The file is transferred in binary/octet mode to the client.

### **Examining the NIM log files**

The `/var/adm/ras` directory contains the NIM master log files. To examine these files, you can use the **alog** command, as shown in Example 3-23. The `/usr/adm/ras/nimlog` file contains information about failed NIM operations. For additional information on NIM related log files, refer to Chapter 7., “Basic NIM problem determination and tuning” on page 575.

*Example 3-23 Using the alog command to view the nimlog file*

---

```
root@master:/: alog -f /usr/adm/ras/nimlog -o
```

---

## **Basic setup of the NIM master resources**

The basic NIM installation resources are: one NIM `lpp_source` and one `spot`. These are needed to perform a NIM `rte` installation. For NIM `mksysb` installation the basic installation resources are one NIM `mksysb` and one `spot`.

### **Create a base level (GA) NIM lpp\_source**

To create a NIM `lpp_source` object you need the GA base level AIX filesets. You can copy the AIX CD/DVD filesets to the local disk of the NIM master (or NIM resources server) in several ways.

The current standard for installation fileset directory structures requires images to be organized into subdirectories according to package type and architecture. For example, Backup File Format (BFF) fileset images reside in the `installp/ppc` directory. The **bfcreate** command is copying from a source containing this structure, the destination is required to conform to the same convention. Both the IBM AIX 5L installation CD/DVD and NIM `lpp_source` directories conform to this standard.

Copying AIX CD/DVD filesets can be done when creating the NIM `lpp_source` object using the source attribute pointing to the AIX base level installation CD/DVD, as in Example 3-24 on page 69.

*Example 3-24 Define `lpp_source` and copy filesets from CD*

---

```
root@master:/: nim -o define -t lpp_source -a server=master
-a location=/export/lpp_source/lpp5300 -a source=/dev/cd0 lpp5300
```

```
Preparing to copy install images (this will take several minutes)...
/export/lpp_source/lpp5300/sysmgtlib.libraries.apps.5.3.0.40.U
. . .
Now checking for missing install images...
All required install images have been found. This lpp_source is now ready.
```

---

Copying can also be done by using the **bfcreate** command or the SMIT **bfcreate** fast path, and saving the BFF filesets, to the intended NIM `lpp_source` directory.

The **bfcreate** command can be issued from the command line as follows:

```
bfcreate -d /dev/cd0 -t /export/lpp_source/lpp5300 all
```

Figure 3-6 shows the corresponding SMIT **bfcreate** fast path panel after accepting `/dev/cd0` as the “INPUT device”.

```

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    [/export/lpp_source/lpp5300]
  DIRECTORY for temporary storage during copying  [/tmp]
  EXTEND file systems if space needed?        yes            +
  Process multiple volumes?                  yes            +

Esc+1=Help      Esc+2=Refresh      Esc+3=Cancel    Esc+4=List
Esc+5=Reset     Esc+6=Command    Esc+7=Edit      Esc+8=Image
Esc+9=Shell     Esc+0=Exit       Enter=Do

```

Figure 3-6 SMIT *nimconfig* fast path with input

But it is also possible to mount the CD/DVD and then copy the files manually using a copy program such as **pax** command. The **cdmount** command in Example 3-25 requires that the *cdromd* (controlled by SRC) daemon is running and that the CD is available in the CD-ROM drive.

Example 3-25 Mounting the AIX CD and copying the files using *pax*

---

```

root@master:/: cdmount
root@master:/: pax -rw /<CD-ROM mountpoint>/installp/ppc
/export/lpp_source/lpp5300/installp/ppc

```

---

Make sure that the *simages* attribute for the NIM *lpp\_source* is set to “yes”, as shown in Example 3-26. The *prev\_state* (Previous State) of “unavailable for use” can be ignored as this point, as this resource is freshly created.

Example 3-26 Attributes of a newly created *lpp\_source*

---

```

root@master:/: lsnim -l lpp5300
lpp53:
  class      = resources
  type       = lpp_source
  arch       = power
  Rstate     = ready for use
  prev_state = unavailable for use
  location   = /export/lpp_source/lpp5300
  simages    = yes
  alloc_count = 0

```



```
server = master
```

---

If you copy the AIX fileset images manually with **bffcreate** or **pax**, you can create the NIM `lpp_source` object directly as shown in Example 3-27.

*Example 3-27 Creating a NIM `lpp_source` from a directory with all AIX filesets*

---

```
root@master:/: nim -o define -t lpp_source -a server=master
-a location=/export/lpp_source/lpp5300 lpp5300
```

Preparing to copy install images (this will take several minutes)...

Now checking for missing install images...

All required install images have been found. This `lpp_source` is now ready.

---

### **Removing a NIM `lpp_source`**

To remove a NIM `lpp_source`, the object definitions will be removed but the directory and filesets will remain. The following command removes the `lpp5300` NIM object:

```
nim -o remove lpp5300
```

### **Check a NIM `lpp_source`**

After adding or removing software, you must run the NIM check operation on the NIM `lpp_source`. The check operation rebuilds the table of contents (`.toc`) file in the `lpp_source` directory. It also determines whether all filesets are included in the resources to qualify the `lpp_source` for the `simages` attribute, which indicates whether the `lpp_source` contains the images necessary to install the Base Operating System images on a machine. See Example 3-28.

*Example 3-28 Check of a NIM `lpp_source`*

---

```
root@master:/: nim -Fo check lpp5304
```

---

### **Valid NIM operations on a NIM `lpp_source`**

Use the “-0” (uppercase “o”) option to the **nim** command to see the valid operations that can be performed on a NIM `lpp_source`, as shown in Example 3-29.

*Example 3-29 Check valid NIM operations for a NIM `lpp_source`*

---

```
root@master:/: lsnim -O lpp5300
lpp5300:
  define = define an object
  change = change an object's attributes
  remove = remove an object
```

```

showres = show contents of a resource
lslpp   = list LPP information about an object
check   = check the status of a NIM object
lppmgr  = eliminate unnecessary software images in an lpp_source
update  = add or remove software to or from an lpp_source

```

---

### ***Creating a NIM spot object***

A SPOT (Shared Product Object Tree) is created through an `installp` process, using a install source which can be either the install CD/DVD or an existing `lpp_source`. The SPOT is a NIM hierarchy of a `/usr` file system used during different NIM operations, such as installation, on a NIM client.

To reduce the overall time needed to create a SPOT resource, we recommend to use an existing `lpp_source` for this purpose. You can either create a “non-`/usr`” SPOT, or, to reduce the SPOT creation time even further, you can create a “`/usr`” SPOT.

Even though using a “`/usr`” SPOT is a quick solution, this means that you can only have the highest AIX level in the environment for all clients, thus, if you need to maintain multiple AIX levels, you need individual “non-`/usr`” SPOT resources for every AIX level.

**Note:** You only need to specify the top directory for the SPOT creation, the `nim command` will create a directory with the same name as the NIM spot object name.

Using a NIM `lpp_source` to create a “non-`/usr`” SPOT is the recommended way, if the SPOT are to be used for NIM client installations and maintenance.

#### *Example 3-30 Creating a non-`/usr` SPOT from a NIM `lpp_source`*

---

```

root@master:/: nim -o define -t spot -a server=master -a location=/export/spot -a
source=lpp5300 -a installp_flags=-aQg spot5300

```

```

Creating SPOT in "/export/spot" on machine "master" from "lpp5300" ...

```

```

Restoring files from BOS image. This may take several minutes ...

```

```

Installing filesets ...

```

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.
```

```
. . .
```

#### WARNINGS

```
-----
```

```
Problems described in this section are not likely to be the source of any
immediate or serious failures, but further actions may be necessary or
desired.
```

```
. . .
```

#### SUCSESSES

```
-----
```

```
Filesets listed in this section passed pre-installation verification
and will be installed.
```

#### Selected Filesets

```
-----
```

```
. . .
```

```
bos.mp 5.3.0.0 # Base Operating System Multip...
```

```
. . .
```

#### Requisites

```
-----
```

```
(being installed automatically; required by filesets listed above)
```

```
. . .
```

```
bos.rte.boot 5.3.0.0 # Boot Commands
```

```
. . .
```

```
<< End of Success Section >>
```

#### FILESET STATISTICS

```
-----
```

```
464 Selected to be installed, of which:
    462 Passed pre-installation verification
      2 FAILED pre-installation verification
    30 Additional requisites to be automatically installed
```

```

-----
492 Total to be installed

```

```

+-----+
|           Installing Software...           |
+-----+

```

```

installp: APPLYING software for:
          bos.rte 5.3.0.0

```

```

. . .

```

```

+-----+
|           Summaries:           |
+-----+

```

#### Installation Summary

```

-----
Name                               Level      Part      Event      Result
-----
. . .
bos.rte                             5.3.0.0   USR       APPLY      SUCCESS
. . .

```

```

Checking filesets and network boot images for SPOT "spot5300".
This may take several minutes ...

```

In Example 3-30, we have excluded most of the several thousand output lines of the installation process. The remainder are mostly the headers to illustrate the SPOT creation flow.

**Note:** The creation of a SPOT, by default, produces a large amount of output. We recommend to visually scan the output to look for non-fatal errors and warnings, to analyze and evaluate their impact.

For more information on creating a SPOT, see “Step 7: Defining basic resources” on page 30 and for more information on creating a “non-/usr” SPOT from a mksysb image, see 5.4.6, “SPOT creation from an existing mksysb” on page 465.

**Note:** With AIX 5300-03 ML, creating a SPOT resources for AIX 4.3.3.0 and subsequent maintenance levels require the environment variable INST\_DEBUG to be set (export INST\_DEBUG=yes).

### **Recreating the NIM spot definition**

If for some reason, the NIM spot definition has been lost (usually when the NIM database has been deleted or corrupted), and the SPOT directory hierarchy is left intact, you can use the command shown in Example 3-31 to recreate the definition.

This can also be used to define the NIM spot definition if you have copied a working SPOT to create a new one, such as when using the **pax** command with the **hardlink recursive write option (pax -lrw <source> <destination>)**.

---

#### *Example 3-31 Recreating the NIM spot definition*

---

```
root@master:/: /usr/lpp/bos.sysmgt/nim/methods/m_mkspot -o -a server=master -a
location=/export/spot -a source=no spot5300
```

---

### **Remove a NIM spot**

To remove a NIM spot, the object definitions will be removed, and the SPOT directory hierarchy as well. The following command removes the spot5300 NIM object:

```
# nim -o remove spot5300
```

**Tip:** If you want to remove a NIM spot resource but you want to keep the directory structure, if the SPOT has been defined in a separate file system, simply unmount the file system and then use the previous command.

### **Reset a NIM spot**

Use the reset operation to change the state of a SPOT, so NIM operations can be performed with it. A reset may be required on a SPOT if an operation was stopped before it completed successfully. The reset operation updates the resource state (Rstate) of the SPOT. After the reset operation is performed, the SPOT's Rstate is set to ready, and you can use the SPOT in subsequent NIM operations:

```
# nim -Fo reset spot5300
```

### **Checking a SPOT**

Use the check operation to verify the usability of the SPOT in the NIM environment. The check operation rebuilds the SPOT's network boot images, if necessary:

```
# nim -o check spot5300
```

### ***Checking the contents of a SPOT***

Use the `lppchk` operation to verify that software was installed successfully on a SPOT resource as shown in Example 3-32.

#### *Example 3-32 Checking the LPPs in a NIM spot*

---

```
root@master:/: nim -o lppchk -a show_progress=yes spot5300
```

```
+-----+
|                                     |
|               Performing "lppchk" Operation               |
|-----+
| Performing lppchk operation on machine 1 of 1: master ... |
|-----+
|               "lppchk" Operation Summary                  |
|-----+
| Target          Result                                     |
|-----+
| master          SUCCESS                                  |
+-----+
```

---

### ***Valid NIM operations on a SPOT***

Use the “-0” option to the `nim` command to find out valid operations that can be performed on a SPOT. See Example 3-33.

#### *Example 3-33 Check valid NIM operations for a SPOT*

---

```
root@master:/: lsnim -0 spot5300
spot5300:
  reset      = reset an object's NIM state
  define     = define an object
  change     = change an object's attributes
  remove     = remove an object
  cust       = perform software customization
  sync_roots = synchronize roots for all clients using specified SPOT
  showres    = show contents of a resource
  maint      = perform software maintenance
  lsipp      = list LPP information about an object
  fix_query  = perform queries on installed fixes
  showlog    = display a log in the NIM environment
  check      = check the status of a NIM object
  lppchk     = verify installed filesets
```

---

## Remove a NIM master configuration

To remove and redefine a NIM master's configuration, we recommend to backup your NIM database first (just in case), using the following command:

```
/usr/lpp/bos.sysmgt/nim/methods/m_backup_db
```

### Example 3-34 Backing up a NIM master configuration

---

```
root@master:/: /usr/lpp/bos.sysmgt/nim/methods/m_backup_db /tmp/nimdb_backup
a ./etc/objrepos/nim_attr 8 blocks
a ./etc/objrepos/nim_attr.vc 8 blocks
a ./etc/objrepos/nim_object 8 blocks
a ./etc/objrepos/nim_object.vc 8 blocks
a ./etc/NIM.level 1 blocks
a ./etc/niminfo 1 blocks
```

---

In Example 3-34, the backup data is stored in the `/tmp/nimdb_backup` file. Unconfigure the NIM master with the `nim` command as shown in Example 3-35.

### Example 3-35 Unconfigure the NIM master

---

```
root@master:/: nim -o unconfig master
0513-044 The nimesis Subsystem was requested to stop.
0513-004 The Subsystem or Group, nimd, is currently inoperative.
0513-083 Subsystem has been Deleted.
0513-083 Subsystem has been Deleted.
4 objects deleted
31 objects deleted
```

---

## Defining NIM clients

Before any NIM operation can be performed on a NIM client, it must be defined as an NIM object ("machines"). The types of NIM client machines that can be managed in the NIM environment are `standalone`, `diskless`, and `dataless` clients. The term `standalone` in the NIM environment means that the client are not dependent on any NIM resources for functioning, whereas a `diskless` or `dataless` client needs certain resources (see 4.8, "Common OS image management (COSI)" on page 299).

To create a *standalone* NIM client you need the hostname and IP address associated with it. The IP address should preferably be on a defined NIM network (if it is not the network can be created when the client is defined).

In Example 3-36, the hostname is `lpar55` and the NIM machines name is `LPAR55` for readability and debug purposes. Only the interface (`if`) attribute is used in this

example, other values are taken from the NIM master definition, such as platform and netboot\_kernel.

*Example 3-36 Create a standalone NIM client*

---

```

root@master:/: nim -o define -t standalone -a if1="net_10_1_1 lpar55 0 ent0" LPAR55
root@master:/: lsnim -l LPAR55
LPAR55:
  class          = machines
  type           = standalone
  connect        = shell
  platform       = chrp
  netboot_kernel = mp
  if1            = net_10_1_1 lpar55 0 ent0
  cable_type1    = N/A
  Cstate         = ready for a NIM operation
  prev_state     = ready for a NIM operation
  Mstate        = not running

```

---

The if attribute stores network interface information for a NIM client, and requires a sequence number when specified, to distinguish between multiple network interfaces (if1 in the previous example). As machines can have multiple network interfaces, NIM allows more than one if attribute per machine.

The value for this attribute consists of three required values and a fourth, optional value (see Example 3-37):

- Value 1 [net\_10\_1\_1] Specifies the name of the NIM network to which this interface connects. If the name of the NIM network is unknown, then the find\_net keyword can be used to match the client's IP address to a defined NIM network. If the find\_net keyword is used, but NIM does not find a matching network, the optional net\_definition attribute should be used to define the network, as well.
- Value 2 [lpar55] Specifies the hostname associated with this interface. This hostname is resolved into an IP address, on the NIM master, and will be used to set the IP address on the NIM clients interface.
- Value 3 [0] Specifies the network adapter hardware MAC address of this interface. If you do not set the MAC address, you can use a value of 0 (unless broadcasting is used for network boot of the client).
- Value 4 [ent0] Specifies the logical device name of the network adapter used for this interface. If this value is not specified, NIM uses a default based on the type of network interface



defined. This field is required when the client is defined on a heterogeneous network.

Example 3-37 does not have the fourth parameter of the `if` attribute.

*Example 3-37 Create a standalone NIM client*

---

```

root@master:/: nim -o define -t standalone -a if1="net_10_1_1 lpar55 0" LPAR55
root@master:/: lsnim -l LPAR55
LPAR55:
  class      = machines
  type       = standalone
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1       = net_10_1_1 lpar55 0
  cable_type1 = N/A
  Cstate     = ready for a NIM operation
  prev_state = ready for a NIM operation
  Mstate     = not running

```

---

Example 3-38 shows some additional machine attributes.

*Example 3-38 Create a standalone NIM client*

---

```

root@master:/: nim -o define -t standalone -a platform=chrp -a if1="net_10_1_1 lpar55
0 ent0" -a cable_type1=tp -a netboot_kernel=mp LPAR55
root@master:/: lsnim -l LPAR55
LPAR55:
  class      = machines
  type       = standalone
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = net_10_1_1 lpar55 0 ent0
  cable_type1 = tp
  Cstate     = ready for a NIM operation
  prev_state = ready for a NIM operation
  Mstate     = not running

```

---

### ***Using SMIT to define NIM machines***

Figure 3-7 shows the first step of the SMIT `nim_mkmac` fast path, if you prefer to use this method instead to create a NIM client.

```

Define a Machine

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.

* Host Name of Machine          [Entry Fields]
  (Primary Network Install Interface)  []

Esc+1=Help      Esc+2=Refresh      Esc+3=Cancel      Esc+4=List
Esc+5=Reset     Esc+6=Command      Esc+7=Edit        Esc+8=Image
Esc+9=Shell     Esc+0=Exit         Enter=Do

```

Figure 3-7 SMIT nim\_mkmac fast path

After typing the NIM client's hostname, you will be presented with the SMIT panel shown in Figure 3-8, and you can change values here or just accept the default values. In our case we have changed the "NIM Machine Name" from the default "lpar55" to "LPAR55" and the "Cable type" from the default "bnc" to "tp".

```

Define a Machine

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

* NIM Machine Name          [Entry Fields]
  [LPAR55]
* Machine Type              [standalone]      +
* Hardware Platform Type    [chrp]          +
  Kernel to use for Network Boot [mp]          +
  Communication Protocol used by client []          +
  Primary Network Install Interface
* Cable Type                tp                +
  Network Speed Setting      []                +
  Network Duplex Setting     []                +
* NIM Network               net_10_1_1
* Host Name                 lpar55
  Network Adapter Hardware Address [0]
  Network Adapter Logical Device Name []
  IPL ROM Emulation Device      []          +/
  CPU Id                       []
  Machine Group                 []          +
  Comments                      []

Esc+1=Help      Esc+2=Refresh      Esc+3=Cancel      Esc+4=List
Esc+5=Reset     Esc+6=Command      Esc+7=Edit        Esc+8=Image
Esc+9=Shell     Esc+0=Exit         Enter=Do

```

Figure 3-8 SMIT nim\_mkmac fast path

### **Removing a NIM client definition**

To remove a NIM client, first the `nim` command removes the information from the NIM master, then it tries to remove the `/etc/niminfo` file from the NIM client itself (Example 3-39).

#### *Example 3-39 Removing a NIM client definition from the NIM master*

---

```
root@master:/: nim -o remove LPAR55
warning: 0042-140 m_rmmac: unable to remove the /etc/niminfo file on
"LPAR55"
```

---

The message after the command is normal when the NIM client is not reachable from the NIM master, such as when it is powered down or otherwise offline.

### **Installing NIM clients**

To install a NIM client you can perform this in several ways:

- Base Operating System (BOS `rte`) installation
- System clone installation (NIM `mksysb`)
- Automated customization after a generic BOS install (5.8.1, “Install time customization” on page 514).

We will show the simple way to perform a BOS `rte` install.

To perform a BOS install with NIM, you need to allocate a NIM `lpp_source` and a NIM `spot` to a NIM client, and then start the NIM `rte` installation.

In this first step we use a NIM `lpp_source` named `lpp5304` and a NIM `spot` named `spot5304` to install a NIM client named `LPAR55`.

```
nim -o allocate -a spot=spot5304 -a lpp_source=lpp5304 LPAR55
```

The second step is to initiate the install, and if the client is already running, NIM it will also set the bootlist to network boot on the client and then reboot the client.

```
nim -o bos_inst -a source=rte -a installp_flags=agX -a
accept_licenses=yes LPAR55
```

The source is `rte` for BOS install from the NIM `lpp_source`, and therefore the `installp_flags` attribute is also used, with the Apply (`a`), Prereqs (`g`) and Expand (`X`) file system flags. The `accept_licenses` attribute must be set to `yes`, otherwise the installation will fail.

In case the installation has not been successful, you need to re-allocate the resources, but first you need to reset and deallocate NIM resources (in this case with the `force` option):

```
nim -Fo reset LPAR55
```

```
nim -Fo deallocate -a subclass=all LPAR55
```

To view the progress during installation and first boot, you can use the showlog operation to the **nim** command:

```
nim -o showlog -a log_type=boot LPAR55
```

**Note:** Other showlog log\_type values are bosinst, devinst, nimerr, boot, niminst, script, lppchk, and alt\_disk\_install.

### ***Basic SMS menu steps***

If the NIM client is not up and running before starting the installation, you need to manually condition (intiate network boot) the client machine.

Reboot the client and access the SMS menus. Set the NIM clients IP address, netmask, gateway (if any), and the IP address of the NIM master. Verify network settings by performing a PING test from the SMS menus. If this is successful, select Network from the Select Install/Boot Device menu.

Exit the SMS and the network installation will start. This starts with the client looking for boot information (BOOTP), then load the kernel and the install programs (TFTP) and then NFS mounting the allocated SPOT. Once the SPOT is mounted, the install process follows the same steps as a CD/DVD install.

For a more detailed description, see “SMS and console flow during NIM client installation” on page 83.

### ***Using SMIT to install a standalone client***

The following list shows the steps you have to perform starting from SMIT nim\_bosinst fast path panel, to install the BOS on a NIM client:

```
root@master:/: smitty nim_bosinst
  Select a TARGET for the operation
    Select the installation TYPE
      Select the LPP_SOURCE to use for the installation
        Select the SPOT to use for the installation
```

Next, you will reach the a SMIT panel depicted in Figure 3-9:

```

Install the Base Operating System on Standalone Clients

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

[TOP]                               [Entry Fields]
* Installation Target                 1par55
* Installation Type                   rte
* SPOT                                530spot_res
LPP_SOURCE                            [530lpp_res]          +
MKSYSB

BOSINST_DATA to use during installation  []          +
IMAGE_DATA to use during installation    []          +
RESOLV_CONF to use for network configuration []        +
Customization SCRIPT to run after installation []      +
Customization FB Script to run at first reboot []      +
ACCEPT new license agreements?         [no]        +
Remain NIM client after install?       [yes]       +
Preserve NIM definitions for resources on [yes]       +
[MORE...35]

Esc+1=Help      Esc+2=Refresh      Esc+3=Cancel    Esc+4=List
Esc+5=Reset     Esc+6=Command    Esc+7=Edit      Esc+8=Image
Esc+9=Shell     Esc+0=Exit       Enter=Do

```

*Figure 3-9 SMIT nim\_bosinst fast path*

Scroll down and change the “**ACCEPT new license agreements?**” to “**yes**”. Then hit **<ENTER>** and the SMIT panel should end with “**OK**”, after which the NIM client will be installed.

### **SMS and console flow during NIM client installation**

Select one (1) from the Initial Program Load (IPL) menu to access the SMS Menu as shown in Figure 3-10 on page 83.

```

IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM

      1 = SMS Menu
      8 = Open Firmware Prompt
      5 = Default Boot List
      6 = Stored Boot List

```

*Figure 3-10 At this point press the “1” key to get to the SMS menu*

Select two (2) from the Main Menu to access the Setup Remote IPL menu, as shown in Figure 3-11.

```
PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Main Menu
1.  Select Language
2.  Setup Remote IPL (Initial Program Load)
3.  Change SCSI Settings
4.  Select Console
5.  Select Boot Options

-----

Navigation Keys:

                                     X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:2
```

Figure 3-11 Setup remote IPL (Initial Program Load) option => 2

Select the desired adapter from the NIC Adapters list, in this case one (1), to access the Network Parameters menu as shown in Figure 3-12 on page 84.

```
PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
NIC Adapters
Device                Location Code          Hardware
                        Location Code          Address
1.  Interpartition Logical LAN  U9131.52A.650E4DG-V4-C30-T1  1a8cf000401e

-----

Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:1
```

Figure 3-12 Interpartition logical LAN option => 1

Select one (1) from the Network Parameters menu to access the IP Parameters menu as shown in Figure 3-13.

```
PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Network Parameters
Interpartition Logical LAN: U9131.52A.650E4DG-V4-C30-T1
1. IP Parameters
2. Adapter Configuration
3. Ping Test
4. Advanced Setup: BOOTP

-----

Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:1
```

Figure 3-13 Network parameters options => 1

1. Select one (1) from the IP Parameters menu, to set the Client IP Address.
2. Select two (2) from the IP Parameters menu, to set the Server IP Address (NIM master).
3. Select three (3) from the IP Parameters menu, to set the Gateway IP Address. Leave it if the NIM master and client are on the same IP subnet, otherwise enter the default router/gateway for the Client IP Address to access the Server IP Address network.
4. Select four (4) from the IP Parameters menu, to set the Subnet Mask for the IP network.
5. Press **<ESC>** and return to the previous menu as shown in Figure 3-14.

```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
IP Parameters
nterpartition Logical LAN: U9131.52A.650E4DG-V4-C30-T1
1. Client IP Address           [10.1.1.55]
2. Server IP Address          [10.1.1.2]
3. Gateway IP Address         [000.000.000.000]
4. Subnet Mask                 [255.255.255.000]

-----

Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:<esc>

```

Figure 3-14 IP parameters => 1, 2, 3, 4 and <esc>

Select three (3) from the Network Parameters menu to select the Ping Test, as shown in Figure 3-15 on page 86.

```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Network Parameters
nterpartition Logical LAN: U9131.52A.650E4DG-V4-C30-T1
1. IP Parameters
2. Adapter Configuration
3. Ping Test
4. Advanced Setup: B00TP

-----

Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:3

```

Figure 3-15 Network parameters option => 3

Select one (1) from the Network Parameters menu to start the Ping Test as shown in Figure 3-16.



```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Ping Test
Interpartition Logical LAN: U9131.52A.650E4DG-V4-C30-T1
Speed, Duplex: auto,auto
Client IP Address: 10.1.1.55
Server IP Address: 10.1.1.2
Gateway IP Address: 000.000.000.000
Subnet Mask: 255.255.255.000
Spanning Tree Enabled: 0
Connector Type: none

1. Execute Ping Test
-----
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:1

```

Figure 3-16 Ping test option #1 => 1

Two screens are shown, the first one with the configuration information and the second screen with the result of the Ping Test. See Figure 3-17 on page 87.

```

$PING: args = /vdevice/l-1an@3000001e:10.1.1.2,10.1.1.55,000.000.000.000

NET: Device String - /vdevice/l-1an@3000001e
NET: Ping timeout - 10000 msec

NET: Ping string - h,10.1.1.2,10.1.1.55,000.000.000.000
PING: chosen-network-type = ethernet,auto,none,auto
PING: client IP = 10.1.1.55
PING: server IP = 10.1.1.2
PING: gateway IP = 0.0.0.0

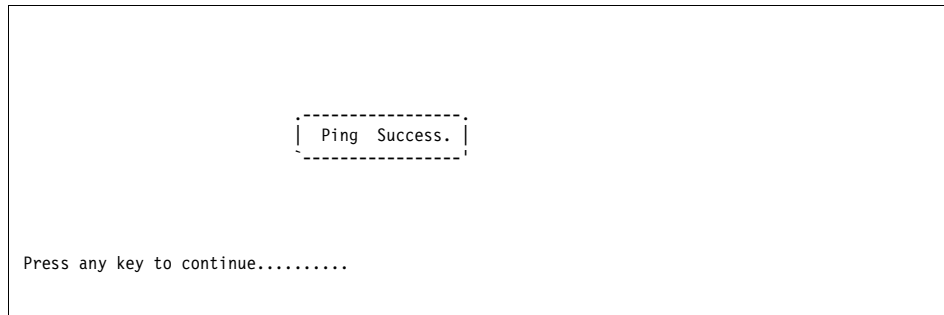
PING: device /vdevice/l-1an@3000001e
PING: loc-code U9131.52A.650E4DG-V4-C30-T1

PING: Ready to ping:
PING: source hardware address is 1a 8c f0 0 40 1e
PING: destination hardware address is 1a 8c f0 0 30 1e
PING: source IP address is 10.1.1.55
PING: destination IP address is 10.1.1.2

```

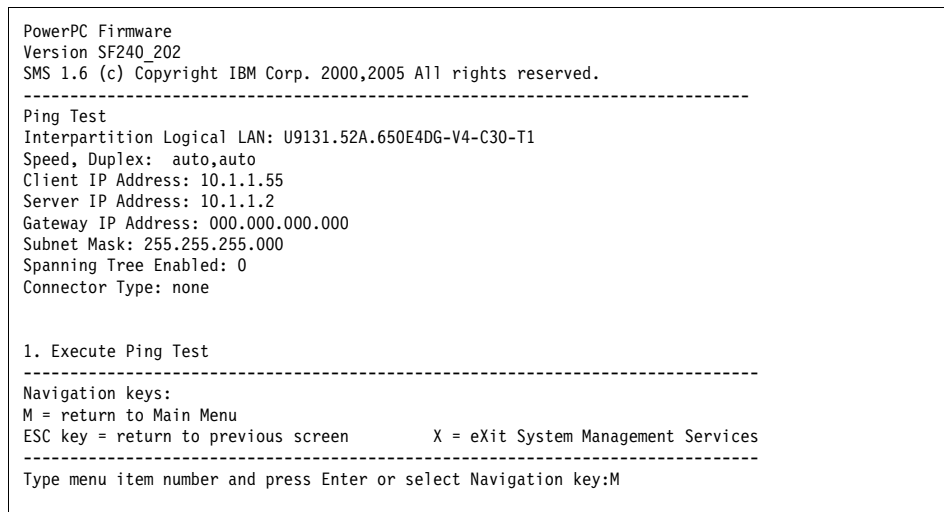
Figure 3-17 Ping test option #2

Press any key to end the Ping Test as shown in Figure 3-18.



*Figure 3-18 Ping test option #3 => Any key*

Next, press M from the Ping Test menu to return to the Main Menu as shown in Figure 3-19.



*Figure 3-19 Ping test option #4 => M*

Select five (5) from the main menu to select the Select Boot Options, as shown in Figure 3-20.

```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Main Menu
1.  Select Language
2.  Setup Remote IPL (Initial Program Load)
3.  Change SCSI Settings
4.  Select Console
5.  Select Boot Options

-----

Navigation Keys:
                                     X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:5

```

Figure 3-20 Select boot options => 5

Select one (1) from the Multiboot menu to select the Select Install/Boot Device as shown in Figure 3-21 on page 89.

```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Multiboot
1.  Select Install/Boot Device
2.  Configure Boot Device Order
3.  Multiboot Startup <OFF>

-----

Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:1

```

Figure 3-21 Select Install/Boot device => 1

Select six (6) from the Select Device Type menu to select the Network as shown in Figure 3-22.

```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Select Device Type
1.  Diskette
2.  Tape
3.  CD/DVD
4.  IDE
5.  Hard Drive
6.  Network
7.  List all Devices

-----

Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:6

```

Figure 3-22 Network selection => 6

Select the desired adapter from the NIC Adapters list, in this case one (1) to access the Select Task menu, as shown in Figure 3-23 on page 90.

```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Select Device
Device Current Device
Number Position Name
1.      -      Virtual Ethernet
                ( loc=U9131.52A.650E4DG-V4-C30-T1 )

-----

Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:1

```

Figure 3-23 Ethernet selection => 1

Select two (2) from the Select Task menu to select the Normal Mode Boot, as shown in Figure 3-24.

```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Select Task

Virtual Ethernet
  ( loc=U9131.52A.650E4DG-V4-C30-T1 )

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

-----

Navigation keys:
M = return to Main Menu
ESC key = return to previous screen      X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:2

```

Figure 3-24 Normal mode boot => 2

Select one (1) from the exit SMS menu to let the network installation begin, as shown in Figure 3-25 on page 91.

```

PowerPC Firmware
Version SF240_202
SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.
-----
Are you sure you want to exit System Management Services?
1.  Yes
2.  No

-----

Navigation Keys:

                                X = eXit System Management Services
-----
Type menu item number and press Enter or select Navigation key:1

```

Figure 3-25 Exit the system management services => 1

Figure 3-26 shows the installation flow which you can expect from an AIX 5L V5.3 LPAR environment. First the network boot (IPL) is initiated for the LPAR.

```

IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM          STARTING SOFTWARE          IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM          PLEASE WAIT...              IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM          IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM

BOOTP: chosen-network-type = ethernet,auto,none,auto
BOOTP: server IP = 10.1.1.2
BOOTP: requested filename =
BOOTP: client IP = 10.1.1.55
BOOTP: client HW addr = 1a 8c f0 0 40 1e
BOOTP: gateway IP = 0.0.0.0
BOOTP: device /vdevice/l-1an@3000001e
BOOTP: loc-code U9131.52A.650E4DG-V4-C30-T1

```

Figure 3-26 Initiating the network boot (IPL)

Second, the LPAR issues BOOTP requests to the server IP-address specified in the IP Parameters menu. It receives the BOOTP record information from the /etc/bootptab on the NIM master. This record specifies the TFTP server for sending a bootable kernel. The LPAR issues TFTP requests to the server listed in the /etc/bootptab sa field from the NIM master. In Figure 3-27 we can see that the image transferred to the client is approximately 12 MB.

```

IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM

BOOTP: chosen-network-type = ethernet,auto,none,auto
BOOTP: server IP = 10.1.1.2
BOOTP: requested filename =
BOOTP: client IP = 10.1.1.55
BOOTP: client HW addr = 1a 8c f0 0 40 1e
BOOTP: gateway IP = 0.0.0.0
BOOTP: device /vdevice/l-1an@3000001e
BOOTP: loc-code U9131.52A.650E4DG-V4-C30-T1

BOOTP R = 1 BOOTP S = 2
FILE: /tftpboot/LPAR55
FINAL Packet Count = 23852
FINAL File Size = 12211712 bytes.
load-base=0x4000
real-base=0x2000000

Elapsed time since release of system processors: 18292 mins 43 secs

-----
Welcome to AIX.
boot image timestamp: 14:57 06/13

```

Figure 3-27 Bootp request and tftp requests

Then control is passed to the transferred and loaded AIX 5L kernel, as shown in Figure 3-28.

```

BOOTP R = 1 BOOTP S = 2
FILE: /tftpboot/LPAR55
FINAL Packet Count = 23852
FINAL File Size = 12211712 bytes.
load-base=0x4000
real-base=0x2000000

Elapsed time since release of system processors: 18292 mins 43 secs

-----
                        Welcome to AIX.
                        boot image timestamp: 14:57 06/13
                        The current time and date: 07:38:53 06/14/2006
                        number of processors: 1   size of memory: 512MB
boot device: /vdevice/l-lan@3000001e:10.1.1.2,,10.1.1.55,000.000.000.00,00,00
                        kernel size: 11008898; 32 bit kernel
-----

```

Figure 3-28 Starting the AIX 5L kernel

In this case the installation is made using a NIM mksysb resource as shown in Figure 3-29.

```

                        Installing Base Operating System

Please wait...

Approximate   Elapsed time
% tasks complete   (in minutes)

16             1       13% of mksysb data restored.

```

Figure 3-29 Installing from a mksysb resource

After installing the software, the boot image is created as shown in Figure 3-30.

```

                                Installing Base Operating System

Please wait...

Approximate      Elapsed time
% tasks complete (in minutes)

    90              13      Creating boot image.
                        Installing Base Operating System

```

*Figure 3-30 Creating boot image*

The installation is finished and the LPAR is restarted as shown in Figure 3-31 on page 94.

```

Licensed Materials - Property of IBM

5765G0300
(C) Copyright International Business Machines Corp. 1985, 2005.
(C) Copyright AT&T 1984, 1985, 1986, 1987, 1988, 1989.
(C) Copyright Regents of the University of California 1980, 1982, 1983, 1985,
1986, 1987, 1988, 1989.
(C) Copyright BULL 1993, 2005.
(C) Copyright Digi International Inc. 1988-1993.
(C) Copyright Interactive Systems Corporation 1985, 1991.
(C) Copyright ISQUARE, Inc. 1990.
(C) Copyright Mentat Inc. 1990, 1991.
(C) Copyright Open Software Foundation, Inc. 1989, 1994.
(C) Copyright Sun Microsystems, Inc. 1984, 1985, 1986, 1987, 1988, 1991.

All rights reserved.
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.
forced unmount of /var
Rebooting . . .

```

*Figure 3-31 Installation is finished and LPAR restarted*

The newly installed kernel is loaded and the LPAR is restarted, as shown in Figure 3-32.



```
-----  
Welcome to AIX.  
boot image timestamp: 07:53 06/14  
The current time and date: 07:55:12 06/14/2006  
number of processors: 1 size of memory: 512MB  
boot device: /vdevice/v-scsi@30000016/disk@810000000000000:2  
kernel size: 13425263; 64 bit kernel  
-----
```

*Figure 3-32 New kernel is loaded and the LPAR restarts*

## 3.2 Web-Based System Manager (WebSM)

This section describes how to configure the Network Installation Manager (NIM) master, create the basic installation resources required to install NIM client machines, and manage the resources for diskless and dataless clients using the Web-Based System Management.

For this procedure you need access from a graphical interface (GUI) such as a local X11 display or a remote X11 client.

### 3.2.1 Configuring a NIM master through WebSM

Use the following procedure to configure the NIM master, and create the basic installation resources using Web-Based System Management:

1. Insert the AIX 5L V5.3, or later, Volume 1 CD into the appropriate drive of the designated master machine.
2. Start the Web-Based System Management application with the `wsm` command. You will see the screen shown in Figure 3-33.



Figure 3-33 Web-Based System Management control panel

In the navigation area (Figure 3-33), select the NIM container. If you are configuring your NIM master from scratch, you will see the screen shown in Figure 3-34.

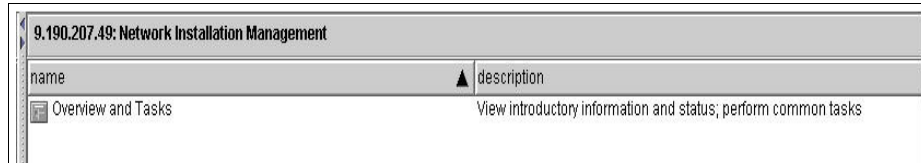


Figure 3-34 Web-Based System Management NIM panel

Select the “**Overview and Tasks**” to open the NIM tasks control panel. The NIM Tasks panel is shown in Figure 3-35.

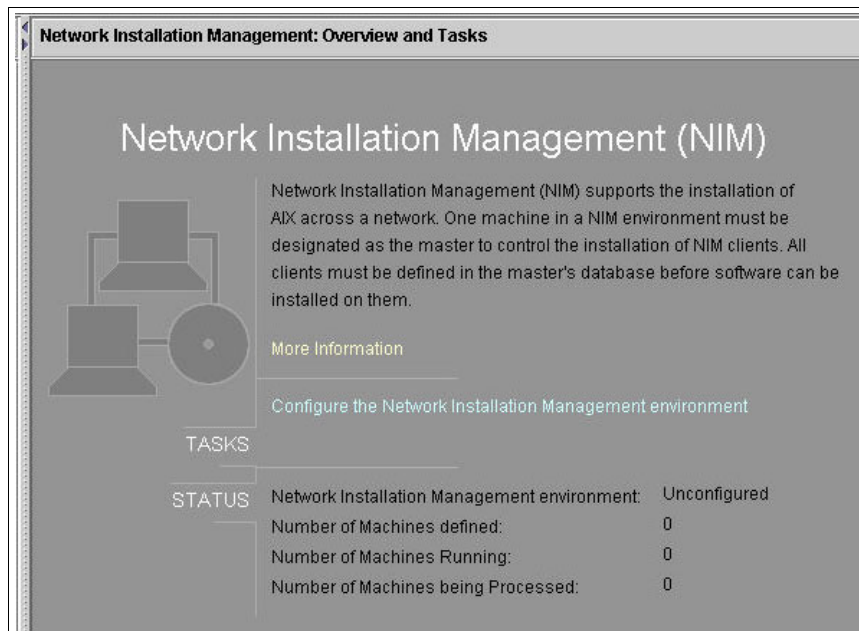


Figure 3-35 WSM NIM tasks panel

In the NIM Tasks panel, select the “**Configure the Network Installation Management environment**” to start the NIM environment configuration wizard. This wizard will take you step-by-step to configure the NIM master. The first screen is shown Figure 3-36.

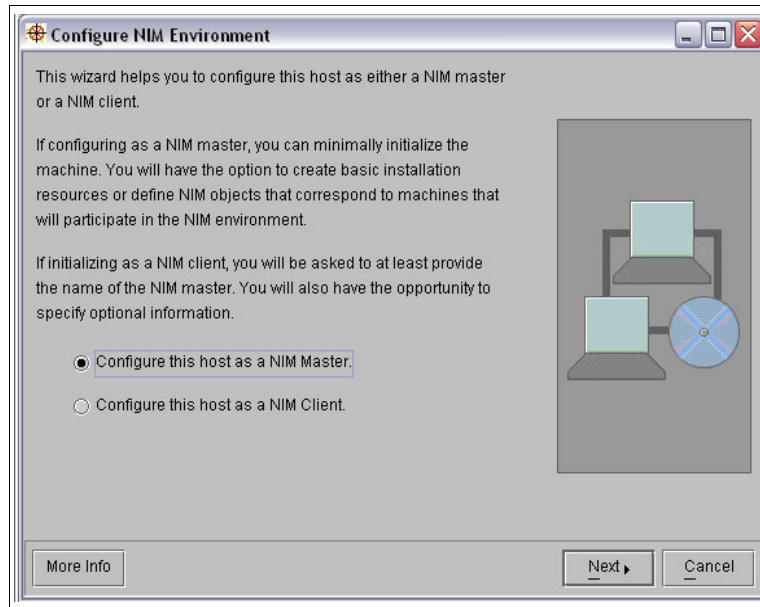


Figure 3-36 Configuring the NIM master

Since we are configuring a NIM Master, select the first option “**Configure this host as a name Master**” and click “Next”. If you are configuring your NIM Master from scratch and the NIM filesets are not installed, the wizard will ask you to insert the AIX media CD in the CD or DVD drive available to the machine you are configuring. The screen shown in Figure 3-37 on page 99 will appear in this case.

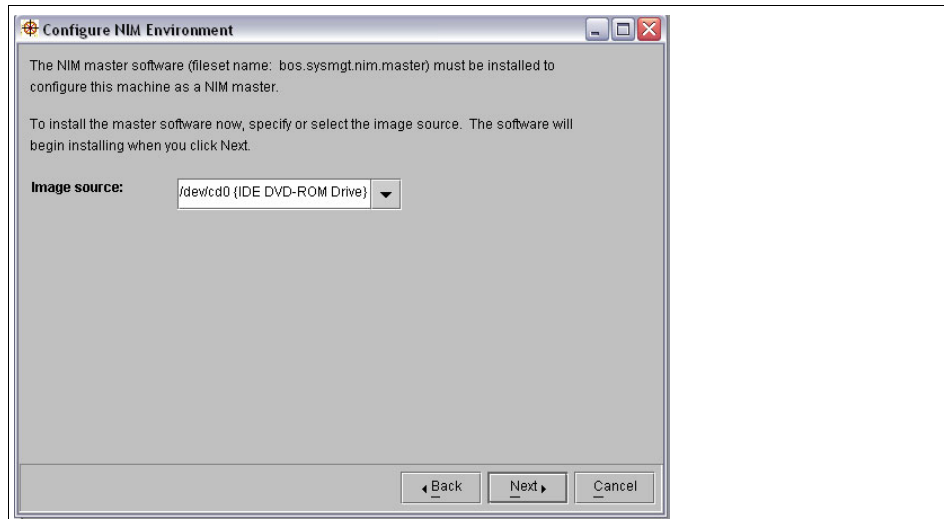


Figure 3-37 Installing the NIM filesets

The wizard will start installing the necessary NIM filesets. The results are similar to Example 3-40.

*Example 3-40 Installing the NIM filesets*

```

+-----+
|           Pre-installation Verification...           |
+-----+
Verifying selections...done
Verifying requisites...Verifying requisites...done
Results...

WARNINGS
-----
Problems described in this section are not likely to be the source of any
immediate or serious failures, but further actions may be necessary or
desired.

FILESET STATISTICS
-----
  2 Selected to be installed, of which:
    1 Passed pre-installation verification
    1 Already installed (directly or via superseding filesets)
-----
  1 Total to be installed
:-----+

```

## Summaries:

+-----+

## Pre-installation Failure/Warning Summary

-----

Name	Level	Pre-installation Failure/Warning
bos.sysmgt.nim.client	5.3.0.10	Already superseded by 5.3.0.30

## Installation Summary

-----

Name	Level	Part	Event	Result
bos.sysmgt.nim.master	5.3.0.10	USR	APPLY	SUCCESS

The next step is to configure the primary network interface of the NIM Master. The screen shown in Figure 3-38 will appear, in which you will define the network name. After you define the network name, press “Next”.

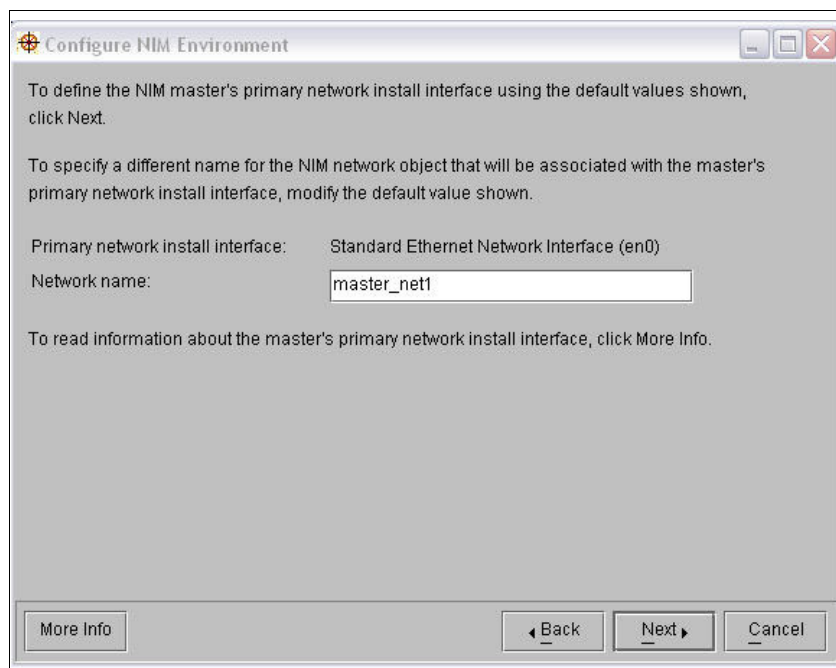


Figure 3-38 Configuring NIM network name.

After the primary NIM network interface and the NIM network name have been configured, the next step is to configure the NIM resources. After clicking “Next” in the previous screen (Figure 3-38), the screen shown in Figure 3-39 allows you

to choose either to configure the NIM resources automatically using the AIX defaults, or to custom configure these resources.

In Figure 3-39, we chose to perform custom configuration of the NIM resources.

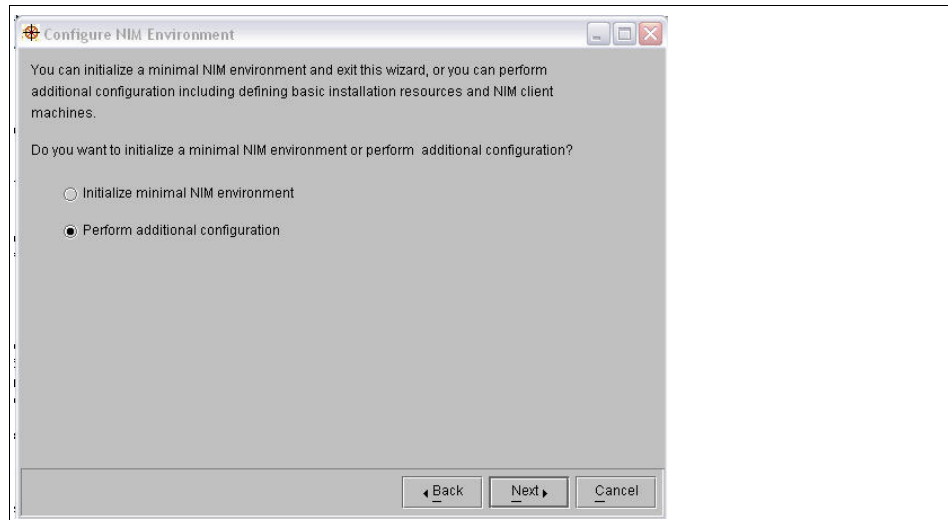


Figure 3-39 NIM master custom configuration

Press “Next” to move to the next couple of screens, where you can select the tasks that are expected to be performed by the NIM master as shown in Figure 3-40 on page 102.

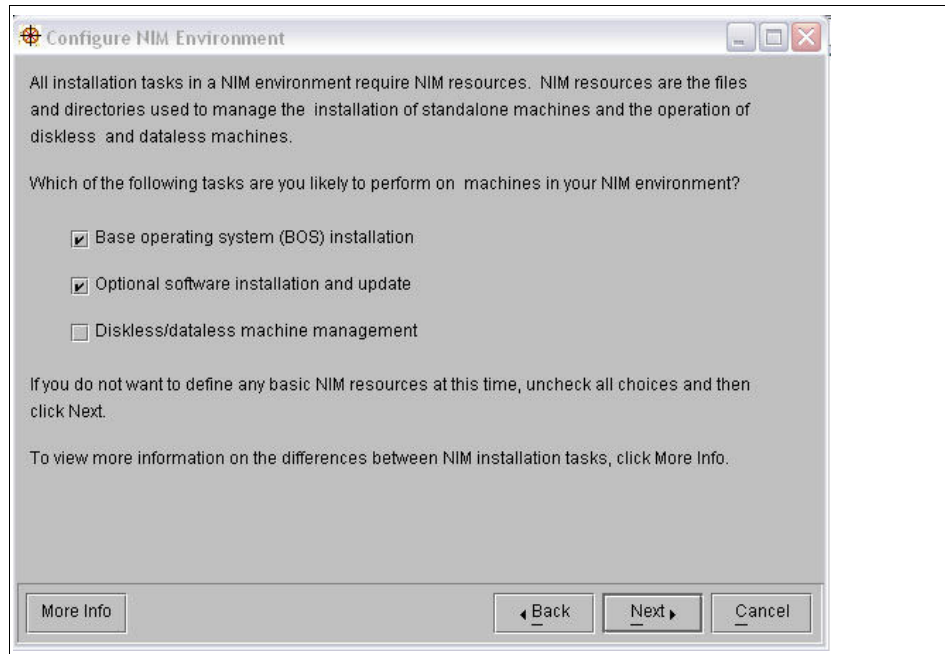


Figure 3-40 NIM master tasks

The first NIM master resource to configure is the NIM “lpp\_source”. In Figure 3-41 on page 102, you configure the location and name of the NIM lpp\_source to build.

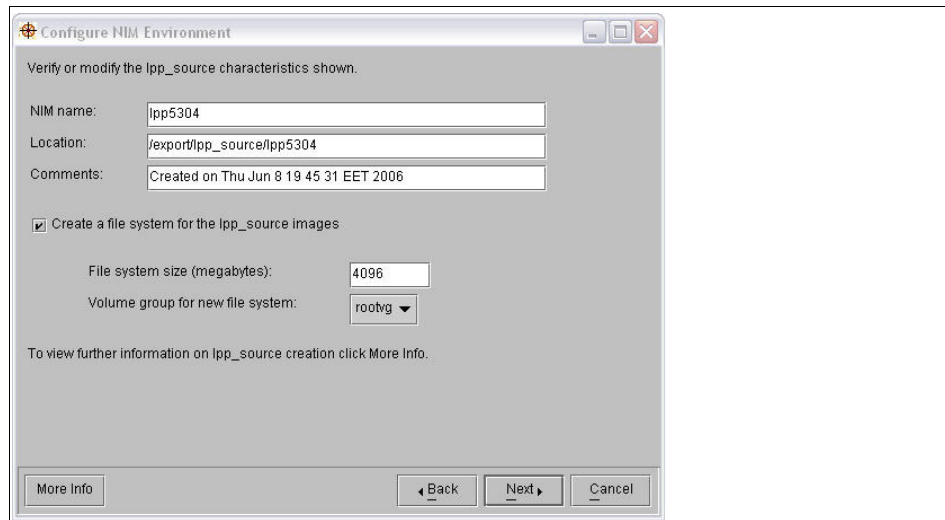


Figure 3-41 NIM master lpp\_source configuration



After configuring the NIM lpp\_source, the next step is to configure the SPOT. In the screen shown in Figure 3-42 on page 103, you configure the NIM spot name and location.

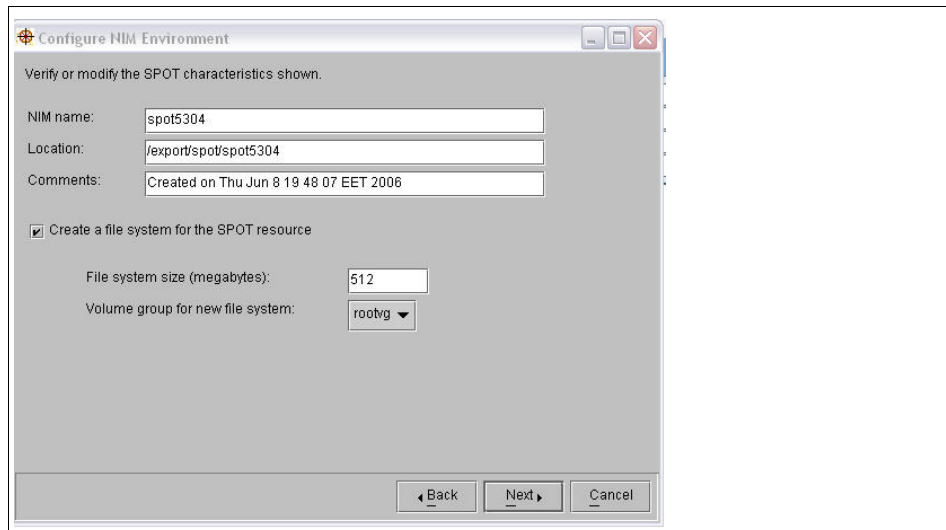


Figure 3-42 NIM master SPOT configuration

After clicking “Next”, you will receive the screen shown in Figure 3-43 on page 103, where you can choose to exit the custom configuration screens, or change the configuration parameters that you have just customized.

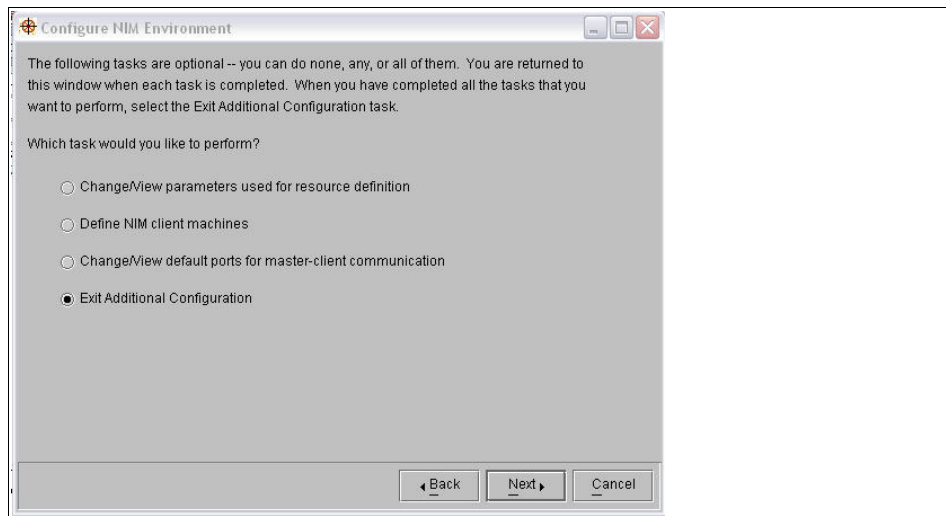


Figure 3-43 Exiting the NIM configuration

In Figure 3-43, you can choose to exit the NIM resources customizing screens. Select **“Exit Additional Configuration”** and click **“Next”**,

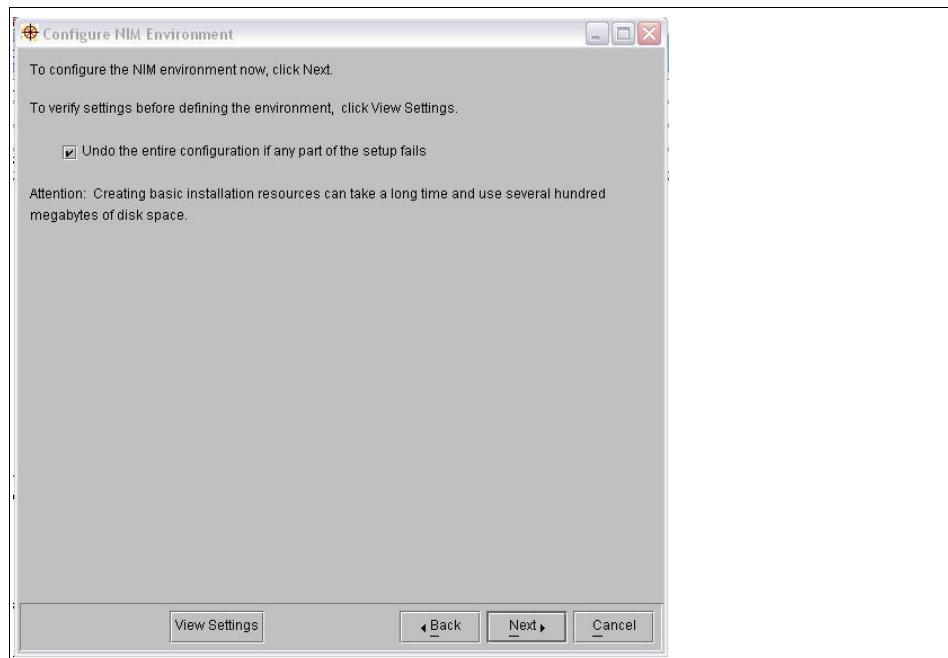


Figure 3-44 WSM NIM configuration wizard - final screen

In this final screen as shown in Figure 3-44, we recommend to select **“Undo the entire configuration if any part of the setup fails”**.

In Figure 3-44, you can choose to view the settings you performed before you start executing the NIM environment setup.

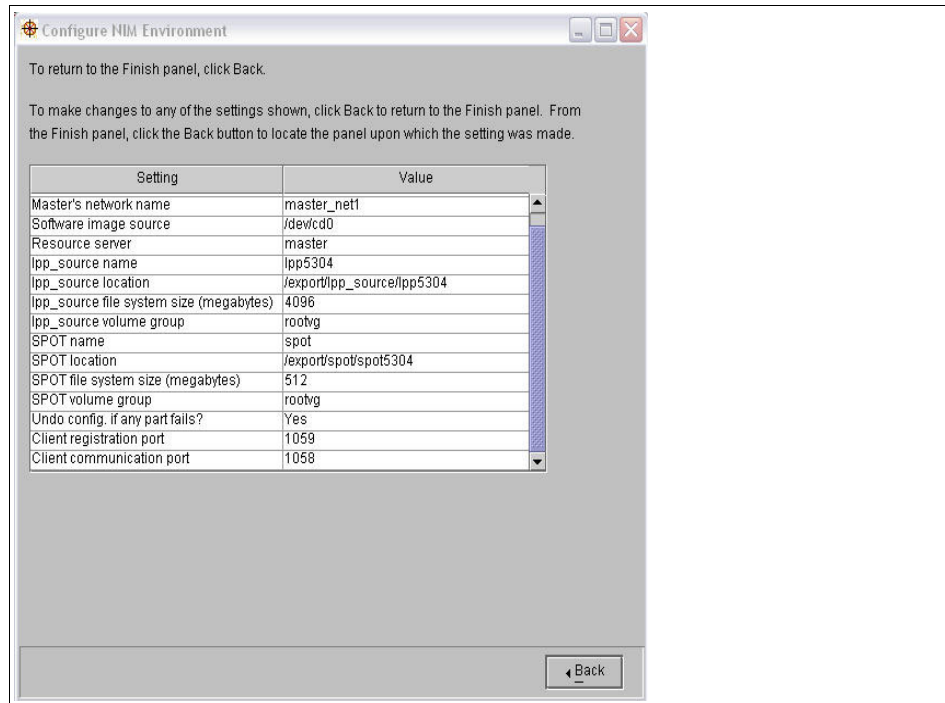


Figure 3-45 Viewing the NIM configuration parameters

In Figure 3-45, you can review the configuration parameters you have setup. Once you are completed with your review, you can click “Back” to return to the previous screen to start executing the setup of the NIM environment.

**Note:** This procedure produces a large amount of output, especially when creating the SPOT resource. Remember to scan through the output to look for nonfatal errors and warnings that may not be evident from a successful return code.

### 3.2.2 Configuring NIM clients using Web-Based System Management

After you complete the setup of the NIM master, the next step is to start defining the NIM clients to your NIM master.

In the Web-Based System Management main panel as shown in Figure 3-46, click the NIM icon to open the NIM panel, then click “**Overview and Tasks**” to open the following screen.

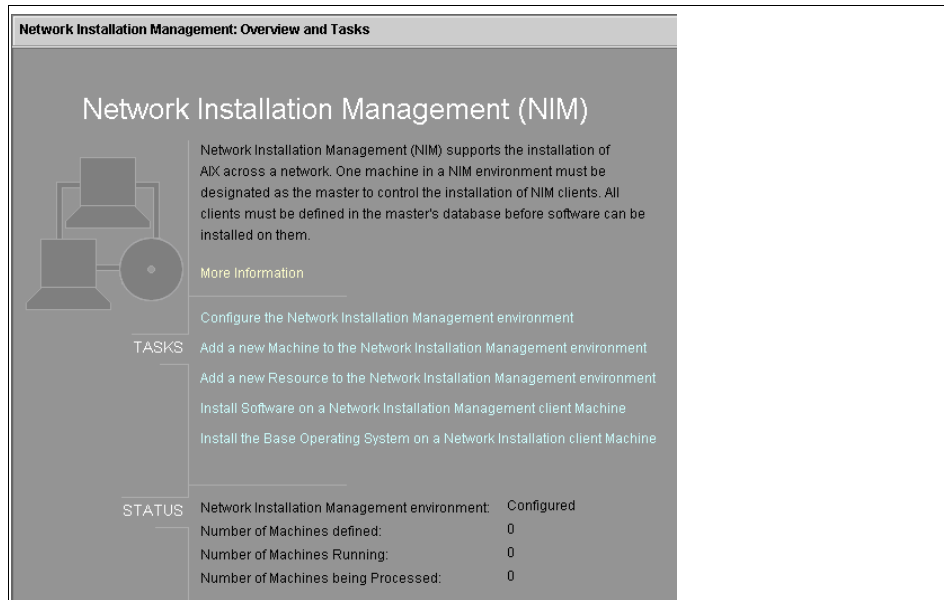


Figure 3-46 NIM tasks panel

In Figure 3-46, click “**Add a new machine to the Network Installation Management environment**” to open the dialogue box shown in Figure 3-46 where you can enter the hostnames of the NIM clients.

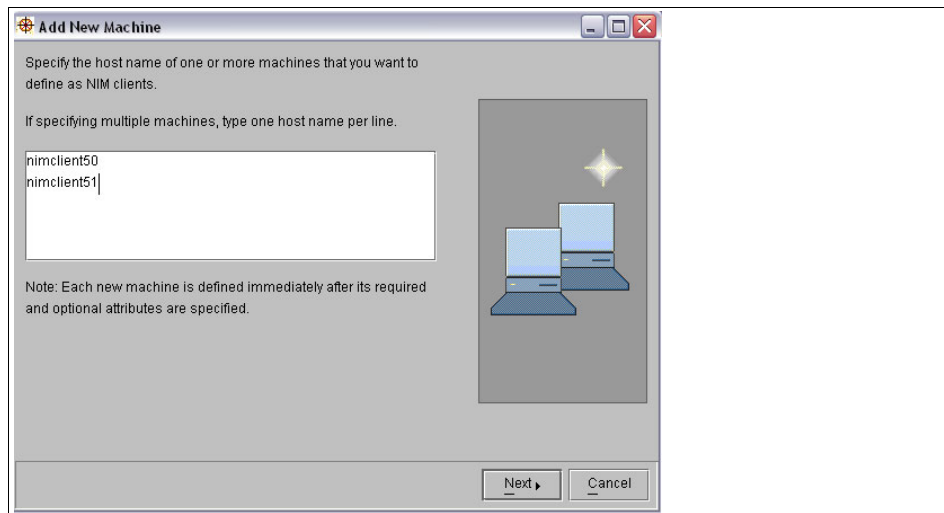


Figure 3-47 NIM clients definition

In the **“Add New Machine”** dialogue box in Figure 3-47, add the hostnames of the NIM Clients you want to define, then click **“Next”**.

**Note:** The hostnames of the NIM clients you want to add must be resolvable and preferably exist in the `/etc/hostname` file. If these hostnames are not defined in the `/etc/hostname` files, you will receive an error message as shown in Figure 3-48.

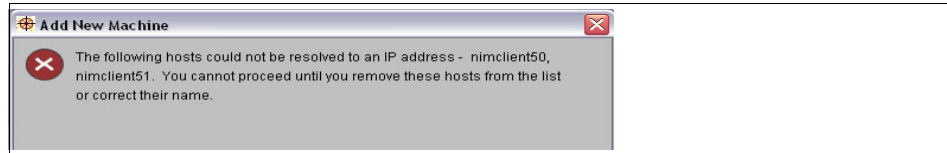


Figure 3-48 Error message when NIM clients' hostnames cannot be resolved

After you click **“Next”** as in Figure 3-47 on page 106, you will receive the screen shown in Figure 3-49 for each NIM client you defined. In this screen, you are required to define the attributes of the NIM client.

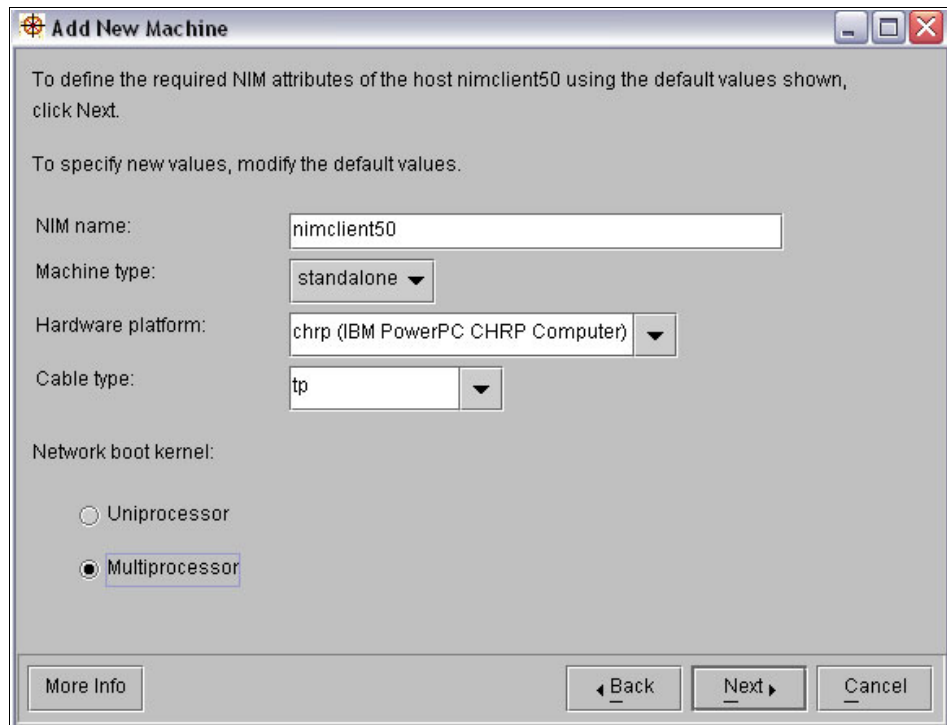
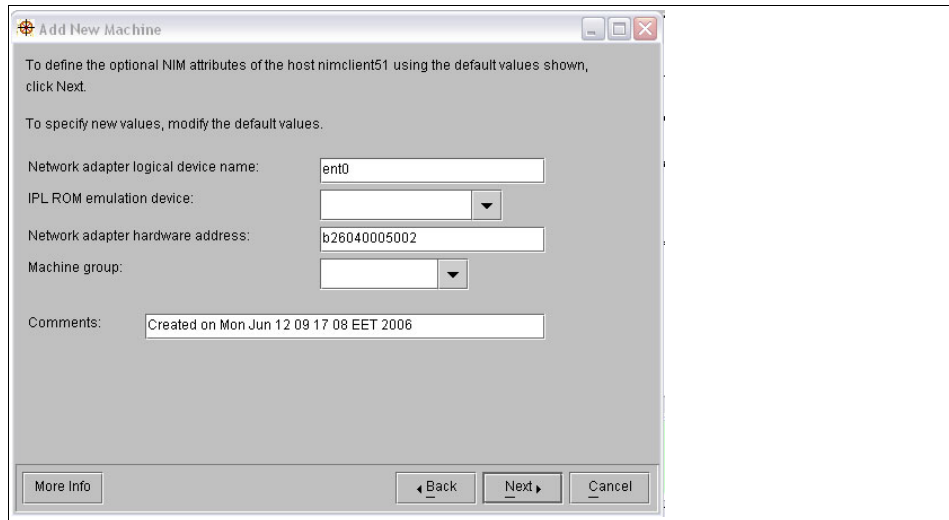


Figure 3-49 NIM client attributes definition

After you click “Next” as shown in Figure 3-49, the screen in Figure 3-50 shows, where you will define the network interface, and its hardware address.



The screenshot shows a window titled "Add New Machine" with the following content:

To define the optional NIM attributes of the host nimclient51 using the default values shown, click Next.

To specify new values, modify the default values.

Network adapter logical device name:

IPL ROM emulation device:

Network adapter hardware address:

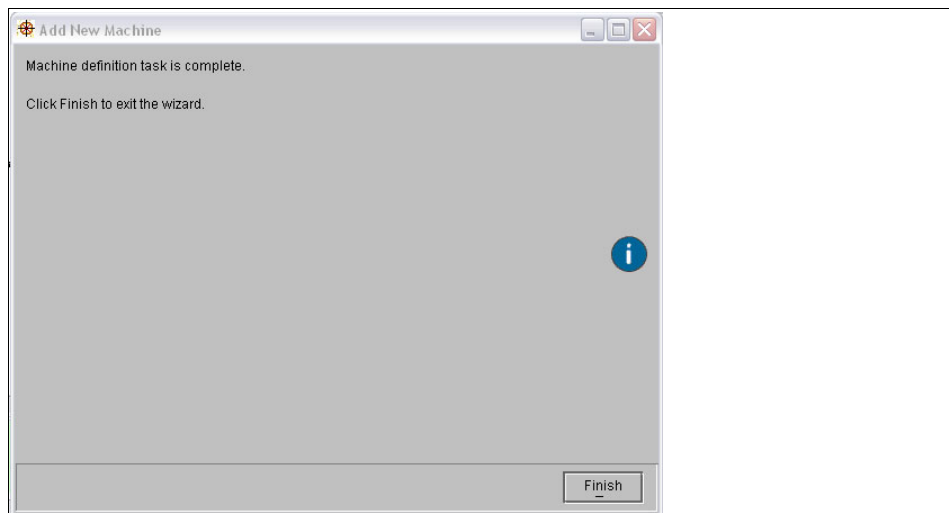
Machine group:

Comments:

Buttons: More Info, Back, Next, Cancel

Figure 3-50 NIM client

After you click “Next”, you get a screen as shown in Figure 3-51 informing you that you completed the configuration of the NIM clients.



The screenshot shows a window titled "Add New Machine" with the following content:

Machine definition task is complete.

Click Finish to exit the wizard.

Buttons: Finish

Figure 3-51 Machine Definition Task is Complete

After completing the NIM clients definition, the next step is to start installing the BOS on these NIM clients.

### 3.2.3 Installing a NIM client using Web-Based System Management

After you have completed the definition of the NIM clients, you are ready to start installing them. This section illustrates the procedure of installing NIM clients using Web-Based System Management.

To install a NIM client using Web-Based System Management, first click on the NIM icon in the Web-Based System Management panel, then click **“Overview and Tasks”**. After you click **“Overview and Tasks”**, you receive the screen shown in Figure 3-52 on page 109. In this screen, click **“Install the Base Operating System on a Network Installation client Machine”**.

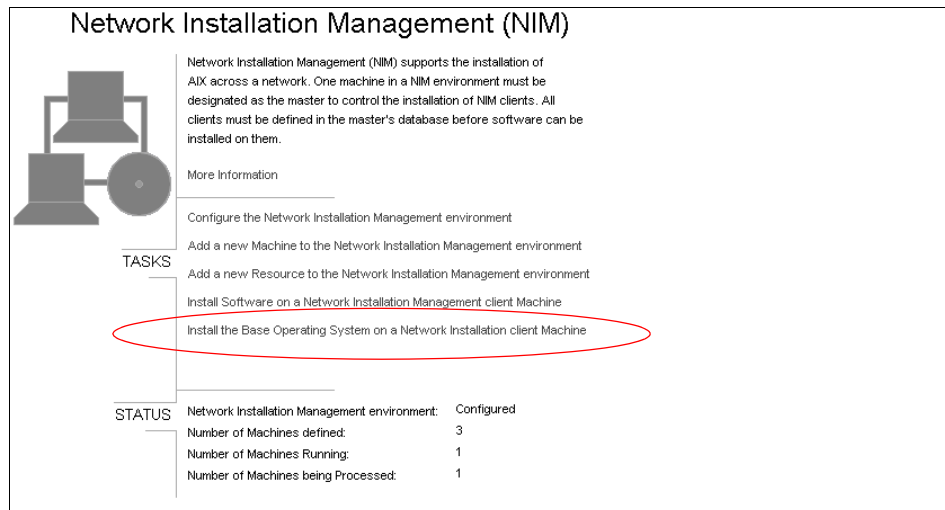


Figure 3-52 Installing the BOS on a NIM client using WebSM

After you click **“Install the BOS on a NIM client Machine”**, you get the screen as shown in Figure 3-53 containing a list of NIM clients already defined. Just select the NIM client you want to install and press **“Next”**.

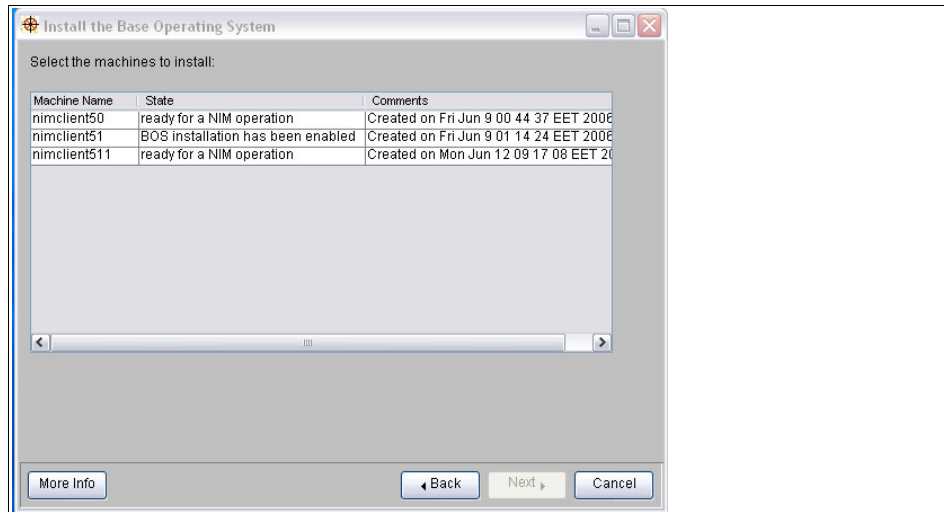


Figure 3-53 Select the NIM client you want to install and click next

The next step is to select the installation method from the screen shown in Figure 3-54 on page 110. In this example, we chose the **“Overwrite the existing system”** since we are installing an LPAR from scratch.

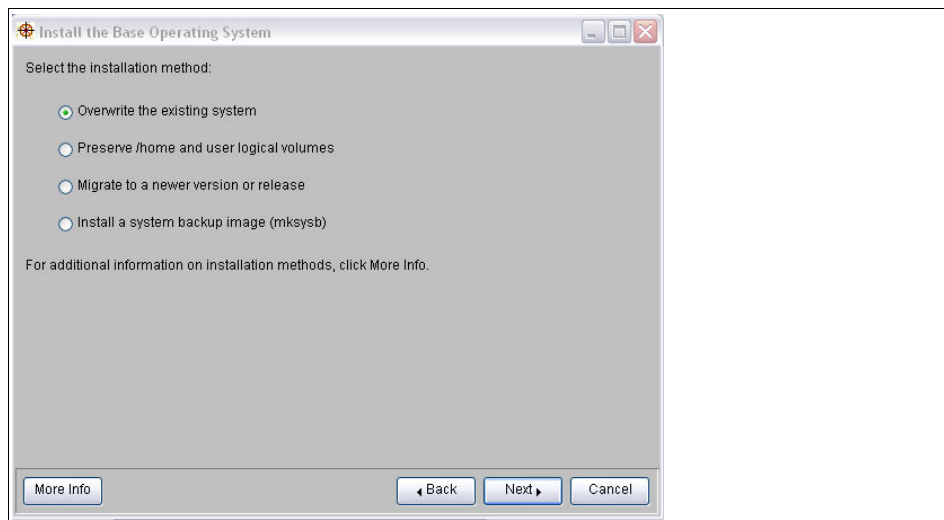


Figure 3-54 Choose the installation method and press next

The next step is to choose the SPOT resource you want to use for the installation. In our case, we defined only one SPOT resource, therefore we have only one option in the list as shown in Figure 3-55 on page 111. In your case, you



might have multiple SPOT resources according to the number of SPOT resources you defined.

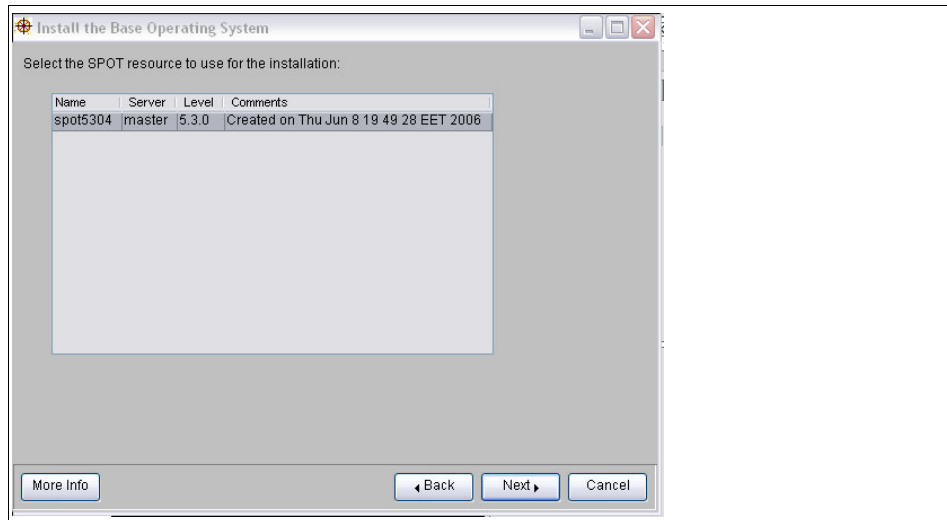


Figure 3-55 Choose the SPOT resource that matches the OS you are installing

In Figure 3-55 we selected spot5304. After pressing "Next", the following step allows you to choose the lpp\_source resource that is relevant to the operating system you are installing. See Figure 3-56.

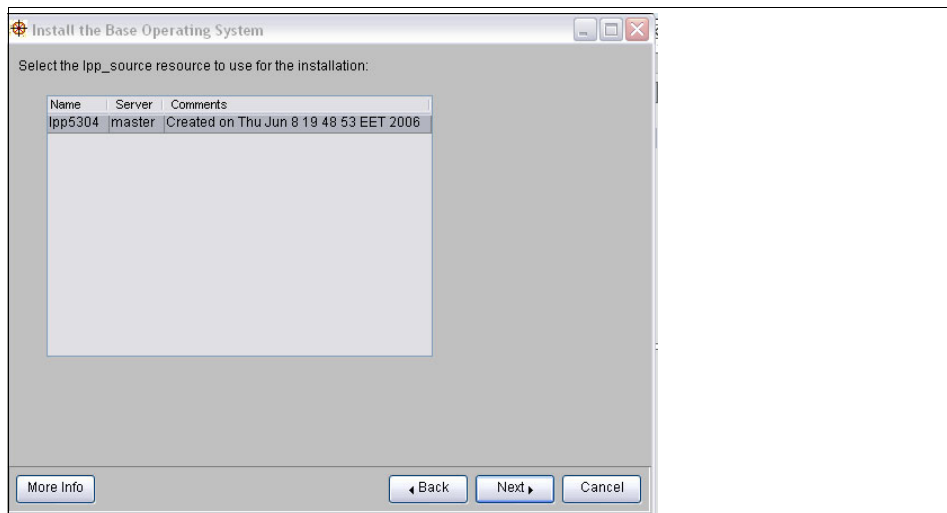


Figure 3-56 Choose the lpp\_source

Select the `lpp_source` resource that matches the operating system you are installing and press “Next”. See Figure 3-56.

In Figure 3-57 on page 112, you can choose to undertake a prompted or a non-prompted installation. The screen presents three options, the first and second options are for a non-prompted installation, while the third option is for a prompted installation.

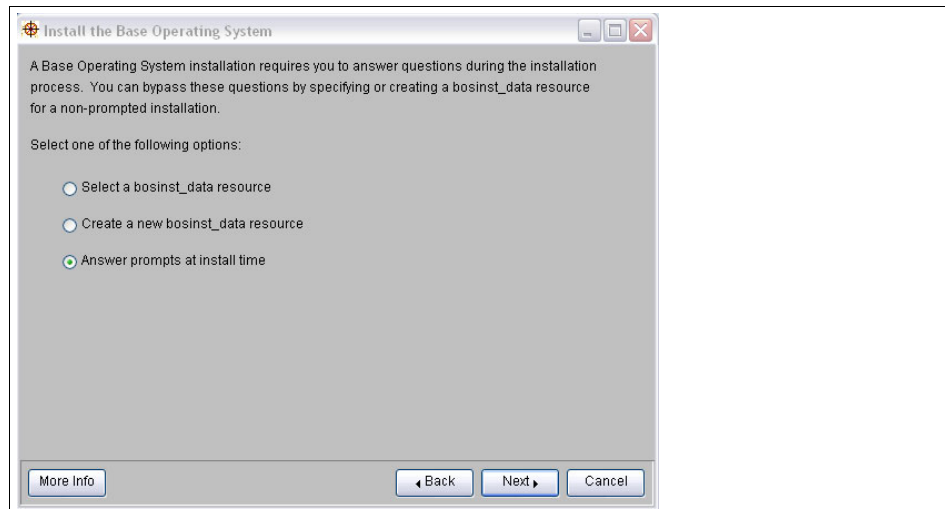


Figure 3-57 Choose between a prompted or a non-prompted installation

In Figure 3-57 you can:

- ▶ Choose “**Select a bosinst\_data resource**”, if you have defined this resource earlier. We recommend to define a “bosinst\_data” resource for the installation scenario you need.
- ▶ Choose “**Create a new bosinst\_data resource**”, if you did not defined one earlier.
- ▶ Choose “**Answer prompts at install time**” if you want to go through defining the installation options manually. Installation options include the console, target LUNs or disks, and BOS Install language.

We choose the third option as we wanted to go through the installation options manually. However, for large environments, we recommend to use a standard `bosinst_data` resource to maintain consistency in your environment.

The next screen (Figure 3-58 on page 113) allows you to perform further customization, such as installing additional software packages and run a customization script.

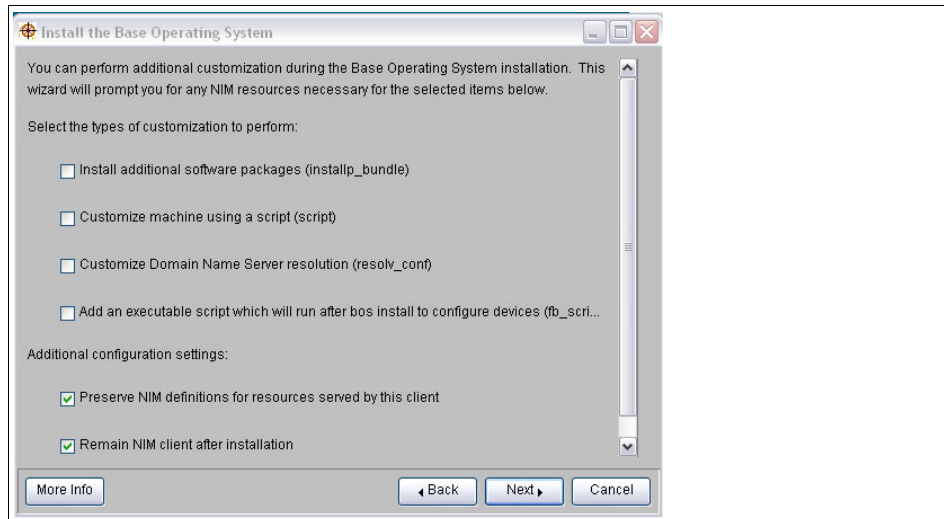


Figure 3-58 Choose the final customization of the installation process

Select the options you want and press “Next” to arrive at the screen shown in Figure 3-59 on page 113.

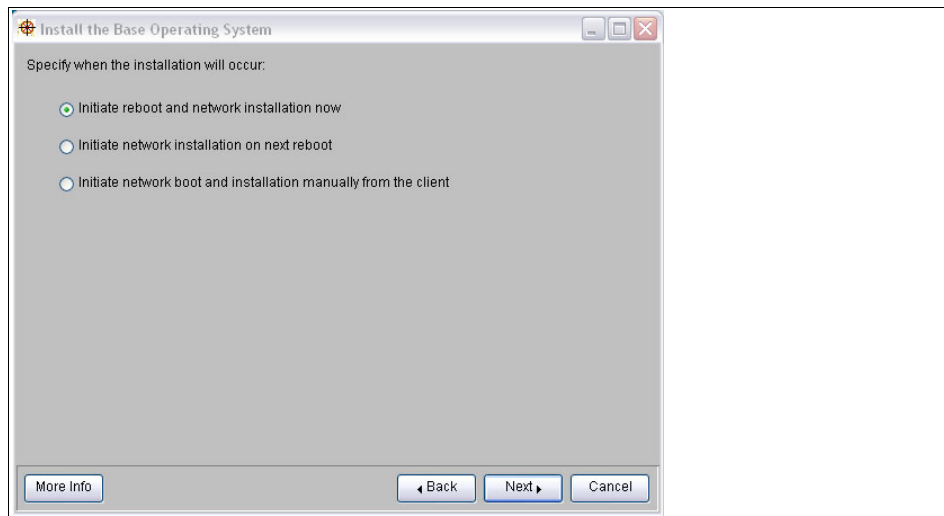


Figure 3-59 Choose the initiate reboot option.

In Figure 3-59 you can choose to start the installation immediately by initiating a client reboot from the NIM master, initiate the installation at the next reboot, or initiate the reboot from the client.

We chose to initiate the reboot and install immediately.

After you press “Next”, the screen with the installation process is shown (see Figure 3-60). In this screen you need to choose to accept the license agreements.

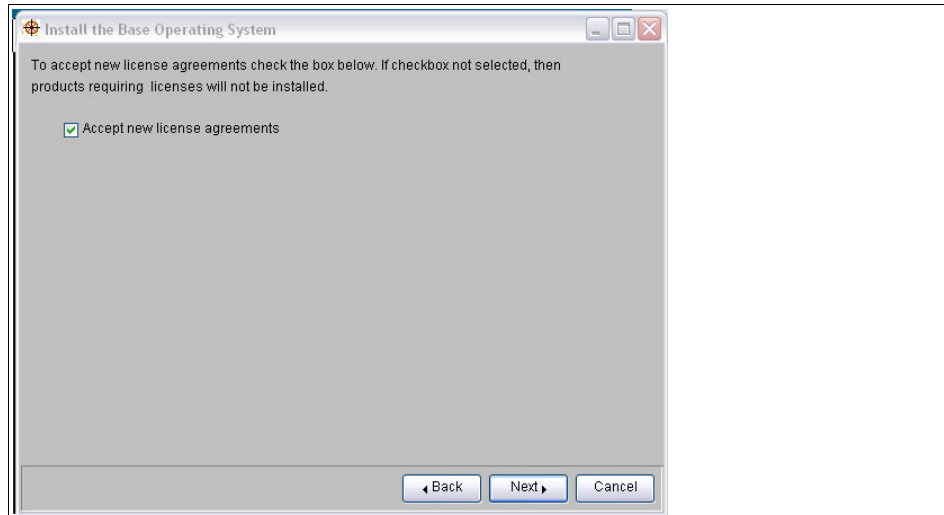


Figure 3-60 Choose to select the accept new license agreements

After you press “Next”, the installation process will start according to the option you defined.

## 3.3 EZ NIM

This section shows you how to use the EZ NIM command scripts to configure the NIM master, create the basic installation resources required to install NIM client machines, and to configure the NIM client machines. For more information on the EZ NIM, refer to the manual *AIX 5L Version 5.3 Installation and migration*, SC23-4887-03.

### 3.3.1 Configuring a NIM master using `nim_master_setup`

The `nim_master_setup` command installs the `bos.sysmgt.nim.master` fileset (if not installed), configures the NIM master, and creates the required resources for installation, including a `mksysb` system backup. You can also use the SMIT `eznim_master_panel` fast path.

The default device to copy the AIX software from is /dev/cd0, the default volume group to create file systems in is rootvg and the default file system is /export/nim.

The script requires that the TCP/IP is configured using either Token ring or an Ethernet interface. There must be 8 MB free space in the /tmp file system, and the volume group needs at least 1 GB for the NIM file system (/export/nim) and 32 MB for the /tftpboot file system. If the NIM file system or /tftpboot do not exist, they are created. The script will create a basic lpp\_source and a SPOT, but also a generic mksysb resource.

For batch creation of NIM clients, a simple client.defs template configuration file is also created, and placed in the /export/nim directory. The client.defs file can be used by the nim\_clients\_setup script.

Example 3-41 assumes that all the AIX software has been saved in the /usr/sys/inst.images directory, for example by using the **bffcreate** command (see “Configuring the NIM master using the nimconfig command” on page 57):

```
# bffcreate -d /dev/cd0 -t /usr/sys/inst.images all
```

---

*Example 3-41 Configuring the NIM master using the nim\_master\_setup command*

```
root@master:/: nim_master_setup
```

```
##### NIM master setup #####
#
# During script execution, lpp_source and spot resource creation times
# may vary. To view the install log at any time during nim_master_setup,
# run the command: tail -f /var/adm/ras/nim.setup in a separate screen.
#
#####
Creating image.data file...done
```

```
Device location is /dev/cd0
Resources will be defined on volume group rootvg
Resources will exist in filesystem /export/nim
Checking for backup software...already installed
Checking /tmp space requirement...done
Installing NIM master fileset...already installed
Defining NIM master...
0513-071 The nimesis Subsystem has been added.
0513-071 The nimd Subsystem has been added.
0513-059 The nimesis Subsystem has been started. Subsystem PID is 663582.
0042-001 nim: processing error encountered on "master":
0042-023 m_chnet: "default " is not a valid NIM routing stanza
```

```

| Located volume group rootvg.
| Creating /export/nim filesystem...File system created successfully.
| 1474308 kilobytes total disk space.
| New File System size is 2949120
| done
| Creating /tftpboot filesystem...File system created successfully.
| 32560 kilobytes total disk space.
| New File System size is 65536
| done
| Checking /export/nim space requirement...done

```

```

| Creating list of files to back up
| Backing up 24085 files....
| 24085 of 24085 files backed up (100%)
| 0512-038 mksysb: Backup Completed Successfully.
| Creating mksysb resource 5300-04master_sysb...done
| Creating bosinst_data resource 5300-04bid_ow...done

```

```

| Please insert AIX 5.2 product media in device /dev/cd0
| If the location for AIX 5.2 product media differs from
| device /dev/cd0, supply the absolute path BEFORE pressing the ENTER key.
| => /usr/sys/inst.images
| Checking /export/nim space requirement...done
| Creating lpp_source resource 530lpp_res...
| Checking /export/nim space requirement...done
| Checking /tftpboot space requirement...done
| Creating spot resource 530spot_res...
| Creating resource group basic_res_grp...done

```

```

| The following resources now exist:
| boot                resources      boot
| nim_script          resources      nim_script
| 5300-04master_sysb  resources      mksysb
| 5300-04bid_ow       resources      bosinst_data
| 530lpp_res          resources      lpp_source
| 530spot_res         resources      spot

```

```

| NIM master setup is complete - enjoy!

```

---

```

| If you get a message stating "Error creating /export/nim filesystem -
| Exiting.", this means that there is probably not enough space available in the
| volume group. In most cases, if the script fails, you can re-run it without removing
| the created configuration.

```

The default NIM environment created by the `nim_master_setup` script is shown in Example 3-42 (AIX 5L V5.3 TL4 on a host named “nimmaster”).

*Example 3-42 nim\_master\_setup NIM environment*

---

```

master:
  class          = machines
  type           = master
  max_nimesis_threads = 20
  comments       = machine which controls the NIM environment
  platform       = chrp
  netboot_kernel = mp
  if1            = master_net nimmaster 1ABCD000301E
  cable_type1    = N/A
  Cstate         = ready for a NIM operation
  prev_state     =
  Mstate         = currently running
  serves         = 5300-04bid_ow
  serves         = 5300-04master_sysb
  serves         = 5301pp_res
  serves         = 530spot_res
  serves         = boot
  serves         = nim_script
  master_port    = 1058
  registration_port = 1059
  reserved       = yes
boot:
  class          = resources
  type           = boot
  comments       = represents the network boot resource
  Rstate         = ready for use
  location       = /tftpboot
  alloc_count    = 0
  server         = master
  reserved       = yes
master_net:
  class          = networks
  type           = ent
  Nstate         = ready for use
  prev_state     = information is missing from this object's definition
  net_addr       = 10.1.1.0
  snm            = 255.255.255.0
5300-04master_sysb:
  class          = resources
  type           = mksysb

```

```
Rstate      = ready for use
prev_state  = unavailable for use
location    = /export/nim/mksysb/generic_sysb
version     = 5
release     = 3
mod         = 0
oslevel_r   = 5300-04
alloc_count = 0
server      = master
5300-04bid_ow:
class       = resources
type        = bosinst_data
Rstate      = ready for use
prev_state  = unavailable for use
location    = /export/nim/5300-04bid_ow
alloc_count = 0
server      = master
530lpp_res:
class       = resources
type        = lpp_source
arch        = power
Rstate      = ready for use
prev_state  = unavailable for use
location    = /export/nim/lpp_source/530lpp_res
simages     = yes
alloc_count = 0
server      = master
530spot_res:
class       = resources
type        = spot
plat_defined = chrp
arch        = power
Rstate      = ready for use
prev_state  = verification is being performed
location    = /export/nim/spot/530spot_res/usr
version     = 5
release     = 3
mod         = 0
oslevel_r   = 5300-04
alloc_count = 0
server      = master
Rstate_result = success
mk_netboot  = yes
mk_netboot  = yes
```

---



### 3.3.2 Configuring NIM clients using `nim_clients_setup`

For the `nim_clients_setup` script to work, the `nim_master_setup` script must have completed properly and there must be a NIM `res_group` named `basic_res_grp` and a `client.defs` file containing client definitions as shown in Example 3-43.

The `nim_clients_setup` command is used to define NIM clients, allocate the installation resources, and initiate a NIM BOS installation on the clients. You can also use the SMIT `eznim_client_panel` fast path.

*Example 3-43 Example client.defs specification file for `nim_clients_setup`*

---

```
# set default values
default:
    machine_type = standalone
    subnet_mask  = 255.255.254.0
    gateway      = 10.1.1.1
    network_type = ent
    cable_type   = tp
    platform     = chrp
    netboot_kernel = mp
```

---

In this scenario, we use the `nimdef` command to create the NIM machines definitions using the `client.defs` file. The `nimdef` command can also be used to display a preview or create a script with NIM commands for later use to create the NIM machines definitions, as shown in Example 3-44.

*Example 3-44 Using `nimdef` to create NIM machines from a `client.defs` file*

---

```
root@master:/export/nim: nimdef -df client.defs
```

Summary

- 1 Machine will be added to the NIM environment.
- 1 Machine group will be created with new members.
- 1 Network will be added to the NIM environment automatically.
- 1 Network will have new NIM machine interfaces added.

---

```
+ nim -o define -t standalone -a if1=find_net lpar55 0 ent -a cable_type1=tp
-anet_definition=ent 255.255.254.0 10.1.1.1 -a netboot_kernel=mp -a platform=chrp
v\lpar2
```

---

By adding the `-c` flag (Example 3-45), the `nimdef` command creates a Korn Shell script and displays it to the standard output.

*Example 3-45 Using the nimdef command to create a script from a client.defs file*

---

```

root@master:/export/nim: nimdef -cf client.defs

#!/bin/ksh
set -x
#####
#
# Summary
#
# 1 Machine will be added to the NIM environment.
# 1 Network will be added to the NIM environment automatically.
# 1 Network will have new NIM machine interfaces added.
#
#####

#####
#
# Commands to define new machines in the NIM environment.
#
#####
nim -o define -t standalone -a if1="find_net lpar55 0 ent" -a cable_type1=tp
-anet_definition="ent 255.255.254.0 10.1.1.1 " -a netboot_kernel=mp -a platform=chrp
v1par2

```

---

If the basic\_res\_grp NIM resource has not been created by the nim\_master\_setup script, it can be created manually as shown in Example 3-46:

*Example 3-46 Creating basic\_res\_grp (manually)*

---

```

root@master:/: nim -o define -t res_group -a bosinst_data=5300-04bid_ow -a
lpp_source=530lpp_res -a spot=530spot_res -a mksysb=5300-04master_sysb basic_res_grp

root@master:/: lsnim -l basic_res_grp
basic_res_grp:
  class   = groups
  type    = res_group
  member1 = 5300-04master_sysb
  member2 = 5300-04bid_ow
  member3 = 530lpp_res
  member4 = 530spot_res

```

---

To allocate and initiate client installation, use the **nim\_clients\_setup** command. We use the **nim\_clients\_setup** command in this case is to allocate the installation resources, and initiate a NIM BOS installation on the clients. To

initiate a change of the NIM client's boot sequence to use network as the first boot device, and to reboot the client, the “-r” option must be used with the `nim_clients_setup` command. Refer to Example 3-47.

*Example 3-47 Using the `nim_clients_setup` script*

---

```
root@master:/export/nim: nim_clients_setup -r
nim_clients_setup
NSORDER=local,bind
```

```
Generating list of client objects in NIM environment...
Locating lpar55...done
```

```
Checking for resource group basic_res_grp...done
```

---

The NIM setup for installing the NIM client is now completed, and the client can be restarted to initiate network boot and BOS installation. Example 3-48 shows the NIM client object and the `/etc/bootptab` entry.

*Example 3-48 Verifying after the `nim_clients_setup` script*

---

```
root@master:/export/nim: lsnim -l v1par2
v1par2:
  class      = machines
  type       = standalone
  default_res = basic_res_grp
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = master_net lpar55 0 ent
  cable_type1 = tp
  Cstate     = BOS installation has been enabled
  prev_state = ready for a NIM operation
  Mstate     = currently running
  boot       = boot
  bosinst_data = 5300-04bid_ow
  lpp_source  = 530lpp_res
  mksysb     = 5300-04master_sysb
  nim_script  = nim_script
  spot       = 530spot_res
  cpuid      = 0000C836D700
  control    = master
```

```
root@master:/: grep lpar55 /etc/bootptab
```

| lpar55:bf=/tftpboot/lpar55:ip=10.1.1.62:ht=ethernet:sa=10.1.1.2:sm=255.  
255.255.0:

---



# Network Installation Manager (NIM) scenarios

This chapter discusses several scenarios and implementation for a NIM environment. As NIM is a highly flexible, feature rich set of tools, these scenarios are considered just a starting point to the real NIM usability. Scenarios covered in this chapter describe:

- ▶ High Availability Network Installation Manager (NIM)
- ▶ Using the OS\_install command
- ▶ NIM environment with additional resource servers
- ▶ Using NIM to perform AIX migrations
- ▶ NIM mkysyb migration and nim\_move\_up POWER5 tools
- ▶ NIM alternate disk migration
- ▶ Network Installation Manager (NIM) and Linux distributions
- ▶ Thin server and common OS image management (COSI)
- ▶ Using HACMP with HA NIM
- ▶ NIM and Service Update Management Assistant (SUMA)
- ▶ Backing up clients with NIM
- ▶ How to create bundles, BFF, and RPM packages
- ▶ NIM in a micro-partition using Virtual I/O (VIO) resources
- ▶ Setting up the VIO server and configuring the NIM master in the VIO client
- ▶ Backup and restore of the VIO Server using NIM
- ▶ Using RSCT Peer Domain with HANIM

## 4.1 High Availability NIM (HA NIM)

AIX 5L V5.3 introduces High Availability NIM (HA NIM) by allowing a backup NIM master on a different machine or LPAR. The backup NIM master takes over when the primary NIM master fails. When the primary NIM master recovers, a fallback operation gives control to the primary NIM master. The failover and fallback operations are initiated by the system administrator.

Unlike a common HA environment, NIM does not perform any heartbeat, nor provides file system takeover. HA NIM only provides a method for replicating NIM database and resources, and a failover/fallback procedure in a coordinated manner.

In order to have the latest NIM configuration when the backup NIM master takeover, the system administrator has to perform regular synchronization of the NIM configuration from the primary master to the backup master.

The synchronization process backs up the NIM database on the primary NIM master and copies it to the backup NIM master. The backup is then restored in the backup master.

**Note:** At the time of this document was created, only rsh/rshd communication is supported for NIM synchronization.

To configure HA NIM, use the following SMIT sequence:

```
# smitty nim
   Perform NIM Administration Tasks
   Manage Alternate Master Environment
```

Or, you can use the **smitty nim\_altmstr** SMIT fast path. Example 4-1, “SMIT screen on Alternate Master Environment” shows the SMIT screen on the Alternate NIM master Environment.

*Example 4-1 SMIT screen on Alternate Master Environment*

---

Manage Alternate Master Environment

Move cursor to desired item and press Enter.

```
Initialize This Machine as an Alternate Master
Define Another Machine as an Alternate Master
Synchronize an Alternate Master's NIM database
Takeover control of NIM clients from an Alternate Master
```

## Remove an Alternate Master

### 4.1.1 Configuring the alternate (backup) NIM master

In our scenario, we have a primary NIM master, lpar1, and we step through the configuration to create the alternate NIM master, which in our case is lpar6. This is shown in Figure 4-1 on page 125.

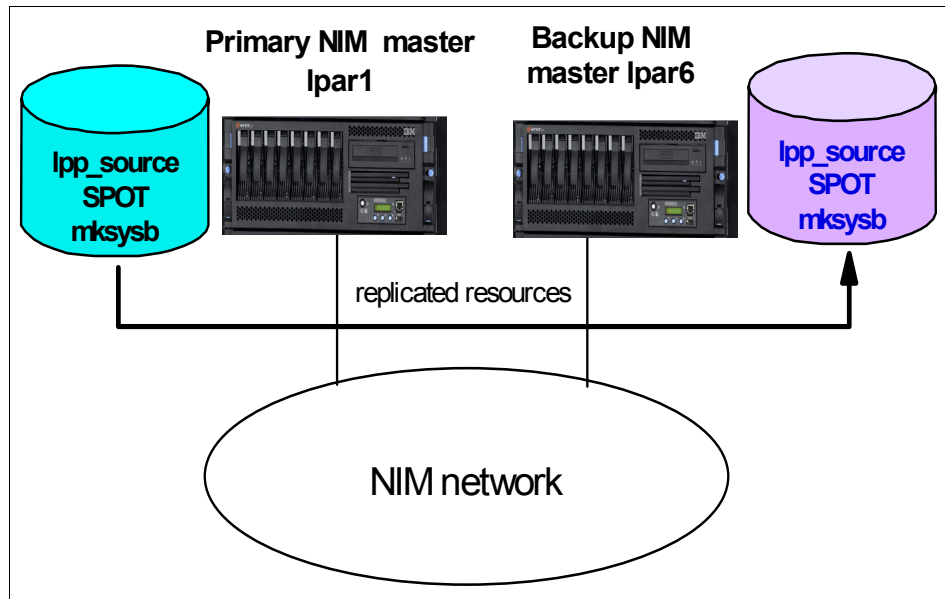


Figure 4-1 High availability NIM

The configuration of the alternate NIM master follows a process similar to the normal NIM master setup:

1. Install the NIM master fileset on the backup NIM master server, lpar6

You need to install the following filesets:

```
bos.sysmgt.nim.master
bos.sysmgt.nim.spot
```

**Note:** The bos.sysmgt.nim.spot package is optional unless you have diskless or dataless clients.

2. Configure the backup NIM master server as the NIM master

On the backup NIM master server, lpar6, you need to configure the NIM environment, just like in the primary NIM master server.

```
# smitty nimconfig
```

**Attention:** Before you create lpar6 as the backup NIM master, make sure that the lpar6 is NOT already a NIM client for the primary NIM master. If so, you need to remove the NIM client definition from the primary NIM master.

3. Copy the lpp\_source to the backup NIM master.

You need to create the same file system structure for the lpp\_source as in the primary NIM master server.

Copy all the lpp\_source directory to the backup NIM master:

From the Primary NIM master,

```
# cd /export/lppsource/lpp-aix5304
# find . -print | backup -iqvf- |rsh lpar6 -l root "cd \
/export/lppsource/lpp-aix5304 ;restore -xqvf-"
```

**Attention:** We recommend you to recreate the .toc file using the `inutoc` command on `/export/lppsource/lpp-aix5304/installp/ppc` directory. Also check that the files have read permission on other user, `chmod 644`.

4. Create the lpp\_source resource on lpar6 as shown in Example 4-2.

```
# smitty nim_mkres
```

- Select “lpp\_source = source device for optional product images”

**Note:** You might not need to create the resource. NIM synchronization process at the later part creates the resource if the lpp\_source is in place. But we recommend to create the resource manually in the backup NIM master.

#### Example 4-2 Defining the lpp\_source resource

---

Define a Resource

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

	[Entry Fields]
* Resource Name	[lpp-aix5304]
* Resource Type	lpp_source
* Server of Resource	[master]



```

* Location of Resource          [/export/lppsource/lpp-aix5304> /
Architecture of Resource      []
Source of Install Images      [] +/
Names of Option Packages      []
Show Progress                  [yes] +
Comments                       []

```

---

You will need to repeat the lpp\_source resource creation if you have more than one resource.

5. Create the SPOT resource on lpar6 as shown in Example 4-3.

```

# smitty nim_mkres
  Select "spot = Shared Product Object Tree - equivalent to /usr
  file"

```

*Example 4-3 Creating the SPOT resource on lpar6*

---

Define a Resource

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

```

                                                    [Entry Fields]
* Resource Name          [spot-aix5304]
* Resource Type          spot
* Server of Resource     [master] +
* Source of Install Images [lpp-aix5304] +
* Location of Resource   [/export/spot] /
Expand file systems if space needed? yes +
Comments                 []

installp Flags
COMMIT software updates? no +
SAVE replaced files?     yes +
AUTOMATICALLY install requisite software? yes +
OVERWRITE same or newer versions? no +
VERIFY install and check file sizes? no +

```

---

**Note:** We recommend to create the SPOT from the previously defined lpp\_source resource, instead of letting the NIM synchronization process create the SPOT. Make sure that you use the same directory as on the primary NIM master.

6. Copy mksysb images, bosinst.data, image.data, firstboot script, /etc/host files  
 You need to copy all the images, for example, bosinst.data, image.data first-boot script files that belong to the NIM resources from the primary NIM master server. The NIM synchronization process creates these resources in the backup NIM master server if the images or files exist. You also need to copy the /etc/hosts file.

## 4.1.2 Initialize the backup server as the alternate NIM master server

After having performed the necessary tasks in section 4.1.1, “Configuring the alternate (backup) NIM master” on page 125, we can initialize the backup server as the alternate NIM master.

We initialize the alternate NIM master from the backup NIM master. This updates the NIM database on the primary NIM master and updates the backup NIM master’s /etc/niminfo file.

In our example, the backup server is lpar6. Follow the next step and see Example 4-4.

Start on lpar6 with the following action:

```
# smitty nim_almstr
   Initialize This Machine as an Alternate Master
```

*Example 4-4 Initializing from the backup NIM master server*

---

```
   Initialize This Machine as an Alternate Master
```

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

```

* This Machine Name                [EntryFields] [lpar6]
* Primary Network Install Interface [en0] +
* Host Name of Master with which to Initialize [lpar1]

Hardware Platform Type             chrp
Kernel to use for Network Boot     [mp] +
Communication Protocol used to communicate with Alternate Master [] +
Comments                           []

Alternate Port Numbers for Network Communications
(reserved values will be used if left blank)
Client Registration                 [] #
```

Once initialization is finished, the NIM database on the primary NIM master is updated, as shown in Example 4-5.

*Example 4-5 lsnim output on primary NIM master*

---

```
lpar1:/ > lsnim -c machines
master      machines      master
lpar7       machines      standalone
lpar6      machines      alternate_master

lpar1:/ > lsnim -l master
master:
  class           = machines
  type            = master
  max_nimesis_threads = 20
  if_defined      = chrp.mp.ent
  comments        = machine which controls the NIM environment
  platform        = chrp
  netboot_kernel  = mp
  if1             = mont_net01 lpar1 0
  cable_type1     = tp
  Cstate          = ready for a NIM operation
  prev_state      = ready for a NIM operation
  Mstate          = currently running
  serves          = BOSINST-lpar7
  serves          = IMAGE-lpar7
  serves          = lpp-aix5304
  serves          = MK-lpar7
  serves          = spot-aix5304
  serves          = alt_disk_install_bnd
  serves          = boot
  serves          = nim_script
  master_port     = 1058
  registration_port = 1059
  reserved        = yes
  is_alternate   = yes
```

---

The `/etc/niminfo` file in the `lpar6` server is updated with the field “export NIM\_ALTERNATE\_MASTER=lpar1”, as shown in Example 4-6.

*Example 4-6 /etc/niminfo file in lpar6 server*

---

```
lpar6:/ > nimconfig
```

```

export NIM_NAME=master
export NIM_CONFIGURATION=master
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_MASTER_HOSTNAME=lpar6
export NIM_ALTERNATE_MASTER="lpar1"

```

---

**Note:** At this point, the NIM database in the backup server, lpar6, does not update lpar1 as its alternate NIM master. This will only be updated after the NIM synchronization is executed.

### 4.1.3 Perform NIM synchronization

NIM synchronization must only be performed after you have completed sections 4.1.1, “Configuring the alternate (backup) NIM master” on page 125 and 4.1.2, “Initialize the backup server as the alternate NIM master server” on page 128. If any of the resource files are not copied to the backup NIM master server, the NIM synchronization process will not create the resource.

From the primary NIM master server (lpar1), follow these steps (Example 4-7):

```

# smitty nim_almstr
  Synchronize an Alternate Master's NIM database
    Select the Backup NIM master server

```

*Example 4-7 Synchronizing an alternate master's NIM database*

---

Synchronize an Alternate Master's NIM database

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

	[Entry Fields]
* Target Name	lpar6
Force	yes +

---

**Note:** You need to force the synchronization. If not, the operation will not be successful because it complains that lpar6 is already a NIM master.

In the future updates, the NIM synchronization will copy the respective resource files and/or directories. For example, the lpp\_source directory will be copied over to the alternate NIM master. You do not need to manually copy them over as explained in 4.1.1, “Configuring the alternate (backup) NIM master” on page 125.

Example 4-8 shows the output when the NIM synchronization is performed. We noticed that the alternate\_disk\_install installp\_bundle resource is not created because the alternate\_disk\_install bundle file was not copied to the backup NIM master. To fix this, copy the alternate\_disk\_install bundle file to the backup NIM master and perform the NIM synchronization again.

*Example 4-8 NIM synchronization output*

---

```
a ./etc/objrepos/nim_attr 16 blocks.
a ./etc/objrepos/nim_attr.vc 16 blocks.
a ./etc/objrepos/nim_object 8 blocks.
a ./etc/objrepos/nim_object.vc 8 blocks.
a ./etc/NIM.level 1 blocks.
a ./etc/niminfo 2 blocks.
The original NIM database was backed up to the
  following location prior to this operation:
    "/export/nim/backups/lpar6.18013606082006.backup"
```

```
0513-044 The nimesis Subsystem was requested to stop.
0513-004 The Subsystem or Group, nimd, is currently inoperative.
0513-083 Subsystem has been Deleted.
0513-083 Subsystem has been Deleted.
0518-307 odmdelete: 14 objects deleted.
0518-307 odmdelete: 130 objects deleted.
Restoring the NIM database from /tmp/_nim_dir_233622/mnt0
x ./etc/NIM.level, 9 bytes, 1 media blocks.
```

```
The level of the NIM master fileset on this machine is: 5.3.0.40
The level of the NIM database backup is: 5.3.0.40
```

Level check is successful.

```
x ./etc/objrepos/nim_attr, 8192 bytes, 16 media blocks.
x ./etc/objrepos/nim_attr.vc, 8192 bytes, 16 media blocks.
x ./etc/objrepos/nim_object, 4096 bytes, 8 media blocks.
x ./etc/objrepos/nim_object.vc, 4096 bytes, 8 media blocks.
x ./etc/NIM.level, 9 bytes, 1 media blocks.
x ./etc/niminfo, 162 bytes, 1 media blocks.
0513-071 The nimesis Subsystem has been added.
0513-071 The nimd Subsystem has been added.
0513-059 The nimesis Subsystem has been started. Subsystem PID is
315556.
Finished restoring the NIM database
Updating master definition in database from lpar6definition
  Updated master attribute platform to chrp
```

```
Updated master attribute netboot_kernel to mp
Updated master attribute if1 to mont_net01 lpar6 000255AF6C06 ent0
Updated master attribute cable_type1 to N/A
Finished updating master definition
Resetting machines
  Reset master
  Reset lpar7
  Reset lpar6
Finished resetting machines
Removing NIM client lpar6
Finished removing lpar6
Resetting NIM resources
Finished resetting NIM resources
Checking NIM resources
  Keeping lpp-aix5304
  Keeping spot-aix5304
Removing alt_disk_install_bnd
  0518-307 odmdelete: 1 objects deleted. from nim_attr (serves attr)
  0518-307 odmdelete: 0 objects deleted. from nim_attr (group
memberships)
0518-307 odmdelete: 6 objects deleted. from nim_attr (resource
attributes)
  0518-307 odmdelete: 1 objects deleted. from nim_object (resource
object)
  Finished removing alt_disk_install_bnd
  Keeping MK-lpar7
  Keeping BOSINST-lpar7
  Keeping IMAGE-lpar7
Finished checking NIM resources
Checking NIM SPOTs
checking spot-aix5304
Finished checking SPOTs
nim_master_recover Complete
```

---

**Note:** You SHOULD \*NOT\* at any time perform a NIM synchronization from the backup (alternate) NIM master. NIM synchronization \*MUST\* always be done at the primary NIM master. You need to be extra careful when you perform the synchronization.

We recommend you to perform the synchronization any time there's a change in the NIM database, or you can perform regular synchronization via crontab. You can add the following NIM synchronization command into the cron jobs table on lpar1 (you can use the **crontab -e** command).

```
# nim -o sync -a force=yes lpar6
```

After the NIM synchronization is done, the NIM database of the backup NIM master is updated. The lpar1 is updated as the alternate NIM master.

Example 4-9 shows the `lsnim` output in the backup NIM master. A “master\_alias” field is added in the NIM database of the lpar6.

*Example 4-9 lsnim output from lpar6 server showing its alternate NIM master*

---

```
lpar6:/ > lsnim -c machines
master          machines          master
lpar7           machines          standalone
lpar1          machines          alternate_master
```

```
lpar6:/ > lsnim -l master
master:
  class          = machines
  type           = master
  max_nimesis_threads = 20
  if_defined     = chrp.mp.ent
  comments       = machine which controls the NIM environment
  platform       = chrp
  netboot_kernel = mp
  if1            = mont_net01 lpar6 0
  cable_type1    = tp
  Cstate         = ready for a NIM operation
  prev_state     = ready for a NIM operation
  Mstate         = currently running
  serves         = BOSINST-lpar7
  serves         = IMAGE-lpar7
  serves         = lpp-aix5304
  serves         = MK-lpar7
  serves         = spot-aix5304
  serves         = boot
  serves         = nim_script
  master_port    = 1058
  registration_port = 1059
  reserved       = yes
  master_alias  = lpar6
```

---

We can also see that the `/etc/niminfo` file in the backup NIM master is again updated, as shown in Example 4-10.

*Example 4-10 /etc/niminfo file in lpar6 server after NIM synchronization*

---

```
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
# machine which controls the NIM environment
export NIM_NAME=master
export NIM_HOSTNAME=lpar6
export NIM_CONFIGURATION=master
export NIM_MASTER_HOSTNAME=lpar6
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.16:lpar6 "
export NIM_MOUNTS=""
export NIM_ALTERNATE_MASTER="lpar1"
```

---

#### 4.1.4 NIM master takeover

If you have problem on the primary NIM master and need to activate the alternate NIM master, you can initiate the takeover from the backup server.

**Note:** Remember to check that the *rsh* communication between the primary and alternate master and the clients is enabled.

From the alternate (backup) server, lpar6, follow these steps (see Example 4-11):

```
# smitty nim_almstr
  Takeover control of NIM clients from an Alternate Master
  Select the Primary Server
```

*Example 4-11 Take over from the backup server*

---

Takeover control of NIM clients from an Alternate Master

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

	[Entry Fields]
* Target Name	lpar1
Force	yes +

---



During the takeover process, the backup server attempts to contact the NIM client to change its controlling master. The clients' /etc/niminfo file will be updated.

The **NIM\_MASTER\_HOSTNAME**, **NIM\_MASTER\_HOSTNAME\_LIST** and the **NIM\_HOSTS** fields in the /etc/niminfo reflect the correct NIM master which this client reports to. Example 4-12 shows the original /etc/niminfo file:

*Example 4-12 NIM client's original /etc/niminfo file*

---

```
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=lpar7
export NIM_HOSTNAME=lpar7
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=lpar1
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_MASTER_HOSTNAME_LIST="lpar1 lpar6"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.74:lpar7 10.1.1.11:lpar1 "
export NIM_MOUNTS=""
```

---

Example 4-13 shows the updated /etc/niminfo file after the NIM takeover. The changes are shown in highlighted text.

*Example 4-13 Updated NIM client's /etc/niminfo file after the takeover*

---

```
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=lpar7
export NIM_HOSTNAME=lpar7
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=lpar6
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_MASTER_HOSTNAME_LIST="lpar6 lpar1"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.74:lpar7 10.1.1.16:lpar6 "
```

---

```
export NIM_MOUNTS=""
```

---

If the NIM client cannot be reached, the backup server updates its database with the “sync\_required” to yes. Example 4-14 shows the result when lpar6 is not able to reach lpar7 during the takeover.

*Example 4-14 lsnim output with sync\_required set to yes on NIM client -lpar7*

---

```
lpar6:/ > lsnim -l lpar7
lpar7:
  class      = machines
  type       = standalone
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = mont_net01 lpar7 0
  net_settings1 = 100 full
  cable_type1 = tp
  Cstate     = ready for a NIM operation
  prev_state = not running
  Mstate     = not running
  Cstate_result = reset
  current_master = lpar6
  sync_required = yes
```

---

The alternate (backup) server also attempts to contact the primary server and update it. However, if the primary server is unreachable, the backup server will update its database with the “sync\_required” to yes. Example 4-15 shows the result when lpar6 is not able to reach lpar1 during the takeover.

Once every NIM client that has been updated without any problem (/etc/niminfo file is updated), the backup NIM master is ready to perform NIM operations as a NIM master.

*Example 4-15 Alternate NIM master lsnim output (sync\_required = yes).*

---

```
lpar6:/ > lsnim -l lpar1
lpar1:
  class      = machines
  type       = alternate_master
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = mont_net01 lpar1 0
  cable_type1 = N/A
  Cstate     = ready for a NIM operation
```

```

prev_state    = ready for a NIM operation
Mstate       = currently running
sync_required = yes

```

---

### 4.1.5 NIM master fallback

Once the original (primary) NIM master has been restored for operation, you can fallback the NIM master control from the backup (alternate) server. You have to follow the same steps described in section 4.1.4, “NIM master takeover” on page 134, but performing the commands from the original (primary) NIM master.

From the original primary server, lpar1 (as shown in Example 4-16), follow these steps:

```

# smitty nim_almstr
  Takeover control of NIM clients from an Alternate Master
  Select the Backup Server

```

#### *Example 4-16 Takeover control of NIM clients*

---

```

Takeover control of NIM clients from an Alternate Master

```

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

	[Entry Fields]
* Target Name	lpar6
Force	yes +

---

## 4.2 Using the OS\_install command

The `OS_install` command can be used to install both AIX 5L clients and Linux clients. To install Linux clients it is required that CSM is installed on the installation server (NIM master).

**Note:** For supporting Linux installation using CSM, there are some new NIM objects such as `linux_source` and `linux_inst`. In this initial release of Linux installation support from NIM, Linux clients will not be able to function as a NIM client in the same capacity as AIX NIM clients are able to.

Since the Linux installation mechanism depends on DHCP, a NIM standalone client that is to be installed with Linux must, in addition to what is required for AIX network installation, have specified a MAC address. In addition, the client and the master must reside in the same subnet.

Also, NIM currently has no capabilities to perform additional system management or bare-metal installation of Linux standalone clients. After an allocation is performed, the client must be manually initiated for network installation using the boot parameters displayed after a successful allocation of a `linux_source` resource.

The supported Linux distributions for installation are Redhat Enterprise Linux (RHEL) V3 and V4, and SUSE Linux Enterprise Server (SLES) V9.

To use the `OS_install` command, you need to first install the `libxml2` RPM and then the `osinstall` RPM, as shown in Example 4-17. The RPM packages are located in `RPMS/ppc` directory of your NIM `lpp_source`.

```
# rpm -ihv libxml2-2.6.17-3.aix5.1.ppc.rpm
# rpm -ihv osinstall-1.0-1.aix5.3.noarch.rpm
```

*Example 4-17 Installed files from the osinstall RPM package*

---

```
root@master:/: rpm -ql osinstall
/opt/osinstall/pm/OSInstall.pm
/opt/osinstall/pm/OSInstall/AIX_Resource.pm
/opt/osinstall/pm/OSInstall/Client.pm
/opt/osinstall/pm/OSInstall/Common.pm
/opt/osinstall/pm/OSInstall/Control_Host.pm
/opt/osinstall/pm/OSInstall/Linux_Resource.pm
/opt/osinstall/pm/OSInstall/OS_Resource.pm
/opt/osinstall/pm/OSInstall/Platform_Tools.pm
/opt/osinstall/pm/OSInstall/XMLHandler.pm
/opt/osinstall/schema/nim1.xsd
```

```
/usr/sbin/OS_install  
/usr/sbin/niml_bootreplyd
```

---

You also need to have OpenSSH and OpenSSL installed on the master to perform netboot operations.

```
rpm -i openssl-0.9.7g-2.aix5.1.ppc.rpm  
installp -d openssl.base all  
installp -d openssl.license all
```

With the **OS\_install** command, it is also possible to use W3C XML to describe the NIM environment for the client installation. The XSD (Extensible Markup Language Schema Definition) specification can be found in the `/opt/osinstall/schema/niml.xsd` file.

The network install schema consists of four fundamental elements: `clients`, `os_resources`, `resource_servers`, and `ctrl_hosts`. In addition, there are also the general `osinstall_config` objects.

## 4.2.1 OS\_install basics

**OS\_install** lets you define a `client` or resource, allocate resources to clients and netboot clients. You can also use the SMIT `nim_linux_inst` fastpath to perform Linux client installations with **OS\_install**.

**Note:** The **OS\_install** command use the `/var/osinstall` directory as root of its file system.

To get started with **OS\_install**, you can use the following procedure:

7. Define installation resources.
8. Define controlling hosts for clients.
9. Define clients.
10. List defined objects.
11. Allocate resources to clients.
12. Perform netboot operations on clients.

Figure 4-2 on page 140 shows the NIM `linux_source` input panel for the SMIT `nim_mkres` fastpath selecting `linux_source` (resource containing Linux installation images) as the Resource Type.

Define a Resource	
Type or select values in entry fields. Press Enter AFTER making all desired changes.	
	[Entry Fields]
* Resource Name	[linux_source]
* Resource Type	linux_source
* Server of Resource	[master] +
* Location of Resource	[/export/linux_source] /
* Source of Install Images	[cd0] +/
* Linux Distribution Name	[RedHatEL-AS] +
* Linux Distribution Version	[ ]
* Linux Distribution Service Level	[GA] +
Custom Configuration File Template	[ ] /
Comments	[ ]

Figure 4-2 SMIT screen for selecting linux\_source as Resource\_Type

## 4.2.2 OS\_install resource definition

Defining an **OS\_install** resource is similar to creating a SPOT or COSI, see “Creating a NIM spot object” on page 72 and 4.8, “Common OS image management (COSI)” on page 299. Before you start, you need to have AIX BOS filesets and AIX 5L V5.3 TL5 filesets available.

In Example 4-18, we have copied the filesets to the /export/lpp\_source/lpp5305 directory. Then you need to decide on how to name your resource, in our example we name the resource BOS5305. Then you must specify the operating system type and the version, in our case AIX and 53TL5 respectively.

To create **OS\_install** resource objects you use the `define_resource` operation:

```
OS_install -o define_resource -a attr=value ... <object_name>
```

Required attributes: type, version

Optional attributes: source, location, configfile

### Example 4-18 OS\_install define\_resource operation

```
root@master:/: OS_install -o define_resource -a location=/export/lpp_source/bos5305
-a source=/export/lpp_source/lpp5305 -a type=AIX -a version=53TL5 BOS5305
```

```
mkdir /export/lpp_source/bos5305
```

```
Mon Jun 26 14:52:28 2006 Executing: (cd /export/lpp_source/lpp5305 && /usr/bin/tar
-cf - ./) | (cd /export/lpp_source/bos5305 && /usr/bin/tar -xf -)
```

### 4.2.3 OS\_install controlling host definition

Every `OS_install` client needs a controlling host. The controlling host performs netboot operations on the client. In Example 4-19, we use the Hardware Management Console (HMC) for the LPAR (Logical Partition) which will be the controlling host and client respectively. The HMC hostname is `hmc5` and the clients hostname is `lpar55`. In Example 4-19, we define the controlling host (`define_ctrl_host`) to be of the type `hmc` and use `ssh` as the `communications_method` (hence the need to install `ssh`), and we name the controlling host `HMC5`.

To create `OS_install` controlling host objects, you use the `define_ctrl_host` operation:

```
OS_install -o define_ctrl_host -a attr=value ... <object_name>
```

Required attributes: `communication_method`, `hostname`, `type`

*Example 4-19 OS\_install define\_ctrl\_host operation*

---

```
root@master:/: OS_install -o define_ctrl_host -a type=hmc -a hostname=hmc5 -a
communication_method=ssh HMC5
Mon Jun 26 14:20:02 2006 Writing HMC5 to repository
```

---

### 4.2.4 OS\_install client definition

After the controlling host have been defined, we can define the clients. It is possible to define the clients first and then connect them to a controlling host.

Just as for standalone NIM machines we need to know the IP-address (10.1.1.55), but also the MAC-address (22:33:80:00:AB:CD), and the NIM networks part with gateway (10.1.1.254) and subnet mask (255.255.255.0). In addition, we need information about the controlling host (HMC5), the LPAR name (lpar55), profile (normal) and managed system (IBM-595-65C5ABC).

To create `OS_install` client objects, you use the `define_client` operation as shown in Example 4-20 on page 142.

```
OS_install -o define_client -a attr=value ... <object_name>
```

Required attributes: `ip_addr`, `mac_addr`, `gateway`, `subnet_mask`

Optional attributes: `adapter_speed`, `adapter_duplex`, `lpar`, `profile`, `managed_system`, `disk_location`, `ctrl_host`

*Example 4-20 OS\_install define\_client operation*


---

```
root@master:/: OS_install -o define_client -a ip_addr=10.1.1.55 -a
mac_addr=22:33:80:00:AB:CD -a gateway=10.1.1.254 -a subnet_mask=255.255.255.0 -a
ctrl_host=HMC5 -a lpar=lpar55 -a profile=normal -a managed_system=IBM-595-65C5ABC
LPAR55
```

```
mkdir /var/osinstall
mkdir /var/osinstall/service_state
mkdir /var/osinstall/service_state/cfg_save
mkdir /var/osinstall/service_state/cfg_data
mkdir /var/osinstall/lock
Mon Jun 26 14:13:59 2006 OSInstall repository not found, create new
Mon Jun 26 14:13:59 2006 Writing LPAR55 to repositor
```

---

To list **OS\_install** objects (list mode) you use the **-l** flag as shown in Example 4-21.

```
OS_install -l [ -v ] [ -t <object_type> | <object_name> ]
```

The **-v** option shows the XML for the object. Object types are shown in the right column in Example 4-21. See also “OS\_install XML repository” on page 144.

*Example 4-21 OS\_install list objects*


---

```
root@master:/: OS_install -l
HMC5          ctrl_host
LPAR55        client
```

---

## 4.2.5 OS\_install resource allocation and installation

To allocate the resource to install a client using **OS\_install**, you use the **allocate** operation:

```
OS_install -o allocate -a os_resource=<resource_name> <client_name>
Required attributes: resource_name, client_name
```

To install a client using **OS\_install**, you use the **netboot** operation:

```
OS_install -o netboot <client_name>
```

**Note:** Do not use the virtual terminal connection from the HMC to the LPAR, this connection will be used by the netboot operation.

In Example 4-22, we install the previously defined LPAR55 with AIX 5L using **OS\_install**. Since we have not configured this NIM master to be able to log in to



the HMC without supplying a password, we will have to enter it at the password prompt.

*Example 4-22 OS\_install netboot of client*

---

```

root@master:/: OS_install -o netboot LPAR55

Tue Jun 27 16:02:03 2006 Netbooting client LPAR55
Tue Jun 27 16:02:03 2006 Executing: /usr/bin/ssh hscroot@hmc5 'lpar_netboot -f -t ent
-m 22338000ABCD -D -s auto -d auto -S 10.1.1.5 -G 10.1.1.254 -C 10.1.1.55 "lpar55"
"normal" "IBM-595-65C5ABC"'
hscroot@hmc5's password:
Tue Jun 27 16:02:10 2006 STDOUT (/usr/bin/ssh): lpar_netboot: Error : Sending Force
close..
Tue Jun 27 16:02:13 2006 STDOUT (/usr/bin/ssh): /bin/stty: standard input: Invalid
argument
Tue Jun 27 16:02:14 2006 STDOUT (/usr/bin/ssh): # Connecting to LPAR55
Tue Jun 27 16:02:14 2006 STDOUT (/usr/bin/ssh): # Connected
Tue Jun 27 16:02:14 2006 STDOUT (/usr/bin/ssh): # Checking for power off.
Tue Jun 27 16:02:14 2006 STDOUT (/usr/bin/ssh): # Power off the node
Tue Jun 27 16:03:48 2006 STDOUT (/usr/bin/ssh): # Wait for power off.
Tue Jun 27 16:03:48 2006 STDOUT (/usr/bin/ssh): # Power off complete.
Tue Jun 27 16:03:59 2006 STDOUT (/usr/bin/ssh): # Power on LPAR55 to Open Firmware.
Tue Jun 27 16:04:02 2006 STDOUT (/usr/bin/ssh): # Power on complete.
Tue Jun 27 16:04:02 2006 STDOUT (/usr/bin/ssh): # Client IP address is 10.1.1.55.
Tue Jun 27 16:04:20 2006 STDOUT (/usr/bin/ssh): # Server IP address is 10.1.1.2
Tue Jun 27 16:04:20 2006 STDOUT (/usr/bin/ssh): # Gateway IP address is 10.1.1.254.
Tue Jun 27 16:04:21 2006 STDOUT (/usr/bin/ssh): # /vdevice/l-lan@30000003 ping
successful.
Tue Jun 27 16:04:22 2006 STDOUT (/usr/bin/ssh): # Network booting install adapter.
Tue Jun 27 16:04:37 2006 STDOUT (/usr/bin/ssh): # bootp sent over network.
Tue Jun 27 16:04:37 2006 STDOUT (/usr/bin/ssh): # Network boot proceeding,
lpar_netboot is exiting.
Tue Jun 27 16:04:37 2006 STDOUT (/usr/bin/ssh): # Finished.
Tue Jun 27 16:04:37 2006 rc for /usr/bin/ssh = 0

```

---

The `OS_install` netboot use the HMC `lpar_netboot EXPECT` script to restart the LPAR through the controlling host (HMC). The script is almost the same as the `bos.sysmgmt.nim.master` script `/usr/sbin/lpar_netboot`.

**Note:** If you define an LPAR as an `OS_install` client and as a NIM client, you can then manage the LPAR by both the `nim` and the `OS_install` commands. Especially the `OS_install` netboot operation are useful when performing LPAR installations, or reinstallations, on regular NIM clients.

To monitor the netboot installation using `OS_install` you can use the `monitor_installation` operation:

```
OS_install -o monitor_installation <client_name> [-d]
```

Should you need to remove a `OS_install` resource or client, you can use the `remove` operation as shown in Example 4-23.

```
OS_install -o remove <object_name>
```

*Example 4-23 OS\_install remove of client*

---

```
root@master:/: OS_install -o remove LPAR55
Tue Jun 27 14:03:24 2006 Removing object LPAR55
Tue Jun 27 14:03:24 2006 Removed LPAR55 from repository
```

---

## 4.2.6 OS\_install XML repository

The `OS_install` information about each client or resource will be stored in the XML repository file `/var/osinstall/xml_repos` as shown in Example 4-24.

*Example 4-24 OS\_install xml\_repos*

---

```
<?xml version="1.0"?>
<osinstall_config xmlns="http://www.ibm.com/nim1" nim1_version="1.0">
<ctrl_host xmlns="http://www.ibm.com/nim1" name="HMC5">
  <hostname xmlns="http://www.ibm.com/nim1">hmc5</hostname>
  <type xmlns="http://www.ibm.com/nim1">hmc</type>
  <commo_method xmlns="http://www.ibm.com/nim1">ssh</commo_method>
</ctrl_host>
<client xmlns="http://www.ibm.com/nim1" name="LPAR55">
  <ip_addr>10.1.1.55</ip_addr>
  <mac_addr>22:33:80:00:AB:CD</mac_addr>
  <gateway>10.1.1.254</gateway>
  <subnet_mask>255.255.255.0</subnet_mask>
  <adapter_speed>auto</adapter_speed>
  <adapter_duplex>auto</adapter_duplex>
  <partition_info lpar="lpar55" profile="normal" managed_system="IBM-595-65C5ABC"/>
  <ref_ctrl_host ctrl_host_name="HMC5"/>
</client>
```

---

Detailed messages from running the `OS_install` command are written to the `/var/osinstall/event_log` file.

## 4.3 NIM environment with additional resource servers

If your environment consists of multiple nodes or consists of a complex network, you can consider configuring NIM resource servers to help off load your NIM master. As mentioned in Chapter 2, “Network Installation Manager (NIM) definitions and concepts” on page 7, a NIM resource server can be any standalone client (machine) configured as a server for a particular resource (for example, lpp\_source or SPOT).

You should consider configuring additional resource servers in case you:

- ▶ Have a complex environment with many nodes, and you need to off load your NIM master.
- ▶ Have a complex network with multiple LANs, routers and/or firewalls for simplifying network administration and security (one resource server per LAN).
- ▶ Have IBM System p5 servers with multiple partitions in each CEC, and you want to use POWER™ hypervisor managed VLANs to off load the external (client) network. You can find specific information on NIM and VIO in 4.13, “NIM and Virtual I/O (VIO)” on page 377.

### 4.3.1 Configuring a resource server

In our scenario, we have an LPAR in the IBM @server p670 server configured as our NIM master, and two NIM clients (LPARs) in one IBM System p p520 server. The two NIM clients are VIO clients as well. We have a Shared Ethernet Adapter (SEA) configured in the VIO server to allow external network communication with the NIM master. Refer to Figure 4-3 on page 146.

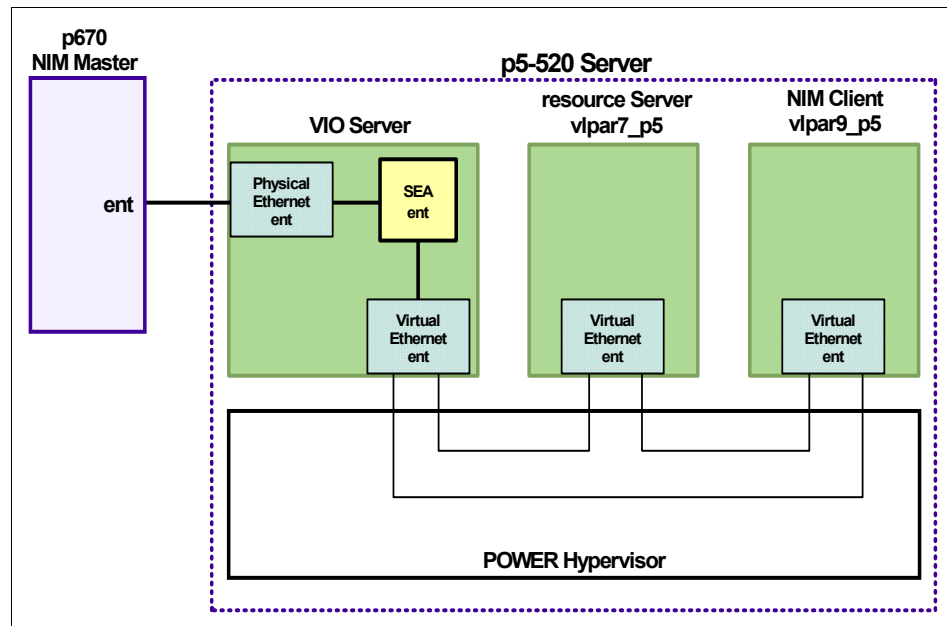


Figure 4-3 NIM master and resource server setup

We configure one of the NIM clients, `vpar7_p5`, as the resource server serving the SPOT and `mksysb` resources to the other NIM client, `vpar9_p5` (both `vpar7_p5` and `vpar9_p5` reside inside the same p5 server).

- ▶ Defining a SPOT resource in a standalone client:

Before creating the SPOT, remember to allocate the disk space for the SPOT creation in the resource server, `vpar7_p5`. We recommend that you create a file system for the SPOT (for example, `/export/spot`).

**Note:** Remember that the NIM master is able to communicate with the resource server either through `rsh` or `nimsh`. Also check that NFS is enabled.

Steps through the SMIT to create the SPOT resource (see Example 4-25 on page 147):

- ▶ From the NIM master, `lpar1`:

```
smitty nim_mkres
  Select "spot = Shared Product Object Tree - equivalent to /usr
  file"
```

*Example 4-25 Define a SPOT resource with v1par7\_p5 as the resource server*


---

```

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

                                [Entry Fields]
* Resource Name                  [SPOT-AIX53-Clust01]
* Resource Type                  spot
* Server of Resource           [v1par7_p5] +
* Source of Install Images       [LPPSRC-AIX53] +
* Location of Resource           [/export/spot] /
Expand file systems if space needed?  yes +
Comments                          []

installp Flags
COMMIT software updates?          no +
SAVE replaced files?              yes +
AUTOMATICALLY install requisite software?  yes +
OVERWRITE same or newer versions? no +
VERIFY install and check file sizes?  no +

```

---

From Example 4-25, observe that v1par7\_p5 is chosen as the *Server of Resource* (bold letters).

In the SPOT creation process, NIM master exports its lpp\_source directory and the standalone client (resource server) NFS mount the NIM master's lpp\_source directory. See Example 4-26 on page 147.

*Example 4-26 lpp\_source directory is exported and nfs mounted on resource server*


---

```

From NIM master, lpar1,
lpar1:/ > exportfs
/export/lppsource/lppsrc-aix53 -ro,root=v1par7_p5:,access=v1par7_p5:

```

```

From the Resource server, v1par7_p5,
v1par7_p5:/ > df
Filesystem      512-blocks  Free   %Used   Iused   %Iused   Mounted on
/dev/hd4         98304      21760   78%     1774    40%     /
/dev/hd2       1507328    134120   92%     9965    54%     /usr
/dev/hd9var     32768      16032   52%      373    17%     /var
/dev/hd3        65536     62808    5%         4     1%     /tmp
/dev/hd1        32768     32064    3%         5     1%     /home
/proc            -           -         -         -     -       /proc
/dev/hd10opt    98304     23032   77%      682    20%     /opt
/dev/spot1v    4194304   2971352   30%    16403    5%     /export/spot
/dev/images1v  4194304  1044904  76%      10     1%     /export/images

```

```
lpar1:/export/lppsource/lppsrc-aix53 24379392 3982728 84%
3061 1% /tmp/_nim_dir_327780/mnt0
```

---

Once the SPOT resource is defined on the NIM master and created on the resource server, you can see that this resource defined into the NIM master's database using `lsnim -l <SPOT_name>`, and observe the resource pointing to the resource server, as highlighted in Example 4-27.

*Example 4-27 lsnim output showing SPOT resource pointing to the resource server*

---

```
lpar1:/ > lsnim -c resources
boot                resources          boot
nim_scri            resources          nim_script
LPPSRC-AIX53        resources          lpp_source
SPOT-AIX53          resources          spot
SPOT-AIX53-Clust01 resourc            spot
```

```
lpar1:/ > lsnim -l SPOT-AIX53-Clust01
SPOT-AIX53-Clust01:
  class      = resources
  type       = spot
  plat_defined = chrp
  arch       = power
  bos_license = yes
  Rstate     = ready for use
  prev_state = verification is being performed
  location   = /export/spot/SPOT-AIX53-Clust01/usr
  version    = 5
  release    = 3
  mod        = 0
  oslevel_r  = 5300-04
  alloc_count = 0
  server    = v1par7_p5
  if_supported = chrp.mp ent
  Rstate_result = success
```

---

► Defining a mksysb resource on a standalone client

In our example, we initiated the mksysb of the NIM client, `v1par9_p5`, from the NIM master. Before you perform this, make sure you have allocated disk space for the resource server. We recommend you create another file system (for example, `/export/images`).

**Note:** Remember to have the following prerequisites in place:

- ▶ Communication between the NIM master and the resource server as well as communication between the NIM master and the NIM client. You can use either `rsh` or `nimsh`.
- ▶ Also check that the NFS between the resource server and the NIM client is enabled.
- ▶ The `ulimit` is set to unlimited on both the resource server and the NIM client (for root user).
- ▶ The file system exported by the resource server is either large file enabled (JFS) or JFS2.

Steps through the SMIT to create the `mksysb` resource as shown in Example 4-28.

- ▶ From the NIM master, `lpar1`:
  - `lpar1:/ > smitty nim_mkres`
  - Select “`mksysb` = a `mksysb` image”

*Example 4-28 Define a `mksysb` resource with `v1par7_p5` as the resource server*

---

```

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

[TOP]                                [Entry Fields]
* Resource Name                       [MK-VLPAR9_P5]
* Resource Type                        mksysb
* Server of Resource                   [v1par7_p5] +
* Location of Resource                 [/export/images/v1par9_p5.mksysb> /
Comments                               []

Source for Replication                 [] +
-OR-
System Backup Image Creation Options:
CREATE system backup image?         yes +
NIM CLIENT to backup                 [v1par9_p5] +
PREVIEW only?                          no +
IGNORE space requireme                 no +

```

---

In the `mksysb` resource creation process, the `v1par7-p5` server's `/export/images` directory is exported to the `v1par9_p5` server to perform the `mksysb`. See Example 4-29.

*Example 4-29 /export/images directory NFS exported*

From Resource server, v1par7\_p5,

```
v1par7_p5:/ > exportfs
```

```
/export/images -root=v1par9_p5:,access=v1par9_p5:
```

From NIM client, v1par9\_p5,

```
v1par9_p5:/ > df
```

Filesystem	512-blocks	Free	%Used	Iused	%Iused	Mounted on
/dev/hd4	49152	15864	68%	1656	46%	/
/dev/hd2	1490944	144224	91%	19527	51%	/usr
/dev/hd9var	32768	15312	54%	363	17%	/var
/dev/hd3	65536	59272	10%	48	1%	/tmp
/dev/hd1	32768	32064	3%	5	1%	/home
/proc	-	-	-	-	-	/proc
/dev/hd10opt	81920	25792	69%	621	17%	/opt
<b>v1par7_p5:/export/images</b>		<b>4194304</b>	<b>2031472</b>	<b>52%</b>	<b>9</b>	<b>1%</b>
<b>/tmp/4056320</b>						

Once the mksysb resource has been defined on the NIM master and created on the resource server, you can see this resource by running (on the NIM master) `lsnim -l <mksysb_name>`. Note that the resource is pointing to the resource server. See the highlighted fields in Example 4-30.

*Example 4-30 lsnim output showing mksysb resource pointing to resource server*

```
lpar1:/ > lsnim -c resources
```

boot	resources	boot
nim_script	resources	nim_script
LPPSRC-AIX53	resources	lpp_source
SPOT-AIX53	resources	spot
SPOT-AIX53-Clust01	resources	spot
<b>MK-VLPAR9_P5</b>	<b>resources</b>	<b>mksysb</b>

```
lpar1:/ > lsnim -l MK-VLPAR9_P5
```

```
MK-VLPAR9_P5:
class      = resources
type       = mksysb
arch       = power
Rstate     = ready for use
prev_state = unavailable for use
location   = /export/images/v1par9_p5.mksysb
version    = 5
release    = 3
mod        = 0
oslevel_r  = 5300-03
```



```
alloc_count = 0
server      = v1par7_p5
```

---

### 4.3.2 Performing a BOS installation

Performing a BOS installation (rte or mksysb), for example, having the resources on the resource server is similar to a BOS installation with all the resources on the NIM master. You just have to allocate the correct resources to perform this task. However, there are a few steps we need to take care of before:

- ▶ Communication between the NIM master and resource server (**rsh** or **nimsh**).
- ▶ Communication between the NIM master and the NIM client, if you need to perform a push operation (**rsh** or **nimsh**).
- ▶ NFS must be enabled between the NIM master and the NIM client; or between the resource server and the NIM client, depending on where the resource is located.
- ▶ If you have a resource server serving the SPOT resource for the NIM client, and performs a pull BOS installation, take note that the database for the bootp server in the `/etc/bootptab` file is in the resource server instead of in the NIM master. You also need to be careful that when you specify the server's IP address in the SMS menu. Specify the resource server's IP instead of the NIM master IP.

We illustrate an example to see what happens when we perform a pull BOS installation with the resource server serving the mksysb and the SPOT resources for the NIM client:

1. Allocate the SPOT and the mksysb resources to the NIM client, v1par9\_p5

From the NIM master, lpar1, this are the steps through the SMIT menus:

```
lpar1:/ > smitty nim_mac_res
Allocate Network Install Resources
```

Then select v1par9\_p5 as the target machine and MK-VLPAR9\_P5 and SPOT-AIX53-Clust01 as resources.

Once the SPOT and mksysb are allocated, the image and the SPOT directories are exported from the v1par7\_p5 to v1par9\_p5 (see Example 4-31).

*Example 4-31 image and SPOT exported in v1par7\_p5*

---

```
v1par7_p5:/ > exportfs
/export/spot/SPOT-AIX53-Clust01/usr
-ro,root=v1par9_p5:,access=v1par9_p5:
```

```
/export/images/vlpar9_p5.mksysb
-ro,root=vlpar9_p5:,access=vlpar9_p5:
```

---

- Initialize the BOS installation on the NIM client, vlpar9\_p5 as shown in Example 4-32.

From the NIM master, this are the steps through the SMIT menu:

```
lpar1:/ > smitty nim_mac_op
```

Then select vlpar9\_p5 as the target machine and bos\_inst = perform a BOS installation.

*Example 4-32 BOS installation - mksysb (pull operation)*

---

#### Perform a Network Install

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

	[Entry Fields]
Target Name	vlpar9_p5
Source for BOS Runtime Files	mksysb +
installp Flags	
	[-agX]
Fileset Names	[]
Remain NIM client after install?	yes +
<b>Initiate Boot Operation on Client?</b>	<b>no +</b>
Set Boot List if Boot not Initiated on Client?	no +
Force Unattended Installation Enablement?	no +
<b>ACCEPT new license agreements?</b>	<b>[yes] +</b>

---

After the BOS installation is initialized, the /etc/bootptab file in the vlpar7\_p5 server is updated (not in the NIM master server). Observe that the server's IP address is pointing to the vlpar7\_p5's IP address instead of the NIM master's IP address as shown in Example 4-33.

*Example 4-33 /etc/bootptab file in vlpar7\_p5*

---

```
# T178 -- (xstation only) -- enable XDMCP
# T179 -- (xstation only) -- XDMCP host
# T180 -- (xstation only) -- enable virtual screen
vlpar9_p5:bf=/tftpboot/vlpar9_p5:ip=10.1.1.74:ht=ethernet:sa=10.1.1.72:
sm=255.255.255.0:
```

---

Also, the NIM script is exported from the vlpar7\_p5 to vlpar9\_p5. See Example 4-34.

*Example 4-34 nim script exported from v1par7\_p5*

```
v1par7_p5:/ > exportfs
/export/spot/SPOT-AIX53-Clust01/usr -ro,root=v1par9_p5:,access=v1par9_p5:
/export/images/v1par9.mksysb -ro,root=v1par9_p5:,access=v1par9_p5:
/export/nim/scripts/VLPAR9_P5.script -ro,root=v1par9_p5:,access=v1par9_p5:
```

### 3. Boot the LPAR in SMS mode

Boot the LPAR in SMS mode to specify the v1par7\_p5's IP address as the server's IP address instead of the NIM master's IP address. Refer to "SMS and console flow during NIM client installation" on page 83 for more details on the SMS menus.

## 4.4 Using NIM to perform AIX migrations of the NIM master and clients

In the following section we discuss using NIM to perform AIX migration on the master and the clients. In particular, we will focus on migration to AIX 5L V5.3. The main topics we will cover are:

- ▶ Migrating a NIM master to AIX 5L V5.3
- ▶ Migrating a NIM client to AIX 5L V5.3

We will cover the conventional method of AIX migration via NIM. Other methods of NIM migration such as **nimadm** are covered in further sections of this book. There are two new methods for AIX migration via NIM: mksysb migration and nim\_move\_up, which are discussed in later sections of this book as well.

Planning to migrate an AIX system (with or without NIM) requires careful planning and testing. We will include some tips for planning a migration, including pre/post migration steps, NIM master and client requirements, disk space requirements, migration testing with NIM clones and more.

### 4.4.1 Migrating a NIM master to AIX 5L V5.3

In this section, we will discuss the migration of a NIM master to AIX 5L V5.3. Although the migration of a NIM master is not performed via a NIM migration procedure, we will cover the specific tasks required when upgrading a NIM master, including tips on pre- and post-migration actions that will ensure a successful, functioning NIM environment.

We will not cover in detail how to perform an AIX migration via CD. This type of migration is already documented in the manual "AIX 5L V5.3 Installation and

Migration, SC23-4887-03" manual and there are no specific differences for a NIM master. Topics covered include:

- ▶ Prerequisites
- ▶ Pre-migration tasks
- ▶ Migration via CD
- ▶ Post-migration tasks
- ▶ Cleanup tasks
- ▶ Alternative migration option

**Note:** In a NIM environment, the NIM master must be running at the highest level of AIX. Therefore it must be migrated to AIX 5L V5.3 before any of its clients. Once migrated, it will be able to serve AIX 5L V5.3 clients as well clients running earlier versions of AIX, for example, AIX V4.3, AIX V5.1 and AIX 5L V5.2.

Before migrating a NIM master check the following:

- ▶ Obtain the latest AIX base install media plus latest technology level (TL) and/or Service Packs (SP) and Concluding Service Packs (CSP).
- ▶ Read the AIX 5L V5.3 release notes. These contain important information that you should know before starting a migration.
- ▶ Check that your client hardware supports AIX 5L V5.3. More information can be found in the release notes for AIX 5L V5.3. Release notes for AIX 5L V5.3 for example, mention that some older hardware is no longer supported (such as MCA).
- ▶ Always perform at least one test migration of the system before doing the real thing. This will at least verify that the AIX migration works as expected and highlight any issues before they occur during a live production upgrade. Make sure you test the back-out plan as well.
- ▶ Ensure that you have all the latest software levels for disk and other storage devices such as SDD, SDDPCM, etc. Check that you have the correct version for AIX 5L V5.3, for example if you are running SDD, you will need SDD for AIX 5L V5.3. After a migration to AIX 5L V5.3 you would need to remove the old version and install the correct version of SDD for AIX 5L V5.3.
- ▶ It is always a good idea to upgrade your system and adapter firmware to the latest level of microcode before performing an AIX migration. Some systems and/or adapters may not be supported or will not function with AIX 5L V5.3 if their microcode is not up-to-date. In particular you should check the levels of your fibre channel (if using SAN-attached storage) and network adapters

(especially when using NIM). Check the IBM support web site for the latest levels of system and adapter microcode and apply them before upgrading:

<http://www14.software.ibm.com/webapp/set2/firmware/gjsn>

- ▶ Check the code level on your Hardware Management Console (HMC) (if applicable). Often there will be corresponding code levels when you upgrade the managed system microcode which may contain important fixes to support AIX 5L V5.3 and the latest technology level.
- ▶ Document the NIM masters current configuration, for example, hardware, AIX and NIM information. You can use the `snap` and `lsnim` commands to document the NIM master's configuration. You could also use a custom script to collect information about your system and store it in a text file. This file can be used later if you need to look back at how the system was configured before the migration.
- ▶ Backup the NIM database. You can use SMIT to backup the NIM database or a custom script. Backup the database before performing a `mksysb`, in this way the NIM database backup will be included in the `mksysb`.
- ▶ Perform a `mksysb` of the NIM master before the migration. You can backup to tape or DVD media. If your NIM resources are kept in a separate volume group (for example, `nimvg`) then you should also perform a volume group backup (`savevg`) to tape, DVD or use the backup tool that exists within your environment (for example, Tivoli Storage Manager).

**Important:** Migration by media (CD) is the only supported way to migrate a NIM master. The NIM master will be unavailable for the duration of the migration

**Note:** Starting with AIX 5L V5.3 there is a new NIM feature called High Availability NIM (HANIM). HANIM will allow you to manually switch your primary master to an alternate. If downtime is an issue for your NIM master (e.g. for disaster recovery purposes), then you can perform your migration without impacting your NIM clients.

Once your NIM master has been migrated to AIX 5L V5.3 you will be able to take advantage of this feature for any future AIX migration (e.g. AIX 5L V5.3 to AIX 5L V5.4) on the NIM master. For details, see 4.1, "High Availability NIM (HA NIM)" on page 124.

In the following section, we will look at upgrading your NIM master from AIX 5L V5.2 to AIX 5L V5.3 via CD.

Prior to migration, make sure that the following prerequisites are met:

- ▶ Ensure you have all the AIX 5L V5.3 media (CD's).
- ▶ Check that you have enough disk space to create file systems for the new AIX 5L V5.3 lpp\_source and SPOT that will be created after the migration of the master.
- ▶ Check the following documentation:
  - AIX 5L Version 5.3 release notes  
[http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304\\_base\\_relnotes.pdf](http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304_base_relnotes.pdf)
  - AIX 5L Version 5.3 Bonus Pack and Expansion Pack Release Notes  
[http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304\\_exp\\_relnotes.pdf](http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304_exp_relnotes.pdf)
- ▶ Check if the hardware is supported. Only CHRP (Common Hardware Reference Platform) machines are supported under AIX 5L V5.3. Verify that the server is a CHRP machine:

```
# bootinfo -p  
chrp
```
- ▶ Check file system space and free PPs in rootvg. Consult the AIX 5L V5.3 release notes for more information about space requirements.
- ▶ Run the AIX 5L V5.3 pre\_migration script and review the results. The pre\_migration script will verify if our system is ready for migration. The pre\_migration script can be found on the AIX 5L V5.3 base install media.
- ▶ Backup /etc/motd and /etc/sendmail.cf. These files are replaced during a migration. Keep copies of the originals on hand so you can merge the old configuration into the newer files manually.
- ▶ Backup the NIM database beforehand.
- ▶ Do you have a CD and/or tape drive on the NIM master? A CD drive will be required for the migration. If using DLPAR, you need to assign one dynamically for use during the migration.
- ▶ Document AIX and NIM master configuration before and after. Use the lsnim and snap commands to generate and capture critical configuration information about your NIM master. This information will be useful if you encounter problems after the migration that require you to rebuild or reconfigure the system.
- ▶ We cloned the NIM master (with “Remain NIM client set to no”) to another test system and performed a test migration to AIX 5L V5.3. We did not test the NIM environment as we only wanted to test and observe the AIX migration process. However, you could clone the master and configure so that it was a fully functional NIM master that could serve clients. You may even consider

this as an alternate way of migrating your NIM master. For more information on cloning a system with NIM refer to 5.4, “Cloning” NIM clients using `mksysb`” on page 460 and 5.10, “How to backup and re-install a NIM master” on page 547.

### Steps to migrate the NIM master to AIX 5L V5.3

The following steps will migrate a NIM master running AIX 5L V5.2 on p670 LPAR to AIX 5L V5.3. We employ the use of an alternate disk install to provide a quick back out should the migration fail for any reason. Once the migration is complete we then verify the NIM master environment is OK and check we can still use NIM to perform a BOS install of the various versions of AIX that our environment supports, for example, AIX 4.3, AIX 5L V5.1, AIX 5L V5.2, and now AIX 5L V5.3.

1. Check for clients that have active NFS mounts from the NIM master before starting the migration.

We have seen situations where clients will hang when the NIM master is shutdown and the clients NFS mount “disappears”. It can be difficult to force umount the NFS cross mount on the client (not impossible but tricky) and may require a reboot of the client if the NIM master is down for a long time.

If you are using CSM you can use the `dsh` command to check all your systems at once for any current NFS mounts from the clients to the NIM master. In the following example, client `lpar4` has NFS mounted `nimmast's /export/images` file system. This should be unmounted prior to migrating the NIM master.

```
{nimmast}:/ # dsh -v mount | grep -i nfs
lpar4: nimmast /export/images /mnt nfs3 Jun 14 15:56
```

2. Document the AIX and NIM master configuration. We use the `snap` and `lsnim` commands to document the system before the migration. The output from both commands would be stored on another system other than the NIM master. We need to be able to access the information whether or not the NIM master is up or down. We also use a custom script to capture other configuration information we would like to see. Check the [additional materials](#).

```
{nimmast}:/ # snap -ac
{nimmast}:/ # lsnim -l
{nimmast}:/ # AIXinfo -pre
```

3. Perform a NIM database backup. We backup the NIM database via SMIT. You can also use a script if you prefer as shown in Example 4-35. The advantage of the script is that it can be run from root's cron every day, giving you several backups to chose from in case of a problem.

```
{nimmast}:/ # smitty nim_backup_db
Backup the NIM Database
* Filename/Device for the Backup
[/etc/objrepos/nimdb.backup.14Jun2006.Prior_to_AIX53_Migration] +
```

*Example 4-35 NIM database backup script*

```

{nimmast}:/usr/local/bin # cat bknimdb
#!/usr/bin/ksh
#
# bknimdb - script to backup the NIM Database
#
# Description
# -----
# This script will backup the NIM database. This backup could be used
# to recover the NIM Database should it become corrupted. The backup
# file will be named nimdb.backup.Day, where Day is the day of the week.
#
# e.g. /etc/objrepos/nimdb.backup.Mon
#
# The NIM database can be restored via procedures documented in the
# "AIX 5L Version 5.3 Installing AIX" Guide (SC23-4887-02).
#
# The restore can be done via SMIT i.e.
#
#       # smitty nim_restore_db
#

PATH=/usr/bin:/usr/sbin

# Directory where the NIM database backup will be created.
nimdbdir=/etc/objrepos

# We can only run this as root.
if [ "$(id -un)" != "root" ]
then
    echo "$(basename $0): must be run as user \"root\""
    exit 1
fi

#
# Backup the NIM Database
#
echo "-----"
echo "Starting NIM Database Backup on $(uname -n ) at $(date)"
echo

/usr/lpp/bos.sysmgt/nim/methods/m_backup_db $nimdbdir/nimdb.backup.$(date +%a)
if [ $? -ne 0 ]
then
    echo
    echo "$(basename $0): Error in backing up NIM Database"
    echo
fi

```



```
echo
echo "Finished NIM Database Backup on $(uname -n ) at $(date)"
echo "-----"
```

```
{nimmast}:/usr/local/bin # bknimdb
```

```
-----
Starting NIM Database Backup on nimmast at Mon Jun 19 10:12:21 DFT 2006
```

```
a ./etc/objrepos/nim_attr 72 blocks.
a ./etc/objrepos/nim_attr.vc 272 blocks.
a ./etc/objrepos/nim_object 16 blocks.
a ./etc/objrepos/nim_object.vc 32 blocks.
a ./etc/NIM.level 1 blocks.
a ./etc/niminfo 1 blocks.
```

```
Finished NIM Database Backup on nimmast at Mon Jun 19 10:12:21 DFT 2006
```

```
-----
{nimmast}:/usr/local/bin # ls -ltr /etc/objrepos/nimdb.backup.Mon
-rw-r--r--  1 root    system      215040 Jun 19 10:12
/etc/objrepos/nimdb.backup.Mon
```

- 
4. Perform a mksysb of the NIM master and backup other volume groups. We saved a mksysb image to tape media and then a savevg of the NIM volume group (nimvg) to another tape media.

```
{nimmast}:/ # mksysb -i /dev/rmt0
{nimmast}:/ # savevg -f /dev/rmt0 nimvg
```

5. Install the bos.alt\_disk\_install filesets. These filesets are required so that we can use the **alt\_disk\_install** command to clone our rootvg before the migration. Refer to Example 4-36.

---

*Example 4-36 Installing the bos.alt\_disk filesets*

```
{nimmast}:/ # smit nim_task_inst
Install and Update from ALL Available Software
* Installation Target                               master
* LPP_SOURCE                                         lpp_sourceaix5204
* Software to Install                               [bos.alt_disk_install]+
```

---

6. Un-mirror rootvg and clone to hdisk1 as shown in Example 4-37. We will use this rootvg clone if our migration fails and we need to go back to AIX 5L V5.2. If rootvg is mirrored, then you can un-mirror and use the spare disk for the alternate rootvg.

If you are in a SAN environment, you can assign a LUN temporarily for the migration and use it as the alternate rootvg. The LUN could be returned some time after the migration is finished and you decide that you do not need to

back out from the migration. The *-B* flag specifies that the bootlist will not be set to the alternate disk after the operation completes.

However, make sure that your system can boot from a SAN attached disk (LUN).

---

*Example 4-37 Cloning rootvg with alt\_disk\_install*

```
{nimmast}:/ # unmirrorvg -c1 rootvg hdisk1
{nimmast}:/ # chpv -c hdisk1
{nimmast}:/ # lspv -l hdisk1 ; migratepv hdisk1 hdisk0 (if required)
{nimmast}:/ # lspv -l hdisk1
{nimmast}:/ # reducevg rootvg hdisk1
{nimmast}:/ # lsvg -p rootvg
{nimmast}:/ # bosboot -a -d /dev/hdisk0
{nimmast}:/ # bootlist -m normal hdisk0
{nimmast}:/ # bootlist -m normal -o
hdisk0
{nimmast}:/ # alt_disk_install -B -C hdisk1
Calling mkszfile to create new /image.data file.
Checking disk sizes.
Creating cloned rootvg volume group and associated logical volumes.
Creating logical volume alt_hd5.
Creating logical volume alt_hd6.
Creating logical volume alt_hd8.
Creating logical volume alt_hd4.
Creating logical volume alt_hd2.
Creating logical volume alt_hd9var.
Creating logical volume alt_hd3.
Creating logical volume alt_hd1.
Creating logical volume alt_hd10opt.
Creating /alt_inst/ file system.
Creating /alt_inst/home file system.
Creating /alt_inst/opt file system.
Creating /alt_inst/tmp file system.
Creating /alt_inst/usr file system.
Creating /alt_inst/var file system.
Generating a list of files
for backup and restore into the alternate file system...
Backing-up the rootvg files and restoring them to the alternate file system...
Modifying ODM on cloned disk.
Building boot image on cloned disk.
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst
forced unmount of /alt_inst
Changing logical volume names in volume group descriptor area.
```

```

Fixing LV control blocks...
Fixing file system superblocks...
{nimmast}:/ # lspv | grep root
hdisk0 00531d9a33ff6ab5          rootvg          active
hdisk1 00531d9a47ed2df6          altinst_rootvg active

```

---

7. Commit all applied software before migrating, see Example 4-38. This will also create additional space in /usr (if there are any filesets in the “applied” state).

*Example 4-38 Committing all applied filesets*

---

```

{nimmast}:/ # smit commit
* SOFTWARE name                [all] +
  PREVIEW only? (commit operation will NOT occur)  no +
  COMMIT requisites?           yes +
  EXTEND file systems if space needed?             yes +
  DETAILED output?              no +

```

---

8. Ensure all users are logged off.
9. Perform migration installation of AIX 5L V5.3. We are now ready to execute the AIX migration. At this point we must go to the NIM master’s console (via the HMC) and prepare to migrate via CD.
10. Insert AIX 5L V5.3 Installation CD Volume 1 into the CD drive.
11. Follow the procedure in the AIX installation and migration guide to migrate via media to AIX 5L V5.3
12. Once the migration is finished remove the AIX 5L V5.3 Installation Volume 1 CD from the CD-ROM drive.
13. Check the system configuration, for example, oslevel, disk, network, AIX error report, etc.
14. Cleanup old AIX 5.2 filesets. It may be necessary to remove old AIX 5.2 filesets after the migration.
15. Check the NIM environment. We perform some quick tests to verify that the NIM environment looks good after the migration. Using the `lsnim` command we check that NIM database is intact. We check the state of the master and clients. We also validate some of our NIM resources.

Check the NIM database:

```
{nimmast}:/ # lsnim
```

Check the status of the NIM master:

```
{nimmast}:/ # lsnim -a Cstate -a Mstate master
```

Check the status of a NIM client:

```
{nimmast}:/ # lsnim -a Cstate -a Mstate LPAR4
```

Validate the NIM resources:

```
{nimmast}:/ # nim -o check LPP_52_ML8
{nimmast}:/ # nim -o check SPOT_52_ML8
```

16. Build an AIX 5L V5.3 lpp\_source and SPOT. You will need to insert the AIX 5L V5.3 install CD number 1 into the CD-ROM drive to build the lpp\_source. After the lpp\_source has been created ensure that **simages** is set to yes as this indicates whether or not it can be used for installations. Check for any errors such as missing filesets when creating the SPOT. See Example 4-39. Also, for details on creating an lpp\_source and a SPOT, see 3.1, “Setting up a basic NIM environment” on page 50

*Example 4-39 AIX 5L V5.3 lpp\_source and SPOT*

---

```
{nimmast}:/ # lsnim -l LPP_53_ML4
LPP_53_ML4:
  class      = resources
  type       = lpp_source
  arch       = power
  Rstate     = ready for use
  prev_state = verification is being performed
  location   = /AIX53ML4/LPP_53_ML4
  simages   = yes
  alloc_count = 0
  server     = master
```

```
{nimmast}:/ # lsnim -l SPOT_53_ML4
SPOT_53_ML4:
  class      = resources
  type       = spot
  plat_defined = chrp
  arch       = power
  bos_license = yes
  Rstate     = ready for use
  prev_state = verification is being performed
  location   = /AIX53ML4/SPOT_53_ML4/usr
  version    = 5
  release    = 3
  mod        = 0
  oslevel_r  = 5300-04
  alloc_count = 1
  server     = master
  if_supported = chrp.mp ent
  Rstate_result = success
```

---

Validate the NIM resources:

```
{nimmast}:/ # nim -o check LPP_53_ML4
{nimmast}:/ # nim -o check SP0T_53_ML4
```

17. Using a test client (LPAR), perform a NIM BOS rte install of AIX 5L V5.3. By performing a NIM BOS install onto our test client we can confirm that our new AIX 5L V5.3 NIM master is functioning correctly.
18. Restore rootvg to a mirrored disk configuration. Now that our new NIM environment is functioning, we can remove the alternate rootvg disk and re-mirror rootvg, as shown in Example 4-40. Finally, reboot the system to turn off quorum for rootvg.

*Example 4-40 Commands to remirror rootvg after alt\_disk\_install*

---

```
{nimmast}:/ # alt_disk_install -X altinst_rootvg
{nimmast}:/ # extendvg rootvg hdisk1
{nimmast}:/ # mirrorvg rootvg hdisk1
{nimmast}:/ # bootlist -m normal hdisk0 hdisk1
{nimmast}:/ # bosboot -a -d /dev/hdisk0
{nimmast}:/ # bosboot -a -d /dev/hdisk1
{nimmast}:/ # shutdown -Fr
```

---

19. The NIM master is now running AIX 5L V5.3 and all NIM resources have been tested. You are now able to deploy AIX 5L V5.3 within your NIM environment.

### ***Back out plan for NIM master***

If there are issues with the NIM master migration we can easily back out by rebooting the master from the alternate rootvg disk. The following procedure can be followed to boot from the cloned rootvg.

1. Set the bootlist to boot from original rootvg. First identify the alternate rootvg disk as shown in Example 4-41.

*Example 4-41 Identifying the alternate rootvg disk*

---

```
{nimmast}:/ # lspv | grep altinst_rootvg
hdisk1 00531d9a47ed2df6 altinst_rootvg active
```

---

2. Check the bootlist is set to hdisk1 as shown in Example 4-42.

*Example 4-42 Verifying the bootlist.*

---

```
{nimmast}:/ # bootlist -m normal -o
hdisk1
```

---

3. Reboot the server. See Example 4-43.

*Example 4-43 Rebooting the system.*

---

```
{nimmast}:/ # shutdown -Fr
```

---

Another way to back out is to restore from the mksysb backup created prior to the migration. This will take more time, so the alternate disk method is the preferred way.

4. Verify the NIM environment as previously shown.

### **Alternative NIM master migration approach**

An alternative migration option for a NIM master would be to build a new NIM master, at AIX 5L V5.3, on new hardware and then to migrate the NIM clients to the new master. If you have additional (newer) hardware, you could build a new AIX 5L V5.3 NIM master on this hardware and then move your existing clients to the new master at your convenience. One advantage of this approach is that there is no downtime for the original NIM master. This is also a good approach if your existing NIM master's hardware does not support AIX 5L V5.3.

## **4.4.2 Migrating a NIM client to AIX 5L V5.3**

In this section we discuss migrating a NIM client to AIX 5L V5.3. Topics include:

- ▶ Prerequisites
- ▶ Pre migration tasks
- ▶ Migration via NIM
- ▶ Post Migration tasks
- ▶ Debugging a NIM migration

**Note:** In a NIM environment, the NIM master must be migrated to AIX 5L V5.3 before any of its clients. It must be running the highest operating system level. Once the NIM master has been migrated the clients can then follow.

We will discuss a conventional NIM migration which also utilizes a standard alternate disk install (i.e. not **nimadm**) for back out.

The advantage of using NIM to migrate a client is that it does not require any installation media (such as CD) to perform the migration. The migration is performed over network. As a result, the administrator can perform and monitor the migration remotely and even automate the migration if desired. Multiple migrations can occur at once.

During a NIM migration, the system will be down for the duration of the upgrade. If downtime is an issue for your site you may consider using the NIM alternate disk migration (**nimadm**) utility. This allows you to migrate on an alternate rootvg disk and then reboot the system to restart it on the newer version of AIX, all under the control of the NIM master. The migration can take place at your convenience and no downtime is required until the reboot is performed at an appropriate time. See 4.6, “NIM alternate disk migration” on page 261.

If HACMP is used in your environment then the downtime required can be minimized. You could execute a controlled failover of the system to a standby node first. Then you would be able to migrate the client without impacting the services that run on that system. Once the migration was complete you could then execute a controlled fallback, restart your services and then test them with AIX 5L V5.3. If there are any issues you could perform another controlled failover to the standby node (which is still running AIX 5L V5.2). This will allow you to investigate the migration issues without impacting the applications on the system.

During the migration process the client boots over the network from the NIM master (bootp/tftp). NFS mounts are then used to update the filesets on the client machine. The network booted client runs in single user mode during the migration and has only limited functionality to support the NIM migration process. As a result client applications cannot be active during the migration.

If the migration process fails, you can either restore a mksysb backup of the client via NIM or reboot the client using the alternate rootvg disk created prior to the migration (which is our preferred method). Once the restore or reboot is complete, you can try the migration again as soon as you have resolved the issues that caused the migration to fail.

It is recommended that the following prerequisites be met before commencing a migration.

Before migrating a client check the following:

- ▶ The NIM master must be running AIX 5L V5.3 with the latest Technology Level. See 3.1, “Setting up a basic NIM environment” on page 50.
- ▶ It is recommended (but not mandatory) that the client be running at the latest level of its current version of AIX. This way your client system will be running the latest code and will be able to take advantage of the latest features and enhancements for that version and level of AIX.
- ▶ Ensure you have enough disk space on the NIM master to backup the NIM client multiple times (for example, check space in /export/images). A mksysb backup of the NIM client will be taken before and after the migration so there must be enough free space for the images.

- ▶ Check that your client hardware supports AIX 5L V5.3. More information can be found in the release notes for AIX 5L V5.3.
- ▶ Check that your applications are supported with AIX 5L V5.3. Use cloning if you want to test an application before performing the real migration. It may be worth putting together an application matrix for all your systems and contacting each vendor to ensure support for AIX 5L V5.3.
- ▶ Perform at least one test migration of the system before doing the real thing. This will at least verify that the AIX migration works as expected and highlight any issues before they occur during a live production upgrade! Clone the client to another test LPAR via NIM. Install the clients mksysb but do not recover its devices (this will avoid IP address conflicts on your network). Once the mksysb is restored you can migrate it to AIX 5L V5.3 and observe/document the results.

**Note:** Make sure you test the back out plan as well.

- ▶ Ensure that you have all the latest software levels for disk and other storage devices such as SDD, SDDPCM, etc. Check that you have the correct version for AIX 5L V5.3 for example SDD for AIX 5L V5.3.
- ▶ It is always a good idea to upgrade your system and adapter firmware to the latest level of microcode before performing an AIX migration. Some systems and/or adapters may not be supported or will not function with AIX 5L V5.3 if their microcode is not up-to-date. In particular you should check the levels of your Fibre Channel (if using SAN disk) and Network adapters (especially when using NIM). Check the IBM support Web site for the latest levels of system and adapter microcode and apply them before upgrading.  
<http://www14.software.ibm.com/webapp/set2/firmware/gjsn>
- ▶ Check the code level on your HMC (if applicable). Often there will be corresponding code levels when you upgrade the managed system microcode. It may even contain important fixes to support AIX 5L V5.3 and the latest technology level.
- ▶ The system to be migrated must be a valid NIM client. It must be registered with the NIM database and should have a valid /etc/niminfo file. You can use **niminit** to register the client if it is not already defined. Also check the current state and resource allocation for the client. Reset the client so it is in a clean state prior to the migration. “Defining NIM clients” on page 77.
- ▶ The NIM master must be able to run remote commands on the NIM client to be migrated.
- ▶ Read the AIX 5L V5.3 release notes. It contains important information that you should know before starting a migration.



- ▶ Install the `bos.alt_disk_install` filesets on the client if you plan on using `alt_disk_install` tools with the conventional NIM migration.

This is the basic process of a NIM client AIX migration:

- ▶ The client boots over the network.
- ▶ It will mount the files required for the migration from the NIM master.
- ▶ A menu will appear called “Migration preparation menu.” This confirms that you are doing a migration and not an overwrite installation.
- ▶ Once the installation is underway the “BOS Installation Menu” appears which displays the progress and status of the migration. At this point your old operating system is removed and replaced with the new version. Therefore if there are any problems with the migration a restore of some kind will be required. This is why you **MUST** take a backup (or backups) of your system **BEFORE** starting the migration.
- ▶ File system allocation and a list of filesets to be updated is displayed next. A lot of this information is also recorded on the client in the `/var/adm/ras` log files such as `devinst.log`.
- ▶ When the message “*Over mounting ./*” appears then the migration is finished and the RAM file system is overmounted by the now migrated root (`/`) file system. Final migration activities are performed like creating a new boot image and setting the client bootlist. The status of the migration can be monitored using `lsnim -a info clientname` command.
- ▶ The NIM master is also informed of the successful migration and resets the NIM client's state (to ensure that the client does not boot from the network again by accident!).
- ▶ The final step in the NIM migration process will be a reboot of the client. Once the reboot is finished your system will be up and running on AIX 5L V5.3. At this point you should start your post migration tasks.

## Troubleshooting a NIM migration

This section describes troubleshooting steps during a NIM migration.

- ▶ If the migration fails for some reason, you will probably experience a “system hang” or see an error appear in the BOS install output. You will probably need to reset the system so that you can recover. As a result the state of the NIM client will not be reset by the NIM master as it never received any indication that the migration was complete. It will be necessary to reset the NIM client in order to perform NIM operations on it in the future. The reset of the NIM client will ensure that the state of the client is cleared and that all NIM resources are deallocated. The reset can be done via SMIT or via the command line (examples will be shown on the following pages). Once the client has been

reset you will be able to perform operations with it again such as re-trying the migration once the reason for the failure has been resolved.

- ▶ The following log files contain useful information when troubleshooting a migration. These log files can also be view from the NIM master via the **nim showlog** operation:

```
/var/adm/ras/conslog - Console output log.
/var/adm/ras/bootlog - System boot log.
/var/adm/ras/bosinstlog - BOS installation log.
/var/adm/ras/nim.script - NIM script output.
/var/adm/ras/nim.installp - NIM filesets install log.
/var/adm/ras/devinst.log - Software installation log.
```

```
# nim -o showlog -a log_type=bosinst LPAR4
# nim -o showlog -a log_type=niminst LPAR4
# nim -o showlog -a log_type=script LPAR4
```

- ▶ If you experience issues during a client network boot, you can try the following to troubleshoot the problem:
  - Check client's IP details in SMS are correct. Perform a maual ping test from the SMS menus.
  - Check adapter speed is correct. Depending on the type of adapter, it may be necessary to set the speed and duplex setting of the adapter (although some newer GB adapters are better set to auto). Incorrect speed and duplex settings can cause hard to find issues, such that the ping test will work but the NIM process will be very slow and may even stop half-way through what appears to be an otherwise successful install (usually during the restore base operating system stage).
  - As previously mentioned, you should also check the microcode levels of the network adapters to avoid any possible issues when booting over the network.
- ▶ If the migration hangs you should check if any of the following situations exist and also if any LED code is displayed that might correspond to the problem.
  - Are the file systems properly exported (NFS) on the NIM master?
  - Can the client access the file systems of the NIM master via NFS?

See the Chapter 7., “Basic NIM problem determination and tuning” on page 575 for more information on problem determination in a NIM environment.

### 4.4.3 NIM client AIX migration example

In this section, we provide an example of migrating a system running AIX 5L V5.2 on a System p670 to AIX 5L V5.3. Our NIM master is a System p615 running AIX 5L V5.3. Both are connected via ethernet on the same VLAN. Our conventional

NIM migration method will require the client to be down during the migration process but also employs an **alt\_disk\_install** (alternate disk install) procedure which will be used as a quick back out if the migration fails.

Although this method requires the most downtime when compared to NIM alternate disk migration, it is considered the most straightforward method, which involves the least amount of effort for the administrator (if planned and tested properly). Unlike nimadm migration, it also allows for TCB support (see 4.6, “NIM alternate disk migration” on page 261 for more information on nimadm TCB support).

Although our test environment uses SCSI disk, we will also include some tips for SAN disk environments were appropriate.

Before starting the migration we check that the following prerequisites are met:

- NIM master already migrated to AIX 5L V5.3.
- System and adapter microcode already upgraded.

Planning for our NIM client migration will involve the following tasks:

- ▶ We have read the AIX 5L V5.3 release notes and checked that our applications are supported.
- ▶ Writing a plan of all the tasks (including command line) that you will perform during the migration:
  - Make sure you have a back out plan
  - Test the back out plan on your test clone.
  - Make it usable by someone else in your team so that another person can execute the plan should you be unable to do so. This plan can be re-used as a template for future migrations.
  - Ask a peer to review the document.
  - Write down all known issues and any fixes required.
  - Briefly explain what the process will be to migrate, for example basic flow of major events: upgrade AIX, post migration tasks, application testing, back out plan if required.
  - Explain what will be upgraded, for example, AIX, HACMP, SDD etc and why.
- ▶ Clone the system to be migrated onto a “test” LPAR and perform the migration. This will at least allow you to review your plan and verify that the AIX migration will work. Resolve and document any issues you might encounter during the “real” migration on the production system. If required clone the entire system i.e. rootvg, datavg, all applications (not just a mkysyb

restore) and do application verification testing. Even test newer software like SDD or SDDPCM for AIX 5L V5.3. Test your back out plan on the test clone.

- ▶ What other software do you need to upgrade? e.g. newer or correct version of SDD, SDDPCM, lsof, nmon, SSH, SSL, rpm, HACMP, ibm2105, device filesets etc. All the necessary fixes to support software on AIX 5L V5.3, for example, HACMP 5.2 required some APARs to support AIX 5L V5.3. Ask your storage administrator to ensure that the LIC level on the storage device you are running supports AIX 5L V5.3 and the newer version of SDD or SDDPCM.

The following pre migration tasks also need to be completed:

- ▶ Document the systems configuration, for example, hardware, AIX, etc. Use **snap** or a custom script. This is important if you run into issues and need to recover or rebuild your system.
- ▶ Run the AIX 5L V5.3 `pre_migration` script and review the output to verify the system is ready for a migration.
- ▶ Check that there is sufficient disk space for the file systems in rootvg plus some free PP's (500MB) for file system expansion if required.
- ▶ Run the **blvset** tool to check the AIX version label for all the disks in rootvg. Correct if necessary.
- ▶ Check the status of the system. Also look for hardware or other errors in the AIX error report.

### ***Performing the migration***

Once the prerequisites have been met, the planning is complete and the pre-migration tasks have been executed successfully, we are ready to perform the migration of the NIM client. The following steps could be used as a template for any migration plan required in the future:

1. Run the AIX 5L V5.3 `pre_migration` script. The `pre_migration` script will save its output to the `/home` file system (ensure there is sufficient space in `/home`). The `pre_migration` script can be found in the AIX 5L V5.3 SPOT on the NIM master, for example, `/AIX53ML4/SPOT_53_ML4/usr/lpp/bos/pre_migration` or on the AIX 5L V5.3 base install media. We will copy the script to our NIM client for execution. It is also recommended that you confirm that the AIX level for all disks in rootvg is correct prior to the migration. You can use the `blvset` command to perform this task as shown in Example 4-44.

*Example 4-44 Output from the `pre_migration` script and `blvset` tool*

---

```
# rcp nimmast:/AIX53ML4/SPOT_53_ML4/usr/lpp/bos/pre_migration /tmp
# chfs -a size=+1 /home
# /tmp/pre_migration
```

All saved information can be found in: `/home/pre_migration.060309123949`

Checking size of boot logical volume (hd5).

Your rootvg has mirrored logical volumes (copies greater than 1)  
 Recommendation: Break existing mirrors before migrating.

Listing software that will be removed from the system.

Listing configuration files that will not be merged.

Listing configuration files that will be merged.

Saving configuration files that will be merged.

Running lppchk commands. This may take awhile.

Please check /home/pre\_migration.060309123949/software\_file\_existence\_check for possible errors.

Please check /home/pre\_migration.060309123949/software\_checksum\_verification for possible errors.

Please check /home/pre\_migration.060309123949/tcbck.output for possible errors.

All saved information can be found in: /home/pre\_migration.060309123949

It is recommended that you create a bootable system backup of your system before migrating.

; Run the blvset command.

; e.g. /usr/lpp/bosinst/blvset -d /dev/hdisk# -glevel <--should show 5.2

```
# /usr/lpp/bosinst/blvset -d /dev/hdisk0 -glevel
5.2
```

```
# /usr/lpp/bosinst/blvset -d /dev/hdisk1 -glevel
5.2
```

; If you do not get 5.2 from the above then run the following command on all rootvg disks.:

```
# /usr/lpp/bosinst/blvset -d /dev/hdisk# -plevel <--enter 5.2 when prompted.
```

2. Check that there is enough disk space to meet the minimum requirements for the file systems in rootvg. Expand file systems if required. Lack of disk space in rootvg file systems is a common cause of migration failure. The AIX 5L V5.3 release notes contain the latest information regarding requirements for disk space. Ensure you have some free physical partitions (PPs) in rootvg in

case file system expansion is required during the migration. Refer to Example 4-45.

---

*Example 4-45 Check free disk space requirements in rootvg file systems*

---

```
; Free space required for rootvg file systems.
/   16 MB
/usr 1256MB;
/var 128 MB;
/tmp 128 MB;
/opt 140 MB;
; Use df command to check free space.
# df -m
;Use lsvg command to check free PP's in rootvg.
# lsvg rootvg | grep FREE
```

---

3. Document the configuration of the server. You can use **snap** (e.g. **snap -ac**) command or a custom script to collect important information about your AIX system. We use a custom script([AIXinfo](#), see [Additional Materials](#)) to collect information about our system. Regardless of the tool used, ensure that the information is stored on a system other than the system to be migrated. It's most likely that the time you need to call on this information will be the time when the client is down and you are trying to recover it. The system configuration information will not be obtainable if the system is not up.

What follows is just some of the system configuration information you would collect and document before the migration.

Review `/etc/tunables/nextboot` file, as shown in Example 4-46.

---

*Example 4-46 Review and document /etc/tunables*

---

```
# cat /etc/tunables/nextboot
# IBM_PROLOG_BEGIN_TAG
# This is an automatically generated prolog.
#
# bos520 src/bos/usr/sbin/perf/tune/nextboot 1.1
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 2002
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# IBM_PROLOG_END_TAG
```

```
info:
    AIX_level = "5.2.0.54"
```

```

Kernel_type = "MP64"
Last_validation = "2005-04-18 11:31:25 EET (current, reboot)"

vmo:
  maxfree = "1536"
  minfree = "1440"

no:
  rfc1323 = "1"
  tcp_recvspace = "262144"
  tcp_sendspace = "262144"
  udp_recvspace = "65536"
  udp_sendspace = "65536"
  use_isno = "0"
  routerevalidate = "1"
  bcastping = "1"

```

---

Collect disk and volume group information as shown in Example 4-47. Capture this information to a file on another system for future reference.

*Example 4-47 Collect disk and volume group information*

---

```

# lsvg
# lsvg -o
# lspv
# lsvpcfg ; Optional. Only for SDD systems.
# lscfg -vpl hdisk0 ;Do this for each boot disk.
# bootinfo -b ; Determine which disk we booted off.
hdisk0

```

---

Collect IP configuration information as shown in Example 4-48 on page 173. Capture this information to a file on another system for future reference.

*Example 4-48 Collecting network and IP information*

---

```

# hostname
# host LPAR4
# ifconfig en0
# lscfg -vpl ent0 | grep Loc
# ifconfig -a
# netstat -nr
# entstat -d entX

```

---

Run **snap** or a script (in our case AIXinfo) to capture information about the system as shown in Example 4-49.

*Example 4-49 Using snap and a script to document the system*

---

```

# snap -ac

```

```
# AIXinfo -pre
```

---

Check the AIX error report. If everything looks fine then it is a good idea to clear the error report and SMIT logs before the migration to avoid confusion. See Example 4-50.

---

*Example 4-50 AIX error report status and SMIT log clean up*

---

```
# errpt
# errpt -a
# mkdir /tmp/errptlogs
# errpt > /tmp/errptlogs/errpt.out
# errpt -a >> /tmp/errptlogs/errpt.out
# errclear 0
# mkdir /tmp/smitlogs
# cd /
# cp smit.* /tmp/smitlogs/
# >smit.log
# >smit.script
# >smit.transaction
```

---

4. Create a mksysb backup of the system as shown in Example 4-51 on page 174. How you do this depends on your environment. Client mksysb backups may be initiated from the NIM master or from the client via a script. In our environment we used a script which runs on the client to create a mksysb backup on the NIM master. Refer to 4.11, “Backing up clients with NIM” on page 353 for more information.

Data in other volume groups also needs to be backed up with savevg or other backup tools such as Tivoli Storage Manager (TSM). We recommend using a script to backup just the structure of the volume group (savevg with all files excluded) and then use TSM to backup the data. To recover the data volume group would then only require a recreation of the volume group structure (i.e. restvg to recreate logical volumes and file systems) and then a complete restore of the data from TSM. We use a custom script in our example to backup the volume group structure. See the [sysbtonim](#) and [skelvg](#) scripts in “Additional Materials”.

---

*Example 4-51 Creating a mksysb backup of the server before the migration*

---

```
# /usr/local/bin/skelvg >> /var/log/skelvg.log 2>&1
# /usr/local/bin/sysbtonim >> /var/log/sysbtonim.log 2>&1
```

---

5. Install the bos.alt\_disk\_install filesets. These filesets are required in order to clone the root volume group. We install the filesets on the client via NIM as shown in Example 4-52.



*Example 4-52 Installing the bos.alt\* filesets via NIM*


---

```
# smit nim_task_inst

        Install and Update from ALL Available Software

* Installation Target                                LPAR4
* LPP_SOURCE                                         lpp_sourceaix5204
* Software to Install                               [bos.alt_disk_install]+
```

---

6. Disable system monitoring. If you have any monitoring software within your environment remember to disable it or schedule downtime for the system, otherwise you may receive unwanted alerts while the system is down during the migration process.
7. Backup the /etc/motd and /etc/sendmail.cf files as shown in Example 4-53. These files are replaced during the migration and you will lose any custom configuration in these files. Back them up to another directory on the system for easy recovery and reference, and also backup them with the backup tools within your environment. For example, the /home directory is not removed during a migration so create a directory in /home and copy the files to that location.

*Example 4-53 Backing up sendmail and other configuration files*


---

```
# mkdir /home/premig
# cp -p /etc/motd /home/premig/
# cp -p /etc/sendmail.cf /home/premig/
```

---

8. Stop any applications running on the system.
9. Disable cron as shown in Example 4-54. This is to avoid the (unlikely) possibility of a cron job running and causing issues or changing data or trying to access applications that are down just prior to migrating the system.

*Example 4-54 Disabling cron before the migration*


---

```
# chitab "cron:23456789:off:/usr/sbin/cron"
# telinit q
```

---

10. Commit applied software. Remove any efixes on the system first as shown in Example 4-55.

*Example 4-55 Commit applied software and check for efixes*


---

```
# emgr -P
# emgr -r -L fixid
# smit commit
```

---

11. In preparation for the cloning of rootvg, we first un-mirror rootvg so that we can use one of the disks for the alt\_disk\_install operation as shown in Example 4-56.

*Example 4-56 Unmirror rootvg*

---

```
# bootinfo -b
hdisk0
# unmirrorvg -cl rootvg hdisk1
# chpv -c hdisk1
# lspv -l hdisk1 ; migratepv hdisk1 hdisk0 (if required).
# lspv -l hdisk1 ; disk should be empty now.
# reducevg rootvg hdisk1
# lsvg -p rootvg
rootvg:
PV_NAME          PV STATE          TOTAL PPs   FREE PPs   FREE DISTRIBUTION
hdisk0 active              521         390        104..17..61..104..104
# bosboot -a -d /dev/hdisk0
# bootlist -m normal hdisk0
# bootlist -m normal -o
hdisk0
```

---

12. Clone rootvg to hdisk1. This alternate rootvg disk could be used if we need to back out our migration and go back to AIX 5L V5.2. The *-B* flags prevent the bootlist from being set after the alternate disk creation is finished as shown in Example 4-57.

*Example 4-57 Cloning rootvg with alt\_disk\_install*

---

```
# alt_disk_install -B -C hdisk1
Calling mkszfile to create new /image.data file.
Checking disk sizes.
Creating cloned rootvg volume group and associated logical volumes.
Creating logical volume alt_hd5.
Creating logical volume alt_hd6.
Creating logical volume alt_hd8.
Creating logical volume alt_hd4.
Creating logical volume alt_hd2.
Creating logical volume alt_hd9var.
Creating logical volume alt_hd3.
Creating logical volume alt_hd1.
Creating logical volume alt_hd10opt.
Creating /alt_inst/ file system.
Creating /alt_inst/home file system.
Creating /alt_inst/opt file system.
Creating /alt_inst/tmp file system.
Creating /alt_inst/usr file system.
Creating /alt_inst/var file system.
Generating a list of files
```

```

for backup and restore into the alternate file system...
Backing-up the rootvg files and restoring them to the alternate file system...
Modifying ODM on cloned disk.
Building boot image on cloned disk.
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst
forced unmount of /alt_inst
Changing logical volume names in volume group descriptor area.
Fixing LV control blocks...
Fixing file system superblocks...

```

```

# lspv | grep root
hdisk0 00531d9a33ff6ab5          rootvg          active
hdisk1 00531d9a47ed2df6          altinst_rootvg  active

```

---

13. Ensure all users are logged off and prevent new user logon as shown in Example 4-58.

*Example 4-58 Checking for active user sessions and preventing new logon*

---

```

# w
# touch /etc/nologin

```

---

14. Setup the NIM environment for the upgrade of the NIM client. We reset the NIM client to ensure that it is in a clean state before attempting the migration. We run a custom script (see Example 4-59 on page 178) to reset the NIM client. The script does a force reset, deallocates resources and clears the clients CPUID.

If a failure occurs during a NIM migration/installation with the message code “0042-008 NIM has attempted to establish socket communications with a remote machine, and it has refused the connection”, the cpuid attribute on the client’s machine definition may be obsolete (for example, if the machine’s system planar was recently replaced or the client definition was recently used for a NIM installation on a different system). To guarantee that this is not the case, erase the cpuid from the machine definition.

**Tip:** The CPU ID of a NIM client is stored in the NIM database so that the master can perform verification that NIM client commands are coming from the machines that were originally registered as clients. A NIM administrator would not want this CPU ID validation to be performed in the following situations:

- ▶ When the hardware of a client machine is changed, giving the client a new CPU ID.
- ▶ When a single client definition is used to install different machines, as on a preinstall assembly line

To disable client CPU ID validation, set the attribute `validate_cpuid=no` on the NIM master:

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

**Attention:** The value of the `validate_cpuid` attribute should not be changed while operations are being performed on NIM clients because it could potentially disrupt client communications for active machines.

---

*Example 4-59 Reset the NIM client before the migration*

```
{nimmast}:/ # cat /work/nim_scripts/resetnimclient
#!/usr/bin/ksh
# Reset a NIM client.

if [[ "$1" = "" ]] ; then
    echo Please specify a NIM client to reset e.g. LPAR4.
else
    if /usr/sbin/lsmim -l $1 > /dev/null 2>&1 ; then
        /usr/sbin/nim -o reset -F $1
        /usr/sbin/nim -Fo deallocate -a subclass=all $1
        /usr/sbin/nim -Fo change -a cpuid= $1
    else
        echo Not a valid NIM client!
    fi
fi

{nimmast}:/ # /work/nim_scripts/resetnimclient LPAR4
```

---

Then we enabled a BOS installation for NIM client LPAR4 using our AIX 5L V5.3 `lpp_source` and `SPOT`. We also chose not to initiate reboot of the client now (Example 4-60).

---

*Example 4-60 Installing the client*

```
{nimmast}:/ # smitty nim_bosinst
```

## Install the Base Operating System on Standalone Clients

```

* Installation Target                LPAR4
* Installation TYPE                  rte
* SPOT                               SPOT_53_ML4
  LPP_SOURCE                         [LPP_53_ML4]

ACCEPT new license agreements?      [yes]
this target?

Initiate reboot and installation now? [no]

```

---

Verify that the NIM resources are now setup to migrate the NIM client. Check the status of the NIM client using `lsnim`, ensure that the AIX 5L V5.3 `lpp_source` and `SPOT` have been allocated, confirm that the clients `Cstate` attribute is set to “BOS installation has been enabled”, review the files in `/tftpboot` and check the entry in `/etc/bootptab` is correct. We executed a script (`chknim`) to show us the current status of the NIM client, as shown in Example 4-61, and Example 4-62 on page 180.

*Example 4-61 Verifying the NIM client is setup to migrate*


---

```

{nimmast}:/usr/local/bin # chknim LPAR4
/tftpboot files.
total 79680
-rw-r--r--  1 root    system    8162755 Jun 02 10:37
spotaix5104.chrp.mp.ent
-rw-r--r--  1 root    system    8720910 Jun 02 11:49
spotaix5204.chrp.mp.ent
-rw-r--r--  1 root    system    11688960 Jun 12 11:13 SPOT-V10.chrp.mp.ent
-rw-r--r--  1 root    system    12211712 Jun 13 13:13
SPOT_53_ML4.chrp.mp.ent
-rw-r--r--  1 root    system          934 Jun 15 12:10 lpar4.info
lrwxrwxrwx  1 root    system          33 Jun 15 12:10 lpar4 ->
/tftpboot/SPOT_53_ML4.chrp.mp.ent

/etc/bootptab entries:
#    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
lpar4:bf=/tftpboot/lpar4:ip=10.1.1.68:ht=ethernet:sa=10.1.1.1:sm=255.255.255.0:

```

```

nim client info:
LPAR4:
  class      = machines
  type       = standalone
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = NET_EN1 lpar4 0
  cable_type1 = tp
  Cstate     = BOS installation has been enabled
  prev_state = ready for a NIM operation
  Mstate     = currently running
  boot       = boot
  lpp_source = LPP_53_ML4
  nim_script = nim_script
  spot       = SPOT_53_ML4
  cpuid      = 001A85D24C00
  control    = master
  current_master = nimmast
  sync_required = yes

```

---

*Example 4-62 chknim script*

---

```

echo "/tftpboot files."
ls -ltr /tftpboot/
echo
echo "/etc/bootptab entries:"
tail /etc/bootptab
echo
if [[ "$1" != "" ]]
then
  echo "NIM client info:"
  lsnim -l $1
fi

```

---

15. Performing the AIX migration. We are now ready to start the AIX migration for the client. We connect to our HMC and open a console window for LPAR4. We will then configure our client (via the SMS menus) to boot from the network.

Login as root via the console and reboot the system as shown in Example 4-63 on page 181.

**Note:** We recommend to reboot the system and check that it restarts OK before the migration. This may uncover any issues that already exist on the system but would be blamed on the migration.

*Example 4-63 Reboot the system.*


---

```
# shutdown -Fr
```

---

- ▶ Press the "1" key to enter the SMS menu as shown in Example 4-64. Once within the SMS menu we can configure the clients IP details via the Remote IPL (Initial Program Load) setup menu.

*Example 4-64 SMS Menu*


---

```
1 = SMS Menu                5 = Default Boot List
6 = Stored Boot List       8 = Open Firmware Prompt
```

```
memory      keyboard    network    scsi      speaker
```

---

- ▶ Select "4. Setup Remote IPL (Initial Program Load)" (Example 4-65):

*Example 4-65 Setup Remote IPL*


---

```
pSeries Firmware
Version RG050215_d79e02_regatta
SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.
```

-----  
Main Menu

1. Select Language
2. Password Utilities NOT available in LPAR mode
3. View Error Log
- 4. Setup Remote IPL (Initial Program Load)**
5. Change SCSI Settings
6. Select Console NOT available in LPAR mode
7. Select Boot Options

-----  
Navigation Keys:

```
X = eXit System Management Services
```

```
Type the number of the menu item and press Enter or select Navigation Key:
```

---

- ▶ Select the network adapter to configure (Example 4-66):

*Example 4-66 NIC selection*


---

```
pSeries Firmware
Version RG050215_d79e02_regatta
SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.
```

-----  
NIC Adapters

```
Device                Slot                Hardware Address
```

1. 10/100 Mbps Ethernet PCI Adapt      2:U1.5-P2-I2/E1      000255afcdf7

-----  
Navigation keys:

M = return to Main Menu

ESC key = return to previous screen      X = eXit System Management Services

-----  
Type the number of the menu item and press Enter or select Navigation Key:  
-----

▶ Select "1. IP Parameters."(Example 4-67 on page 182):

*Example 4-67 IP Parameters menu*

pSeries Firmware

Version RG050215\_d79e02\_regatta

SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.

-----  
Network Parameters

10/100 Mbps Ethernet PCI Adapter II: U1.5-P2-I2/E1

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

-----  
Navigation keys:

M = return to Main Menu

ESC key = return to previous screen      X = eXit System Management Services

-----  
Type the number of the menu item and press Enter or select Navigation Key:  
-----

▶ Enter the *client, server* and *gateway* IP address and the *subnet mask* (Example 4-68):

*Example 4-68 Configuring IP parameters*

pSeries Firmware

Version RG050215\_d79e02\_regatta

SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.

-----  
IP Parameters

10/100 Mbps Ethernet PCI Adapter II: U1.5-P2-I2/E1

1. Client IP Address      [10.1.1.68]
2. Server IP Address      [10.1.1.1]
3. Gateway IP Address      [000.000.000.000]
4. Subnet Mask      [255.255.255.000]

-----  
Navigation keys:

M = return to Main Menu



ESC key = return to previous screen      X = eXit System Management Services

---

Type the number of the menu item and press Enter or select Navigation Key:

---

- ▶ Execute a “Ping Test “ to ensure client can communicate with the NIM master (Example 4-69):

*Example 4-69 Selecting Ping Test menu*

---

pSeries Firmware  
Version RG050215\_d79e02\_regatta  
SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.

---

Ping Test  
10/100 Mbps Ethernet PCI Adapter II: U1.5-P2-I2/E1  
Speed, Duplex: 100,full  
Client IP Address: 10.1.1.68  
Server IP Address: 10.1.1.1  
Gateway IP Address: 000.000.000.000  
Subnet Mask IP Address: 255.255.255.000  
Protocol: Standard  
Spanning Tree Enabled: No  
Connector Type: rj45

**1. Execute Ping Test**

---

Navigation keys:  
M = return to Main Menu  
ESC key = return to previous screen      X = eXit System Management Services

---

Type the number of the menu item and press Enter or select Navigation Key:

---

Ping Test results are shown in Example 4-70:

*Example 4-70 Successful ping test output*

---

```

-----
| Attempting Ping... |
-----

```

```

PING: chosen-network-type = ethernet,100,rj45,full
PING: client      IP = 10.1.1.68
PING: server      IP = 10.1.1.1
PING: gateway     IP = 0.0.0.0
PING: device      /pci@3fffbe09000/pci@b,2/ethernet@1
PING: loc-code    U1.5-P2-I2/E1

```

PING: Ready to ping:

PING: source hardware address is 0 2 55 af cd f7  
 PING: destination hardware address is 0 9 6b 2e 26 1d  
 PING: source IP address is 10.1.1.68  
 PING: destination IP address is 10.1.1.1

```

-----
| Ping Success. |
-----
  
```

Press any key to continue.....

- ▶ Exit back (*ESC*) to the Main Menu (Example 4-71 on page 184):

*Example 4-71 Back to SMS Main Menu*

```

pSeries Firmware
Version RG050215_d79e02_regatta
SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.
-----
  
```

Main Menu

1. Select Language
2. Password Utilities NOT available in LPAR mode
3. View Error Log
4. Setup Remote IPL (Initial Program Load)
5. Change SCSI Settings
6. Select Console NOT available in LPAR mode
7. Select Boot Options

-----  
 Navigation Keys:

X = eXit System Management Services

-----  
 Type the number of the menu item and press Enter or select Navigation Key:  
 -----

- ▶ Select “7. *Select Boot Options*.” (Example 4-72):

*Example 4-72 Boot Options*

```

pSeries Firmware
Version RG050215_d79e02_regatta
SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.
-----
  
```

Main Menu

1. Select Language
2. Password Utilities NOT available in LPAR mode
3. View Error Log
4. Setup Remote IPL (Initial Program Load)
5. Change SCSI Settings
6. Select Console NOT available in LPAR mode

## 7. Select Boot Options

---

 Navigation Keys:

X = eXit System Management Services

---

 Type the number of the menu item and press Enter or select Navigation Key:
 

---

► Select “1. *Select Install or Boot Device*”:

*Example 4-73 Boot Device selection*


---

 pSeries Firmware

Version RG050215\_d79e02\_regatta

SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.

- 
1. **Select Install or Boot Device**
  2. Configure Boot Device Order
  3. Multiboot Startup <OFF>
- 

---

 Navigation keys:

M = return to Main Menu

ESC key = return to previous screen

X = eXit System Management Services

---

 Type the number of the menu item and press Enter or select Navigation Key:
 

---

► Select “6. *Network*” (Example 4-74):

*Example 4-74 Network boot selection*


---

 pSeries Firmware

Version RG050215\_d79e02\_regatta

SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.

---

 Select Device Type

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

---

 Navigation keys:

M = return to Main Menu

ESC key = return to previous screen

X = eXit System Management Services

Type the number of the menu item and press Enter or select Navigation Key:

---

- ▶ Select the network adapter to boot from (Example 4-75):

*Example 4-75 Ethernet device selection*

---

pSeries Firmware  
 Version RG050215\_d79e02\_regatta  
 SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.

---

Select Device  
 Device Current Device  
 Number Position Name  
 1. 1 Ethernet ( loc=2:U1.5-P2-I2/E1 )

---

Navigation keys:  
 M = return to Main Menu  
 ESC key = return to previous screen X = eXit System Management Services

---

Type the number of the menu item and press Enter or select Navigation Key:

---

- ▶ Select "2. Normal Mode Boot"(Example 4-76):

*Example 4-76 Selecting Boot Mode*

---

pSeries Firmware  
 Version RG050215\_d79e02\_regatta  
 SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.

---

Select Task  
 Ethernet ( loc=2:U1.5-P2-I2/E1 )

1. Information
  2. **Normal Mode Boot**
  3. Service Mode Boot
- 

Navigation keys:  
 M = return to Main Menu  
 ESC key = return to previous screen X = eXit System Management Services

---

Type the number of the menu item and press Enter or select Navigation Key:

---

- ▶ Select "1. Yes" (Example 4-77):

*Example 4-77 Accepting boot parameters*


---

```
pSeries Firmware
Version RG050215_d79e02_regatta
SMS 1.3 (c) Copyright IBM Corp. 2000,2003 All rights reserved.
```

```
-----
Are you sure you want to exit System Management Services?
```

1. Yes
2. No

```
-----
Navigation Keys:
```

```
X = eXit System Management Services
```

```
-----
Type the number of the menu item and press Enter or select Navigation Key:
```

---

- The system will now perform a network boot as shown in Example 4-78.

*Example 4-78 System network boot*


---

```
STARTING SOFTWARE
PLEASE WAIT...
```

```
BOOTP: chosen-network-type = ethernet,100,rj45,full
BOOTP: server IP = 10.1.1.1
BOOTP: requested filename =
BOOTP: client IP = 10.1.1.14
BOOTP: client HW addr = 0 2 55 af cd f7
BOOTP: gateway IP = 0.0.0.0
BOOTP: device /pci@3fffbe09000/pci@b,2/ethernet@1
BOOTP: loc-code U1.5-P2-I2/E1
BOOTP: Cancel = ctrl-C
BOOTP R = 1 BOOTP S = 1
```

```
FILE: /tftpboot/lpar4
Load Addr=0x4000 Max Size=0xffffc000
FINAL Packet Count = 17034 FINAL File Size = 8720910 bytes.
```

```
Elapsed time since release of system processors: 345447 mins 24 secs
```

```
-----
Welcome to AIX.
boot image timestamp: 09:49 06/02
The current time and date: 12:13:35 06/15/2006
number of processors: 2 size of memory: 1024Mb
```

```
boot device:
/pci@3ffffbe09000/pci@b,2/ethernet@1:speed=100,duplex=full,bootp,10.1.1.1,,10.1.
1.14,10.1.1.1
closing stdin and stdout...
```

---

**Tip:** You also can force the system to boot from the network during next reboot with the **following** command:

```
# bootlist -m normal en0 bserver=10.1.1.1 gateway=10.1.1.1 \
client=10.1.1.68
# shutdown -Fr
```

This will boot the system from the network via en0. If there are issues with the network boot you will need to go into the SMS menus and perform a ping test in order to troubleshoot the problem.

- ▶ Next, you will be prompted to define the system console for the migration as shown in Example 4-79. Type 1 and press enter.

*Example 4-79 Defining the system console*

---

```
***** Please define the System Console. *****
```

Type a 1 and press Enter to use this terminal as the system console.

Pour definir ce terminal comme console systeme, appuyez sur 1 puis sur Entree.

Taste 1 und anschliessend die Eingabetaste druecken, um diese Datenstation als Systemkonsole zu verwenden.

Premere il tasto 1 ed Invio per usare questo terminal come console.

Escriba 1 y pulse Intro para utilizar esta terminal como consola del sistema.

Escriviu 1 i premeu Intro per utilitzar aquest terminal com a consola del sistema.

Digite um 1 e pressione Enter para utilizar este terminal como console do sistema.

```
>>> 1 Type 1 and press Enter to have English during install.
```

```
88 Help ?
```

```
>>> Choice [1]:
```

---

- ▶ The “Welcome to Base Operation System Installation and Maintenance” menu will appear next as shown in Example 4-80 on page 189. Select “2 *Change/Show Installation Settings and Install*”.

*Example 4-80 BOS install welcome menu*

---

Welcome to Base Operating System  
Installation and Maintenance

Type the number of your choice and press Enter. Choice is indicated by >>>.

```
>>> 1 Start Install Now with Default Settings
      2 Change/Show Installation Settings and Install
      3 Start Maintenance Mode for System Recovery

      88 Help ?
      99 Previous Menu
```

>>> Choice [2]:

---

- ▶ The installation and settings menu appears as shown in Example 4-82. Ensure that the Method of Installation is Migration. Check that the correct disk will be used for the migration.

For systems that boot from SAN (for example, rootvg resides on SAN disk), the hdisk numbers may not reflect the numbers recorded prior to the migration. To confirm that the correct disk is being selected for migration you should confirm the LUN ID of the rootvg disk is correct. You can do this by exiting out of the Migration menu and selecting the "New and Complete Overwrite" Menu. Then select the disks to install onto and Select "77 Display Alternative Disk Attributes". This will give you the LUN ID which you can then match against the lscfg output you captured earlier. Once you have confirmed the LUN ID to hdisk mapping is correct you can then re-enter the migration menu and continue with the migration. Refer to Example 4-81.

*Example 4-81 Alternative disk attributes for LUN IDs*

---

Name	Device	Adapter	Connection	Location
or Physical Location Code				
>>> 1	hdisk0	...	-W500507630303427E-	L4018400000000000
2	hdisk4	...	-W500507630308427E-	L400E400000000000
3	hdisk1	...	-W500507630303427E-	L4007400300000000

---

Select “3 *More Options*” as shown in Example 4-82

*Example 4-82 BOS installation settings menu.*

---

Installation and Settings

Either type 0 and press Enter to install with current settings, or type the number of the setting you want to change and press Enter.

- ```

1 System Settings:
  Method of Installation.....Migration
  Disk Where You Want to Install.....hdisk0

2 Primary Language Environment Settings (AFTER Install):
  Cultural Convention.....English (United States)
  Language.....English (United States)
  Keyboard.....English (United States)
  Keyboard Type.....Default

3 More Options (Desktop, Security, Kernel, Software, ...)

>>> 0 Install with the settings listed above.
```

```

88 Help ?      | +-----+
89 Previous Menu | | WARNING: Base Operating System Installation will
                | | destroy or impair recovery of SOME data on the
                | | destination disk hdisk0.
```

---

- a. The menu shown in Example 4-83 is displayed.

*Example 4-83 Select other options for the migration*

- 
- ```

1. Desktop ..... NONE
2. Enable Trusted Computing Base ..... No
3. Import User Volume Groups ..... Yes
4. Enable System Backups to install any System .... No
   (Install all devices and kernels)
5. Remove Java 1.1.8 Software ..... No
```
- 

- b. Select "1 Install with current settings". The migration installation confirmation menus follows as shown in Example 4-84.

*Example 4-84 Migration installation confirmation menu*

---

Migration Installation Confirmation

```

Disks: hdisk0
Cultural Convention: en_US
Language: en_US
Keyboard: en_US
Import User Volume Groups: yes
Enable System Backups to install any system: no
```



Remove Java 1.1.8 Software: no

>>> 1 Continue with Install

```

      88 Help ?          | +-----+
      99 Previous Menu | | WARNING: Base Operating System Installation will
                        | | destroy or impair recovery of SOME data on the
                        | | destination disk hdisk0.

```

>>> Choice [1]:

Select 1, "Install with current settings"

---

- c. Migration confirmation screens. Review the output from each selection on the confirmation menu (for example, 1, 2 and 3) as shown in Example 4-85. If you want to abandon the migration for any reason you can do so at this point by selecting option 4 which will reboot the system without migrating.

Select "0 Continue with the migration".

*Example 4-85 Migration confirmation menu*

---

Migration Confirmation

Either type 0 and press Enter to continue the installation, or type the number of your choice and press Enter.

- 1 List the saved Base System configuration files which will not be merged into the system. These files are saved in /tmp/bos.
- 2 List the filesets which will be removed and not replaced.
- 3 List directories which will have all current contents removed.
- 4 Reboot without migrating.

Acceptance of license agreements is required before using system. You will be prompted to accept after the system reboots.

>>> 0 Continue with the migration.

88 Help ?

```

+-----+ WARNING:
Selected files, directories, and filesets (installable options) from the Base
System will be removed. Choose 2 or 3 for more information.

```

>>> Choice[0]: 1

```

/etc/motd
/etc/rc
/etc/tsh_profile
/sbin/rc.boot

```

```

/usr/sbin/skulker

End of list. Press Enter

>>> Choice[0]: 2

End of list. Press Enter

>>> Choice[0]: 3
/lpp/bos
/tmp
/usr/lpp/bos/bos.rte
/usr/lpp/bos/bos.rte.*
/usr/lpp/bos/deinst1

End of list. Press Enter

>>> Choice[0]: 0

```

---

```

Saving system configuration files in /tmp/bos...
Removing obsolete filesets, directories, and files...

```

---

- ▶ Migration should now commence. Depending on your environment, the process should take 30 to 60 minutes. The system will reboot on completion of the migration process.

16. Monitor the status of the migration as shown in Example 4-86. You can monitor the status by watching the console output via a *vterm* session on the HMC or you can run the `lsnim -a info` command on the NIM master to view the progress of the client's migration.

*Example 4-86 Monitoring the migration status of LPAR4 (partial output)*

---

```

{nimmast}:/ # lsniminst LPAR4
LPAR4:
  info = LED 610: mount -r nimmast:/AIX53ML4/SPOT_53_ML4/usr /SPOT/usr
LPAR4:
  info = setting_console
LPAR4:
  info = verifying_data_files
LPAR4:
  info = prompting_for_data_at_console
LPAR4:
  info = BOS install 1% complete : Importing root volume group.
LPAR4:
  info = BOS install 4% complete : Copying old files
LPAR4:
  info = BOS install 7% complete : Restoring base operating system.
LPAR4:

```

```

    info = BOS install 12% complete : Initializing disk environment.
LPAR4:
    info = BOS install 13% complete : Over mounting /.
LPAR4:
    info = BOS install 16% complete : Merging.
LPAR4:
    info = BOS install 19% complete : Installing additional software.
LPAR4:
    info = BOS install 96% complete : Initializing dump device.
LPAR4:
    info = BOS install 97% complete : Network Install Manager customization.
LPAR4:
    info = BOS install 98% complete : Creating boot image.
LPAR4:
    info = BOS install 100% complete

```

---

We ran a script (Example 4-87) on the NIM master which continuously displayed the progress of our LPAR's migration.

*Example 4-87 Isniminst script*

---

```

host=$1
while true
do
    lsnim -a info $host
    sleep 5
done

```

---

17. Once the migration is finished and the client has rebooted we check the NIM client state to ensure it has been reset correctly after the NIM migration. Notice that the current state (Cstate) is now showing “*ready for a NIM operation*” as shown in Example 4-88.

*Example 4-88 Check the NIM client status after the migration*

---

```

{nimmast}:/ # lsnim -l LPAR4
LPAR4:
class          = machines
type           = standalone
connect        = shell
platform       = chrp
netboot_kernel = mp
if1            = NET_EN1 lpar4 0
cable_type1    = tp
Cstate        = ready for a NIM operation
prev_state     = not running
Mstate        = currently running
cpuid         = 001A85D24C00
Cstate_result  = success

```

```
current_master = nimmast
sync_required = yes
```

---

18. Check the system configuration after the migration completes. See Example 4-89 on page 194.

*Example 4-89 Checking the system after a migration*

---

```
# oslevel -r
# oslevel -s
# instfix -i | grep AIX
# instfix -icqk ML-LEVEL | grep “:-:”
# lppchk -m3 -v
# lppchk -m3 -c
# lspv
# lsvg
# lsvg -o
# lsdev -Cc disk
# lsvg -l rootvg
```

---

19. Check the files in /var/adm/ras as shown in Example 4-90. These log files can also be view from the NIM master via the NIM showlog operation.

*Example 4-90 Console, NIM and BOS installation/migration log files*

---

```
# alog -f /var/adm/ras/conslog -o
# more /var/adm/ras/bootlog
# more /var/adm/ras/bosinstlog
# more /var/adm/ras/nim.script
# more /var/adm/ras/devinst.log.
```

```
{nimmast}:/ # nim -o showlog -a log_type=bosinst LPAR4
{nimmast}:/ # nim -o showlog -a log_type=niminst LPAR4
{nimmast}:/ # nim -o showlog -a log_type=script LPAR4
```

---

20. Restore & Update /etc/motd and /etc/sendmail.cf as shown in Example 4-91. Do not just copy the sendmail.cf file over the new one as the configuration file changes with each version of sendmail. Review the file and determine which information needs to be migrated to the new sendmail.cf file.

*Example 4-91 /etc/motd and /etc/sendmail.cf files after a migration*

---

```
# cp -p /home/premig/motd /etc/motd
# vi /etc/motd
; change AIX version info etc.
# grep ^DS /home/premig/sendmail.cf
xyz-mailserver.com
# vi /etc/sendmail.cf
; Change DS line xyz-mailserver.com
```

```
; Send a test email.
# mail -v somebody@xyzcompany.com
# mailq
# tail /var/log/maillog
```

---

21. Cleanup any old AIX 5L V5.2 filesets as shown in Example 4-92 on page 195.

*Example 4-92 Cleanup AIX 5L V5.2 filesets if appropriate*

---

```
# lspp -l | grep 5.2
# smit deinstall
```

Remove Installed Software

```
* SOFTWARE name                                [bos.docsearch*]

PREVIEW only? (remove operation will NOT occur)      yes
REMOVE dependent software?                          yes
EXTEND file systems if space needed?                  no
DETAILED output?                                     no
```

---

22. Install SDD for AIX 5L V5.3. (Optional, only for systems using SDD). At this stage you would install the correct version of SDD for AIX 5L V5.3. You would need to remove the previous version for AIX 5.2 first. Please consult the SDD user's guide for more information.

23. Verify AIX (and optionally, SDD) levels are correct, as shown in Example 4-93.

*Example 4-93 Check the AIX version and level after the migration*

---

```
# oslevel -s
5300-04-01
# lppchk -m3 -v
# instfix -i | grep AIX
  All filesets for 5.3.0.0_AIX_ML were found.
  All filesets for 5300-01_AIX_ML were found.
  All filesets for 5300-02_AIX_ML were found.
  All filesets for 5300-03_AIX_ML were found.
  All filesets for 5300-04_AIX_ML were found.
```

```
# lspp -l devices.sdd*
Fileset                                Level State      Description
-----
Path: /usr/lib/objrepos
devices.sdd.53.rte 1.6.0.5 COMMITTED IBM Subsystem Device Driver
                                     for AIX V53

Path: /etc/objrepos
devices.sdd.53.rte 1.6.0.5 COMMITTED IBM Subsystem Device Driver
                                     for AIX V53
```

---

24. Check and clean the error report. Check the AIX error report for any hardware or other issues after the migration. If everything looks fine then clear the errpt if desired. See Example 4-94.

*Example 4-94 Checking the AIX error report after the migration*

---

```
# errpt
# errpt -a
# errclear 0
```

---

25. Optional - SDD was upgraded then a reboot must be performed now.

26. Optional - SDD - Verify data path's and volume groups as shown in Example 4-95. SDD will configure the altinst\_rootvg volume group as a vpath device. Rectify this situation.

*Example 4-95 Check the SDD configuration after the migration*

---

```
# datapath query adapter
# datapath query device
# datapath query wwpn

# lsvg
# lsvg -o
# lspv
# vp2hd altinst_rootvg
# rmdev -dl vpathX
```

---

27. Verify IP configuration as shown in Example 4-96.

*Example 4-96 Checking network configuration after the migration*

---

```
# ifconfig -a
# netstat -nr
# entstat -d entX
```

---

28. Verify that the date, time and certain environment variables (for example, LANG and TZ) are correct for your environment after the migration. Refer to Example 4-97.

*Example 4-97 Check environment variables and date/time after the migration*

---

```
# date
# echo $TZ
# echo $LANG
```

---

29. Upgrade SSH, SSL, Isosf and nmon to the correct version for AIX 5L V5.3 (if required).

30. Enable cron as shown in Example 4-98.

*Example 4-98 Enabling cron*


---

```
# chitab "cron:23456789:respawn:/usr/sbin/cron"
# telinit q
```

---

**Post-migration tasks**

This section describes the post migration tasks.

1. Run the `post_migration` script as shown in Example 4-99. The output is saved to `/home` directory. Review the files and ensure installation verification is OK.

*Example 4-99 Running the post\_migration script*


---

```
# rcp nimmast:/AIX53ML4/SPOT_53_ML4/usr/lpp/bos/post_migration /tmp
# /tmp/post_migration
```

---

2. Create a `mksysb` backup of the client as shown in Example 4-100.

*Example 4-100 Creating a mksysb backup of the client after the migration*


---

```
# /usr/local/bin/sysbtonim -s rootvg >> /var/log/sysbtonim.log 2>&1
```

---

3. Remove the pre & post migration directories in `/home`.
4. Start applications and perform verification testing.
5. Restore `rootvg` to a mirrored disk configuration as shown in Example 4-101. You should then reboot the system to turn off quorum for `rootvg`.

*Example 4-101 Mirroring rootvg after migration*


---

```
# alt_disk_install -X altinst_rootvg
# extendvg rootvg hdisk1
# mirrorvg rootvg hdisk1
# bosboot -a -d /dev/hdisk0
# bosboot -a -d /dev/hdisk1
# bootlist -m normal hdisk0 hdisk1
# bootlist -m normal -o
hdisk0
hdisk1
# shutdown -Fr
```

---

**Backing out a failed AIX migration**

We can follow the procedures in this section to back out our AIX 5L V5.3 migration. Using an alternate `rootvg` allows us to reboot on the original disk and restart the system with our previous version of AIX.

1. Set the `bootlist` to boot from original `rootvg` as shown in Example 4-102.

*Example 4-102 Setting the bootlist to the alternate rootvg*


---

```
# lspv | grep altinst_rootvg
hdisk1
# bootlist -m normal hdisk1
```

---

2. Check the bootlist only has hdisk1 as shown in Example 4-103.

*Example 4-103 Checking the bootlist*


---

```
# bootlist -m normal -o
hdisk1
```

---

3. Reboot the server as shown in Example 4-104.

*Example 4-104 Reboot the system*


---

```
# shutdown -Fr
```

---

4. Restore the mksysb if issues with the alternate rootvg or if it no longer exists. If required, restore rootvg to a mirrored disk configuration as shown in Example 4-105. You should then reboot the system to turn off quorum for rootvg.

*Example 4-105 Mirroring rootvg after the migration*


---

```
# alt_disk_install -X altinst_rootvg
# extendvg rootvg hdisk1
# mirrorvg rootvg hdisk1
# bosboot -a -d /dev/hdisk0
# bosboot -a -d /dev/hdisk1
# bootlist -m normal hdisk0 hdisk1
# bootlist -m normal -o
hdisk0
hdisk1
# shutdown -Fr
```

---

5. Reboot the server:
 

```
# shutdown -Fr
```
6. Verify the system and start the applications.

Our NIM migration to AIX 5L V5.3 is now complete. Our client has been successfully upgraded from AIX 5L V5.2 to AIX 5L V5.3.

#### 4.4.4 Debugging a NIM AIX migration

Enabling BOS install debug output is very useful if you need to troubleshoot a migration or installation via NIM. With debug information turned on, all the



commands and the output from the installation are displayed on the console or tty. This is can be useful, because you will be able to see the commands or operations that failed and then take corrective action. Using BOS debug output can save you time in determining the root cause of a problem and can reduce the scope of your investigation.

You will see the **showled** command and its output within the debug output. This command displays the systems LED value at each stage of the process. This can also be useful as many known problems can be referenced by the LED value displayed when a problem occurs.

There are two methods for enabling debug output during a BOS installation and/or migration:

1. The first method involves entering a special value at one of the installation menus.
2. The second method uses a `bosinst_data` resource to tell the installation program to display debug output.

### **Manually producing debug output**

Use this procedure to produce debug output without using a `bosinst_data` resource. To enable debugging for the BOS installation program, start by performing all the processing you would normally do to install or migrate a client. Because you are not using a `bosinst_data` resource, you will be prompted to supply information about the installation to the BOS installation program.

1. Select your console.
2. Select your language.
3. The Welcome to Base Operating System Installation and Maintenance menu is displayed. Instead of selecting one of the options, type **911** at the prompt and press Enter.
4. Continue the normal procedure for selecting options and specifying data until the installation begins. Debug output will be sent to the client's display while the installation proceeds.

### ***Automatically producing debug output***

Use this procedure to produce debug output when using a `bosinst_data` resource.

1. To enable debugging for the BOS installation program, set the value `BOSINST_DEBUG = yes` in the `control_flow` stanza of the `bosinst.data` file that you are using for your `bosinst_data` resource.

A minimum bosinst.data file for debugging purposes would contain the following lines:

```
control_flow:
    BOSINST_DEBUG = yes
```

2. In addition to the processing you would normally do to install or migrate a client, include the modified bosinst\_data resource as a resource for the operation.

After the client boots over the network, it will use the bosinst\_data resource to obtain settings for the installation. If the only data specified in your bosinst.data file is BOSINST\_DEBUG = yes, you will be prompted for the remaining required information before the installation will continue. Debug output will be sent to the client's display while the installation continues.

In Example 4-106, we use a combination of the **script** command and the **vtmenu** tool on the HMC to capture all of the debug output to a file for further analysis.

- a. We prepare our migration for the client as shown previously.
- b. Run the **script** command to capture session output.

```
{nimmast}:/ # cd /tmp ; script bosdebug.out
```

- c. Connect to the HMC and run **vtmenu** to connect to the LPARs console (in our case lpar4 is the system to be migrated so we entered "3" to open a virtual terminal).

*Example 4-106 Using vtmenu to open a console*

---

```
{nimmast}:/tmp # ssh hscroot@hmcvico
[hscroot@hmcvico hscroot]$ vtmenu
```

```
Retrieving name of managed system(s) . . . vico
```

```
-----
Partitions On Managed System: vico
-----
```

```
1)   lpar6           Running:
2)   lpar5           Running:
3)   lpar4           Running:
4)   lpar3           Running:
5)   lpar2           Running:
6)   lpar1           Running:
7)   lpar7           Ready:
8)   lpar8           Ready:
9)   lpar9           Ready:
```

```
Enter Number of Running Partition (q to quit): 3
```

Opening Virtual Terminal On Partition lpar4 . . .

NVTS hmcvico 9734 004\*7040-671\*01A85D2 1 004\*7040-671\*01A85D2 \_VT\_

AIX Version 5

(C) Copyrights by IBM and by others 1982, 2004.

Console login:

- 
- d. Start the normal network boot process. When presented with the "Welcome to Base Operating System" menu, we entered **911**. A message appeared at the bottom of the screen letting us know that debug is enabled, for example, *BOSINST\_DEBUG enabled*. See Example 4-107.

---

*Example 4-107 Enabling BOS installation debug output*

---

```

Welcome to Base Operating System
Installation and Maintenance

```

Type the number of your choice and press Enter. Choice is indicated by >>>.

```

>>> 1 Start Install Now with Default Settings

      2 Change/Show Installation Settings and Install

      3 Start Maintenance Mode for System Recovery

      88 Help ?
      99 Previous Menu

```

```

>>> Choice [1]: 911

```

```

Welcome to Base Operating System
Installation and Maintenance

```

Type the number of your choice and press Enter. Choice is indicated by >>>.

```

>>> 1 Start Install Now with Default Settings

      2 Change/Show Installation Settings and Install

      3 Start Maintenance Mode for System Recovery

```

```

BOSINST_DEBUG enabled

```

```

88 Help ?
99 Previous Menu

```

>>> Choice [1]:

---

- e. We continue with migration as normal.
- f. All debug output is displayed to the screen (tty) as shown in Example 4-108.

*Example 4-108 BOS debug partial output*

---

```
+ [[ migrate = overwrite ]]
+ [[ migrate = erase_only ]]
+ [ -z ]
+ Log Shrink_It
+ + ../SPOT/usr/lpp/bosinst/bidata -i -g logical_volume_policy -f SHRINK
SHR=
+ + ../SPOT/usr/lpp/bosinst/bidata -i -g image_data -f PRODUCT_TAPE
PT=yes
+ [ = yes -a yes = no ]
+ [ = yes -a 0 = 1 ]
+ return 0
+ + ../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f ENABLE_64BIT_KERNEL
ENABLE_64BIT=Default
+ + ../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f CREATE_JFS2_FS
CREATE_JFS2_FS=Default
+ bootinfo -y
+ 2> /dev/null
+ [ 64 = 64 ]
+ bootinfo -a
+ 2> /dev/null
+ [ 3 = 3 ]
+ [[ Default = [Yy][Ee][Ss] ]]
+ [[ Default = [Dd][Ee][Ff][Aa][Uu][Ll][Tt] ]]
+ [[ yes = [Yy][Ee][Ss] ]]
+ KERNEL64ENABLE=no
+ [[ Default = [Yy][Ee][Ss] ]]
+ [[ Default = [Dd][Ee][Ff][Aa][Uu][Ll][Tt] ]]
+ [ yes = yes ]
+ CREATEJFS2=no
+ Log Prepare_Target_Disks
+ typeset CURR_PVID=
+ typeset NEW_PVID=
+ typeset PAGELV=
+ typeset RLV=
+ typeset FS=
+ + Get_Primary_blv
+ typeset ERROR=eval return 1
+ typeset BLV=
+ typeset TYPE=
```

```

+ + ../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c LABEL -v
primary_bootlv
BLV=
+ [[ -z ]]
+ + awk {print $1}
+ ../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c TYPE -v
boot
BLV=hd5
+ print hd5
+ return 0
PRIMARY_BLV=hd5

```

---

- g. Once the migration has completed successfully and the system has rebooted, we disconnect from the HMC (use `~.` to close the vtmenu session).

The **script** command output file (`bosdebug.out`) in `/tmp` on the NIM master contains all the debug output including the **showled** command. See Example 4-109.

*Example 4-109 Partial debug output including showled command*

---

```

{nimmast}:/ # more bosdebug.out
.....
+ ../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
IM=migrate
+ [ migrate = overwrite ]
+ + expr 5 + 1
PERCENT=6
+ Change_Status 6
+ return 0
+ + expr 6 + 1
PERCENT=7
+ Change_Status 7
+ return 0
+ [ -z ]
+ /usr/lib/methods/showled 0xA54
+ Log Restore_System
+ [ 5 -eq 5 ]
+ grep loc_lpp_src
+ lsvg -l rootvg
+ 1> /dev/null 2>& 1
+ rc=1
+ [ 1 -eq 0 ]
+ dspmsg ../usr/lib/nls/msg/C/BosMenus.cat -s 10 59 Restoring base operating
system.\n
.....
+ + ../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f CUSTOMIZATION_FILE
CF=

```

```

+ [ -z ]
+ + ../SPOT/usr/lpp/bosinst/bidata -i -g post_install_data -f BOSINST_FILE
CF=
+ cd /
+ [ -f ../ ]
+ [ -f ]
+ return 0
+ /usr/lib/methods/showled 0xA46
+ Finish
+ Change_Status 100
+ alog -t bosinst -q -s 16384
+ cat ../var/adm/ras/bi.log /var/adm/ras/bi.log
cat: cannot open /var/adm/ras/bi.log
+ cp ../var/adm/ras/bi.log.* /var/adm/ras
+ 2> /dev/null
+ [ -z ]
+ [[ -s ../var/adm/ras/BosMenus.log0 ]]
+ cp ../var/adm/ras/BosMenus.log /var/adm/ras
+ 2> /dev/null
+ mv /bosinst.data /var/adm/ras
+ 2> /dev/null
+ mv /image.data /var/adm/ras
+ 2> /dev/null
+ cp ../Ch_Stat.log /var/adm/ras
+ 2> /dev/null
+ [ = true ]
+ + pwd
CWD=/
+ cd /tmp
+ ar x /usr/lpp/bos/liblpp.a
+ echo

+ cat bos.rte.copyright
  Licensed Materials - Property of IBM
  .....
+ ar t /usr/lpp/bos/liblpp.a
+ rm -f bos.rte.copyright bos.rte.usr.rmlist bos.rte.root.rmlist
bos.rte.cfgfiles bos.rte.post_i bos.rte.pre_i incompat.pkgs
productid bos.rte.inventory bos.rte.al bos.rte.size bos.rte.tcb
+ cd /
+ rm -f /liblpp.a /lpp_name /SPOT ../Update_Status.pn
+ [ -n ]
+ rm -f /var/adm/ras/bi.log
+ 1> /dev/null
+ [ 5 -eq 5 ]
+ nimclient -R success
+ nimclient -S shutdown
+ [ 5 -eq 3 ]
+ [ 5 -eq 3 ]

```

```
+ [[ 5 -eq 3 ]]
+ [[ 5 -eq 3 ]]
+ mv /SPOT.save.7242 /SPOT
+ 1> /dev/null 2>& 1
+ [ migrate = migrate -a 5.2 != 3.2 ]
+ [ -z ]
+ rm -rf /dev
+ 1> /dev/null 2>& 1
+ mkdir /dev
+ chown root.system /dev
+ chmod 775 /dev
+ cd /dev
+ restore -xqf /tmp/bos/dev.disk.bff
+ 1> /dev/null 2>& 1
+ [[ -L /tmp/bos/dev.disk.bff ]]
+ rm -f /tmp/bos/dev.disk.bff
+ [ yes = no ]
+ sync
+ sync
+ sync
+ umount -f /var
+ 2> /dev/null
+ umount -f /usr
+ 2> /dev/null
+ umount -f /
+ 2> /dev/null
+ reboot -q
Rebooting . .
```

---

## 4.5 NIM mksysb migration and nim\_move\_up POWER5 tools

Two new features for Network Installation Manager (NIM) and AIX migrations were introduced with AIX 5L V5.3. These new features are:

- ▶ mksysb migration
- ▶ `nim_move_up`

Both are designed to simplify and assist with the migration of older (possibly unsupported) levels of AIX running on older IBM RS/6000 or IBM @server pSeries hardware. Both processes assist you with migrating these “older” systems to AIX 5L V5.3 on POWER5 hardware. In this section, we describe and demonstrate the features of both utilities.

### 4.5.1 mksysb migration

A mksysb migration can be extremely valuable when you want to move a system to a new POWER5 machine running AIX 5L V5.3. For example, you may have an old AIX 4.3 system running on a IBM System p 6H1 server that you want to move to a POWER5 LPAR on a IBM System p p595. Given that AIX 4.3 is not supported on the POWER5 platform, in the days before “mksysb migrations”, the only course of action would have been to upgrade the AIX 4.3.3 system to AIX 5L V5.3 on the existing hardware (for example, the 6H1) and then clone the system via a mksysb to the new POWER5 LPAR. This process is now simplified with the mksysb migration.

A mksysb migration allows you to move a old AIX system (for example, AIX 4.3.3 or AIX 5L V5.1) to POWER5 without upgrading AIX on the existing server first. Essentially you boot of AIX 5L V5.3 media (in our case we use NIM) and recover the old AIX 4.3.3 or AIX 5L V5.1 mksysb image followed by an immediate migration to AIX 5L V5.3. This was not possible with previous versions of AIX and pSeries hardware. A mksysb migration is now the recommended way of moving unsupported hardware configurations of AIX 4.3 and AIX 5L V5.1 to new supported hardware and AIX 5L V5.3.

Of course this method is only intended for configurations that will not support a traditional migration method and bypasses the previous hardware limitation by restoring a mksysb on the new hardware and then performing the migration without needing to actually run AIX 4.3 on the new platform. Systems that can be migrated with a conventional migration should do so.

The end result is that the new system will be a clone of the existing system but now running AIX 5L V5.3 on a POWER5 platform. The existing system remains



the same, for example, running AIX 4.3. You may choose to use this method to perform a test migration of a system and certify the applications, databases and/or code against AIX 5L V5.3 on the clone before executing the real mksysb migration at some later stage.

### Requirements for a mksysb migration

- ▶ A customized bosinst.data file is required to perform a mksysb migration installation. Your customized bosinst.data file must meet the following requirements to be used with a mksysb migration:
  - The file must be provided using the supplementary diskette method or using the client file method (NIM).
  - A new variable called *MKSYSB\_MIGRATION\_DEVICE* must exist within the file. This variable specifies the name of the device that contains the mksysb. For a network installation, the valid value is the word *network*. Valid values are */dev/cddevice* number for a mksysb image on a CD-DVD, and */dev/rmtdevice* number for a mksysb image on tape.
  - The following variables in the CONTROL\_FLOW stanza must be set as follows:
    - PROMPT must be set to *no*.
    - INSTALL\_METHOD must be set to *migrate*.
    - EXISTING\_SYSTEM\_OVERWRITE must be set to *yes*.
    - RECOVER\_DEVICES must be set to *no*. A mksysb migration attempts to recover the sys0 attributed for the source system as specified in the mksysb ODM, but no other device-specific data is recovered from the source system.
    - Any user-supplied values for these variables is ignored.
- ▶ The file should list the disks to be installed in the TARGET\_DISK\_DATA stanza to ensure that only those disks are used. A mksysb migration is a combination of an overwrite installation and a migration installation. The overwrite portion destroys all of the data on the target disks. The TARGET\_DISK\_DATA stanza must have enough information to clearly single out a disk. If you supply an empty TARGET\_DISK\_DATA stanza, the default disk for the platform is used, if available. Example 4-110 shows possible values for the TARGET\_DISK\_DATA stanza.

*Example 4-110 Possible values for the TARGET\_DISK\_DATA stanza*

---

Sample 1. Disk names only (two disks)

```
target_disk_data:
PVID =
```

```

PHYSICAL_LOCATION =
CONNECTION =
LOCATION =
SIZE_MB =
HDISKNAME = hdisk0

```

```

target_disk_data:
PVID =
PHYSICAL_LOCATION =
CONNECTION =
LOCATION =
SIZE_MB =
HDISKNAME = hdisk1

```

#### Sample 2. Physical location specified (1 disk)

```

target_disk_data:
PVID =
PHYSICAL_LOCATION = U0.1-P2/Z1-A8
CONNECTION =
LOCATION =
SIZE_MB =
HDISKNAME =

```

#### Sample 3. By physical volume ID (PVID)(2 disks)

```

target_disk_data:
PVID = 0007245fc49bfe3e
PHYSICAL_LOCATION =
CONNECTION =
LOCATION =
SIZE_MB =
HDISKNAME =
target_disk_data:
PVID = 00000000a472476f
PHYSICAL_LOCATION =
CONNECTION =
LOCATION =
SIZE_MB =
HDISKNAME =

```

#### Sample 4. By SAN disk ID (ww\_name//lun\_id - 2 disks)

```
target_disk_data:
  SAN_DISKID = 0x500507630308427e//0x4021401a00000000

target_disk_data:
  SAN_DISKID = 0x500507630303427e//0x4010400000000000
```

---

Some basic prerequisites should be met before attempting a mksysb migration. All requisite hardware, including any external devices (such as tape, CD, or DVD-ROM drives), must be physically connected. Before you begin the installation, other users who have access to your system must be logged off. Verify that your applications run on AIX 5L V5.3. Also, verify that your applications are binary-compatible with AIX 5L V5.3. If your system is an application server, verify that there are no licensing issues. Refer to your application documentation or provider to verify on which levels of AIX your applications are supported and licensed. You can also check the AIX application availability guide at the following Web site address:

<http://www.ibm.com/servers/aix/products/ibmsw/list>

Verify that your hardware microcode is up-to-date. There must be adequate disk space and memory available. AIX 5L V5.3 requires 128 MB of memory and 2.2 GB of physical disk space. For additional release information, see the *AIX 5L V5.3 Release Notes* at the Web site:

[http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304\\_base\\_relnotes.pdf](http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304_base_relnotes.pdf)

Make a backup copy of your system software and data. If the source system is available, run the pre-migration script on it. Ignore any messages that pertain to the hardware configuration of the source system because the migration takes place on the target system. Correct any other problems as recommended by the script.

## 4.5.2 Performing a mksysb migration with NIM

The following section describes how to perform a mksysb migration with NIM. We will migrate an AIX 5L V5.1 system currently running as an LPAR on a p670 (lpar4) to a new LPAR (v1par4) running on a p520 with AIX 5L V5.3 (see Figure 4-4 on page 210 and Figure 4-5 on page 211).

```

# prtconf | head
System Model: IBM,7040-671
Machine Serial Number: 01A85D2
Processor Type: PowerPC_POWER4
Number Of Processors: 2
Processor Clock Speed: 1000 MHz
CPU Type: 64-bit
Kernel Type: 32-bit
LPAR Info: 4 lpar4
Memory Size: 1024 MB

# oslevel -r
5100-04

# lsdev -Cc disk
hdisk0 Available 57-08-00-8,0 16 Bit LVD SCSI Disk Drive
hdisk1 Available 5M-08-00-8,0 16 Bit LVD SCSI Disk Drive

# lspv
hdisk0          001a85d27c3dc850          rootvg
hdisk1          001a071f9a12dbaf          None
# lscfg -l hdisk0
DEVICE LOCATION      DESCRIPTION
hdisk0 57-08-00-8,0    16 Bit LVD SCSI Disk Drive (18200MB)

# lsdev -Cc adapter
ent0 Available 4p-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
scsi0 Available 57-08 Wide/Ultra-3 SCSI I/O Controller
scsi1 Available 5M-08 Wide/Ultra-3 SCSI I/O Controller
sa0 Available      LPAR Virtual Serial Adapter

```

Figure 4-4 Current hardware configuration for lpar4

These steps assume that the new LPAR has already been configured on the p520 prior to attempting the migration and that all hardware has been setup, configured and verified, for example, CPU, memory, hard disks, Ethernet, SCSI and/or fibre channel adapters. The new LPAR on the p520 is a micro-partition using VIO resources as shown in Figure 4-5 on page 211.

```
# prtconf | head
System Model: IBM,9131-52A
Machine Serial Number: 650E4DG
Processor Type: PowerPC_POWER5
Number Of Processors: 1
Processor Clock Speed: 1499 MHz
CPU Type: 64-bit
Kernel Type: 32-bit
LPAR Info: 6 v1par4
Memory Size: 512 MB

System configuration:
type=Shared
mode=Capped
smt=On
lcpu=2
mem=512
psize=2
ent=0.10

# lsdev -Cc disk
hdisk0 Available Virtual SCSI Disk Drive

# lspv
hdisk0 0000c836a465153e rootvg active

# lscfg -l hdisk0
hdisk0 U9131.52A.650E4DG-V6-C24-T1-L810000000000 Virtual SCSI Disk Drive

# lspv hdisk0 | grep TOTAL
TOTAL PPs:          319 (10208 megabytes)    VG DESCRIPTORS:    2

# lsdev -Cc adapter
ent0 Available Virtual I/O Ethernet Adapter (1-lan)
vsa0 Available LPAR Virtual Serial Adapter
vscsi0 Available Virtual SCSI Client Adapter
```

Figure 4-5 Hardware configuration for v1par4.

#### 1. Preparing the system for a mksysb migration

Verify which hard disks will be used for the mksysb migration on the target system. In our test, environment we only had internal SCSI disks so we only needed to include hdisk0 and/or hdisk1 in our custom bosinst.data file. However, if you are working in a SAN disk environment it may be necessary to try and determine which disks (for example, ww\_name and lun\_id) will be used for rootvg in advance of the migration. One way to do this is to NIM

install AIX 5L V5.3 onto the new systems rootvg disks and run the commands (shown in Example 4-111) to obtain the `lun_id` and `ww_name` for each rootvg disk. This information is used to complete the `SAN_DISKID` tag in the custom `bosinst.data`.

---

*Example 4-111 Commands to obtain the `lun_id` and `ww_name`*

---

```
# lsattr -EH -l hdisk6 | egrep 'lun_id|ww_name'
lun_id 0x4021401a00000000 Logical Unit Number ID True
ww_name 0x500507630308427e FC World Wide Name False

# lsattr -EH -l hdisk7 | egrep 'lun_id|ww_name'
lun_id 0x4010400000000000 Logical Unit Number ID True
ww_name 0x500507630303427e FC World Wide Name False
```

---

Prepare the custom `bosinst.data` file for the `mksysb` migration. Edit the `bosinst.data` file and change the options in the `bosinst.data` file (as shown in Example 4-112). This file was located on our NIM master (`/other_res/bid.np.hd0.mkmig`). We then checked the contents of our file by using the `bicheck` command to ensure there were no obvious errors.

---

*Example 4-112 Changing the `bosinst.data` file*

---

```
{nimmast}:/ # vi /other_res/bid.np.hd0.mkmig

INSTALL_METHOD = migrate
PROMPT = no
EXISTING_SYSTEM_OVERWRITE = yes
RECOVER_DEVICES = no
MKSYSB_MIGRATION_DEVICE = network

target_disk_data:
PVID =
PHYSICAL_LOCATION =
CONNECTION =
LOCATION =
SIZE_MB =
HDISKNAME = hdisk0

{nimmast}:/ # /usr/lpp/bosinst/bicheck /other_res/bid.np.hd0.mkmig
{nimmast}:/ # echo $?
0
```

---

**Note:** If you are in a SAN disk environment for rootvg, the stanza would look similar to the following:

```
target_disk_data:
SAN_DISKID = 0x500507630308427e//0x4021401a00000000

target_disk_data:
SAN_DISKID = 0x500507630303427e//0x4010400000000000
```

## 2. Define the custom bosinst.data file

Example 4-113 shows how we defined the custom bosinst.data file as a NIM resource.

### *Example 4-113 Defining the bosinst.data file as a NIM resource*

---

```
{nimmast}:/ # smit nim_mkres
bosinst_data = config file used during base system installation
* Resource Name [BID_NP_HDO_MKMIG]
* Resource Type bosinst_data
* Server of Resource [master]
* Location of Resource [/other_res/bid.np.hd0.mkmig]
```

---

## 3. Define the source clients AIX 5L V5.1 mksysb as a NIM mksysb resource

We already had an existing mksysb for the client, which we then defined as a mksysb resource for use with the mksysb migration, as shown in Example 4-114.

### *Example 4-114 Defining the mksysb as a resource for NIM*

---

```
{nimmast}:/ # smit nim_mkres
mksysb = a mksysb image
* Resource Name [lpar4-mksysb-mig]
* Resource Type mksysb
* Server of Resource [master]
* Location of Resource [/export/images/lpar4-mksysb.5104]
```

---

## 4. Allocate the NIM mksysb resource to the client, VLPAR4.

In Example 4-115, we allocated the mksysb resource to the target NIM client called VLPAR4.

### *Example 4-115 Allocating the NIM mksysb resource to the client*

---

```
{nimmast}:/ # smit nim
Perform NIM Administration Tasks
Manage Machines
Manage Network Install Resource Allocation
```

```
Allocate Network Install Resources
VLPAR4 machines      standalone
lpar4-mksysb-mig    mksysb
```

---

#### 5. Prepare for the AIX migration on the target NIM client, VLPAR4

We selected an `rte` install, the AIX 5L V5.3 `lppsource` and `SPOT` resources, our custom `bosinst_data` (`BID_NP_HDO_MKMIG`) and to accept the license agreements. We selected `no` to 'initiate a reboot and the installation now' as shown in Example 4-116.

#### Example 4-116 Preparing the client VLPAR4 for migration

---

```
{nimast}:/ # smit nim_bosinst
VLPAR4      machines      standalone
rte - Install from installation images
LPP_53_ML4  resources      lpp_source
SPOT_53_ML4 resources      spot
```

#### Install the Base Operating System on Standalone Clients

```
* Installation Target      VLPAR4
* Installation TYPE       rte
* SPOT                    SPOT_53_ML4
LPP_SOURCE                [LPP_53_ML4]
BOSINST_DATA to use during installation [BID_NP_HDO_MKMIG]
ACCEPT new license agreements? [yes]
Initiate reboot and installation now? [no]
```

---

#### 6. We then checked that our target client was ready for the NIM mksysb migration process

Example 4-117 shows the status of the NIM client, in particular the allocated `bosinst_data`, `lpp_source`, `spot` and `mksysb` resources.

#### Example 4-117 Status of the NIM client

---

```
{nimast}:/ # lsnim -l VLPAR4
VLPAR4:
class      = machines
type       = standalone
platform   = chrp
netboot_kernel = up
if1        = network1 LPAR4 0
cable_type1 = tp
Cstate    = BOS installation has been enabled
prev_state = ready for a NIM operation
Mstate     = not running
boot       = boot
bosinst_data = BID_NP_HDO_MKMIG
lpp_source   = LPP_53_ML4
```



```

mksysb          = lpar4-mksysb-mig
nim_script      = nim_script
spot           = SPOT_53_ML4
cpuid          = 00C69B9E4C00
control        = master

```

---

7. Next, we initiated a standard (manual) network boot of the target LPAR.

The mksysb migration then proceeded as an unattended installation i.e. non-prompted. Once the “*Installing Base Operating System*” screen appears the installation continues. The progress is displayed and the percentage complete field and elapsed time is incremented to indicate the overall status of the installation.

The mksysb from the source client is restored first. Once this is finished the actual migration to AIX 5L V5.3 is performed (this includes such tasks as merging configuration data, installing the base run-time environment and installing additional software).

Example 4-118 and Example 4-119 on page 216 show the output that we observed during the mksysb migration.

**Note:** Much of the output was removed to save space. Note that the BOS menus are disabled during the upgrade hence the non-prompted migration. If an error occurs the installation displays a message stating that the migration cannot continue, and *088* will be shown on the service display.

We monitored the mksysb migration status by watching the output on the console in one window (via vtmenu on the HMC - see Example 4-118) and observing the NIM installation information in another terminal, as shown in Example 4-119 on page 216 (using our custom *lsniminst* script that we introduced earlier in Example 4-87 on page 193).

*Example 4-118 Output of the mksysb migration*

---

```

{nimmast}:/ # ssh hscroot@hmc520
Last login: Tue May 30 15:56:55 2006 from 10.1.1.1
hscroot@hmc520:~> vtmenu

```

```

Retrieving name of managed system(s) . . . YLI894-2-650E4DG

```

```

-----
Partitions On Managed System: YLI894-2-650E4DG
0S/400 Partitions not listed

```

```

-----
1)  VI01                      Running
2)  vlpar1                    Running
3)  vlpar2                    Running

```

```

4)    v1par3                Running
5)    v1par5                Not Activated
6)    LPAR4_1par           Running
7)    v1par4                Running
8)    v1par6                Running

```

Enter Number of Running Partition (q to quit): 7

Opening Virtual Terminal On Partition v1par4 . . .

Open in progress..

```

*****
Mksysb\Migration Enabled
*****

```

```

Running: Init_Target_Disks...
Running: Other_Initialization...
Running: Fill_Target_Stanzas...
Running: Check_Other_Stanzas...

```

Installing Base Operating System

Please wait...

Approximate % tasks complete	Elapsed time (in minutes)
0	0

---

#### Example 4-119 Migration output

---

```

{nimmast}:/ # lsniminst VLPAR4
info = extract_diskette_data
info = BOS install 2% complete : Running: MnM_Restore_NIM_Required_Files...
info = BOS install 3% complete : Making paging logical volumes.
info = BOS install 3% complete : Running: MnM_Create_Rootvg_LVS...
info = BOS install 6% complete : Making file systems.
info = BOS install 6% complete : Making file systems.
info = BOS install 6% complete : Running: MnM_Restore_NIM_Mksysb...
info = BOS install 6% complete : Running: MnM_Restore_NIM_Mksysb...
info = BOS install 6% complete : 4% of mksysb data restored.
info = BOS install 6% complete : 4% of mksysb data restored.
info = BOS install 6% complete : 4% of mksysb data restored.

```

```

info = BOS install 6% complete : 4% of mksysb data restored.
info = BOS install 6% complete : 7% of mksysb data restored.
info = BOS install 6% complete : 51% of mksysb data restored.
info = BOS install 6% complete : 51% of mksysb data restored.
info = BOS install 6% complete : 98% of mksysb data restored.
info = BOS install 6% complete : Unmounting file systems...
info = BOS install 8% complete : Preserving old data.
info = BOS install 9% complete : Copying old files
info = BOS install 12% complete : Restoring base operating system.
info = BOS install 18% complete : Over mounting /.
info = BOS install 21% complete : Merging.
info = installing additional software
info = BOS install 98% complete : Installing additional software.
info = BOS install 98% complete : Installing additional
info = BOS install 100% complete : Network Install Manager customization.
info = BOS install 100% complete : Creating boot image.
info = BOS install 100% complete : Creating boot image.
info = BOS install 100% complete

```

---

After the migration completes, the system will reboot. The target LPAR on the p520 will now be a clone of the AIX 5L V5.1 source system but now running AIX 5L V5.3. You can now run the post-migration script to review the migration and perform any post migration steps that are necessary.

As mentioned earlier, device specific configuration from the source system is not recovered from the mksysb restore (*RECOVER\_DEVICES* was set to *no* in our custom bosinst.data file). Therefore you may need to perform some configuration tasks for devices such as network adapters or volume groups (although we encourage you to automate this via a NIM first boot customization script that runs after the system reboots post installation).

Once you are satisfied that the migration has been successful you can begin taking any action required to test your applications with AIX 5L V5.3 such as copying data across to the new system for testing.

### 4.5.3 Migrating a NIM client to a POWER5 logical partition using `nim_move_up`

The `nim_move_up` tool allows you to easily migrate a back-level AIX system onto a logical partition (LPAR) residing on a POWER5 (or newer) server. It automates much of the manual processes associated with migrating older systems to a current level. It is designed to simplify and assist with the migration of older (possibly unsupported) levels of AIX running on older IBM RS/6000 and IBM @server pSeries hardware. In this section, we describe and demonstrate the features of this tool.

The basic flow of a `nim_move_up` operation is as follows:

- ▶ A system backup (mksysb) of the original machine's rootvg is taken.
- ▶ An LPAR is created with equivalent hardware resources on the POWER5 machine.
- ▶ The original mksysb backup is migrated to AIX 5.3
- ▶ The migrated backup is installed onto the new LPAR on the POWER5 machine.

Before you can run the `nim_move_up` tool the NIM master must meet the following prerequisites :

- ▶ A configured NIM master running AIX 5L V5.3 with 5300-03 or above.
- ▶ Perl 5.6 or above.
- ▶ OpenSSH (obtainable from the Linux Toolbox CD).
- ▶ At least one stand-alone NIM client running AIX 4.3.3.75 or above. It must be a registered NIM client and have a valid `/etc/niminfo` file.
- ▶ AIX product media version AIX 5L with 5200-04 or higher, or AIX 5L product media version 5.3 or higher, or equivalent `lpp_source` and SPOT NIM resources.

You must also have the following hardware resources in your environment:

- ▶ A POWER5 server with sufficient hardware resources to support the target clients' equivalent POWER5 configuration.
- ▶ If virtual resources will be used to migrate the clients, an installed and configured Virtual I/O Server is required.
- ▶ An HMC controlling the POWER5 server, along with sufficient privileges to start, stop, and create LPARs.

**Note:** You will also require root user authority to run the `nim_move_up` command.

This `nim_move_up` process requires no downtime on the part of the original client. In addition, `nim_move_up` is capable of migrating a client onto virtualized hardware, such as virtual disks, using the Virtual I/O capabilities of the POWER5 server.

This migration process can be completed by the `nim_move_up` application in phases to allow more control over the process, or it can be completed all at once without any user interaction required. It is possible to migrate either a single system (a NIM client) or a group of systems (a NIM machine group) at the same time.

After a successful `nim_move_up` migration, the final configuration (in terms of the NIM master, source and target clients) will be similar to the following:

- ▶ The original NIM master.
- ▶ The original (source) NIM client unchanged.
- ▶ A new (target) LPAR(s) on the POWER5 server that corresponds to the original (source) NIM client(s) and are controlled by the NIM master.
- ▶ An HMC to control the LPAR(s) on the POWER5 servers.

### The different phases of `nim_move_up`

The `nim_move_up` migration process is completed in ten phases. To allow some control over the process, each phase can be executed individually or all at once. This phased approach allows for more control (if required) over how and when the migration executes.

**Phase 1.** The *Setup NIM resources* phase creates the needed NIM resources to perform the migration steps if they do not already exist or are not provided beforehand.

**Phase 2.** The *Pre-migration Software Assessment* phase performs an assessment on each target client to determine what software is installed and can be migrated. Any software that is missing from the `lpp_source` will be added from the source of installation images that should be provided to `nim_move_up`.

**Phase 3.** The *Client Hardware and Utilization Data Gathering* phase gathers data about each target client's hardware resources and attempts to assess how much of those resources are utilized on average over a given amount of time.

**Phase 4.** The *p5 Resource Availability Data Gathering and Client Resource Data Translation* phase searches the given managed system for available hardware resources. It uses the data gathered in the previous phase to create an equivalent LPAR configuration that utilizes the managed system's available resources. Creates the client LPARs with virtual I/O resources instead of physical I/O resources if `nim_move_up` was provided a Virtual I/O Server LPAR to work with. Creates the appropriate adapters and configuration on the Virtual I/O Server as they are needed.

**Phase 5.** The *Create System Backups* phase creates an installable image of each target client and its resources using the `mksysb` command.

**Phase 6.** The *Migrate Each System Backup* phase uses the `nimadm` command to migrate the newly-created installable images to the new level of AIX 5L.

**Phase 7.** The *Configure NIM Definitions of Client LPARs* phase uses the network information provided to the `nim_move_up` tool to create NIM standalone client

objects for the new LPARs created in the *p5 Resource Availability Data Gathering and Client Resource Data Translation* phase. Allocates the appropriate NIM resources and runs a **bos\_inst** pull operation (For example, NIM will not attempt to boot the client) on each NIM client.

**Phase 8.** The *Initiate LPAR Network Installations* phase reboots each LPAR via the service console (HMC) and initiates the installation.

**Note:** This phase ends when the installation begins. The actual progress of the installation is not monitored.

**Phase 9.** The *Post-migration Software Assessment* phase assesses the overall success of the migration after each installation, and reports on any software migration issues. It may be necessary to manually correct the errors reported for filesets that fail to migrate.

**Phase 10.** The optional *Post-installation Customization* phase performs a NIM customization operation on each client with the values provided if an alternate lpp\_source, fileset list, or customization script was provided to the nim\_move\_up application. This allows for the optional installation of additional software applications or for any additional customization that may be needed.

### ***nim\_move\_up and the mig2p5 tools.***

The LPAR sizing and creation process is performed by a set of scripts known as the mig2p5 tools. These tools have been developed to work with **nim\_move\_up** and also for standalone use. Appendix A, “General LPAR sizing and creation with mig2p5 tools” on page 625 for more information.

**Note:** The tools described in the following list can be retrieved from the Additional Materials accompanying this redbook.

The **nim\_move\_up** and mig2p5 tools (from Additional Materials) work together in the following way (refer to Figure 4-6):

1. Collect the source LPARs hardware configuration and CPU/memory utilization. This is performed by the *mig2p5* tools, **getSrcCfg** and **getSrcUtil**.
2. Capture the target POWER5 system’s hardware configuration. The *mig2p5* tool, **getTgtRsrc**, performs this task.
3. Generate the target LPAR’s required configuration, based on the hardware requirements and CPU/memory utilization collected in step one. The **genTgtCfg**, *mig2p5* tool, executes this task.

4. Create the target LPAR on the POWER5 system via the **createTgtLPAR**, mig2p5 tool.
5. A mksysb of the source system is created by **nim\_move\_up**.
6. The mksysb is migrated to AIX 5L V5.3 and installed to the new POWER5 LPAR via **nim\_move\_up**.

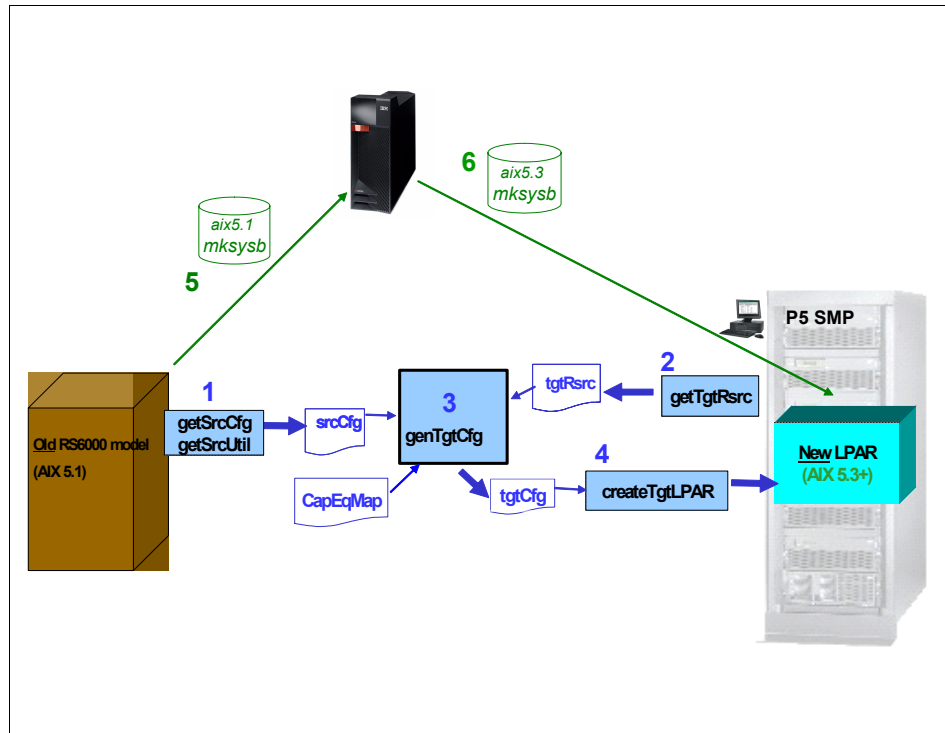


Figure 4-6 *nim\_move\_up* and *mig2p5* tools operation

#### 4.5.4 Example *nim\_move\_up* operation

In the following section, we describe how we used the **nim\_move\_up** tool to migrate a NIM client to AIX 5L V5.3. Our test environment consisted of an AIX 5L V5.1 LPAR (lpar4) on a p670 which was then migrated to an AIX 5L V5.3 LPAR on a p520. In our environment we used Virtual I/O (VIO) resources for the new POWER5 LPAR. A working VIO server had already been configured on the IBM System p p520.

**Tip:** For more information on configuring a VIO server please visit the following Web site:

<http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/topic/iphbl/iphblkickoff.htm>.

In our example, we chose to execute each phase of the “move up” operation individually instead of executing them all at once. This decision was made so that we could discuss each phase in some detail.

The end result is somewhat similar to the mksysb migration. For example, the original client will be unchanged and the new LPAR will be a clone of the original, only now it is running AIX 5L V5.3 on a POWER5 server. The big difference between the two methods is that `nim_move_up` automates the entire process, taking care of the creation of the client mksysb, creating the necessary NIM resources if they do not exist already, creating the LPAR definition (either with or without VIO resources), performing the AIX migration via `nimadm` and finally initiating the mksysb installation to the new POWER5 LPAR.

Before commencing the `nim_move_up` operation, we performed several “pre-flight” checks to confirm that our test environment met all the necessary requirements and was ready to support the migration.

1. We verified that the following software was installed at the correct levels on our NIM master.

Confirm the version of Perl is 5.6 or later:

```
{nimmast}:/ # lspp -l perl.rte
perl.rte 5.8.2.30 COMMITTED Perl Version 5 Runtime
```

Confirm that OpenSSH has been installed:

```
{nimmast}:/ # lspp -l openssh*
openssh.base.client 4.1.0.5301 COMMITTED Open Secure Shell Commands
openssh.base.server 4.1.0.5301 COMMITTED Open Secure Shell Server
openssh.license 4.1.0.5301 COMMITTED Open Secure Shell License
openssh.man.en_US 4.1.0.5301 COMMITTED Open Secure Shell
```

Check what version and maintenance/technology level of AIX is running on the NIM master. It must be at least AIX 5L V5.3 ML3:

```
{nimmast}:/ # oslevel -r
5300-04
```

2. Confirm that the source client is running AIX 4.3 .3.75 or later:

```
{nimmast}:/ # rsh lpar4 oslevel -r
5100-04
```



3. Review the available CPU, memory, network and storage adapter resources on the target POWER5 system.

Ensure that there will be enough spare capacity to meet the needs of the client. This can be done via your HMC and viewing the properties of the managed system. Click on each tab (Processor, Memory, etc.) and review the current available resources. If you intended to use VIO resources, ensure that there is already a VIO server configured and running. Logon to the VIO server and review the storage availability.

4. Now that we are satisfied that our environment meets all the requirements we are ready to start the `nim_move_up` configuration process.

First we will display and configure the `nim_move_up` input values. We run the `nim_move_up` command with the `-S` flag to display the current status of the operation, as shown in Example 4-120.

*Example 4-120 `nim_move_up -S` command*

---

```
{nimmast}:/ # nim_move_up -S
```

```
nim_move_up Status:
```

```
=====
```

```
Tue Jun 6 10:24:24 2006
```

```
Next phase to execute: 1 - Setup NIM Resources
```

---

Next we enter the `nim_move_up` configuration information (see Example 4-121 on page 225) based on the following information:

- ▶ Our existing NIM client is named LPAR4. If you want to migrate more than one client at a time, you need to specify a NIM machine group. For example, `NIM_MAC_GRP_1`.
- ▶ The new POWER5 LPAR address will be 10.1.1.65. If you have specified to migrate a NIM machine group you must supply a sufficient range of IP addresses to cover the number of clients you intend to migrate, for example, `10.1.1.65-10.1.1.75`. The `/etc/hosts` file should be updated with this information also.

**Important:** Before starting a `nim_move_up` operation you need to allocate a new IP address for each target client involved in the operation.

- ▶ Our POWER5 HMC hostname is `hmc520`.
- ▶ Our managed system name is `YLI894-2-650E4DG`.
- ▶ The source install images are located in `/moveup/AIX5304`. It is recommended that you set the source to something like a product DVD with all the required

software. In this way, if your lpp\_source is missing any filesets that the client will attempt to migrate, it can be added to the lpp\_source in phase 2. In our test environment, /moveup/AIX5304 contained a copy of the AIX 5L V5.3 install CDs.

- ▶ The location of the new NIM resources will be located in the file system /moveup. This is a file system that we created which will be used to contain any new NIM resources that **nim\_move\_up** must create. Our file system was 6G in size but the size will depend on the number of and size of the client mksysb images and whether or not any new lppsource or SPOT resources need to be created.
- ▶ The Virtual I/O Server LPAR name will be VIO1 as we will be using VIO resources for the new POWER5 LPAR.

**Attention:** Ensure that your VIO server hostname and LPAR name match in terms of upper and lowercase. During our tests we encountered the following error:

```
Tue Jun  6 11:24:55 2006 OUTPUT
(/usr/lpp/bos.sysmgt/nim/methods/getTgtRsrc): ssh hscroot@hmcp520
viosvrcmd -m YLI894-2-650E4DG -p viol -c "\lsvg\"|
Tue Jun  6 11:24:56 2006 OUTPUT
(/usr/lpp/bos.sysmgt/nim/methods/getTgtRsrc): Invalid volume group
HSCL8012 The partition named viol was not found. Please check your entry
and retry the command.
```

Upon further investigation we found that the LPAR name for the VIO server was in uppercase, for example VIO1. However, in the nim\_move\_up configuration the name had been entered in lowercase. Once the correct case was used the command completed OK.

```
hscroot@hmcp520:~> viosvrcmd -m YLI894-2-650E4DG -p VIO1 -c "\lsvg\"
rootvg
rootvg_clients
```

- ▶ The LPP\_SOURCE name will be LPP\_53\_ML4 and the SPOT name will be SPOT\_53\_ML4.
- ▶ We have not specified the bosinst\_data resource, exclude\_files resource or the Temporary Volume Group for NIMADM information. This is OK as nim\_move\_up will create the resources as required and it will use the rootvg as the temporary volume group for the **nimadm** cache file systems.

**Important:** Ensure you have enough space in rootvg on the NIM master for the extraction of the client mksysb plus at least 500MB of free space for the migration tasks. For optimum performance, it is recommended that you use a different volume group that is separate from the volume group that contains the other NIM resources. To reduce the size of the mksysb image you can exclude files from the mksysb via an EXCLUDE\_FILES resource which can be specified in the nim\_move\_up configuration.

*Example 4-121 smit nim\_move\_up configuration parameters*

---

```
{nimmast}:/ # smit nim_move_up

Configure nim_move_up Input Values

* Existing NIM Client or Machine Group          [LPAR4]
* New LPAR IP Address                          [10.1.1.65]
* New LPAR Subnet Mask                        [255.255.255.0]
* New LPAR Default Gateway                   [10.1.1.1]
* Hostname of HMC                            [hmcp520]
* Managed System Name                        [YLI894-2-650E4DG]
* Source of Install Images                   [/moveup/AIX5304]
* Location of New NIM Resources              [/moveup]
Virtual I/O Server LPAR Name                 [VIO1]
Force the Use of Physical Network Adapter?   [no]
Force the Use of Physical Storage Controller? [no]
Accept All New License Agreements?          [yes]

LPP_SOURCE Name                             [LPP_53_ML4]
SPOT Name                                     [SPOT_53_ML4]
BOSINST_DATA                                [ ]
EXCLUDE_FILES resource                      [ ]
Customization SCRIPT resource               [ ]
INSTALLP BUNDLE containing packages to add  [ ]
FIX_BUNDLE to install                       [ ]
Temporary Volume Group for NIMADM          [ ]
Number of Loops to Run on Client            [ ]#
Seconds for Each Loop                       [ ]#
```

---

In Example 4-122 we check the current status of the nim\_move\_up operation which also shows us the configuration information we just entered.

*Example 4-122 Status of the nim\_move\_up operation*

---

```
{nimmast}:/ # nim_move_up -S

nim_move_up Status:
=====
```

Tue Jun 6 10:41:12 2006

Next phase to execute: 1 - Setup NIM Resources

Saved parameters:

```

accept_licenses = yes
images = /moveup/AIX5304
hmc = hmcp520
location = /moveup
gateway = 10.1.1.1
managed_sys = YLI894-2-650E4DG
subnet_mask = 255.255.255.0
spot = SPOT_53_ML4
lpp_source = LPP_53_ML4
vioserver = VI01
target_client = LPAR4
min_ip = 10.1.1.65

```

---

We are now ready to execute phase 1 of the `nim_move_up` process.

Before moving onto the first phase, it is worth mentioning a few details regarding the `nim_move_up` configuration database and log files. All `nim_move_up` operations are logged to the directory `/var/mig2p5`. This directory serves several purposes as shown in Table 4-1. It records state information so that the `nim_move_up` operation can remember what phases have been executed. The directory is created by the `nim_move_up` command and its contents are removed once `nim_move_up` is unconfigured. It also contains several log files which can be very helpful in troubleshooting.

With the exception of the log files and the contents of the `client_data` directory, the files in `/var/mig2p5` can be read and modified so that `nim_move_up` can perform tasks that it would not do through its command line and SMIT interfaces. Users are encouraged to manipulate the `mig2p5` environment to meet any specific needs and to aid in the troubleshooting of any problems that might arise during the `nim_move_up` process. Modifying the `mig2p5` environment is an advanced task and users must ensure they fully understand the changes being performed and their effect on the `nim_move_up` tool.

Table 4-1 .Description and purpose of the files/directories in `/var/mig2p5`

File/directory	Purpose
config_db - file	Contains all of the saved configuration options passed to <code>nim_move_up</code> .
current_phase - file	Contains the number of the phase that will execute next.

File/directory	Purpose
global_log - file	Contains the output of all the phases that have been run since the last time the mig2p5 directory was initialized. This log is very helpful in troubleshooting a nim_move_up failure.
client_data/ - directory	Contains files that are generated by nim_move_up during phases 3 and 4, in which each of the original clients' system resources and utilization are monitored and quantified into configuration files. The available resources in the POWER5 server are also quantified into corresponding text files. All the data in these files will be taken into account when determining the hardware profile of the newly derived LPARs on the POWER5 server.  NOTE: These files are intended to be machine-readable data files for the nim_move_up command's internal use. <b>Do not manually modify or create them yourself.</b>
phase#/ - directory phase1-9	Contain data specific to the corresponding phase denoted by the number in its name e.g. phase4/. Every phase has a directory, for example, phase1, phase2, and so on.
phase#/log - file	A log file that contains all the output displayed during a phase's execution. If a phase is executed multiple times (such as after an error has been corrected), all new output is appended to the log file. This log file is helpful in investigating failures related to this phase after they have occurred.
phase#/status - file	Indicates whether a phase has been successful or not. This file is used by nim_move_up to determine whether a subsequent phase can be run. A phase can only run if all previous phases' status files contain the string <i>success</i> . If a phase encounters an error that causes it to fail the status file will contain the string <i>failure</i> .
pid - file	Contains the nim_move_up process ID number when nim_move_up is running in the background. This file is cleaned up when the process finishes. As long as this file exists and contains a process ID, nim_move_up cannot run phases because concurrent runs of nim_move_up are not supported.

The following Example 4-123 shows the directory listing of /var/mig2p5 from our test system:

*Example 4-123 Migration directory sample*

---

```
{nimmast}:/var/mig2p5 $ ls -ltr
```

```
total 8
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase9
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase8
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase7
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase6
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase5
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase4
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase3
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase2
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase10
drwxr-xr-x  2 root    system    256 Jun 06 16:17 phase1
-rw-r--r--  1 root    system      2 Jun 06 16:17 current_phase
drwxr-xr-x  2 root    system    256 Jun 06 16:17 client_data
-rw-r--r--  1 root    system    236 Jun 06 16:20 config_db
```

---

At this stage, the *config\_db* file contains our configuration information (Example 4-124):

*Example 4-124 Sample config\_db file*

---

```
{nimmast}:/var/mig2p5 # cat config_db
min_ip:10.1.1.65
subnet_mask:255.255.255.0
gateway:10.1.1.1
hmc:hmcp520
vioserver:VI01
managed_sys:YLI894-2-650E4DG
target_client:LPAR4
location:/moveup
images:/moveup/AIX5304
accept_licenses:yes
lpp_source:LPP_53_ML4
spot:SPOT_53_ML4
```

---

5. Configure the SSH keys on the target HMC.

SSH keys needed to be configured on our HMC to allow for unattended remote execution of the commands from the NIM master. We executed the commands shown in Example 4-125 to configure the SSH keys for the root user on the NIM master and append the public key to the *authorized\_keys2* file for the *hscroot* user on the HMC. This allows the root user on the NIM master to run commands remotely on the HMC without entering a password. After configuring our SSH keys, we checked that we could run a command on the HMC without being prompted for a password.

*Example 4-125 Configuring the SSH keys*


---

```
{nimmast}:/ # nim_move_up -K
Connecting to host hmcp520...
hscroot@hmcp520's password:
id_rsa.pub
100% 222    0.2KB/s   00:00
{nimmast}:/ # ssh hscroot@hmcp520 date
Tue May 30 15:56:59 CEST 2006
```

---

**6. Execute Phase 1 - Setting up NIM resources.**

In Example 4-126, we executed the `nim_move_up` command with the `-n` flag, which instructs `nim_move_up` to execute the next phase in the operation.

*Example 4-126 Executing the nim\_move\_up -n command*


---

```
{nimmast}:/ # nim_move_up -n
Tue Jun  6 10:43:40 2006 Starting Phase 1 : Setup NIM Resources...
Tue Jun  6 10:43:40 2006 Creating bosinst_data resource.
Tue Jun  6 10:43:40 2006 Executing: /usr/sbin/nim -o define -t bosinst_data -a
server=master -a location=/moveup/bosinst.data nim_move_up_bid
Tue Jun  6 10:43:41 2006 rc for /usr/sbin/nim = 0
Tue Jun  6 10:43:41 2006 Creating exclude_files resource.
Tue Jun  6 10:43:41 2006 Executing: /usr/sbin/nim -o define -t exclude_files -a
server=master -a location=/moveup/exclude_files nim_move_up_exclude
Tue Jun  6 10:43:41 2006 rc for /usr/sbin/nim = 0
Tue Jun  6 10:43:41 2006 STATUS for phase 1: success
```

---

The `bosinst_data` and `exclude_files` NIM resources were created in this phase. Example 4-127 shows details of both NIM resources, the contents of the files used to create the NIM resources and the directory structure. These resources are not removed after a `nim_move_up` operation.

*Example 4-127 Details of the NIM resources*


---

```
{nimmast}:/ # lsnim -l nim_move_up_bid nim_move_up_exclude
nim_move_up_bid:
  class      = resources
  type       = bosinst_data
  Rstate     = ready for use
  prev_state = unavailable for use
  location   = /moveup/bosinst.data
  alloc_count = 0
  server     = master
nim_move_up_exclude:
  class      = resources
  type       = exclude_files
  Rstate     = ready for use
```

```

prev_state = unavailable for use
location   = /moveup/exclude_files
alloc_count = 0
server     = master

{nimmast}:/ # cd /moveup
{nimmast}:/moveup # ls -ltr
total 8
drwxr-xr-x  2 root    system      256 Jun 06 10:33 lost+found
-rw-r--r--  1 root    system      914 Jun 06 10:43 bosinst.data
-rw-r--r--  1 root    system         0 Jun 06 10:43 exclude_files

{nimmast}:/moveup # cat bosinst.data
  CONSOLE = Default
  INSTALL_METHOD = overwrite
  PROMPT = no
  EXISTING_SYSTEM_OVERWRITE = yes
  INSTALL_X_IF_ADAPTER = yes
  RUN_STARTUP = no
  RM_INST_ROOTS = no
  ERROR_EXIT =
  CUSTOMIZATION_FILE =
  TCB = no
  INSTALL_TYPE =
  BUNDLES =
  RECOVER_DEVICES = Default
  BOSINST_DEBUG = no
  ACCEPT_LICENSES = yes
  DESKTOP = CDE
  INSTALL_DEVICES_AND_UPDATES = yes
  IMPORT_USER_VGS =
  ENABLE_64BIT_KERNEL = Default
  CREATE_JFS2_FS = Default
  ALL_DEVICES_KERNELS = yes
  GRAPHICS_BUNDLE = yes
  MOZILLA_BUNDLE = no
  KERBEROS_5_BUNDLE = no
  SERVER_BUNDLE = no
  REMOVE_JAVA_118 = no
  HARDWARE_DUMP = yes
  ADD_CDE = no
  ADD_GNOME = no
  ADD_KDE = no
  ERASE_ITERATIONS = 0
  ERASE_PATTERNS =

target_disk_data:
  LOCATION =
  SIZE_MB =

```



```
HDISKNAME =
```

```
locale:
```

```
BOSINST_LANG =
CULTURAL_CONVENTION =
MESSAGES =
KEYBOARD =
```

---

**Phase 2** will run next. We check the status before we execute the next step and we are informed that some pre-migration tasks will be performed as shown in Example 4-128.

*Example 4-128 Pre-migration tasks*

---

```
{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun  6 10:44:29 2006
```

Next phase to execute: 2 - Pre-Migration

```
Saved parameters:
accept_licenses = yes
images = /moveup/AIX5304
hmc = hmcp520
location = /moveup
gateway = 10.1.1.1
managed_sys = YLI894-2-650E4DG
subnet_mask = 255.255.255.0
spot = SPOT_53_ML4
lpp_source = LPP_53_ML4
vioserver = VI01
target_client = LPAR4
min_ip = 10.1.1.65
```

---

## 7. Run **Phase 2** - Pre-Migration.

The `pre_migration` script is run on the source client. Commands such as **lppchk** are run to assess what software is installed and which software cannot be migrated. The `pre_migration` script saves its output to `/home` and should be reviewed prior to migrating. Any software that is missing from the `lpp_source` is then copied from the source installation images (which in our case is `/moveup/AIX5304`) to the `lpp_source` directory (which in our case is `/AIX53ML4/LPP_53_ML4`). See Example 4-129.

*Example 4-129 Pre-migration checks*

---

```
{nimmast}:/ # nim_move_up -n
Tue Jun  6 10:44:53 2006 Starting Phase 2 : Pre-Migration...
```

```
Tue Jun 6 10:44:53 2006 Running command /usr/lpp/bos.sysmgmt/nim/methods/c_rsh
lpar4 /usr/lpp/bos/pre_migration
Tue Jun 6 10:44:53 2006 OUTPUT:
All saved information can be found in: /home/pre_migration.060606104453
```

```
Checking size of boot logical volume (hd5).
```

```
Listing software that will be removed from the system.
```

```
Listing configuration files that will not be merged.
```

```
Listing configuration files that will be merged.
```

```
Saving configuration files that will be merged.
```

```
Running lppchk commands. This may take awhile.
```

```
Please check /home/pre_migration.060606104834/software_consistency for possible
errors.
```

```
Please check /home/pre_migration.060606104834/tcbck.output for possible errors.
```

```
All saved information can be found in: /home/pre_migration.060606104834
```

```
It is recommended that you create a bootable system backup of your system
before migrating.
```

```
Tue Jun 6 10:48:34 2006 Obtaining list of installed software on client LPAR4
Tue Jun 6 10:48:34 2006 Attempting to update lpp_source LPP_53_ML4 with
software installed on clients.
Tue Jun 6 10:48:34 2006 Executing: /usr/sbin/gencopy -bqv -X -d /moveup/AIX5304
-t /AIX53ML4/LPP_53_ML4 -f /tmp/gcpyargs.553090
/AIX53ML4/LPP_53_ML4/installp/ppc/printers.rte.5.3.0.30.I
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.util.5.3.0.10.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.rte.5.3.0.10.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.com.5.3.0.10.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.util.5.3.0.10.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.rte.5.3.0.10.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.com.5.3.0.10.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.util.5.3.0.40.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.rte.5.3.0.40.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.com.5.3.0.40.U
/AIX53ML4/LPP_53_ML4/installp/ppc/bos.diag.5.3.0.30.I
/AIX53ML4/LPP_53_ML4/installp/ppc/X11.apps.config.5.3.0.40.U
/AIX53ML4/LPP_53_ML4/installp/ppc/X11.apps.5.3.0.30.I
Tue Jun 6 10:54:52 2006 STATUS for phase 2: success
```

---

Example 4-130 shows the collection information on the clients hardware and system usage.

*Example 4-130 Clients' hardware and system usage information collection*


---

```
{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun  6 10:45:13 2006
```

Next phase to execute: 3 - Client Hardware and Utilization Data Gathering

```
Saved parameters:
  accept_licenses = yes
  images = /moveup/AIX5304
  hmc = hmcp520
  location = /moveup
  gateway = 10.1.1.1
  managed_sys = YLI894-2-650E4DG
  subnet_mask = 255.255.255.0
  spot = SPOT_53_ML4
  lpp_source = LPP_53_ML4
  vioserver = VI01
  target_client = LPAR4
  min_ip = 10.1.1.65
```

---

#### 8. Phase 3 - Client hardware and utilization and data gathering.

This phase collects information about each client's current hardware resources and also attempts to assess the average use of those resources (in terms of CPU and memory) over a given amount of time.

To collect usage statistics, the **vmstat** command is run on the client for a period of time based on the values we entered for "Number of Loops to Run on Client" and "Seconds for Each Loop" during the configuration stage. Changing these values will allow you to gather more data over a greater space of time which may provide a better picture of the overall utilization of the system and can result in a better final configuration for the target LPAR on the POWER5 system.

In Example 4-131 on page 233, the number of loops and seconds for each loop in the initial `nim_move_up` configuration were changed to 30 and 3 respectively. This means that the **vmstat** command will run on the client 30 times every 3 seconds.

*Example 4-131 Number of loops parameter change*


---

Number of Loops to Run on Client	[30]#
Seconds for Each Loop	[3]#

---

Example 4-132 shows the output the from phase 3.

*Example 4-132 Output of phase 3*

```

{nimmast}:/ # nim_move_up -n
Tue Jun  6 10:45:33 2006 Starting Phase 3 : Client Hardware and Utilization
Data Gathering...
Tue Jun  6 10:45:33 2006 Running getSrcUtil on client LPAR4
Tue Jun  6 10:45:33 2006 Executing: /usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil
-s lpar4 -o /var/mig2p5/client_data/srcUtil_LPAR4
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil):
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil):
The number of vmstat loops :10
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil):
The Average of Active virtual pages: 127 MB
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil):
The Average of Size of the free list: 582 MB
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil):
The sum of the CPU %idle: 990
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil):
The Average of CPU idle: 99.00
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil):
The Average of CPU utilization: 1.00
Tue Jun  6 10:45:42 2006 rc for /usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil = 0
Tue Jun  6 10:45:42 2006 Running getSrcCfg on client LPAR4
Tue Jun  6 10:45:42 2006 Executing: /usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg
-s lpar4 -o /var/mig2p5/client_data/srcCfg_LPAR4 -util
/var/mig2p5/client_data/srcUtil_LPAR4
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
INPUT:
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
Source Hostname           = lpar4
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
File name for source configuration= /var/mig2p5/client_data/srcCfg_LPAR4
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
File containing utilization data = /var/mig2p5/client_data/srcUtil_LPAR4
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
-----
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
Getting the type & model of the given source machine.....
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
-----
Tue Jun  6 10:45:42 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
Getting the info about CPUs and Memory in the source machine....
Tue Jun  6 10:45:44 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
-----
Tue Jun  6 10:45:44 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
Adding utilization info from the /var/mig2p5/client_data/srcUtil_LPAR4
File.....
Tue Jun  6 10:45:44 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
-----

```

```
Tue Jun  6 10:45:44 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
Getting the info about volume groups in the source machine.....
Tue Jun  6 10:45:45 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
-----
Tue Jun  6 10:45:45 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg):
Getting the info about network adapters in the source machine....
Tue Jun  6 10:45:45 2006 rc for /usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg = 0
Tue Jun  6 10:45:45 2006 STATUS for phase 3: success
```

---

The scripts in Example 4-133 are used to collect hardware information and system usage data on the source client system.

*Example 4-133 Scripts used to collect hardware and system usage information*

---

```
/usr/lpp/bos.sysmgmt/nim/methods/getSrcCfg
/usr/lpp/bos.sysmgmt/nim/methods/getSrcUtil
```

---

Upon successful completion of this phase, we found the following files had been created in the /var/mig2p5/client\_data directory as shown in Example 4-134.

*Example 4-134 New files created in the /var/mig2p5/client\_data*

---

```
{nimmast}:/var/mig2p5/client_data # ls -l
total 16
-rw-r--r--  1 root    system      62 Jun 06 10:45 srcUtil_LPAR4
-rw-r--r--  1 root    system     253 Jun 06 10:45 srcCfg_LPAR4

{nimmast}:/var/mig2p5/client_data # cat srcUtil_LPAR4
CPU   UTILIZATION in %   = 1
MEMORY UTILIZATION in MBs = 442

{nimmast}:/var/mig2p5/client_data # cat srcCfg_LPAR4
MACHINE = 7040-671
CPUS = PowerPC_POWER4, 1000 MHz, 2
MEMORY = 1024 MB
CPU   UTILIZATION in %   = 1
MEMORY UTILIZATION in MBs = 442
# VG = <vg name>,<size in MB>
VG = rootvg,17344
# NETWORK = <description>
NETWORK = 10/100 Mbps Ethernet PCI Adapter II
```

---

The files in Example 4-134 contain the CPU and memory utilization statistics for the LPAR and the current hardware configuration of the system. The `nim_move_up` operation will use these files to determine what the system configuration will need to look like on the new POWER5 LPAR.

Now, we move into the POWER5 resource availability and LPAR creation stage as shown in Example 4-135.

*Example 4-135 LPAR creation stage*

---

```
{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun  6 10:46:08 2006
```

Next phase to execute: 4 - p5 Resource Availability Data Gathering and LPAR Creation

```
Saved parameters:
  accept_licenses = yes
  images = /moveup/AIX5304
  hmc = hmcp520
  location = /moveup
  gateway = 10.1.1.1
  managed_sys = YLI894-2-650E4DG
  subnet_mask = 255.255.255.0
  spot = SPOT_53_ML4
  lpp_source = LPP_53_ML4
  vioserver = VIO1
  target_client = LPAR4
  min_ip = 10.1.1.65
```

---

## 9. Phase 4 - POWER5 resource availability data gathering and LPAR creation.

In this phase, `nim_move_up` will examine the POWER5 system for available hardware resources. The data gathered in the previous phase is used to derive an equivalent LPAR configuration on the POWER5 system. We specified a VIO server so our target LPAR will be configured using VIO resources instead of physical I/O resources. The creation of the necessary VIO adapters, storage devices and other configuration on the VIO server is managed by the `nim_move_up` operation.

Lets execute this phase now. Note there is significant amount of output, as you can see in Example 4-136, but it is very useful for helping us to understand what is accomplished in this phase.

*Example 4-136 Output of phase 4*

---

```
{nimmast}:/ # nim_move_up -n
Tue Jun  6 11:30:54 2006 Starting Phase 4 : p5 Resource Availability Data
Gathering and LPAR Creation...
Tue Jun  6 11:30:54 2006 Obtaining available resources on p5 server
YLI894-2-650E4DG
```

```

Tue Jun  6 11:30:54 2006 Executing: /usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc
-hmc hmcp520 -u hscroot -m "YLI894-2-650E4DG" -o
"/var/mig2p5/client_data/tgtRsrc_YLI894-2-650E4DG" -viosDisk "VIO1" -viosEth
"VIO1"
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
INPUT:
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
HMC Host = hmcp520
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
HMC Username = hscroot
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Managed System name = YLI894-2-650E4DG
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Ethernet VIO Server(s) = VIO1
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Disk VIO Server(s) = VIO1
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Output file =
/var/mig2p5/client_data/tgtRsrc_YLI894-2-650E4DG
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
-----
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Getting the type & model of the given P5 machine.....
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
cmd = ssh hscroot@hmcp520 lssyscfg -m YLI894-2-650E4DG -r sys -Ftype_model
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Type-Model = 9131-52A
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
-----
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Getting the number of processing units in the given P5 machine.....
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
cmd = ssh hscroot@hmcp520 lshwres -m YLI894-2-650E4DG -r proc --level sys
-Fconfigurable_sys_proc_units
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
available_proc_units = 4.0
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
-----
Tue Jun  6 11:30:54 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Getting the amount of memory in MBs in the given P5 machine.....
Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Configurable Memory = 8192
Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Available memory = 1248
Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
LMB Size = 32
Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
-----

```

```

Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Getting the available physical I/O slots.....
Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
-----
Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Getting the volume groups info in the VIO Disk server(s).....
Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Command:
Tue Jun  6 11:30:55 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
ssh hscroot@hmcp520 viosvr cmd -m YLI894-2-650E4DG -p VI01 -c "\lsvg\"|
Tue Jun  6 11:30:56 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Command:
Tue Jun  6 11:30:56 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
ssh hscroot@hmcp520 viosvr cmd -m YLI894-2-650E4DG -p VI01 -c "\lsvg rootvg\"|
Tue Jun  6 11:30:57 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Command:
Tue Jun  6 11:30:57 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
ssh hscroot@hmcp520 viosvr cmd -m YLI894-2-650E4DG -p VI01 -c "\lsvg
rootvg_clients\"|
Tue Jun  6 11:30:58 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
VIO Server = VI01; VG = rootvg; Size = 53120MB
Tue Jun  6 11:30:58 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
VIO Server = VI01; VG = rootvg_clients; Size = 98816MB
Tue Jun  6 11:30:58 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
-----
Tue Jun  6 11:30:58 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Getting the available VLANs info in the VIO server(s).....
Tue Jun  6 11:30:58 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
Command:
Tue Jun  6 11:30:58 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
ssh hscroot@hmcp520 lshwres -m YLI894-2-650E4DG -r virtualio --subtype eth
--level lpar --filter "lpar_names=VI01"
-Flpar_name:slot_num:port_vlan_id:is_trunk
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc):
VIO Server = VI01; VLAN Id = 1; Slot number = 30
Tue Jun  6 11:30:59 2006 rc for /usr/lpp/bos.sysmgmt/nim/methods/getTgtRsrc = 0
Tue Jun  6 11:30:59 2006 Creating p5 configuration for client LPAR4

Tue Jun  6 11:30:59 2006 Executing: /usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg
-s /var/mig2p5/client_data/srcCfg_LPAR4 -m
"/var/mig2p5/client_data/tgtRsrc_YLI894-2-650E4DG" -e
/usr/lpp/bos.sysmgmt/nim/methods/capEqMaps -lpar "LPAR4_lpar" -prof
"LPAR4_profile" -o /var/mig2p5/client_data/tgtCfg_LPAR4
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
INPUT:
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Source Configuration file : /var/mig2p5/client_data/srcCfg_LPAR4

```



```

Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Target Machine information file      :
/var/mig2p5/client_data/tgtRsrc_YLI894-2-650E4DG
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Capacity equivalence maps file      :
/usr/lpp/bos.sysmgmt/nim/methods/capEqMaps
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Target LPAR name                     : LPAR4_lpar
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Name of the profile to be created in LPAR: LPAR4_profile
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Target LPAR file                     : /var/mig2p5/client_data/tgtCfg_LPAR4
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Use dedicated CPUs?                 : No
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Use dedicated Ethernet adapters?    : No
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Use dedicated Hard disks?          : No
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
-----
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Reading the Source Configure from the given file.....
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
10002
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
src_mem_util = 43
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Machine Type-model of the source machine = 7040-671
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Number of CPUs in the source machine   = 2
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Amount of memory in the source machine = 1024
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Network slots in source machine       = 10/100 Mbps Ethernet PCI Adapter II
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
-----
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Reading the target P5 managed system configuration.....
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Available processing units = 4.0
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):
Available memory             = 1248
Tue Jun  6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/genTgtCfg):

```

```

Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg): IO
slots = 21010002/none/Fibre Channel Serial Bus:21030003/none/PCI 1Gbps
Ethernet Fiber 2-port:21040003/none/Storage controller:21050003/none/PCI 1Gbps
Ethernet Fiber 2-port:21010004/none/Universal Serial Bus UHC
Spec:21030004/none/Fibre Channel Serial Bus
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Storage IO slots = 21040003/none/Storage controller
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Network IO slots = 21030003/none/PCI 1Gbps Ethernet Fiber
2-port:21050003/none/PCI 1Gbps Ethernet Fiber 2-port
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
VIO Volume groups = rootvg,VI01,53120
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
rootvg_clients,VI01,98816
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
VIO Hard disks =
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
VIO VLANs = 1,VI01,30
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
src_freq = 1000, tgt_freq = 1500
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
-----
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Calculating the resource requirements of target LPAR.....
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
des_procs = 0.1
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
des_memory = 448
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
New Physical io slots =
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
-----
Tue Jun 6 11:30:59 2006 OUTPUT (/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg):
Creating the target LPAR configuration file.....
Tue Jun 6 11:30:59 2006 rc for /usr/lpp/bos.sysmgt/nim/methods/genTgtCfg = 0
Tue Jun 6 11:30:59 2006 Executing:
/usr/lpp/bos.sysmgt/nim/methods/createTgtLPAR -m
"/var/mig2p5/client_data/tgtRsrc_YLI894-2-650E4DG" -t
/var/mig2p5/client_data/tgtCfg_LPAR4
Tue Jun 6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgt/nim/methods/createTgtLPAR): INPUT:

```

```

Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Target Machine configuration
file = /var/mig2p5/client_data/tgtRsrc_YLI894-2-650E4DG
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Target LPAR configuration file
= /var/mig2p5/client_data/tgtCfg_LPAR4
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Managed system =
YLI894-2-650E4DG
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): HMC Hostname = hmcp520
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): User configured for remote ssh
to HMC = hscroot
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): LPAR name: LPAR4_lpar
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Profile name: LPAR4_profile
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Proc units: Min=0.1,
Desired=0.1, Max=0.2
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): CPUs: Min=1, Desired=1, Max=1
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Memory: Min=256, Desired=448,
Max=896
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Virtual IO VGs:
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): rootvg,17344,rootvg,VI01
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Virtual IO VLANs:
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): 1,VI01,30
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
-----
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Creating the target LPAR
profile with proc units and memory.....
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:30:59 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): ssh hscroot@hmcp520 mkssyscfg
-r lpar -m YLI894-2-650E4DG -i

```

```
name=LPAR4_lpar,profile_name=LPAR4_profile,lpar_env=aixlinux,proc_mode=shared,sh
aring_mode=cap,min_proc_units=0.1,desired_proc_units=0.1,max_proc_units=0.2,mi
n_procs=1,desired_procs=1,max_procs=1,min_mem=256,desired_mem=448,max_mem=896,b
oot_mode=norm
```

```
Tue Jun 6 11:31:07 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
```

```
-----
Tue Jun 6 11:31:07 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Discovering the LPAR id
assigned to the above LPAR.....
```

```
Tue Jun 6 11:31:07 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
```

```
Tue Jun 6 11:31:07 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): ssh hscroot@hmcp520 lssyscfg
-r lpar -m YLI894-2-650E4DG --filter lpar_names=LPAR4_lpar -Flpar_id
```

```
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): LPAR ID = 1
```

```
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
```

```
-----
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Adding Physical I/O slots to
the LPAR profile.....
```

```
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): .....No physical I/O slots to
add to the LPAR profile.
```

```
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
```

```
-----
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Adding Virtual I/O to the LPAR
profile.....
```

```
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
```

```
-----
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Getting the list of existing
vscsi devices in the VIO server.....
```

```
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
```

```
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): ssh hscroot@hmcp520 viosvr cmd
-m YLI894-2-650E4DG -p VIO1 -c "\lsdev -virtual\""
```

```
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
```

```
-----
Tue Jun 6 11:31:08 2006 OUTPUT
```

```
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Determining the free virtual
slot number in the VIO server.....
```

```

Tue Jun  6 11:31:08 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:08 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):  ssh hscroot@hmcp520 lshwres
-r virtualio --rsubtype slot -m YLI894-2-650E4DG --level slot --filter
"lpar_names=VI01" -Fslot_num
Tue Jun  6 11:31:13 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
-----
Tue Jun  6 11:31:13 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Creating virtual scsi adapter
in the VIO server.....
Tue Jun  6 11:31:13 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Adding the vscsi adapter to
the VIO server using DR operation.
Tue Jun  6 11:31:13 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command = ssh hscroot@hmcp520
chhwres -r virtualio --rsubtype scsi -m YLI894-2-650E4DG -o a -p VI01 -s 31 -a
"adapter_type=server,remote_slot_num=2,remote_lpar_name=LPAR4_lpar"
Tue Jun  6 11:31:15 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
-----
Tue Jun  6 11:31:15 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Getting the default and
current profile names of VIO server...
Tue Jun  6 11:31:15 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:15 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):  ssh hscroot@hmcp520 lssyscfg
-r lpar -m YLI894-2-650E4DG --filter "lpar_names=VI01"
-Fdefault_profile:curr_profile
Tue Jun  6 11:31:15 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
-----
Tue Jun  6 11:31:15 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Adding the vscsi adapter to
the default profile of VIO server....
Tue Jun  6 11:31:15 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:15 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):  ssh hscroot@hmcp520 chsyscfg
-r prof -m YLI894-2-650E4DG -i
"name=normal,lpar_name=VI01,virtual_scsi_adapters+=31/server/1/LPAR4_lpar/2/1"
Tue Jun  6 11:31:21 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
-----
Tue Jun  6 11:31:21 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Configuring the virtual scsi
adapter in the VIO server.....

```

```

Tue Jun  6 11:31:21 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:21 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): ssh hscroot@hmcp520 viosvr cmd
-m YLI894-2-650E4DG -p VI01 -c "\"cfgdev\""
Tue Jun  6 11:31:22 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
:-----
Tue Jun  6 11:31:22 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Determining the OS-generated
vscsi device name for the new adapter....
Tue Jun  6 11:31:22 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:22 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): ssh hscroot@hmcp520 viosvr cmd
-m YLI894-2-650E4DG -p VI01 -c "\"lsdev -virtual\""
Tue Jun  6 11:31:23 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
:-----
Tue Jun  6 11:31:23 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Creating a logical volume in
the VIO server.....
Tue Jun  6 11:31:23 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:23 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): ssh hscroot@hmcp520 viosvr cmd
-m YLI894-2-650E4DG -p VI01 -c "\"mklv -lv rootvg_1 rootvg 17G\""
Tue Jun  6 11:31:24 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): rootvg_1
Tue Jun  6 11:31:24 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
:-----
Tue Jun  6 11:31:24 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Attaching the above LV to the
vscsi device created earlier.....
Tue Jun  6 11:31:24 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:24 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): ssh hscroot@hmcp520 viosvr cmd
-m YLI894-2-650E4DG -p VI01 -c "\"mkvdev -vdev rootvg_1 -vadapter vhost4 -dev
vtscsi4\""
Tue Jun  6 11:31:26 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): vtscsi4 Available
Tue Jun  6 11:31:26 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
:-----
Tue Jun  6 11:31:26 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Creating the virtual scsi
adapter for client LPAR.....

```

```

Tue Jun  6 11:31:26 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:26 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):  ssh hscroot@hmcp520 chsyscfg
-r prof -m YLI894-2-650E4DG -i
name=LPAR4_profile,lpar_name=LPAR4_lpar,virtual_scsi_adapters+=2/client//VI01/3
1/1
Tue Jun  6 11:31:29 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):
-----
Tue Jun  6 11:31:29 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Creating the virtual ethernet
adapters for client LPAR.....
Tue Jun  6 11:31:29 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR): Command:
Tue Jun  6 11:31:29 2006 OUTPUT
(/usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR):  ssh hscroot@hmcp520 chsyscfg
-r prof -m YLI894-2-650E4DG -i
name=LPAR4_profile,lpar_name=LPAR4_lpar,virtual_eth_adapters+=3/0/1//0/1
Tue Jun  6 11:31:32 2006 rc for /usr/lpp/bos.sysmgmt/nim/methods/createTgtLPAR =
0
Tue Jun  6 11:31:32 2006 STATUS for phase 4: success

```

---

In summary, **Phase 4** performed the following:

- ▶ Obtained information regarding the POWER5 target system, i.e. type and model, available processing units, available memory, available physical I/O slots, VIO volume group information, and VIO VLAN information.
- ▶ Creation of the new LPAR on the POWER5 machine such as LPAR name, name of the LPAR profile, whether or not dedicated or shared CPU was needed, whether or not physical adapters were required (i.e. Ethernet, Fibre Channel, SCSI or VSCSI), LPAR profile configuration (min/desired/max for CPU and memory), virtual I/O resources on the VIO server for virtual SCSI disk, virtual Ethernet, virtual SCSI adapters and so on. Example 4-137 shows the virtual SCSI adapter and disk resources that were created on the VIO server for the client.

**Note:** If the overall creation process of the new LPAR fails but the VIO resources were created successfully, or if you wish to restart the `nim_move_up` process from the beginning, you may need to remove the virtual SCSI adapter and disk resources on the VIO server. Before you can try phase 4 again you should remove the vhost adapter (for example, `vhost4`) and then the client logical volume ( `rootvg_1`). The `nim_move_up` program is currently unable to undo changes made to the POWER5 server (LPAR creation) and a VIO server. These changes must be done manually even if the `nim_move_up -r` command has been run to unconfigure the operation.

*Example 4-137 Virtual SCSI and disk resources created for the client*

```

$ lsmap -vadapter vhost4
SVSA      Physloc                                Client Partition ID
-----
vhost4    U9131.52A.650E4DG-V2-C31    0x00000001

VTD                vtscsi4
LUN                0x8100000000000000
Backing device    rootvg_1
Physloc

$ lsvg -lv rootvg
rootvg:
LV NAME      TYPE      LPs  PPs  PVs  LV STATE    MOUNT POINT
hd5          boot      1    1    1    closed/syncd  N/A
hd6          paging    4    4    1    open/syncd    N/A
paging00     paging    8    8    1    open/syncd    N/A
hd8          jfs2log   1    1    1    open/syncd    N/A
hd4          jfs2      2    2    1    open/syncd    /
hd2          jfs2      10   10   1    open/syncd    /usr
hd9var       jfs2      5    5    1    open/syncd    /var
hd3          jfs2      11   11   1    open/syncd    /tmp
hd1          jfs2      80   80   1    open/syncd    /home
hd10opt      jfs2      1    1    1    open/syncd    /opt
lg_dump1v    sysdump   8    8    1    open/syncd    N/A
rootvg_1     jfs       136  136  1    open/syncd    N/A

```

On completion of **Phase 4**, a new LPAR is created on the p520. The LPAR is called LPAR4\_lpar.

The following file was used by nim\_move\_up to automatically create the new POWER5 LPAR as shown in Example 4-138.

*Example 4-138 File used to create the POWER 5 LPAR*

```

{nimmast}:/var/mig2p5/client_data # cat tgtCfg_LPAR4
LPAR_NAME = LPAR4_lpar
PROFILE_NAME = LPAR4_profile
# PROC_UNITS = <min> <desired> <max>
PROC_UNITS = 0.1 0.1 0.2
# MEMORY = <min MB> <desired MB> <max MB>
MEMORY = 256 448 896
# VIO_VG_INFO = <vgname_src>,<size in MB>,<vgname_vio>,<lpar_name>
#   Where <vgname_src> is the VG name in source machine, and
#       <vgname_vio> is the VG name in VIO server LPAR
VIO_VG_INFO = rootvg,17344,rootvg,VI01
# VIO_VLAN_INFO = <vlan id>,<lpar name>,<slot number>

```



```
VIO_VLAN_INFO = 1,VI01,30
```

---

The file in Example 4-139 is used to determine what resources are available on the POWER5 system.

*Example 4-139 Determining the resources available on the POWER5 system*

---

```
{nimmast}:/var/mig2p5/client_data # cat tgtRsrc_YLI894-2-650E4DG
MACHINE = 9131-52A
MANAGED SYSTEM = YLI894-2-650E4DG
HMC = hmcp520
HMC_SSH_USER = hscroot
AVAILABLE PROC UNITS = 4.0
CONFIGURABLE MEMORY = 8192MB
AVAILABLE MEMORY = 1248MB
LMB SIZE = 32MB
# IOSLOT = <drc_index>/<slot_pool_id>/<description>
IOSLOT = 21010002/none/Fibre Channel Serial Bus
IOSLOT = 21030003/none/PCI 1Gbps Ethernet Fiber 2-port
IOSLOT = 21040003/none/Storage controller
IOSLOT = 21050003/none/PCI 1Gbps Ethernet Fiber 2-port
IOSLOT = 21010004/none/Universal Serial Bus UHC Spec
IOSLOT = 21030004/none/Fibre Channel Serial Bus
# VIO VG = <vg name>,<lpar name>,<size in MB>
VIO VG = rootvg,VI01,53120
VIO VG = rootvg_clients,VI01,98816
# VIO VLAN = <vlan id>,<lpar name>,<slot number>
VIO VLAN = 1,VI01,30
```

---

## 10.. Phase 5 - Create system backups.

The next step, shown in Example 4-140, creates a mksysb backup of the source client.

**Note:** Ensure that the file size limit (ulimit) for the root user on the client and the master is set to unlimited for mksysb images larger than 2GB , for example:

```
# chuser fsize=-1 root
# lsuser -a fsize root
root fsize=-1
```

*Example 4-140 Creation of the mksysb backup*

---

```
{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun 6 11:32:04 2006
```

Next phase to execute: 5 - Create System Backups

Saved parameters:

```

accept_licenses = yes
images = /moveup/AIX5304
hmc = hmcp520
location = /moveup
gateway = 10.1.1.1
managed_sys = YLI894-2-650E4DG
subnet_mask = 255.255.255.0
spot = SPOT_53_ML4
lpp_source = LPP_53_ML4
vioserver = VI01
target_client = LPAR4
min_ip = 10.1.1.65

```

---

**Phase 5** execution has begun. Once the mksysb for the client is finished, a new mksysb resource definition is created (which in our case was named LPAR4\_phys\_mksysb). See Example 4-141.

*Example 4-141 New mksysb resource definition*

---

```

{nimmast}:/ # nim_move_up -n
Tue Jun 6 11:39:20 2006 Starting Phase 5 : Create System Backups...
Tue Jun 6 11:39:20 2006 Creating mksysb for client LPAR4
Tue Jun 6 11:39:20 2006 Executing: /usr/sbin/nim -o define -t mksysb -a
server=master -a source=LPAR4 -a location=/moveup/mksysb/LPAR4/LPAR4_mksysb -a
mk_image=yes LPAR4_phys_mksysb
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim):
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim):
+-----+
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim): System Backup
Image Space Information
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim): (Sizes are
displayed in 1024-byte blocks.)
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim):
+-----+
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim):
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim): Required = 327520 (328 MB)
Available = 8452488 (8452 MB)
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim):
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim): Data compression will be used
by the system backup utilities which
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim): create the system backup
image. This may reduce the required space
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim): by up to 50 percent.
Tue Jun 6 11:39:22 2006 OUTPUT (/usr/sbin/nim):
Tue Jun 6 11:39:24 2006 OUTPUT (/usr/sbin/nim):

```

```

Tue Jun  6 11:39:33 2006 OUTPUT (/usr/sbin/nim): Creating information file
(/image.data) for rootvg.
Tue Jun  6 11:39:33 2006 OUTPUT (/usr/sbin/nim):
Tue Jun  6 11:39:34 2006 OUTPUT (/usr/sbin/nim): Creating list of files to back
up.
Tue Jun  6 11:39:55 2006 OUTPUT (/usr/sbin/nim): Backing up 12723 files..
Tue Jun  6 11:39:55 2006 OUTPUT (/usr/sbin/nim): 12723 of 12723 files (100%)
Tue Jun  6 11:39:55 2006 OUTPUT (/usr/sbin/nim): 0512-038 savevg: Backup
Completed Successfully.
Tue Jun  6 11:40:05 2006 rc for /usr/sbin/nim = 0
Tue Jun  6 11:40:05 2006 STATUS for phase 5: success

```

---

### 11. Phase 6 - Migrate the system backup.

In this phase, the `nimadm` command is executed to perform an AIX 5L V5.3 migration of each clients mksysb image as shown in Example 4-142.

#### *Example 4-142 Migration the system backups*

---

```

{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun  6 11:41:19 2006

```

Next phase to execute: 6 - Migrate Each System Backup

```

Saved parameters:
  accept_licenses = yes
  images = /moveup/AIX5304
  hmc = hmcp520
  location = /moveup
  gateway = 10.1.1.1
  managed_sys = YLI894-2-650E4DG
  subnet_mask = 255.255.255.0
  spot = SPOT_53_ML4
  lpp_source = LPP_53_ML4
  vioserver = VI01
  target_client = LPAR4
  min_ip = 10.1.1.65

```

---

Begin the execution of **Phase 6** and monitor the output. Note that the output has been reduced to save space as shown in Example 4-143.

#### *Example 4-143 Execution and monitoring of phase 6*

---

```

{nimmast}:/ # nim_move_up -n
Tue Jun  6 11:41:32 2006 Starting Phase 6 : Migrate Each System Backup...
Tue Jun  6 11:41:32 2006 Executing: /usr/sbin/nimadm -HM -s SPOT_53_ML4 -l
LPP_53_ML4

```

```

Tue Jun  6 11:41:32 2006 OUTPUT (/usr/sbin/nimadm):
+-----
Tue Jun  6 11:41:32 2006 OUTPUT (/usr/sbin/nimadm): Gathering System
Information
Tue Jun  6 11:41:32 2006 OUTPUT (/usr/sbin/nimadm):
+-----
Tue Jun  6 11:41:32 2006 OUTPUT (/usr/sbin/nimadm): Determining level of the
"bos.alt_disk_install.rte" fileset for "rootvg" ...
Tue Jun  6 11:41:33 2006 OUTPUT (/usr/sbin/nimadm): Determining level of the
"bos.alt_disk_install.rte" fileset for SPOT NIM
Tue Jun  6 11:41:33 2006 OUTPUT (/usr/sbin/nimadm): resource "SPOT_53_ML4" ...
Tue Jun  6 11:41:36 2006 OUTPUT (/usr/sbin/nimadm): Determining level of the
"bos.alt_disk_install.rte" fileset for lpp_source NIM
Tue Jun  6 11:41:36 2006 OUTPUT (/usr/sbin/nimadm): resource "LPP_53_ML4" ...
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm):
+-----
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm): Checking Software
Synchronization
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm):
+-----
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm): The
bos.alt_disk_install.rte fileset level for "rootvg" is 5.3.0.40
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm): The
bos.alt_disk_install.rte fileset level for "SPOT_53_ML4" is 5.3.0.40
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm): The
bos.alt_disk_install.rte fileset level for "LPP_53_ML4" is 5.3.0.40
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm): Software is currently
synchronized.
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:41:37 2006 OUTPUT (/usr/sbin/nimadm): Log file is:
/var/adm/ras/nimadm.log
Tue Jun  6 11:41:37 2006 rc for /usr/sbin/nimadm = 0
Tue Jun  6 11:41:37 2006 Migrating mkysyb for client LPAR4
Tue Jun  6 11:41:37 2006 Executing: /usr/sbin/nimadm -s SPOT_53_ML4 -l
LPP_53_ML4 -T LPAR4_phys_mkysyb -O /moveup/mkysyb/LPAR4p5/LPAR4_mkysyb -j
rootvg -N LPAR4_p5_mkysyb -Y
Tue Jun  6 11:41:38 2006 OUTPUT (/usr/sbin/nimadm): Initializing the NIM
master.
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Verifying
alt_disk_migration eligibility.
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Initializing log:
/var/adm/ras/alt_mig/LPAR4_phys_mkysyb_alt_mig.log
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Starting Alternate Disk
Migration.
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm):
+-----

```

```

Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Executing nimadm phase 1.
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Processing target mksysb
resource "LPAR4_phys_mksysb".
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Restoring /image.data from
mksysb image.
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Executing nimadm phase 2.
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Creating nimadm cache file
systems on volume group rootvg.
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Checking for initial
required migration space.
Tue Jun  6 11:41:42 2006 OUTPUT (/usr/sbin/nimadm): Creating cache file system
/LPAR4_phys_mksysb_mm_alt/alt_inst
Tue Jun  6 11:41:45 2006 OUTPUT (/usr/sbin/nimadm): Creating cache file system
/LPAR4_phys_mksysb_mm_alt/alt_inst/usr
Tue Jun  6 11:41:48 2006 OUTPUT (/usr/sbin/nimadm): Creating cache file system
/LPAR4_phys_mksysb_mm_alt/alt_inst/var
Tue Jun  6 11:41:50 2006 OUTPUT (/usr/sbin/nimadm): Creating cache file system
/LPAR4_phys_mksysb_mm_alt/alt_inst/tmp
Tue Jun  6 11:41:52 2006 OUTPUT (/usr/sbin/nimadm): Creating cache file system
/LPAR4_phys_mksysb_mm_alt/alt_inst/home
Tue Jun  6 11:41:54 2006 OUTPUT (/usr/sbin/nimadm): Creating cache file system
/LPAR4_phys_mksysb_mm_alt/alt_inst/opt
Tue Jun  6 11:41:56 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:41:56 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:41:56 2006 OUTPUT (/usr/sbin/nimadm): Executing nimadm phase 3.
Tue Jun  6 11:41:56 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:41:56 2006 OUTPUT (/usr/sbin/nimadm): Syncing mksysb data to
cache ...
Tue Jun  6 11:47:05 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm): Executing nimadm phase 4.
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm): nimadm: There is no user
customization script specified for this phase.
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm): Executing nimadm phase 5.

```

```
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm):
+-----
Tue Jun  6 11:47:06 2006 OUTPUT (/usr/sbin/nimadm): Saving system configuration
files.
Tue Jun  6 11:47:41 2006 OUTPUT (/usr/sbin/nimadm): Checking for initial
required migration space.
Tue Jun  6 11:47:41 2006 OUTPUT (/usr/sbin/nimadm): Expanding
/LPAR4_phys_mkysyb_mm_alt/alt_inst/usr local filesystem.
Tue Jun  6 11:47:52 2006 OUTPUT (/usr/sbin/nimadm): Filesystem size changed to
786432
Tue Jun  6 11:47:52 2006 OUTPUT (/usr/sbin/nimadm): Setting up for base
operating system restore.
Tue Jun  6 11:47:52 2006 OUTPUT (/usr/sbin/nimadm): Restoring base operating
system.
Tue Jun  6 11:48:19 2006 OUTPUT (/usr/sbin/nimadm): Merging system
configuration files.
Tue Jun  6 11:48:19 2006 OUTPUT (/usr/sbin/nimadm): Rebuilding inventory
database.
Tue Jun  6 11:48:27 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge Config_Rules.
Tue Jun  6 11:48:27 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge SRCextmeth.
Tue Jun  6 11:48:27 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge SRCsubsys.
Tue Jun  6 11:48:28 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge SWservAt.
Tue Jun  6 11:48:28 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge pse.conf.
Tue Jun  6 11:48:29 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge xtiso.conf.
Tue Jun  6 11:48:30 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge PdDv.
Tue Jun  6 11:48:30 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: convert_errnotify.
Tue Jun  6 11:48:30 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: passwd_mig.
Tue Jun  6 11:48:30 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: login_mig.
Tue Jun  6 11:48:30 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: user_mrg.
Tue Jun  6 11:48:30 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: secur_mig.
Tue Jun  6 11:48:31 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: mkusr_mig.
Tue Jun  6 11:48:31 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: group_mig.
Tue Jun  6 11:48:31 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ldapcfg_mig.
```

```

Tue Jun  6 11:48:31 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: convert_errlog.
Tue Jun  6 11:48:31 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge GAI.
Tue Jun  6 11:48:31 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge PdAt.
Tue Jun  6 11:48:32 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: merge_smit_db.
Tue Jun  6 11:48:32 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: ODM_merge fix.
Tue Jun  6 11:48:33 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: merge_swpvds.
Tue Jun  6 11:48:38 2006 OUTPUT (/usr/sbin/nimadm): Running migration merge
method: SysckMerge.
Tue Jun  6 11:48:42 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 11:48:42 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:48:42 2006 OUTPUT (/usr/sbin/nimadm): Executing nimadm phase 6.
Tue Jun  6 11:48:42 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 11:48:42 2006 OUTPUT (/usr/sbin/nimadm): Installing and migrating
software.
Tue Jun  6 11:48:47 2006 OUTPUT (/usr/sbin/nimadm): Updating install utilities.
Tue Jun  6 11:48:47 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
+-----+
Tue Jun  6 12:23:07 2006 OUTPUT (/usr/sbin/nimadm): Executing nimadm phase 11.
Tue Jun  6 12:23:07 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 12:23:07 2006 OUTPUT (/usr/sbin/nimadm): Defining NIM mksysb
resource ...
Tue Jun  6 12:23:08 2006 OUTPUT (/usr/sbin/nimadm): New NIM mksysb resource
name is "LPAR4_p5_mksysb"
Tue Jun  6 12:23:08 2006 OUTPUT (/usr/sbin/nimadm):
Tue Jun  6 12:23:08 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 12:23:08 2006 OUTPUT (/usr/sbin/nimadm): Executing nimadm phase 12.
Tue Jun  6 12:23:08 2006 OUTPUT (/usr/sbin/nimadm):
+-----+
Tue Jun  6 12:23:08 2006 OUTPUT (/usr/sbin/nimadm): Cleaning up
alt_disk_migration on the NIM master.
Tue Jun  6 12:23:08 2006 rc for /usr/sbin/nimadm = 0
Tue Jun  6 12:23:08 2006 STATUS for phase 6: success

```

---

This phase migrates the client mksysb images to AIX 5L V5.3 using the nimadm utility.

**Note:** The `nimadm` log files are located in `/var/adm/ras/nimadm.log` and the `/var/adm/ras/alt_mig/` directory.

## 12. Phase 7 - Configure NIM definitions for client LPARs.

We run the `nim_move_up -S` command again to review what the next phase will be, see Example 4-144. In this phase (7) new NIM objects are created for each LPAR. The necessary NIM resources are allocated and a `bos_inst` pull operation is run on each client.

### Example 4-144 Configuring NIM definitions for clients

---

```
{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun  6 13:11:03 2006
```

Next phase to execute: 7 - Configure NIM Definitions of Client LPARs

```
Saved parameters:
  accept_licenses = yes
  images = /moveup/AIX5304
  hmc = hmcp520
  location = /moveup
  gateway = 10.1.1.1
  managed_sys = YLI894-2-650E4DG
  subnet_mask = 255.255.255.0
  spot = SPOT_53_ML4
  lpp_source = LPP_53_ML4
  vioserver = VI01
  target_client = LPAR4
  min_ip = 10.1.1.65
```

---

**Phase 7** executes and creates NIM client definitions for the target system. A new NIM client was created named `LPAR4_p5` as shown in Example 4-145.

### Example 4-145 Phase 7 execution and creation of NIM client definitions

---

```
{nimmast}:/ # nim_move_up -n
Tue Jun  6 13:12:10 2006 Starting Phase 7 : Configure NIM Definitions of Client
LPARs...
Tue Jun  6 13:12:10 2006 Adding the following entry into /etc/hosts: LPAR4_p5
10.1.1.65
Tue Jun  6 13:12:10 2006 Executing: /usr/sbin/nim -o define -t standalone -a
if1="find_net LPAR4_p5 0" LPAR4_p5
Tue Jun  6 13:12:20 2006 rc for /usr/sbin/nim = 0
```



```

Tue Jun  6 13:12:20 2006 Executing: /usr/sbin/nim -o bos_inst -a source=mksysb
-a mksysb=LPAR4_p5_mksysb -a spot=SPOT_53_ML4 -a bosinst_data=nim_move_up_bid
-a boot_client=no LPAR4_p5
Tue Jun  6 13:12:23 2006 rc for /usr/sbin/nim = 0
Tue Jun  6 13:12:23 2006 STATUS for phase 7: success

```

**Note:** If you do not update your /etc/hosts file with the new systems hostname and IP address, nim\_move\_up will do this for you. See Figure 4-7.

```

#####
##                                     ##
##  PARTITIONS  VICO                                     ##
##                                     ##
#####

10.1.1.11  lpar1
10.1.1.12  lpar2
10.1.1.13  lpar3
10.1.1.14  lpar4
10.1.1.15  lpar5
10.1.1.16  lpar6
10.1.1.17  lpar7
10.1.1.18  lpar8
10.1.1.50  hmcp520
10.1.1.51  VI01
10.1.1.61  v\lpar1
10.1.1.62  v\lpar2
10.1.1.63  v\lpar3
10.1.1.64  v\lpar4

9.212.133.42  alp3bfs

9.212.133.57  nimmstbfs  nimmast
10.1.1.65    LPAR4_p5

```

Figure 4-7 nim\_move\_up has updated our /etc/hosts

### 13. Phase 8 - Initiate LPAR network installation.

Now it is time for us to install the new POWER5 LPARs as shown in Example 4-146 on page 256.

*Example 4-146 Initiating LPAR network installations*


---

```
{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun  6 13:12:45 2006

Next phase to execute: 8 - Initiate LPAR Network Installations

Saved parameters:
  accept_licenses = yes
  images = /moveup/AIX5304
  hmc = hmcp520
  location = /moveup
  gateway = 10.1.1.1
  managed_sys = YLI894-2-650E4DG
  subnet_mask = 255.255.255.0
  spot = SPOT_53_ML4
  lpp_source = LPP_53_ML4
  vioserver = VI01
  target_client = LPAR4
  min_ip = 10.1.1.65
```

---

Example 4-147 shows the status of our new NIM client. You can see from its current state that it is now ready for a BOS installation via a mksysb (named LPAR4\_p5\_mksysb).

*Example 4-147 Showing the status of our new NIM client*


---

```
{nimmast}:/ # lsnim -l LPAR4_p5
LPAR4_p5:
  class           = machines
  type            = standalone
  connect         = shell
  platform        = chrp
  netboot_kernel  = mp
  if1             = NET_EN1 LPAR4_p5 0
  cable_type1     = N/A
  Cstate         = BOS installation has been enabled
  prev_state      = ready for a NIM operation
  Mstate          = not running
  boot            = boot
  bosinst_data    = nim_move_up_bid
  mksysb         = LPAR4_p5_mksysb
  nim_script      = nim_script
  spot            = SPOT_53_ML4
  control         = master
```

---

The LPAR network installation is initiated (via the `/usr/hmcrbin/lpar_netboot` command on the HMC) as shown in Example 4-148. Once the installation has begun this phase is complete. The installation is not monitored by `nim_move_up`. We monitored the installation (by observing the output on the LPARs console and using the `lsnim` command) to ensure that the installation completed successfully.

*Example 4-148 LPAR network installation initiated*

---

```
{nimmast}:/ # nim_move_up -n
Tue Jun  6 13:23:50 2006 Starting Phase 8 : Initiate LPAR Network
Installations...
Tue Jun  6 13:23:50 2006 Executing: /usr/bin/ssh -l hscroot hmcp520
"lpar_netboot -A -D -t ent -s auto -d auto -S 10.1.1.1 -C 10.1.1.65 -G 10.1.1.1
\"LPAR4_lpar\" \"LPAR4_profile\" \"YLI894-2-650E4DG\""
Tue Jun  6 13:23:54 2006 OUTPUT (/usr/bin/ssh): # Connecting to LPAR4_lpar
Tue Jun  6 13:23:54 2006 OUTPUT (/usr/bin/ssh): # Connected
Tue Jun  6 13:23:54 2006 OUTPUT (/usr/bin/ssh): # Checking for power off.
Tue Jun  6 13:23:55 2006 OUTPUT (/usr/bin/ssh): # Power off complete.
Tue Jun  6 13:23:55 2006 OUTPUT (/usr/bin/ssh): # Power on LPAR4_lpar to Open
Firmware.
Tue Jun  6 13:25:59 2006 OUTPUT (/usr/bin/ssh): # Power on complete.
Tue Jun  6 13:26:01 2006 OUTPUT (/usr/bin/ssh): # Client IP address is
10.1.1.65.
Tue Jun  6 13:26:01 2006 OUTPUT (/usr/bin/ssh): # Server IP address is
10.1.1.1.
Tue Jun  6 13:26:01 2006 OUTPUT (/usr/bin/ssh): # Gateway IP address is
10.1.1.1.
Tue Jun  6 13:27:16 2006 OUTPUT (/usr/bin/ssh): # /vdevice/l-lan@30000003 ping
successful.
Tue Jun  6 13:27:17 2006 OUTPUT (/usr/bin/ssh): # Network booting install
adapter.
Tue Jun  6 13:27:19 2006 OUTPUT (/usr/bin/ssh): # bootp sent over network.
Tue Jun  6 13:31:47 2006 OUTPUT (/usr/bin/ssh): # Network boot proceeding,
lpar_netboot is exiting.
Tue Jun  6 13:31:47 2006 OUTPUT (/usr/bin/ssh): # Finished.
Tue Jun  6 13:31:47 2006 rc for /usr/bin/ssh = 0
Tue Jun  6 13:31:47 2006 STATUS for phase 8: success
```

---

#### 14. Phase 9 - post-migration.

Before executing **Phase 9**, make sure that the installation that was initiated in **Phase 8** has completed successfully first. If you try and execute the next phase before the installation is complete you will receive an error message stating that the NIM client has not finished installing as shown in Example 4-149 on page 258. Use the `lsnim` command to check the clients status. When the installation is finished it's Cstate will show `'ready for a NIM operation'`.

*Example 4-149 Error message if phase 8 has not completed*


---

```
{nimmast}:/ # nim_move_up -n
Tue Jun  6 13:34:08 2006 Starting Phase 9 : post-migration...
Tue Jun  6 13:34:08 2006 ERROR: The NIM client LPAR4_p5 has not finished
installing. Try again later.
Tue Jun  6 13:34:08 2006 STATUS for phase 9: failure
```

---

Now that our LPAR has been successfully installed, we are ready to move into **Phase 9**. Refer to Example 4-150.

*Example 4-150 Executing phase 9*


---

```
{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun  6 13:32:36 2006
```

Next phase to execute: 9 - post-migration

Saved parameters:

```
accept_licenses = yes
images = /moveup/AIX5304
hmc = hmcp520
location = /moveup
gateway = 10.1.1.1
managed_sys = YLI894-2-650E4DG
subnet_mask = 255.255.255.0
spot = SPOT_53_ML4
lpp_source = LPP_53_ML4
vioserver = VI01
target_client = LPAR4
min_ip = 10.1.1.65
```

---

The `post_migration` script is run on LPAR4\_p5 as shown in Example 4-151. This script runs a commands such as **lppchk** to verify the installed filesets. The output from the `post_migration` script is saved to `/home` and should be reviewed after the migration completes.

*Example 4-151 Post migration script execution*


---

```
{nimmast}:/ # nim_move_up -n
Tue Jun  6 14:04:33 2006 Starting Phase 9 : post-migration...
Tue Jun  6 14:04:33 2006 Running post_migration on client LPAR4_p5
Tue Jun  6 14:04:33 2006 Executing: /usr/lpp/bos.sysmgmt/nim/methods/c_rsh
LPAR4_p5 /usr/lpp/bos/post_migration
Tue Jun  6 14:04:33 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/c_rsh):
Tue Jun  6 14:04:33 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/c_rsh):
Running lppchk commands. This may take awhile.
Tue Jun  6 14:11:20 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/c_rsh):
```

---

```
Tue Jun  6 14:11:20 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/c_rsh): All
saved information can be found in: /home/post_migration.060606140432
Tue Jun  6 14:11:20 2006 OUTPUT (/usr/lpp/bos.sysmgmt/nim/methods/c_rsh):
Tue Jun  6 14:11:20 2006 rc for /usr/lpp/bos.sysmgmt/nim/methods/c_rsh = 0
Tue Jun  6 14:11:20 2006 STATUS for phase 9: success
```

---

### 15. Phase 10 - Post installation customization.

This phase is optional. This phase allows for the installation of additional software (LPPs/filesets) or for any additional customization (using a script) that may be needed. We did not perform any customization on the client. Refer to Example 4-152.

#### *Example 4-152 Post installation and customization phase (optional)*

---

```
{nimmast}:/ # nim_move_up -n
Tue Jun  6 14:16:05 2006 Starting Phase 10 : Post-Installation Customization
(Optional)...
Tue Jun  6 14:16:05 2006 No customization resources provided- skipping
customization.
Tue Jun  6 14:16:05 2006 STATUS for phase 10: success
```

---

All phases have now been executed. Our `nim_move_up` operation has completed each phase successfully. All that is left to do is unconfigure the environment in preparation for any future operations we may need to execute. This step is shown in Example 4-153 on page 259.

#### *Example 4-153 Unconfiguring and saving the environment parameters*

---

```
{nimmast}:/ # nim_move_up -S
nim_move_up Status:
=====
Tue Jun  6 14:16:25 2006
All phases have executed.

Saved parameters:
  accept_licenses = yes
  images = /moveup/AIX5304
  hmc = hmcp520
  location = /moveup
  gateway = 10.1.1.1
  managed_sys = YLI894-2-650E4DG
  subnet_mask = 255.255.255.0
  spot = SPOT_53_ML4
  lpp_source = LPP_53_ML4
  vioserver = VI01
  target_client = LPAR4
  min_ip = 10.1.1.65
```

```
{nimmast}:/ # nim_move_up -n
```

All phases have been executed. Run `nim_move_up -r` to clear `nim_move_up` data.

```
{nimmast}:/ # nim_move_up -r
```

Reset of `nim_move_up` environment completed successfully.

---

Our goal has been achieved. We have migrated our AIX 5L V5.1 p670 LPAR to an AIX 5L V5.3 p520 LPAR (see Figure 4-8). Although we have spent a lot of time reviewing each phase of the `nim_move_up` operation the actual time and labour required was minimal when compared to previous platform migration methods.

```
# prtconf | head
System Model: IBM,7040-671
Machine Serial Number: 01A85D2
Processor Type: PowerPC_POWER4
Number Of Processors: 2
Processor Clock Speed: 1000 MHz
CPU Type: 64-bit
Kernel Type: 32-bit
LPAR Info: 4 lpar4
Memory Size: 1024 MB
Good Memory Size: 1024 MB

# oslevel -r
5100-04

# prtconf | head
System Model: IBM,9131-52A
Machine Serial Number: 650E4DG
Processor Type: PowerPC_POWER5
Number Of Processors: 1
Processor Clock Speed: 1499 MHz
CPU Type: 64-bit
Kernel Type: 64-bit
LPAR Info: 1 LPAR4_lpar
Memory Size: 448 MB
Good Memory Size: 448 MB

# oslevel -r
5300-04
```

Figure 4-8 Hardware configuration after `nim_move_up` operation

## 4.5.5 POWER5 to POWER5 nim\_move\_up

It is possible to use the **nim\_move\_up** tool to move an existing POWER5 LPAR running AIX 5L V5.3 to another POWER5 system. During the writing of this redbook we needed to move some AIX 5L V5.3 LPARs from one POWER5 system to another. We were able to use **nim\_move\_up** for this task by making some minor changes to phase 6.

As our clients were already running AIX 5L V5.3 it was not necessary to migrate to AIX 5L V5.3 with **nim\_move\_up** (the migration is performed in phase 6). For example, we moved an LPAR (VLPAR1) from one p520 to another p520. In order for us to move our LPARs, without performing phase 6, we needed to make the following changes immediately after phase 5:

1. We made sure that the string within the *phase6/status* file was set to *success*.
2. We changed the phase number in the *current\_phase* file to 7.
3. We also copied the mksysb image *VLPAR1\_mksysb* to *VLPAR1\_p5\_mksysb* (which is the name of the mksysb image that **nim\_move\_up** would have created after the migration and is required for phase 7).

This allowed us to skip phase 6, continue on with the remaining phases (for example, 7, 8, 9 and 10) and move VLPAR1 to our new p520 to a new LPAR named VLPAR1\_p5.

## 4.6 NIM alternate disk migration

In this section, we describe the Network Installation Manager (NIM) alternate disk migration (**nimadm**) tool and show an example of how to use it to perform an AIX migration to AIX 5L V5.3.

NIM alternate disk migration (**nimadm**) utility gives you several advantages for migrating to a later level of AIX, over a conventional migration. A system administrator can use **nimadm** to create a copy of a clients rootvg to a spare disk on the client (similar to a standard alternate disk install with **alt\_disk\_install**) and migrate the disk to a newer version or release of AIX. All of this can be done without disruption to the client, for example, no outages to perform the upgrade. Once the upgrade is finished, the only downtime required will be a scheduled reboot of the system.

There are several advantages to using **nimadm** over a standard NIM migration, such as:

- ▶ Reduced downtime for the client. The migration is executed while the system is up and running as normal. There is no disruption to any of the applications

or services running on the client therefore the upgrade can be done at a time convenient to the administrator. At a later stage a reboot can be scheduled in order to restart the system at the later level of AIX.

- ▶ The nimadm process is very flexible and it can be customized using some of the optional NIM customization resources such image\_data, bosinst\_data, pre/post\_migration scripts, exclude\_files, etc.
- ▶ Quick recovery from migration failures. All changes are performed on the rootvg copy (altinst\_rootvg). If there are any serious problems with the migration the original rootvg is still available and the system has not been impacted. If a migration fails or terminates at any stage, nimadm is able to quickly recover from the event and cleanup afterwards. There is little for the administrator to do except determine why the migration failed, rectify the situation and attempt the nimadm process again. If the migration process completed but issues were discovered after the system was restarted on the later level of AIX, then the administrator can back-out easily by booting from the original rootvg disk.

The following requirements must be met before attempting to use nimadm:

1. You must have a NIM master running AIX 5L V5.1 or higher with the latest recommended AIX 5L maintenance/technology level or higher.
2. The NIM master must have the *bos.alt\_disk\_install.rte* fileset installed in its own rootvg and in the SPOT which will be used for the migration. Both need to be at the same level.

**Note:** It is not necessary to install the alternate disk utilities on the client.

3. The lpp\_source and SPOT NIM resources which have been selected for the migration MUST match the AIX level to which you are migrating.
4. The NIM master (as always) should be at the same or higher AIX level than the level you are migrating to on the client.
5. The client system to be migrated must be at AIX 4.3.3 or higher.
6. The client must have disk(s) large enough to clone the existing rootvg plus an additional 500 MB of free space for the migration. This amount will depend on the client system configuration.
7. The target client must be registered with the NIM master as a standalone NIM client.
8. The NIM master must be able to execute remote commands (**rsh**) on the client using the rshd protocol.
9. The NIM master and client must have a minimum of 128 MB of RAM.



- 10.. A reliable network connection between the master and client is required. A large amount of NFS and rsh traffic will occur as the master and client will perform NFS mounts and read/write operations during the migration.
11. The client's hardware and software should support the AIX level that is being migrated to and meet the requirements of a conventional migration.

Some limitations apply when using the **nimadm** utility:

1. If the client's rootvg has TCB turned on, you will need to either disable it (permanently) or perform a conventional migration (for example, using CD or NIM. Refer to 4.4, "Using NIM to perform AIX migrations of the NIM master and clients" on page 153). This limitation exists because TCB needs to access file metadata which is not visible over NFS.
2. All NIM resources used by nimadm must be local to the NIM master.
3. The client may experience some minor performance decrease due to the increase in disk I/O and NFS activity, accompanied by some additional CPU usage from the alt\_disk\_install cloning.
4. NFS tuning may be required to optimize nimadm performance, especially on slow networks.
5. If configuration changes are made to the system after rootvg has been cloned then you will need to either redo the alt\_disk\_install phase of the nimadm operation or implement them manually or via a customization script after the migration. For example if new users are created or system tuning options are changed after the **alt\_disk\_install** is run, then you will need to ensure that these changes are performed once the migration is complete, for example, user-related: /etc/passwd, /etc/group, /etc/security/passwd, etc., and system tuning related: /etc/tunables/nextboot.

#### 4.6.1 Local disk caching versus NFS

Before we look at an example of using **nimadm**, there is one last feature we should point out. As already mentioned, nimadm uses NFS for many of the tasks during the migration. This can be a problem on slower networks as NFS writes can be very expensive. To avoid using NFS there is a "Local Disk Caching" option which may provide some performance advantages.

Local disk caching allows the NIM master to avoid having to use NFS to write to the client. This can be useful if the nimadm operation is not performing well due to an NFS write bottleneck. If the "Local Disk Caching" function is invoked, nimadm will create the client file systems in a volume group on the NIM master. It will then use streams (via rshd) to cache all of the data from the client to the file systems on the NIM master.

Advantages of local disk caching:

1. Improved performance for nimadm operations on relatively slow networks.
2. Improved performance for nimadm operations that are bottlenecked in NFS writes.
3. Decreased CPU usage on the client.
4. Client file systems not exported.

Disadvantages of local disk caching:

1. Cache file systems take up space on the NIM master. You must have enough disk space in a volume group on the NIM master to host the client's rootvg file systems plus some space for the migration of each client.
2. Increased CPU usage on the master.
3. Increased I/O on the master. For best performance, use a volume group on the NIM master that does not contain the NIM resources being used for the AIX migration.

We will deploy 'Local Disk Caching' in our **nimadm** test environment, as you will observe shortly.

## 4.6.2 The 12 phases of nimadm migration

The **nimadm** command performs a migration in 12 phases. It is a good idea to have a good understanding of the nimadm process before performing a migration. Each phase can be executed individually using the **-P** flag which can be useful if you are trying to pinpoint a problem with the process, for example, you can run each phase and analyze the result before moving onto the next step.

The nimadm phases are as follows:

- ▶ **Phase 1.** The master issues the **alt\_disk\_install** command to the client, which makes a copy of the rootvg to the target disks (this is Phase 1 of the **alt\_disk\_install** process). In this phase, **altinst\_rootvg** (alternate rootvg) is created. If a target **mksysb** has been specified, the **mksysb** is used to create a rootvg using local disk caching on the NIM master.
- ▶ **Phase 2.** The master runs remote client commands to export all of the **/alt\_inst** file systems to the master. The file systems are exported as read/write with root access to the master.\*\*
- ▶ **Phase 3.** The master NFS mounts the file systems exported in Phase 2. If a target **mksysb** has been specified, the **mksysb** archive is restored into the cache file systems created in Phase 2.\*\*

- ▶ **Phase 4.** If a premigration script resource has been specified, it is executed at this time.
- ▶ **Phase 5.** System configuration files are saved. Initial migration space is calculated and appropriate file system expansions are made. The bos image is restored and the device database is merged (similar to a conventional migration). All of the migration merge methods are executed and some miscellaneous processing takes place.
- ▶ **Phase 6.** All system filesets are migrated using `installp`. Any required RPM images are also installed during this phase.
- ▶ **Phase 7.** If a post-migration script resource has been specified, it is executed at this time.
- ▶ **Phase 8.** The `bosboot` command is run to create a client boot image, which is written to the client's alternate boot logical volume (`alt_hd5`).
- ▶ **Phase 9.** All mounts made on the master in phase 3 are removed<sup>1</sup>.
- ▶ **Phase 10.** All client exports created in phase 2 are removed<sup>2</sup>.
- ▶ **Phase 11.** The `alt_disk_install` command is called again (phase 3 of `alt_disk_install`) to make final adjustments and put `altinst_rootvg` to sleep. The `bootlist` is set to the target disk (unless the `-B` flag is used). If an output `mksysb` has been specified, the cache is archived into a `mksysb` file, and a NIM `mksysb` resource created.
- ▶ **Phase 12.** Cleanup is executed to end the migration. The client is rebooted, if the `-r` flag is specified.

If you are unable to meet the requirements for phases 1 to 10 then you will need to perform a conventional migration, see 4.4, “Using NIM to perform AIX migrations of the NIM master and clients” on page 153.

**Tip:** The `nimadm` command supports migrating several clients at the same time.

<sup>1</sup> When using the “Local Disk Caching” option with `nimadm`, the following phases are modified (all other phases remain the same):

Phase 2 - The NIM master creates local cache file systems in the specified target volume group (on the NIM master).

Phase 3 - The NIM master populates the cache file systems with the client's `rootvg` data.

Phase 9 - The NIM master writes all the migrated data to the clients alternate `rootvg`.

Phase 10 - The NIM master cleans up and removes the local cache file systems.

<sup>2</sup> See 1

### 4.6.3 nimadm migration example

Example 4-154 demonstrates a typical NIM alternate disk migration. We will migrate a system from AIX 5L V5.2 to AIX 5L V5.3. This environment consists of a NIM master running AIX 5L V5.3 on a p615 and a NIM client running AIX 5L V5.2 as an LPAR on a p670. Our NIM client name is LPAR4 and the NIM master hostname is nimmast.

First, we check that our NIM master and client are ready to take part in a nimadm operation.

NIM master checklist:

- ▶ Ensure that the bos.alt\_disk\_install.rte fileset is installed on the NIM master and also in the SPOT that will be used for this migration.

#### *Example 4-154 Alternate disk migration example*

```
{nimmast}:/ # lsllpp -l bos.alt_disk_install.rte
Fileset                Level  State      Description
-----
bos.alt_disk_install.rte 5.3.0.40 COMMITTED  Alternate Disk Installation
Runtime
```

```
{nimmast}:/ # nim -o lsllpp -a filesets='bos.alt_disk_install.rte' SPOT_53_ML4
Fileset                Level  State      Description
-----
Path: /usr/lib/objrepos
bos.alt_disk_install.rte 5.3.0.30 COMMITTED  Alternate Disk Installation
Runtime
                    5.3.0.40 APPLIED    Alternate Disk Installation
Runtime
```

Example 4-155 shows you how to install the bos.alt\* filesets into the SPOT.

#### *Example 4-155 Installing bos.alt\_disk\_install filesets into the SPOT*

```
{nimmast}:/ # smit nim_task_inst
Install and Update from ALL Available Software
SPOT_53_ML4          resources      spot
LPP_53_ML4          resources      lpp_source
Install and Update from ALL Available Software

* Installation Target                SPOT_53_ML4
* LPP_SOURCE                          LPP_53_ML4
* Software to Install                 [bos.alt_disk_install+

Customization SCRIPT to run after installation  [] +
(not applicable to SPOTs)
```

```
Force                                no +

installp Flags
  PREVIEW only?                       [no] +
  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] +
```

#### COMMAND STATUS

```
Command: OK          stdout: yes          stderr: no
```

Before command completion, additional instructions may appear below.

[TOP]

Installing filesets ...

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...
```

#### SUCSESSES

```
-----
Filesets listed in this section passed pre-installation verification
and will be installed.
```

#### Selected Filesets

```
-----
bos.alt_disk_install.boot_images 5.2.0.40 # Alternate Disk Installation ...
bos.alt_disk_install.rte 5.2.0.40      # Alternate Disk Installation ...
```

<< End of Success Section >>

#### FILESET STATISTICS

```
-----
  2 Selected to be installed, of which:
    2 Passed pre-installation verification
  ----
```

[MORE...39]

---

### ***NIM client check list***

- Is the client at AIX 4.3.3 or higher? .
- Does the client have a disk large enough to clone the existing rootvg plus ~500MB of free space for the migration?  
If your rootvg is mirrored you may consider breaking the mirror and using one of the disks for the alternate rootvg. In a SAN environment you may consider allocating a LUN temporarily for the nimadm operation and removing it afterwards.

```
# lspv
hdisk0 001a85d27c3dc850          rootvg          active
hdisk1 001a071f9a12dbaf          None
```

- No need to install the alt\_disk\_install filesets on the client as the **alt\_disk\_install** command which will be run on the client will be executed from the SPOT which is mounted from the NIM master.
- Is the client already registered with the NIM master as a valid NIM client?  
If not, the **niminit** command can be used to configure it from the client side.
- Can the master execute commands on the client? Yes.

```
{nimmast}:/ # rsh lpar4 oslevel -r
5200-04
```

The ~/.rhosts file must exist on the client so that it allows the NIM master remote root access. The correct ownership and permissions must also be set. For example:

```
{nimmast}:/ # rsh lpar4 cat /.rhosts
nimmast root
{nimmast}:/ # rsh lpar4 ls -l /.rhosts
-rw-----  1 root    system          13 Jun 05 14:56 /.rhosts
```

Is the client TCB enabled? If it is enabled, then you have two choices. The first is to disable TCB (*permanently*). Example 4-156 shows you how to disable TCB permanently. Please note the emphasis on “permanently”. If you try to enable TCB again after the migration you may run into some trouble with files being deactivated which may cause havoc to your running system.

Disabling TCB can be accomplished following the procedure shown in Example 4-156:

*Example 4-156 Disabling TCB permanently*

---

```
# odmget -q attribute=TCB_STATE PdAt > /tmp/odmout
# cd /tmp
# cat odmout
PdAt:
    uniquetype = ""
    attribute = "TCB_STATE"
    deflt = "tcb_enabled"
    values = ""
    width = ""
    type = ""
    generic = ""
    rep = ""
    nls_index = 0

# vi odmout
PdAt:
    uniquetype = ""
    attribute = "TCB_STATE"
    deflt = "tcb_disabled"
    values = ""
    width = ""
    type = ""
    generic = ""
    rep = ""
    nls_index = 0

# odmdelete -o PdAt -q attribute=TCB_STATE
0518-307 odmdelete: 1 objects deleted.

# odmget -q attribute=TCB_STATE PdAt
# odmadd /tmp/odmout
# odmget -q attribute=TCB_STATE PdAt

PdAt:
    uniquetype = ""
    attribute = "TCB_STATE"
    deflt = "tcb_disabled"
    values = ""
    width = ""
    type = ""
    generic = ""
    rep = ""
    nls_index = 0
```

---

**Note:** If you do not disable TCB, the nimadm process will terminate and will not perform the migration as shown in Example 4-157.

*Example 4-157 Message if TCB is not disabled*


---

```
{nimmast}:/ # nimadm -j nimvg -c LPAR4 -s SPOT_53_ML4 -l LPP_53_ML4 -d "hdisk1"
-Y
Initializing the NIM master.
Initializing NIM client lpar4.
Verifying alt_disk_migration eligibility.
0505-193 nimadm: The client, lpar4, is TCB enabled.
You must either disable TCB or perform a conventional migration.
Cleaning up alt_disk_migration on the NIM master.
```

---

In our test environment, we attempted the procedure then we re-enabled TCB after a migration as a test only. The results are shown in Example 4-158. It appears obvious that attempting to enable TCB again after a migration will damage your system.

**Note:** "The second option is to perform a conventional migration which supports systems with TCB enabled."

*Example 4-158 tcbck after a migration*


---

```
; Check all the files in the TCB database.
# tcbck -y ALL
; Check is OK.
; Migrate to AIX 5.3
; After migration attempt to enable TCB.
# odmget -q attribute=TCB_STATE PdAt > /tmp/odmout
# cd /tmp
# cat odmout
PdAt:
    uniquetype = ""
    attribute = "TCB_STATE"
    deflt = "tcb_disabled"
    values = ""
    width = ""
    type = ""
    generic = ""
    rep = ""
    nls_index = 0

# vi odmout
PdAt:
    uniquetype = ""
    attribute = "TCB_STATE"
    deflt = "tcb_enabled"
    values = ""
    width = ""
    type = ""
```



```

        generic = ""
        rep = ""
        nls_index = 0

# odmdelete -o PdAt -q attribute=TCB_STATE
0518-307 odmdelete: 1 objects deleted.

# odmget -q attribute=TCB_STATE PdAt
# odmadd /tmp/odmout
# odmget -q attribute=TCB_STATE PdAt

PdAt:
    uniquetype = ""
    attribute = "TCB_STATE"
    deflt = "tcb_enabled"
    values = ""
    width = ""
    type = ""
    generic = ""
    rep = ""
    nls_index = 0

; Check all the files in the TCB database.
# tcbck -y ALL
3001-023 The file /usr/sbin/mirscan has the wrong file mode.
3001-028 The file /usr/sbin/mirscan has the wrong checksum value.
3001-049 The file /usr/sbin/mirscan has the wrong file size.
3001-023 The file /usr/sbin/mklvcopy has the wrong file mode.
3001-028 The file /usr/sbin/mklvcopy has the wrong checksum value.
3001-049 The file /usr/sbin/mklvcopy has the wrong file size.
.....
; Lot's of tcbck errors are reported.
; Now a lot of system commands no longer function.
# lspv
ksh: lspv: 0403-006 Execute permission denied.
; Permissions on lot's of files/commands have changed. For example:
----- 1 root    system    26538 Aug 25 2005 varyonvg
----- 1 root    system    27744 Aug 25 2005 lspv
----- 1 root    system    10216 Aug 25 2005 lchangelv
----- 1 root    system    13506 Aug 25 2005 lresynclv
----- 1 root    system    19070 Aug 25 2005 lchangevg
----- 1 root    system    38796 Aug 28 2005 mklv
----- 1 root    system    14418 Sep 01 2005 lscons
----- 1 root    system    113686 Sep 01 2005 restbyname
----- 1 root    system    15720 Sep 01 2005 swcons
; A mksysb restore may be required to resolve this situation.

```

---

We are now ready to migrate our client from AIX 5L V5.2 ML4 to AIX 5L V5.3 TL4. Our nimadm operation will be performed as shown in Example 4-159.

*Example 4-159 nimadm operation*


---

```

{nimmast}:/ # smit nimadm
Perform NIM Alternate Disk Migration

* Target NIM Client                [LPAR4] +
* NIM LPP_SOURCE resource          [LPP_53_ML4] +
* NIM SPOT resource                [SPOT_53_ML4] +
* Target Disk(s) to install        [hdisk1]
  DISK CACHE volume group name    [nimvg] +

NIM IMAGE_DATA resource           [] +
NIM BOSINST_DATA resource         [] +
NIM EXCLUDE_FILES resource        [] +
NIM INSTALLP_BUNDLE resource      [] +
NIM PRE-MIGRATION SCRIPT resource [] +
NIM POST-MIGRATION SCRIPT resource [] +

Phase to execute                   [all] +
NFS mounting options               []
Set Client bootlist to alternate disk?  yes +
Reboot NIM Client when complete?      no +
Verbose output?                      no +
Debug output?                         no +

ACCEPT new license agreements?       yes +

```

---

We have chosen to use the “local disk caching” option for this migration. The options/flags we supplied to the **nimadm** command are explained in Table 3-1.

**Note:** You may want to consider using at least a custom BOSINST\_DATA resource with **nimadm**. This will give you greater control over the migration process. For example, you can change ALL\_DEVICES\_KERNELS from the default of yes to no which will only install the devices and kernel type for your system configuration thus reducing the number of filesets installed and the time taken to perform the migration.

Performing the operation from the command line would look like:

```

{nimmast}:/ # nimadm -j nimvg -c LPAR4 -s SPOT_53_ML4 -l LPP_53_ML4 -d "hdisk1"
-Y

```

Table 4-2 *nimadm* flags used in our example

Flag	Purpose
-j	This flag, followed by a volume group name, will instruct the <i>nimadm</i> operation to create file systems on the specified volume group (on the NIM master). <i>Nimadm</i> will then use streams to cache all of the data from the client to these file systems.
-c	The name of the NIM client which will be the target of the migration operation.
-s	The SPOT resource to be used for the migration.
-l	The name of the <i>lpp_source</i> to be used for the migration.
-d	Specifies the target disks on the client which will be used to create the <i>altinst_rootvg</i> volume group. This volume group will be the target of the migration.
-Y	Specifies that any software license agreements will be accepted. Before starting a <i>nimadm</i> migration you will be required to accept all software license agreements for the software to be installed. This flag will automatically answer <i>yes</i> to all licenses.

**Note:** When *nimadm* is run via SMIT, you will notice that an undocumented flag (-H) appears in the command line (press F6).

```
/usr/sbin/nimadm -H -cLPAR4 -lLPP_53_ML4 -sSPOT_53_ML4 -dhdisk1 -jnimgv -Y
```

This -H flag is used internally (passed between the SMIT scripts to *nimadm*) so that *nimadm* knows it is called via SMIT. This will then change the way some messages appear, for example, a message like "*Choose this menu option*" would be shown as "*use this flag*" for SMIT versus command line.

There are two other options which should also be considered carefully. The first is whether or not you would like the bootlist on the NIM client set to the alternate disk after the migration is complete. The default action is to set the bootlist. This may be desirable if you are planning on rebooting the client soon after the upgrade.

However, if it is not possible to reboot a short time after (for example, within the next day or so) then you may not prefer this approach. It is possible that if the bootlist is set and the system is left running for several days or even weeks after the migration that you may encounter a undesirable situation. For example, if the system suffers from an unscheduled outage due to a power failure or a system crash and the system comes up on the alternate disk.

If there are any issues as a result of the AIX migration this may cause further unscheduled downtime for the system. Most administrators would prefer to deal with these situations during a scheduled outage.

To prevent the boot list from being set/changed after a nimadm migration, specify the **-B** flag.

The second option to consider is whether or not you would like the NIM client rebooted after the migration completes. The default is set to no. Depending on your environment this may not be appropriate. In our example, we chose not to reboot immediately after the migration as we had applications running on the system that would have been impacted by a reboot of the system.

If you wish to reboot the client after nimadm migration, specify the **-r** flag.

It may be more appropriate for your production environment to schedule an outage at an agreed time and then to stop all applications on the server, manually set the bootlist and reboot the client.

Once the nimadm operation commences the status of each phase will be displayed on the screen. We will pause at each phase and comment on what is happening behind the scenes.

It is possible to execute **nimadm** one phase at a time. You may want to do this in order to perform some troubleshooting if the process is failing. Generally speaking however this won't be required and you will execute all phases at once. Once the process is underway there should be no need for the administrator to intervene.

**Note:** All output from the nimadm operation is logged to the file `/var/adm/ras/alt_mig/nim_client_name_alt_mig.log`, where `nim_client_name` is the name of the NIM client being migrated. In our case, the name of this file was: `/var/adm/ras/alt_mig/LPAR4_alt_mig.log`.

In Example 4-160, **Phase 1** has commenced. During this phase an alternate disk copy of the clients rootvg was created via the `alt_disk_install` utility.

*Example 4-160 Phase 1 starts*

```
{nimmast}:/ # nimadm -j nimvg -c LPAR4 -s SPOT_53_ML4 -l LPP_53_ML4 -d "hdisk1"
-Y
Initializing the NIM master.
Initializing NIM client lpar4.
Verifying alt_disk_migration eligibility.
Initializing log: /var/adm/ras/alt_mig/LPAR4_alt_mig.log
Starting Alternate Disk Migration.
```

```

+-----+
Executing nimadm phase 1.
+-----+
Cloning altinst_rootvg on client, Phase 1.
Client alt_disk_install command: alt_disk_copy -j -M 5.3 -P1 -d "hdisk1"
Calling mkszfile to create new /image.data file.
Checking disk sizes.
Creating cloned rootvg volume group and associated logical volumes.
Creating logical volume alt_hd5
Creating logical volume alt_hd6
Creating logical volume alt_hd8
Creating logical volume alt_hd4
Creating logical volume alt_hd2
Creating logical volume alt_hd9var
Creating logical volume alt_hd3
Creating logical volume alt_hd1
Creating logical volume alt_hd10opt
Creating /alt_inst/ file system.
Creating /alt_inst/home file system.
Creating /alt_inst/opt file system.
Creating /alt_inst/tmp file system.
Creating /alt_inst/usr file system.
Creating /alt_inst/var file system.
Generating a list of files
for backup and restore into the alternate file system...
Phase 1 complete.

```

---

In **Phase 2**, the NIM master creates the cache file systems in the nimvg volume group. Some initial checks for required migration disk space were also performed. See Example 4-161.

*Example 4-161 Executing nimadm phase 2*

```

+-----+
Executing nimadm phase 2.
+-----+
Creating nimadm cache file systems on volume group nimvg.
Checking for initial required migration space.
Creating cache file system /LPAR4_alt/alt_inst
Creating cache file system /LPAR4_alt/alt_inst/usr
Creating cache file system /LPAR4_alt/alt_inst/var
Creating cache file system /LPAR4_alt/alt_inst/tmp
Creating cache file system /LPAR4_alt/alt_inst/home
Creating cache file system /LPAR4_alt/alt_inst/opt
Creating /alt_inst/var file system.

```

---

In **Phase 3**, as shown in Example 4-162, the NIM master copies the NIM client's data to the cache file systems in nimvg. We are using the "local disk cache" option so this data copy is done via rsh instead of using NFS mounts of the client's file systems to the master.

*Example 4-162 Executing nimadm phase 3*

---

```
+-----+
Executing nimadm phase 3.
+-----+
Syncing client data to cache ...
```

---

Viewing the status of the NIM client from the NIM master shows that an alternate disk migration is being performed. See Example 4-163.

*Example 4-163 Checking the status of the NIM client*

---

```
{nimmast}:/ # lsrim -l LPAR4
LPAR4:
  class      = machines
  type       = standalone
  locked     = 397474
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = NET_EN1 lpar4 0
  cable_type1 = tp
  Cstate     = alt_disk_mig operation is being performed
  prev_state = ready for a NIM operation
  Mstate     = currently running
  lpp_source = LPP_53_ML4
  spot       = SPOT_53_ML4
  cpuid      = 001A85D24C00
  control    = master
  Cstate_result = success
```

---

In Example 4-164, we can see the cache file systems that nimadm has created to support the migration operation. Each of the */LPAR4\_alt/alt\_inst\** file systems corresponds to the real file systems on the client's alternate disk copy. On the client the file systems are also mounted with a */alt\_inst* prefix. The */alt\_inst* prefix gives these file systems a unique name which prevents any conflict between the real and the alternate. For example, you cannot have two file systems called */usr* so the alternate is called */alt\_inst/usr*.

*Example 4-164 Cache file system created by nimadm*

---

```
{nimmast}:/ # lsvg -l nimvg
nimvg:
```

LV NAME	TYPE	LPs	PPs	PVs	LV STATE	MOUNT POINT
lppsrc1v	jfs2	97	97	1	open/syncd	/export/lpp_source
loglv00	jfs2log	1	1	1	open/syncd	N/A
spotlv	jfs2	11	11	1	open/syncd	/export/spot
imageslv	jfs2	113	113	1	open/syncd	/export/images
loglv01	jfslog	1	1	1	open/syncd	N/A
lv00	jfs	1	1	1	open/syncd	/LPAR4_alt/alt_inst
lv01	jfs	17	17	1	open/syncd	/LPAR4_alt/alt_inst/usr
lv02	jfs	1	1	1	open/syncd	/LPAR4_alt/alt_inst/var
lv03	jfs	1	1	1	open/syncd	/LPAR4_alt/alt_inst/tmp
lv04	jfs	1	1	1	open/syncd	/LPAR4_alt/alt_inst/home
lv05	jfs	1	1	1	open/syncd	/LPAR4_alt/alt_inst/opt

```
# lsvg -l altinst_rootvg
altinst_rootvg:
```

LV NAME	TYPE	LPs	PPs	PVs	LV STATE	MOUNT POINT
alt_hd5	boot	1	1	1	closed/syncd	N/A
alt_hd6	paging	8	8	1	closed/syncd	N/A
alt_hd8	jfslog	1	1	1	open/syncd	N/A
alt_hd4	jfs	1	1	1	open/syncd	/alt_inst
alt_hd2	jfs	6	6	1	open/syncd	/alt_inst/usr
alt_hd9var	jfs	1	1	1	open/syncd	/alt_inst/var
alt_hd3	jfs	1	1	1	open/syncd	/alt_inst/tmp
alt_hd1	jfs	1	1	1	open/syncd	/alt_inst/home
alt_hd10opt	jfs	1	1	1	open/syncd	/alt_inst/opt

On the client you can see that hdisk1 now belongs to the altinst\_rootvg volume group:

```
{lpar4}:/ # lspv
hdisk0      001a85d27c3dc850    rootvg      active
hdisk1      001a071f9a12dbaf    altinst_rootvg active
```

In **Phase 4**, if a premigration script was specified to run on the client, it would be executed now. We did not execute any premigration script in our test environment. See Example 4-165.

*Example 4-165 Phase 4 pre-migration scripts execution*

```
+-----+
Executing nimadm phase 4.
+-----+
nimadm: There is no user customization script specified for this phase.
```

**Phase 5** will save system configuration files and then perform an initial check for disk space required for the migration as shown in Example 4-166. If required file

systems will be expanded at this point. The BOS run-time environment is restored and the device database (ODM) is merged (similar to a conventional migration).

*Example 4-166 Execution of nimadm phase 5*

---

```

+-----+
Executing nimadm phase 5.
+-----+
Saving system configuration files.
Checking for initial required migration space.
Setting up for base operating system restore.
Restoring base operating system.
Merging system configuration files.
Rebuilding inventory database.
Running migration merge method: ODM_merge Config_Rules.
Running migration merge method: ODM_merge SRCextmeth.
Running migration merge method: ODM_merge SRCsubsys.
Running migration merge method: ODM_merge SWservAt.
Running migration merge method: ODM_merge pse.conf.
Running migration merge method: ODM_merge xtiso.conf.
Running migration merge method: ODM_merge PdDv.
Running migration merge method: convert_errnotify.
Running migration merge method: passwd_mig.
Running migration merge method: login_mig.
Running migration merge method: user_mrg.
Running migration merge method: secur_mig.
Running migration merge method: mkusr_mig.
Running migration merge method: group_mig.
Running migration merge method: ldapcfg_mig.
Running migration merge method: convert_errlog.
Running migration merge method: ODM_merge GAI.
Running migration merge method: ODM_merge PdAt.
Running migration merge method: merge_smit_db.
Running migration merge method: ODM_merge fix.
Running migration merge method: merge_swpds.
Running migration merge method: SysckMerge.

```

---

The migration to AIX 5L V5.3 commences in **Phase 6**. All the client's filesets on the local cache file systems will be migrated via installp. At this point all the activity is taking place on the NIM master not on the client. Output has been removed to save space in Example 4-167.

*Example 4-167 Executing nimadm phase 6 - migration commences*

---

```

+-----+
Executing nimadm phase 6.
+-----+
Installing and migrating software.

```



```

Updating install utilities.
+-----+
          Pre-installation Verification...
+-----+
Verifying selections...done
Verifying requisites...done
Results...
....
....
Checking space requirements for installp install.
Expanding /LPAR4_alt/alt_inst/usr local filesystem.
Filesystem size changed to 2097152
Installing software with the installp installer.
+-----+
          Pre-installation Verification...
+-----+
Verifying selections...done
Verifying requisites...done
Results...
.....
FILESET STATISTICS
-----
  709 Selected to be installed, of which:
      611 Passed pre-installation verification
       60 Replaced by superseding updates
       36 Already installed (directly or via superseding filesets)
        1 Additional requisites to be automatically installed
-----
  612 Total to be installed
.....
install_all_updates: Checking for recommended maintenance level 5300-04.
install_all_updates: Executing /usr/bin/oslevel -rf, Result = 5300-04
install_all_updates: Verification completed.
install_all_updates: Log file is /var/adm/ras/install_all_updates.log
install_all_updates: Result = SUCCESS
Restoring device ODM database.

```

If a post-migration script was specified for the NIM client, then it will be executed in **Phase 7**. We did not execute any post-migration script in our example. See Example 4-168.

*Example 4-168 Executing post-migration scripts if any - phase 7*

```

+-----+
Executing nimadm phase 7.
+-----+
nimadm: There is no user customization script specified for this phase.

```

The **bosboot** command is run to create a new boot image which is written to the clients alternate boot logical volume `alt_hd5`. as shown in Example 4-169.

*Example 4-169 Running the bosboot command during phase 8*

---

```
+-----+
Executing nimadm phase 8.
+-----+
Creating client boot image.
bosboot: Boot image is 25216 512 byte blocks.
Writing boot image to client's alternate boot disk hdisk1.
```

---

On the client (`lpar4`), we could see the alternate boot image being created by observing the current processes on the system as shown in Example 4-170.

*Example 4-170 Alternate boot image is created*

---

```
{lpar4}:/ #ps -ef | grep alt_hd5
  root 188456 258226  1 10:42:37    - 0:00 /usr/bin/dd
if=/alt_inst/usr/lpp/bos.alt_disk_install/boot_images/alt_mig.client.boot
of=/dev/alt_hd5 seek=2
  root 258226 303174  0 10:42:37    - 0:00 ksh -c /usr/bin/dd
if=/alt_inst/usr/lpp/bos.alt_disk_install/boot_images/alt_mig.client.boot
of=/dev/alt_hd5 seek=2 2>&1 && echo rmc_cmd_RET0 || echo rmc_cmd_RET1
  root 274464 188456  1 10:42:37    - 0:00 /usr/bin/dd
if=/alt_inst/usr/lpp/bos.alt_disk_install/boot_images/alt_mig.client.boot
of=/dev/alt_hd5 seek=2
```

---

All the migrated data is now copied from the NIM master's local cache file to the client's alternate rootvg via **rsh** as shown in Example 4-171.

*Example 4-171 Migrated data is now copied during nimadm phase 9*

---

```
+-----+
Executing nimadm phase 9.
+-----+
Adjusting client file system sizes ...
Adjusting size for /
Adjusting size for /usr
Expanding /alt_inst/usr client filesystem.
Filesystem size changed to 2490368
Adjusting size for /var
Adjusting size for /tmp
Adjusting size for /home
Adjusting size for /opt
Syncing cache data to client ...
```

---

Observing the process table on the client and the master (`ps -ef` command) revealed the processes involved in the master-to-client data synchronization, as shown in Example 4-172.

**Note:** The `/alt_inst/alt_disk_mig_rfail` log file which captures status information regarding the syncing of the cache data to the client can be useful for troubleshooting.

*Example 4-172 Data synchronization*

---

```
{lpar4}:/ # ps -ef | grep restore
root 188482 303202  0 10:46:05      - 0:00 ksh -c cd /alt_inst/usr &&
/usr/sbin/restore -xqf- > /dev/null || > /alt_inst/alt_disk_mig_rfail.397474

{nimmast}:/ # ps -ef | grep restore
root 545008 397474  0 10:46:02 pts/3  0:00 /usr/bin/rsh lpar4 cd
/alt_inst/usr && /usr/sbin/restore -xqf- > /dev/null || >
/alt_inst/alt_disk_mig_rfail.397474
root 548984 545008  0 10:46:02 pts/3  0:01 /usr/bin/rsh lpar4 cd
/alt_inst/usr && /usr/sbin/restore -xqf- > /dev/null || >
/alt_inst/alt_disk_mig_rfail.397474
```

---

The NIM master cleans up and removes the local cache file systems in **Phase 10** as they are no longer required as shown in Example 4-173.

*Example 4-173 Cleaning up by the NIM master - nimadm phase 10*

---

```
+-----+
Executing nimadm phase 10.
+-----+

Unmounting client mounts on the NIM master.
forced unmount of /LPAR4_alt/alt_inst/var
forced unmount of /LPAR4_alt/alt_inst/usr
forced unmount of /LPAR4_alt/alt_inst/tmp
forced unmount of /LPAR4_alt/alt_inst/opt
forced unmount of /LPAR4_alt/alt_inst/home
forced unmount of /LPAR4_alt/alt_inst
Removing nimadm cache file systems.
Removing cache file system /LPAR4_alt/alt_inst
Removing cache file system /LPAR4_alt/alt_inst/usr
Removing cache file system /LPAR4_alt/alt_inst/var
Removing cache file system /LPAR4_alt/alt_inst/tmp
Removing cache file system /LPAR4_alt/alt_inst/home
Removing cache file system /LPAR4_alt/alt_inst/opt
```

---

The `alt_disk_install` command is called again, as it was in **Phase 3** to verify the `altinst_rootvg` and to put the volume group to sleep, for example, unmounting

the /alt\_inst file systems as shown in Example 4-174. The bootlist is set to the alternate disk (unless the **-B** flag is specified which we did not).

*Example 4-174 Verifying the alternate rootvg installation*

---

```
+-----+
Executing nimadm phase 11.
+-----+
Cloning altinst_rootvg on client, Phase 3.
Client alt_disk_install command: alt_disk_copy -j -M 5.3 -P3 -d "hdisk1"
## Phase 3 #####
Verifying altinst_rootvg...
Modifying ODM on cloned disk.
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst
forced unmount of /alt_inst
Changing logical volume names in volume group descriptor area.
Fixing LV control blocks...
Fixing file system superblocks...
Bootlist is set to the boot disk: hdisk1
```

---

In **Phase 12** several clean up tasks are performed on the NIM master and client. If the **-r** flag was specified with nimadm, then the NIM client would be rebooted now. We chose not to reboot, which is the default action as shown in Example 4-175.

*Example 4-175 Cleaning continues on phase 12*

---

```
+-----+
Executing nimadm phase 12.
+-----+
Cleaning up alt_disk_migration on the NIM master.
Cleaning up alt_disk_migration on client LPAR4.
```

---

Now that all nimadm phases have been completed successfully we are ready to reboot our client.

Once a system outage time window has been agreed upon and scheduled, all that would be required to “migrate” our system to AIX 5L V5.3 would be a reboot. The following tasks would be performed to reboot from our alternate disk:

- ▶ Once all applications have been stopped and all users logged off. Confirm that the client’s bootlist is set to the alternate rootvg and reboot the system:

```
{lpar4}:/ # bootlist -m normal -o
```

```

hdisk1
{lp4}:/ # lspv
hdisk0      001a85d27c3dc850      rootvg      active
hdisk1      001a071f9a12dbaf      altinst_rootvg
{lp4}:/ # shutdown -Fr

```

- ▶ After the reboot confirm that the correct AIX 5L is shown, for example AIX 5L V5.3. At this point you would perform verification testing to ensure that the system is functioning as expected. For example:

```

{lp4}:/ # oslevel -r
5300-04

```

- ▶ If required the administrator can easily “back out” the upgrade by booting from the original rootvg disk as shown below:

```

{lp4}:/ # bootlist -m normal hdisk0
{lp4}:/ # lspv
hdisk0      001a85d27c3dc850      old_rootvg
hdisk1      001a071f9a12dbaf      rootvg      active
# bootlist -m normal -o
hdisk0 blv=hd5
{lp4}:/ # shutdown -Fr

```

- ▶ After the reboot, confirm that the correct version of AIX is running. For example, the version prior to the migration to AIX 5L V5.3 which was AIX 5L V5.2. Use the following command:

```

{lp4}:/ # oslevel -r
5200-04

```

- ▶ If required, rootvg should be remirrored after the migration. You should then reboot the system to turn off quorum for rootvg.

#### 4.6.4 Cleaning up after a failed nimadm operation

If a nimadm operation fails during a migration, the nimadm utility can take care of cleaning up the system(s) without user intervention. However, in some cases it may be necessary to ask nimadm to rerun the cleanup process. This can be done by running the **nimadm** command with the **-C** (clean up) and/or **-F** (unlock NIM client) flags or via SMIT, as shown in Example 4-176 on page 283.

*Example 4-176 Cleaning after a failed nimadm operation*

---

```

{nimmast}:/ # smit nimadm
                Clean up NIM Alternate Disk Migration

```

```

* Target NIM Client                [LPAR4]+
* NIM SPOT resource                 [SPOT_53_ML4]+
  Debug output?                     no+

```

```
FORCE cleanup to reset all locks?                yes+
```

#### COMMAND STATUS

```
Command: OK          stdout: yes          stderr: no
```

Before command completion, additional instructions may appear below.

```
Initializing the NIM master.
Initializing NIM client lpar4.
Cleaning up alt_disk_migration on the NIM master.
Cleaning up alt_disk_migration on client LPAR4.
Client alt_disk_install command: alt_disk_install -M all -X
Bootlist is set to the boot disk: hdisk0
```

---

It may also be necessary to perform other manual tasks in the event of a nimadm failure:

- ▶ Unmounting all /alt\_\* file systems
- ▶ Removing the /alt\_\* file systems
- ▶ Varying off the altinst\_rootvg volume group
- ▶ Removing the altinst\_rootvg volume group with the **alt\_disk\_install -x altinst\_rootvg** command.

**Important:** It is not recommended to use **exportvg** on an alternate rootvg. You should always use the **-x** option to **alt\_disk\_install**. Using **exportvg** may result in some required entries in **/etc/filesystems** being removed. Our tests have shown that this can and does happen.

Once you are satisfied that the failed nimadm process has been successfully cleaned up, you should attempt the migration operation again.

## 4.7 Network Installation Manager (NIM) and Linux distributions

IBM System p machines traditionally use BOOTP protocol for booting over the network, while Intel® based computers (Windows/Linux) typically use DHCP and PXE protocol.

To be able to serve both AIX and Linux client network installations from one network install server, DHCP must be used for both types, as BOOTP and DHCP

cannot be active on the same server at the same time. This is not a problem since DHCP on AIX 5L V5.3 supports handling of BOOTP requests as well as DHCP requests. But a conversion has to be made for all existing NIM clients to use DHCP configuration files and syntax.

As with AIX 5L V5.3, RFC 2349 is implemented in the AIX 5L **tftp** subsystem, making it possible to use an AIX 5L server as a complete network install server for multiple platforms.

Intel clients typically use DHCP and PXE (Pre-boot eXecution Environment) for booting over the network, and since x86 based clients cannot load more than 512kb of code in the first boot stage, we need a special kernel called 'Linux PXE kernel' for the initial load when installing Linux these platforms. This kernel is responsible for transferring the network install kernel (and optionally a ramdisk) via TFTP and start the installation.

For IBM System p using BOOTP there is only one file to load since IBM System p hardware does not have the 512kb limitation.

## 4.7.1 Configuring DHCP

Before starting to configure the DHCP server, make sure that the `bos.net.tcp.server` fileset is installed. The DHCP server configuration file is located in `/etc/dhcpd.conf`. This configuration example is for a netboot DHCP server only, not to be used to serve other DHCP clients.

By default, almost all parameters in `/etc/dhcpd.conf` are commented out. Example 4-177 shows which parameters have to be activated to setup DHCP as a netboot server. For the DHCP server to start, there has to be at least one network and one subnet defined. To solve this we define a dummy network and a dummy subnet. One of the most important parameters is 'supportBOOTP', as this will enable the DHCP server to answer BOOTP requests, hence support installation of IBM System p servers.

*Example 4-177 Example dhcpd.conf*

---

```
numLogFiles      4
logFileSize      100
logFileName      /var/log/dhcpd.log
logItem          ACTION
logItem          INFO

leaseTimeDefault -1
supportBOOTP    yes
```

# For DHCP to start, at least one network and one subnet is required

```
# Dummy network
network 172.16.0.0 255.255.255.253
```

```
# Dummy subnet
subnet 172.16.0.0      172.16.0.1-172.16.0.2
```

---

Do not start the DHCP server daemon yet, as we must disable BOOTP and convert any existing NIM clients to use DHCP syntax.

## Disabling BOOTP

Bootp is a service started (default) by the `inetd` subsystem. To list the services `inetd` is handling, run:

```
# lssrc -ls inetd
```

If BOOTP is enabled, you should see a line starting with 'bootps'. If you do, edit `/etc/inetd.conf`, and comment out the following line:

```
bootps dgram udp wait root /usr/sbin/bootpd bootpd /etc/bootptab
```

Let the `inetd` daemon be aware of the change by issuing:

```
# refresh -s inetd.
```

Run `lssrc -ls inetd` again, you should not see 'bootps' in the list anymore.

## Converting to DHCP

To enable the DHCP server to handle the existing NIM clients BOOTP requests, their definitions must be converted to use DHCP instead of BOOTP. Before this is done, verify that the MAC (hardware) address is in the NIM definition for all clients as this is required for DHCP. See Example 4-178.

*Example 4-178 Checking the NIM definition of lpar5304*

---

```
root@master:/: lsnim -l lpar5304
Lpar5304:
class          = machines
type           = standalone
connect        = shell
platform       = chrp
netboot_kernel = mp
if1            = eth_nim lpar5304 0006298426AD
cable_type1    = tp
Cstate         = ready for a NIM operation
prev_state     = ready for a NIM operation
Mstate         = currently running
```

---



The MAC address shown in Example 4-178 is 0006298426AD.

If the MAC address is not present, the client has to be updated with this information before the conversion is made. To get the MAC address of an AIX 5L client, issue the `lscfg` command (assuming `ent0` is the network install interface) as shown in Example 4-179.

---

*Example 4-179 Getting the MAC address of an AIX 5L client*

---

```
root@master:/: lscfg -vl ent0
ent0    P1/E1  IBM 10/100 Mbps Ethernet PCI Adapter (23100020)

Network Address.....0006298426AD
Displayable Message.....PCI Ethernet Adapter (23100020)
Device Specific.(YL).....P1/E1
```

---

To update the client with the MAC address information, enter:

```
# nim -o change -a "if1=eth_nim lpar5304 0006298426AD" lpar5304
```

Alternatively, use SMIT to update this information, though this can be time consuming if many clients have to be updated. See Example 4-180.

---

*Example 4-180 SMIT Change/Show characteristics of a machine*

---

Change/Show Characteristics of a Machine

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

	[Entry Fields]	
Machine Name	[lpar5304]	
* Hardware Platform Type	[chrp]	+
* Kernel to use for Network Boot	[mp]	+
Machine Type	standalone	
Network Install Machine State	currently running	
Network Install Control State	ready for a NIM operation	
Primary Network Install Interface		
Network Name	eth_nim	
Host Name	[lpar5304]	
Network Adapter Hardware Address	[0006298426AD]	
Network Adapter Logical Device Name	[ent]	
Cable Type	tp	+
Network Speed Setting	[]	+
Network Duplex Setting	[]	+
IPL ROM Emulation Device	[]	+/
CPU Id	[]	
Communication Protocol used by client	[shell]	+

Comments	[]		
Force	no		+
F1=Help	F2=Refresh	F3=Cancel	F4=List
F5=Reset	F6=Command	F7=Edit	F8=Image
F9=Shell	F10=Exit	Enter=Do	

---

When all NIM clients have their MAC addresses entered into their definitions, some kind of `net_boot` must be activated to make NIM update the `/etc/bootptab` file with the information we will later convert. For our example, we will enable a DIAG boot on all NIM clients. For one client issue the following command:

```
# nim -o diag_boot -a spot=spot5304 <client>
```

If there are many clients, a simple 'awk' script will do the trick:

```
for i in $(lsnim -t standalone|awk '{print $1}'); do
    nim -o diag_boot -a spot=spotxxx $i
done
```

This may not work if some of the clients are currently in a state preventing operations on them. Reset the clients by issuing this command:

```
nim -Fo reset <client>
```

To reset all clients defined use the following:

```
for i in $(lsnim -t standalone|awk '{print $1}'); do
    nim -Fo reset $i
done
```

Then rerun the previous command to enable DIAG boot on all clients:

```
# nim -o diag_boot -a spot=spot5304 <client>
```

When all the NIM clients are allocated for network boot, use the AIX command **bootptodhcp** (part of `bos.net.tcp.server` fileset) to convert the `/etc/bootptab` file to DHCP format. See the man page of **bootptodhcp** for a detailed description of parameters. For our example, use the `-d` flag to dump the output to a file other than `/etc/dhcpsd.cnf`. For example:

```
bootptodhcp -d /tmp/my_nim_clients.dhcp
```

The file `/tmp/my_nim_clients.dhcp` looks similar to the one in Example 4-181.

You should see a list of all your NIM clients in a similar way. The file pointed to by option `bf "/tftpboot/lpar5304"` does not need to exist as NIM will create this as a symbolic link when performing an install, diag or maintenance operation.

Now we have all our NIM clients BOOTP definitions converted to DHCP and saved in this file, which we will use in the next section.

Before we continue, we will reset the state of the NIM clients, and deallocate the resources which were allocated during our DIAG command:

Reset a single client:

```
# nim -Fo reset <client>
```

Reset all NIM clients defined:

```
for i in $(lsnim -t standalone|awk '{print $1}'); do
    nim -Fo reset $i
done
```

Deallocate all resources from a single client:

```
# nim -o deallocate -a subclass=all <client>
```

Deallocate all resources from all NIM clients defined:

```
for i in $(lsnim -t standalone|awk '{print $1}'); do
    nim -o deallocate -a subclass=all $i
done
```

When the resources are deallocated, no more entries should exist in `/etc/bootptab` file other than the default comments.

If new NIM clients are defined, they are NOT automatically added to `dhcpsd.cnf`, this has to be done manually or automated with scripts.

## Merging files

Next step is to merge the client definitions (`/tmp/my_nim_clients.dhcp`) with the `dhcpsd` configuration file. Append this file to `/etc/dhcpsd.cnf`:

```
cat /tmp/my_nim_clients.dhcp >> /etc/dhcpsd.cnf
```

**Note:** the double '>', otherwise your configuration file will be overwritten.

The configuration file looks like Example 4-181.

*Example 4-181 Example dhcpsd.cnf*

---

```
numLogFiles      4
```

```

logFileSize      100
logFileName      /var/log/dhcpd.log
logItem          ACTION
logItem          INFO

leaseTimeDefault      -1
#leaseExpireInterval  3 minutes
supportBOOTP          yes

# For DHCP to start, at least one network and one subnet is required
# Dummy network
network 172.16.0.0 255.255.255.253

# Dummy subnet
subnet 172.16.0.0      172.16.0.1-172.16.0.2

# BOOTP CLIENT: lpar5304
client 1 0006298426AD 192.168.234.19
{
    option 1 255.255.255.0
    option sa 192.168.234.10
    option bf "/tftpboot/lpar5304"
}

```

---

## Enabling DHCP

Now that BOOTP is disabled, we can safely enable the DHCP server. As we already configured `dhcpd.conf`, all we have to do is start the server. **dhcpd** is controlled by **srcmstr**. Its configuration is shown in Example 4-182.

### *Example 4-182 Checking the dhcpd configuration*

```

root@master:/: lssrc -s dhcpd
Subsystem      Group          PID           Status
dhcpd          tcpip         327750        inoperative

```

---

Start the DHCP server as shown in Example 4-183.

### *Example 4-183 startsrc -s dhcpd*

```

root@master:/: startsrc -s dhcpd
0513-059 The dhcpd Subsystem has been started. Subsystem PID is 327750.

```

---

Check if the server started as shown in Example 4-184.

*Example 4-184 lssrc -s dhcpsd*


---

```
root@master:/: lssrc -s dhcpsd
Subsystem      Group          PID           Status
dhcpsd        tcpip          327750        active
```

---

Check if the clients are active for DHCP netboot, as shown in Example 4-185.

*Example 4-185 lssrc -ls dhcpsd*


---

```
root@master:/: lssrc -ls dhcpsd
Log File:                /var/log/dhcpsd.log
Log Level:                0x8c0
Client Expire Interval:   3600
Reserve Expire Interval:  900
Bad Addr Reclaim Interval: 4294967295
Database Save Interval:   3600
```

---

IP Address	Status	Duration	Time Stamp	Client ID
172.16.0.1	Free			
172.16.0.2	Free			
192.168.234.19	Free			

---

We can see here that our client (192.168.234.19) is activated for netboot via DHCP.

To enable the DHCP server to start at boot time, edit `/etc/rc.tcpip` and uncomment the line starting with: `#start /usr/sbin/dhcpsd "$src_running"`.

NIM does not handle DHCP, so all clients will still be able to boot from DHCP (BOOTP) as long as they are defined in `/etc/dhcpsd.cnf` and **dhcpsd** daemon is running. This is not normally a problem with IBM System p systems, as the bootlist can be altered from the SMS menu, remotely from the service processor, or using the **bootlist** command (in AIX).

Intel systems on the other hand do not have the same flexibility, and the boot order is often fixed with the netboot device before disk in the bootlist. This can cause a problem when the system boots and finds a DHCP server responding.

To get around this problem, the `dhcpsd.cnf` configuration file has to be altered to prevent the client from getting a response from the server. This can be done manually by editing the configuration file, or automated with scripts.

Another solution would be to stop the **dhcpsd** server, but this would make installations for other clients impossible as well.

## Enabling tftp

TFTP (Trivial File Transfer Protocol) is used to fetch the actual netboot kernel from a server after the boot has completed. This is done by BOOTP (AIX 5L, IBM System p or DHCP (pxeboot, Linux)).

IBM System p can fetch the entire kernel at once, but for Intel based systems, the size is limited to 512kb and we need a special PXE kernel to handle the rest. Anyway, tftp has to be enabled. This is also a service provided by the **inetd** daemon.

Check if **tftpd** is active by running the command shown in Example 4-186 and look for a line starting with:

```
tftp /usr/sbin/tftpd tftpd -n active
```

*Example 4-186 Checking if tftpd is active (in this case, it is NOT)*

```
root@master:/: lssrsc -ls inetd
```

Subsystem	Group	PID	Status
inetd	tcpip	147528	active

```
Debug          Not active
```

```
Signal         Purpose
```

```
SIGALRM        Establishes socket connections for failed services.
```

```
SIGHUP         Rereads the configuration database and reconfigures services.
```

```
SIGCHLD        Restarts the service in case the service ends abnormally.
```

Service	Command	Description	Status
xmquery	/usr/bin/xmtopas	xmtopas -p3	active
wsmserver	/usr/websm/bin/wsmserver	wsmserver -start	active
time	internal		active
daytime	internal		active
time	internal		active
daytime	internal		active
ntalk	/usr/sbin/talkd	talkd	active
exec	/usr/sbin/rexecd	rexecd	active
login	/usr/sbin/rlogind	rlogind	active
shell	/usr/sbin/rshd	rshd	active
telnet	/usr/sbin/telnetd	telnetd -a	active
ftp	/usr/sbin/ftpd	ftpd	active

If you do not find such a line, edit the `/etc/inetd.conf` and uncomment the line starting with `#tftpd`, then refresh **inetd**:

```
refresh -s inetd
```

Run `lssrc -ls inetd` again to verify that inetd has started. See Example 4-187.

*Example 4-187 Checking for tftp service (here, it is active)*

```
root@master:/: lssrc -ls inetd
```

Subsystem	Group	PID	Status
inetd	tcpip	147528	active

Debug Not active

Signal	Purpose
SIGALRM	Establishes socket connections for failed services.
SIGHUP	Rereads the configuration database and reconfigures services.

SIGCHLD	Restarts the service in case the service ends abnormally.
---------	---

Service	Command	Description	Status
xmquery	/usr/bin/xmtopas	xmtopas -p3	active
wsmserver	/usr/websm/bin/wsmserver	wsmserver -start	active
time	internal		active
daytime	internal		active
time	internal		active
daytime	internal		active
ntalk	/usr/sbin/talkd	talkd	active
<b>tftp</b>	<b>/usr/sbin/tftpd</b>	<b>tftpd -n</b>	<b>active</b>
exec	/usr/sbin/rexecd	rexecd	active
login	/usr/sbin/rlogind	rlogind	active
shell	/usr/sbin/rshd	rshd	active
telnet	/usr/sbin/telnetd	telnetd -a	active
ftp	/usr/sbin/ftpd	ftpd	active

## Directory structure setup

In this section, we are going to concentrate on installing Fedora5 Linux on an Intel PC and on a IBM System p system. The way to do it should not differ much from other Linux distributions.

Figure 4-9 on page 294 shows the directory structure which was used for the installation.

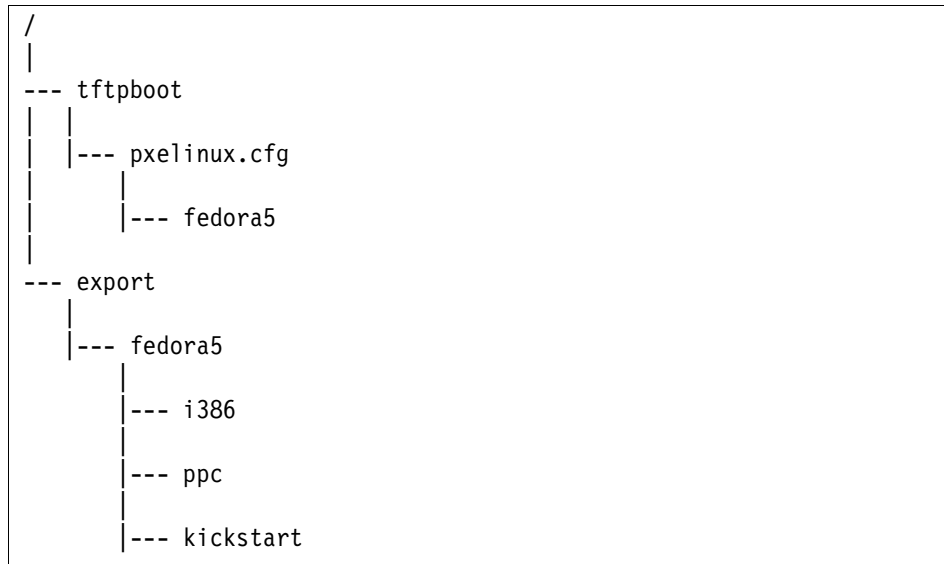


Figure 4-9 Directory structure

It is recommended that each Linux distribution has its own file system for the install sources. About 3-5GB in size depending on the distribution.

To create one file system for Fedora5/i386 and one for Fedora5/ppc on volume group vgnim, enter:

```

crfs -v jfs2 -g vgnim -m /export/fedora5/i386 -A yes -a size=5G
crfs -v jfs2 -g vgnim -m /export/fedora5/ppc -A yes -a size=5G
mount /export/fedora5/i386
mount /export/fedora5/ppc

```

### Get the PXE Linux kernel

The Linux PXE kernel can be obtained from:

<http://www.kernel.org/pub/linux/utils/boot/syslinux>

The actual file you need (pxelinux.0) is included in a package called syslinux. Be sure to get a stable release of syslinux. For our examples, version 2.00

<http://www.kernel.org/pub/linux/utils/boot/syslinux/01d/syslinux-2.00.tar.gz>

was used as it seems to work well.

Unpack the needed pxeboot.0 file using the **gunzip** utility found on the AIX 5L Toolbox for Linux Applications.



```
gunzip -c syslinux-2.00.tar.gz|tar -xf - syslinux-2.00/pxelinux.0
```

Copy the pxelinux.0 file to the /tftpboot/pxelinux.0.200 and create a symbolic link to the pxelinux.0 as shown in Example 4-188.

*Example 4-188 Creating link to copied pxelinux.0 file*

---

```
root@master:/: cp syslinux-2.00/pxelinux.0 /tftpboot/pxelinux.0.200
root@master:/: cd /tftpboot
root@master:/: ln -s pxelinux.0.200 pxelinux.0
ls -l
total 24
drwxr-xr-x  2 root    system      256 Jun 12 13:50 fedora5
lrwxrwxrwx  1 root    system         14 Jun 26 10:15 pxelinux.0 -> pxelinux.0.200
-rw-r--r--  1 root    system    10820 Jun 12 12:00 pxelinux.0.200
drwxr-xr-x  2 root    system      256 Jun 12 12:00 pxelinux.cfg
```

---

## Linux install source creation

The next thing to do is to copy the installation files to the NIM server. Copy the whole directory structure from the installation CDs to the file system we created earlier.

For each CD (or DVD) in the distribution, assuming /cdrom is the CD mount point, do:

For Fedora5/i386™:

```
mount /cdrom
cp -Rhp /cdrom/* /export/fedora5/i386
umount /cdrom
```

For Fedora5/ppc:

```
mount /cdrom
cp -Rhp /cdrom/* /export/fedora5/ppc
umount /cdrom
```

For Fedora5/i386 the netboot kernel (installer) and ramdisk are located in directory images/pxeboot on the installation CD/DVD. Since we copied the whole contents of the CD:s/DVD, they are now in /export/fedora5/i386/images/pxeboot.

These files needs to be copied to /tftpboot/fedora5 directory, preferably to a name the reflects the distribution and version.

Ownership, group and permissions should also be checked. These files will be loaded by the Linux PXE kernel, when the initial boot stage is completed. See Example 4-189.

---

*Example 4-189 Copying the Linux distribution files*

---

```
cp /export/fedora5/i386/images/pxeboot/vmlinuz /tftpboot/fedora5/vmlinuz.fc5.i386
cp /export/fedora5/i386/images/pxeboot/initrd.img
/tftpboot/fedora5/initrd.fc5.i386.img
chown root.system /tftpboot/fedora5/vmlinuz.fc5.i386
chown root.system /tftpboot/fedora5/initrd.fc5.i386.img
chmod 444 /tftpboot/fedora5/vmlinuz.fc5.i386
chmod 444 /tftpboot/fedora5/initrd.fc5.i386.img
```

---

For Fedora/ppc clients using BOOTP only one kernel file has to be loaded during netboot, and the name of the file is the clients' hostname just as with NIM. The netboot kernel file is, after we copied the installation CDs/DVD, located in /export/fedora5/ppc/images/netboot.

There are two files in this directory, ppc32.img and ppc64.img to be used for 32-bit kernel and 64-bit kernel respectively. One of these files (or both) needs to be copied to /tftpboot directory, preferably to a name that reflects the distribution, version and kernel.

Also check ownership, group and permission on the file. A symbolic link must be created from this file to the clients hostname when a network installation is going to be performed. Refer to Example 4-190,

---

*Example 4-190 Copying Linux distribution files and changing ownership permissions*

---

```
cp /export/fedora5/ppc/images/netboot/ppc32.img /tftpboot/ppc32.fc5.img
cp /export/fedora5/ppc/images/pxeboot/ppc64.img /tftpboot/ppc64.fc5.img
chown root.system /tftpboot/ppc32.fc5.img
chown root.system /tftpboot/ppc64.fc5.img
chmod 444 /tftpboot/ppc32.fc5.img
chmod 444 /tftpboot/ppc64.fc5.img
```

---

For the clients to access the installation source over NFS, the corresponding directory need to be exported. Add the following lines to the /etc/exports file:

```
/export/fedora5/i386 -sec=sys,ro
/export/fedora5/ppc -sec=sys,ro
```

Or execute these commands:

```
mknfsxp -d /export/fedora5/i386 -B -S sys -t ro
mknfsxp -d /export/fedora5/ppc -B -S sys -t ro
```

## 4.7.2 Intel PXE configuration

When booting an Intel client with the Linux PXE kernel, a configuration file is needed on the installation server to tell the small Linux PXE kernel where to find the actual netboot kernel and ramdisk image.

**Note:** This configuration file *must* be located in a directory called `/tftpboot/pxelinux.cfg`.

The name of the configuration file is the client's IP-address translated to upper case hexadecimal format. For example, if our client's IP-address is 192.168.234.19, the configuration filename would be 'COA8EA13' as shown in Table 4-3.

Table 4-3 Example decimal to hex conversion

Decimal	Hexadecimal
192	C0
168	A8
234	EA
19	13

Newer versions of the Linux PXE kernel will first look for a filename of the client's ARP type code and hardware address in lowercase hexadecimal with dash separators. For example, with an ethernet (ARP type 1) adapter with address 88:99:AA:BB:CC:DD, the Linux PXE kernel will search for a filename like '01-88-99-aa-bb-cc-dd'.

Refer to the documentation included with the Linux PXE kernel for more details.

It is recommended that the configuration file is a symbolic link to a more easy to understand filename. For example 'fedora5.manual', or 'fedora5.auto'.

```
cd /tftpboot/pxelinux.cfg
ln -s COA8EA13 fedora5.manual
```

The format of the configuration file for Fedora5/i386 is explained in Example 4-191.

Example 4-191 `/tftpboot/pxelinux.cfg/fedora5.manual`

---

```
Install
SERIAL 0 9600
DEFAULT fedora5
```

```

LABEL fedora5
KERNEL fedora5/vmlinuz.fc5.i386
APPEND root=/dev/ram console=/dev/tty1 initrd=fedora5/initrd.fc5.i386.img

```

---

This is a basic configuration file just loading the netboot kernel and the ramdisk image. The installation on the client will be manual just as if installed with CD/DVD. Table 4-4 shows the configuration line entry and its description.

*Table 4-4 Basic configuration file line entry and its description*

Line entry	Description
# Install	Comment
SERIAL 0 9600	Enables serial port to act as a console
DEFAULT fedora5	Sets the default command line
LABEL fedora5	A label of the kernel to boot
KERNEL fedora5/vmlinuz.fc5.i386	Kernel image to boot
APPEND root=/dev/ram cons...	Options appended to the kernel command line

It is possible to specify a more options on the 'APPEND' line, for example to use a kickstart installation for a completely unattended installation. Kickstart installations is out scope for this document, however a sample PXE configuration file using kickstart is provided in Example 4-192.

*Example 4-192 /tftpboot/pxelinux.cfg/fedora5.auto*

```

# Install
SERIAL 0 9600
DEFAULT fedora5
LABEL fedora5
KERNEL fedora5/vmlinuz.fc5.i386
APPEND root=/dev/ram console=/dev/tty1 initrd=fedora5/initrd.fc5.i386.img
ks=nfs:192.168.234.10:/export/fedora5/kickstart/kickstart.i386.fedora5 ksdevice=eth0
PROMPT 0

```

---

This requires a kickstart configuration file. For information on Anaconda, the Fedora Linux installer, and Kickstart, see:

<http://www.fedora.redhat.com>.

## PPC configuration

As Fedora/ppc uses BOOTP for booting over the network, there is no configuration files to edit. When the system boots a manual installation can to be performed just as booting with CD/DVD.

As an alternative, kickstart can be used to automate the installation also on ppc systems, but since it is currently not possible to append any command line parameters to the netboot kernel, they have to be specified at the firmware prompt. The Open Firmware prompt as shown in Example 4-193 can be accessed by pressing F8 at boot time.

*Example 4-193 Open firmware prompt*

---

1 = SMS Menu	5 = Default Boot List
6 = Stored Boot List	8 = Open Firmware Prompt

```
memory      keyboard      network      scsi      speaker      ok
```

```
0>boot net ks=nfs://192.168.234.10:/export/fedora5/kickstart/kickstart.i386.fedora5
```

---

The installation proceeds in kickstart mode just as with the automated PXE configuration above.

## 4.8 Common OS image management (COSI)

With the emergence of IBM On-Demand Virtualization Engine™, the desire to create partitions using shared processors, virtual network interfaces, virtual disks and using NIM as the focal point for system management is becoming ever so important.

Also, the appeals to maximize hardware and software resource utilization have prompted the re-use of NIM diskless and dataless clients as a more cost-effective solution to represent the logical partition carved up from the managed system.

Diskless clients are machines without disk whereas dataless clients are machines that have disks, but lack the necessary information to operate independently. Neither of these types of machines have the capability to boot up by themselves, and must rely on a common boot image along with a NIM SPOT and other file system resources that resides on some server on the network, to operate correctly. The same resources can be used amongst different clients.

The original concept of diskless and dataless clients for use as graphical workstations has changed, and as such, the term thin server is being used to describe these type of machines for its limited and remote resource requirement and the term *common image* is used to describe the SPOT resource used by the thin servers.

## 4.8.1 Thin server, diskless, dataless, and common OS image

### ***Thin server***

A *thin server* is a machine that lacks the physical hardware or software to allow it to operate independently. This machine must rely on remote resources such as a common OS image to operate correctly. The thin server will encompass both a diskless and a dataless machine.

### ***Diskless machine***

A *diskless machine* is a machine that lacks a physical disk and must rely on a spot resource, a root resource, a dump resource, and a paging resource to run. Other optional resources that a diskless machine uses are tmp, home, and shared\_home.

### ***Dataless machine***

A *dataless machine* is a machine that has a physical disk but the disk holds no local operating system. This machine must rely on a spot resource for its OS. It also requires a root resource and a dump resource. The paging resource is not required because the local disk can be used for paging. Other optional resources that a dataless machine uses are tmp, home, and shared\_home.

### **Thin server resources**

The following are resources provided by a NIM master or NIM resource server, for the thin server clients to function, in either diskless or dataless mode:

<b>root</b>	Defined as a directory which will be NFS mounted by thin server clients for storing a clients' "/" (root) directories. When a client is initialized, the root resource will be populated with a client's configuration files. These configuration files are copied from the SPOT/COSI resource used by a client to boot up and as its operating system.
<b>dump</b>	Defined as a directory which will be NFS mounted by thin server clients for used as a dump device by the client for storing a clients dump files.
<b>paging</b>	Defined as a directory which will be NFS mounted by thin server clients for used as a paging device by the client.

<b>home</b>	Defined as a directory which will be NFS mounted by thin server clients for storing a client's specific /home directories.
<b>shared_home</b>	Defined as a /home directory shared by all clients. All thin server clients that use a shared_home resource will mount the same directory as the /home file system.
<b>tmp</b>	Defined as a directory which will be NFS mounted by thin server clients for use as a /tmp file system on the clients.
<b>cosi</b>	A common OS image (COSI) is a repository that contains all the necessary software to bring thin server clients up to a functional state. For AIX systems it is a SPOT (Single Product Object Tree) and provides a /usr file system.

**Note:** The COSI term replaces the SPOT term for the tasks of managing thin servers and common images without requiring user to understand NIM.

### Thin server commands

With AIX 5L V5.3 TL5, the following new commands to manage common OS images and thin servers are introduced:

<b>mkcosi</b>	Makes a common operating system image (COSI) for use with thin servers.
<b>lscosi</b>	List information related to a common operating system image (COSI).
<b>chcosi</b>	Change a common operating system image (COSI).
<b>cpcosi</b>	Clone a common operating system image (COSI).
<b>rmcosi</b>	Remove a common operating system image (COSI).
<b>mkts</b>	Make a thin server.
<b>swts</b>	Switch a thin server to a different COSI.
<b>lsts</b>	List information related to a thin server.
<b>rmts</b>	Remove a thin server.
<b>dbts</b>	Debug a thin server.

The COSI and thin server management can also be accessed through the SMIT `tscosi_client` fast path.

**Note:** The thin server commands use the NIM master database and the **nim command** can be used to manipulate the thin server NIM environments, just as if they were diskless or dataless NIM clients.

You can also use the **nim** command directly for several thin server operations such as to reboot a thin Server named LPAR55 in NIM:

```
# nim -o reboot LPAR55
```

On a NIM COSI object these are the operations you can perform using the `nim` command directly:

<code>reset</code>	Reset the COSI's NIM state
<code>define</code>	Define a COSI
<code>change</code>	Change a COSI's attributes
<code>remove</code>	Remove a COSI
<code>cust</code>	Perform software customization
<code>sync_roots</code>	Synchronize roots for all clients using specified COSI
<code>showres</code>	Show contents of a COSI
<code>maint</code>	Perform software maintenance
<code>lslpp</code>	List LPP information about the COSI
<code>fix_query</code>	Perform queries on installed fixes
<code>showlog</code>	Display a log in the NIM environment
<code>check</code>	Check the status of the COSI
<code>lppchk</code>	Verify installed filesets in the COSI

On a NIM thin server object, these are the operations you can perform using the `nim` command directly, depending on if it is diskless or dataless:

<code>define</code>	Define a diskless or dataless thin server
<code>change</code>	Change a thin server's attributes
<code>remove</code>	Remove a thin server
<code>allocate</code>	Allocate a resource for use by a thin server
<code>deallocate</code>	Deallocate a resource
<code>dkls_init</code>	Initialize a diskless thin server environment
<code>dtls_init</code>	Initialize a dataless thin server environment
<code>diag</code>	Enable a thin server to boot a diagnostic image
<code>reset</code>	Reset a thin server's NIM state
<code>check</code>	Check the status of a thin server
<code>reboot</code>	Reboot specified thin server
<code>showlog</code>	Display a log in the NIM environment
<code>dbts</code>	Perform a dbts operation on a thin server
<code>swts</code>	Perform a swts operation on a thin server

## 4.8.2 Creating a common OS image (COSI)

Assuming we have stored all necessary filesets for AIX 5L V5.3 with TL5 in `/usr/sys/inst.images` directory, you use the `mkcosi` command to create a COSI, as shown in Example 4-194.

*Example 4-194 Creating a common operating system image (COSI) resource*

```
root@master:/: mkcosi -s /usr/sys/inst.images -l /export/cosi cosi5305
```



Defining cosi5305\_106646\_lpp object...Preparing to copy install images (this will take several minutes)...

```
| /export/cosi/cosi5305_106646_lpp/RPMS/ppc/cdrecord-1.9-7.aix5.2.ppc.rpm
| /export/cosi/cosi5305_106646_lpp/RPMS/ppc/mkisofs-1.13-4.aix4.3.ppc.rpm
| /export/cosi/cosi5305_106646_lpp/installp/ppc/xlC.aix50.8.0.0.I
```

...

Now checking for missing install images...

```
| All required install images have been found. This lpp_source is now ready.
done
```

Defining cosi5305 object...

```
Creating SPOT in "/export/cosi" on machine "master" from "cosi5305_106646_lpp" ...
```

```
| Restoring files from BOS image. This may take several minutes ...
```

```
| Installing filesets ...
```

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.

...

done

```
Removing cosi5305_106646_lpp object definition...done
```

---

```
|
| When the mkcosi command starts, it creates a temporary NIM lpp_source, so you must ensure that the location file system has enough space to accommodate this requirement. Example 4-195 shows the temporary NIM lpp_source created in Example 4-194 on page 302.
```

*Example 4-195 Displaying information on temporary lpp\_source for COSI creation*

---

```
cosi5305_106646_lpp:
  class      = resources
  type       = lpp_source
  locked     = 258218
  Rstate    = unavailable for use
  prev_state =
  location   = /export/cosi/cosi5305_106646_lpp
  alloc_count = 0
  server     = master
```

---

```
|
| To display information about a COSI, use the lscosi command. It has both a debugging verbose flag -v, and a detail verbose level flag -l, with levels 1, 2 and
```

3, to provide the user with detailed information. The `lscosi` command, with level one, is equivalent to the `lsnim` command with the `-l` flag. See Example 4-196.

*Example 4-196 Using the lscosi command*

---

```
root@master:/: lscosi -l1 cosi5305
cosi5305:
  class      = resources
  type       = spot
  plat_defined = chrp
  arch       = power
  Rstate     = ready for use
  prev_state = verification is being performed
  location   = /export/cosi/cosi5305/usr
  version    = 5
  release    = 3
  mod        = 0
  oslevel_r  = 5300-05
  alloc_count = 1
  server     = master
  if_supported = chrp.mp ent
  Rstate_result = success
```

---

To remove a COSI, use the `rmcosi` command:

```
root@master:/: rmcosi cosi5305
```

### 4.8.3 Creating a thin server

Use the `mkts` command to create resources to bring up a thin server, except the COSI which has to be already available. The `mkts` command will initialize the thin server and create the `dump`, `tmp`, `root`, `paging`, `home` and `shared_home` NIM resources, if the resources does not exist and their respective flags were specified.

The `mkts` command uses the `/export/nim` directory to create the necessary resources for the client, and the equivalent of a dataless machine. To create a diskless thin server, use the `-l` option, as shown in Example 4-197.

*Example 4-197 Using mkts to define a thin server definition*

---

```
root@master:/: mkts -i 10.1.1.55 -m 255.255.255.0 -g 10.1.1.1 -c cosi5305 -l LPAR55
```

```
Checking /export/nim space requirement...Defining LPAR55 object...done
```

```
Installing software in the root directories of any diskless or dataless
```

clients served by this SPOT. This may take several minutes ...

---

If you get a message similar to the following one, you should create a /export/nim file system with enough space for the NIM resources that **mkts** is going to create:

```
dspmsg: Invalid argument index in message. May need more arguments
on the command line.
Checking define_ts[31]: 9.27 >
blocks^J143.70^J71.14^J21.98^J32.63^J13.13^J-^J1
7.72^J39.52^J2821.70^J1579.36 : syntax error
```

To display information about a thin server, use the **lst**s command as shown in Example 4-198. It has both a debugging flag (verbose, -v), and a detail verbose level flag (-l), with levels 1, 2 and 3, to provide the user with detailed information. The **lst**s command, with level one, is equivalent to the **lsnim** command with the -l flag.

*Example 4-198 Using lsts to display thin server definition*

---

```
root@master:/: lsts -ll LPAR55
LPAR55:
  class      = machines
  type       = dataless
  comments   = dataless client defined with mkts
  platform   = chrp
  netboot_kernel = mp
  if1        = net_10_1_1 lpar55 0
  cable_type1 = N/A
  Cstate     = dataless install has been enabled
  prev_state =
  Mstate     = currently running
  boot       = boot
  dump       = dump
  root       = root
  spot       = cosi5305
  control    = master
```

---

For our thin server, the /usr-COSI (formerly known as SPOT) together with the root, dump and paging file systems have been exported (/etc/exports), and can be viewed using the **exportfs** command as follows:

```
/export/cosi/cosi5305/usr -ro,root=1par55:,access=1par55:
/export/nim/root/LPAR55 -root=1par55:,access=1par55:
/export/nim/dump/LPAR55 -root=1par55:,access=1par55:
/export/nim/paging/LPAR55 -root=1par55:,access=1par55:
```

And in the `/etc/bootptab` file, there is a created network boot entry as follows:

```
lpar55:bf=/tftpboot/lpar55:ip=10.1.1.55:ht=ethernet:sa=10.1.1.2:sm=2
55.255.255.0:
```

```

BOOTP R = 1 BOOTP S = 2
FILE: /tftpboot/lpar55
FINAL Packet Count = 24729
FINAL File Size = 12660736 bytes.
load-base=0x4000
real-base=0x2000000
. . .

***** Please define the System Console. *****

. . .
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
  Performing all automatic mounts

```

*Figure 4-10 Booting a thin server*

Now the thin server should be up and running using the NIM resource server for each of its needed resources as shown in Figure 4-10.

Example 4-199 shows sample commands to illustrate the file system environment in the thin server. This particular thin server is created as a diskless thin server, but it has one disk allocated even though it is not currently used.

*Example 4-199 Commands on the thin server*

---

```

root@master:/: rsh lpar55
*****
*
*
* Welcome to AIX Version 5.3!
*
*
* Please see the README file in /usr/lpp/bos for information pertinent to
* this release of the AIX Operating System.
*

```

```

*
*****

# mount
node          mounted          mounted over    vfs      date          options
-----
master /export/nim/root/LPAR55 /          nfs3    Jun 27 16:04 rw,hard,intr,llock,ac1
master /export/cosi/cosi5305/usr /usr      nfs3    Jun 27 16:04 ro,hard,intr,llock,ac1

# lps -a
Page Space      Physical Volume  Volume Group    Size %Used Active  Auto  Type
swapnfs0        ---             ---             64MB   2   yes   yes   nfs

# ipl_varyon -i
PVNAME          BOOT DEVICE      PVID              VOLUME GROUP ID
hdisk0          NO               00cc544ecd1145720000000000000000 00cc544e00004c00

# lspv
hdisk0          00cc544ecd114572          None

# lsvg
0516-318 lsvg: No volume groups found.

```

---

To modify the / (root) file system for the thin server shown in Example 4-199, edit files in /export/nim/root/LPAR55 directory, and for the thin server /usr file system, edit files in the /export/cosi/cosi5305/usr directory.

#### 4.8.4 Remove a thin server

To remove a thin server definition, use the `rmts` command. In Example 4-200, we first try to remove the LPAR55 thin server. Since it is not online in the first example, the command fails and we rerun it with the force flag `-f`.

*Example 4-200 Using rmts to remove a thin server definition*

---

```

root@master:/: rmts LPAR55
Removing LPAR55 object definition...failed
rmts: Unable to remove LPAR55.
Try using the force flag.

root@master:/: rmts -f LPAR55
Uninitializing LPAR55.
Removing LPAR55 object definition...done

```

---

Using the `nim` command operation `deallocate` in the following example, will cause all remotely mounted resources to be removed from use by the thin server, in this case the LPAR55:

```
# nim -Fo deallocate -a subclass=all LPAR55
```

This will cause all I/O to be blocked since the NFS file systems will no longer be available.

## 4.9 Using HACMP with HANIM

Starting with AIX 5L V5.3, Network Installation Manager (NIM) provides an integrated high availability feature called HANIM, which is presented in 4.1, “High Availability NIM (HA NIM)” on page 124.

However, when using HANIM, the NIM master and the alternate NIM master are not aware of each others state (as there is no keep alive information between primary and alternate NIM masters). The takeover of the role as acting NIM master is based on whether NIM clients wish to use a specific system as their NIM master or not. The NIM master for a NIM client is specified in the clients’ `/etc/niminfo` file.

In this section, we show how IBM High Availability Cluster Multi-Processing (HACMP) can be used to automate HANIM takeover operations. The HACMP configuration shown is minimal, and the purpose is to automate the start and stop of the `nimesis` daemon under the control of SRC, and issuing the appropriate `nim` commands to perform a takeover of NIM clients.

**Note:** As HANIM does not make use of shared storage, HACMP is not involved in any disk takeover operation.

In this section, we use the following three POWER5 logical partitions as shown in Figure 4-11 on page 309:

lpar55	This is the first and original NIM master (HACMP cluster node one).
lpar56	This is the first and alternate NIM master (HACMP cluster node two).
lpar6	This is the first and only NIM client (already running as a NIM client to another NIM master environment).

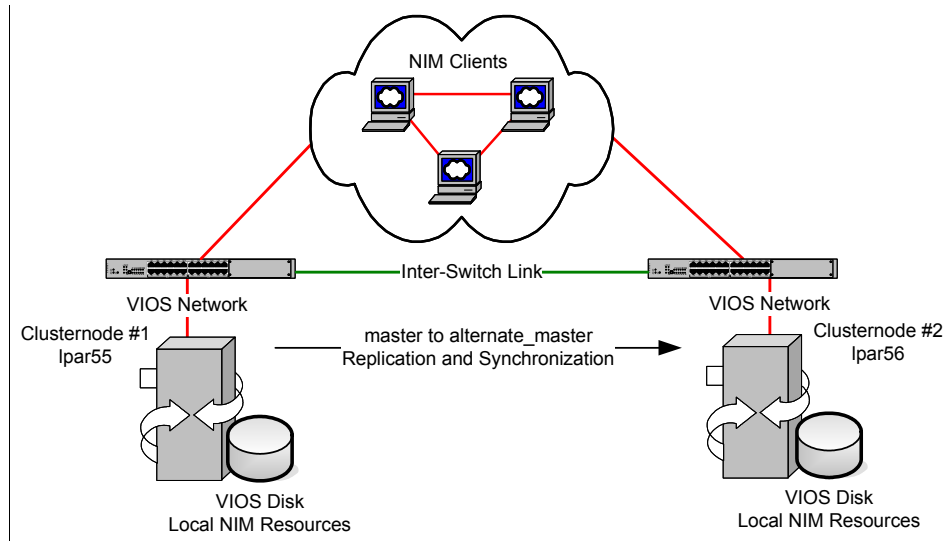


Figure 4-11 Graphic layout of our HACMP cluster setup for HANIM

All LPARs are configured using a VIO server for disk and network resources, and CPU micro-partitioning, as shown in Example 4-201.

*Example 4-201 Basic LPAR config for HACMP HANIM examples*

```

root@lpar56:/: lscfg -vl hdisk0
hdisk0 U9111.520.65C544E-V6-C2-T1-L810000000000 Virtual SCSI Disk Drive

root@lpar56:/: lscfg -vl ent0
ent0 U9111.520.65C544E-V6-C3-T1 Virtual I/O Ethernet Adapter (1-lan)

Network Address.....223380006003
Displayable Message.....Virtual I/O Ethernet Adapter (1-lan)
Device Specific.(YL).....U9111.520.65C544E-V6-C3-T1

root@lpar56:/: lparstat -h
System configuration: type=Shared mode=Capped smt=0n lcpu=2 mem=512 psize=2 ent=0.10
. . .

```

The HACMP configuration will only use the ent0 network virtual network adapter on the cluster nodes, and all NIM database and resource file systems will be local to each of the two cluster nodes. HANIM functionality will be used to keep the database synchronized and resources replicated.

**Note:** In this scenario, we assume that one NIM master has all server resources that are needed to service the network (there are no additional resource servers).

The HACMP configuration is made using these assumptions and requirements regarding availability:

- ▶ The systems management personnel shall not have to restore and reconfigure the NIM server in case of hardware failure.
- ▶ The longest downtime for the NIM server shall be two hours.
- ▶ Hardware failures might occur during the NIM servers lifetime, but is unlikely.
- ▶ Planned downtime will occur during the NIM servers' lifetime to install new firmware or system updates.
- ▶ In case of network failure, the NIM master can failover to the alternate master.
- ▶ There will not be any service IP address, since HANIM will manage the NIM environment.
- ▶ In case of disk failure, the NIM master can failover to the alternate master.

Based on the assumptions and requirements, we used the following software levels: AIX 5L V5.3 TL5, RSCT 2.4.5, and HACMP 5.4 (use SUMA to download updates to AIX 5L, RSCT and HACMP respectively).

After the NIM master completes the BOS “rte” install of the LPARs used in this section, they are customized with the `nim` commands in Example 4-202 (becoming HACMP nodes).

The NIM client name in Example 4-202 is LPAR55. First, it is defined in the acting NIM master (hostname “master”), then an AIX 5L V5.3 TL5 SPOT (`cosi5305`) and NIM `lpp_source` (`lpp5305`) are allocated, and finally the `bos_inst` operation are performed to reinstall the NIM client. After this steps, some additional `bos` filesets are installed together with the `bos.sysmgt.nim.master` fileset for NIM master functionality itself. Then the `rsct.basic.hacmp` and `rsct.basic.rte` and `cluster.es.server` filesets for HACMP are installed. The same steps are done for LPAR56.

*Example 4-202 NIM master additional software updates of HACMP nodes*

```
root@master:/: nim -o define -t standalone -a if1="net_10_1_1 lpar55 0" LPAR55
root@master:/: nim -o allocate -a spot=cosi5305 -a lpp_source=lpp5305 LPAR55
root@master:/: nim -o bos_inst -a source=rte LPAR55
root@master:/: nim -o cust -a lpp_source=lpp5305 -a installp_flags=agXY -a
filesets="bos.adt.libm bos.adt.syscalls bos.sysmgt.nim.master" LPAR55
```



```
root@master:/: nim -o cust -a lpp_source=hacmp54 -a installp_flags=agXY -a
filesets="rsct.basic.hacmp rsct.basic.rte cluster.es.server" LPAR55
```

---

Now we configure HACMP, and synchronize the HACMP nodes (lpar55 and lpar56), and bring up the cluster.

First we make sure that /etc/hosts on each node contain the IP-address of both HACMP cluster nodes and of the client, lpar6. For our purposes, we use the following IP-addresses:

```
10.1.1.55 lpar55
10.1.1.56 lpar56
10.1.1.6 lpar6
```

We also adds the NSORDER variable to /etc/environment to ensure that we use the local systems /etc/hosts file to resolve hostnames to IP-addresses (we could also use the /etc/netsvc.conf or /etc/irs.conf files for the same purpose, see “Setting up IP name resolution” on page 62):

```
NSORDER=local,bind
```

Since we use rsh to communicate between the HACMP cluster nodes, we update the /.rhosts and /usr/sbin/cluster/etc/rhosts file on each node. We include the IP-address of the other nodes IP-address in each file.

The /.rhosts looks like this in HACMP node lpar55:

```
10.1.1.56 root
```

And for /usr/sbin/cluster/etc/rhosts on lpar55 looks like this:

```
10.1.1.56
```

.rhosts on lpar56 looks like this:

```
10.1.1.55 root
```

And for /usr/sbin/cluster/etc/rhosts on lpar56 like this:

```
10.1.1.55
```

We update the /usr/sbin/cluster/netmon.cf file and put IP addresses of known hosts to PING (since we only use a single network interface on each HACMP cluster node and this will let our local node determine if it has a working network, when it can not communicate with the other cluster node). In our case we use the default gateway only:

```
10.1.1.254
```

**Note:** The HACMP cluster commands to create a HACMP cluster are in the `/usr/es/sbin/cluster/utilities/` directory. Include this in your path, or prefix each “cl” command below with the full path, or it will not work.

## HACMP configuration

Now we are ready to issue the first HACMP cluster command to initiate the HACMP cluster itself. Call it NIMESIS.

```
claddclstr -n NIMESIS
```

We continue by adding our two nodes, the NIM master (lpar55) and alternate NIM master (lpar56), picking up the communications interface (-p) from `/etc/hosts`:

```
clnodename -a lpar55 -p lpar55
clnodename -a lpar56 -p lpar56
```

We let HACMP collect information about disks on the nodes (but we will not use this information further):

```
clharvest_vg -w
```

Now we can create the default network and allow IP-address takeover to be performed via IP aliases (even if we will not use any specific service IP-address in the cluster):

```
clmodnetwork -a -n master_net -i ether -s 255.255.255.0 -l yes
```

We add the boot IP-addresses for each node to the HACMP `master_net` Ethernet network:

```
claddnode -a lpar55:ether:master_net::boot: -n lpar55
claddnode -a lpar56:ether:master_net::boot: -n lpar56
```

To create the HACMP resource group, for the NIM master functionality, we first create the resource group itself:

```
claddgrp -g NIMESIS -n "lpar55 lpar56" -S "OHN" -O "FNPN" -B "NFB"
```

We create the HACMP application server for automating the HANIM takeover functionality (see “HACMP start and stop scripts for HANIM” on page 318 for more details about the scripts):

```
claddserv -s"NIMESIS" -b /local/rg/start.nimesis -e
/local/rg/stop.nimesis
```

We add the HACMP application server (NIMESIS) to the HACMP resource group (NIMESIS):

```
claddress -g NIMESIS APLICATIONS=NIMESIS
```

The additional configuration is to add the HACMP application server scripts on both nodes to be kept in sync by HACMP. This can easily be done by creating a file collection and adding the script files to this collection:

```
clfilecollection -o coll -a start_stop 'application start and stop
scripts' yes yes
clfilecollection -o file -a start_stop /local/rg/start.nimesis
clfilecollection -o file -a start_stop /local/rg/stop.nimesis
```

Now the cluster is configured. We just need to verify and synchronize with the peer node (lpar56) correcting minor errors if there are any. See Example 4-203.

```
clclare -rt -i -b -C yes
```

*Example 4-203 Partial output from clclare -rt -i -b -C yes*

---

The following file collections will be processed:

```
start_stop
```

```
Starting file propagation to remote node lpar56.
```

```
Successfully propagated file /local/rg/start.nimesis to node lpar56.
```

```
Successfully propagated file /local/rg/stop.nimesis to node lpar56.
```

```
Total number of files propagated to node lpar56: 2
```

```
Verification to be performed on the following:
```

```
Cluster Topology
```

```
Cluster Resources
```

```
Verification will automatically correct verification errors.
```

```
Reading Cluster Topology Configuration...
```

```
Retrieving data from available cluster nodes. This could take a few minutes....
```

```
VERIFICATION NAME: base.ver_determine_active_cluster
```

```
Thu Jun 29 16:49:13 2006
```

```
Determine if one or more nodes in the cluster are active.
```

```
. . .
```

```
Committing any changes, as required, to all available nodes...
```

```
Verification has completed normally.
```

---

## Checking the HACMP cluster configuration

Now we have a HACMP cluster with two nodes, one Ethernet interface each, nothing shared, and with one resource group. Example 4-204 shows our configuration displayed with the `cltopinfo` and `cllsres` commands.

*Example 4-204 HACMP cluster displayed with cltopinfo -i and cllsres*

```
root@lpar55:/: cltopinfo -i
```

IP Label	Network	Type	Node	Address	If	Netmask
----------	---------	------	------	---------	----	---------

```

=====      =====  =====  =====      =====  =====
lpar55      master_net ether  lpar55      10.1.1.55   en0  255.255.255.0
lpar56      master_net ether  lpar56      10.1.1.56   en0  255.255.255.0

```

```

root@lpar55:/: cllsres
APPLICATIONS="NIMESIS"
FILESYSTEM=""
FORCED_VARYON="false"
FSCHECK_TOOL="fsck"
FS_BEFORE_IPADDR="false"
RECOVERY_METHOD="sequential"
SSA_DISK_FENCING="false"
VG_AUTO_IMPORT="false"

```

---

### HANIM takeover operation

If a NIM client's `/etc/niminfo` file looks like the first example (lpar6 here), after the takeover operation are performed (from lpar55 here), it will be changed to look like in the second example. See Example 4-205.

*Example 4-205 Changes to a NIM clients `/etc/niminfo` file by `nim -o takeover`*

---

```

{lpar6}:/ # cat /etc/niminfo
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=lpar6
export NIM_HOSTNAME=lpar6
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=lpar56
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_MASTER_HOSTNAME_LIST="lpar56 lpar55"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.6:lpar6  10.1.1.56:lpar56 "
export NIM_MOUNTS=""

```

```

root@lpar55:/: nim -o takeover lpar56

```

```

{lpar6}:/ # cat /etc/niminfo
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM

```

```

export NIM_NAME=lpar6
export NIM_HOSTNAME=lpar6
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=lpar55
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_MASTER_HOSTNAME_LIST="lpar55 lpar56"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.6:lpar6 10.1.1.55:lpar55 "
export NIM_MOUNTS=""

```

The `nim` command with the takeover operation are the core of what we use in our HACMP application start script.

### Starting the HACMP HANIM cluster

To start the HACMP HANIM cluster, you can use the SMIT `clstart` fast path as shown in Figure 4-12. Select the node you want to start the cluster manager on, in this case `lpar55` (we prefer to start the cluster managers sequentially and one at a time, at least until the primary node is up).

Start Cluster Services		
Type or select values in entry fields. Press Enter AFTER making all desired changes.		
	[Entry Fields]	
* Start now, on system restart or both	now	+
Start Cluster Services on these nodes	[lpar55]	+
* Manage Resource Groups	Automatically	+
BROADCAST message at startup?	false	+
Startup Cluster Information Daemon?	true	+
Ignore verification errors?	false	+
Automatically correct errors found during cluster start?	Interactively	+

Figure 4-12 Starting HACMP cluster node

After the same `clstart` has been performed for the second HACMP node (`lpar56`), we check the status of our NIMESIS resource group with the `clRGinfo` or `clfindres` commands as shown in Example 4-206.

#### Example 4-206 Using `clRGinfo`

```

root@lpar55:/: clRGinfo
-----

```

Group Name	Group State	Node
NIMESIS	ONLINE	lpar55
	OFFLINE	lpar56

Now, we will stop the NIMESIS resource group with the **c1RGmove** command, on lpar55 as shown in Example 4-207.

*Example 4-207 Using c1RGmove to stop the HANIM resource group*

```
root@lpar55:/: c1RGmove -g NIMESIS -n lpar55 -d
Attempting to bring group NIMESIS offline on node lpar55.

Waiting for the cluster to process the resource group movement request....

Waiting for the cluster to stabilize.....

Resource group movement successful.
Resource group NIMESIS is offline on node lpar55.
```

We can start the NIMESIS resource group with the **c1RGmove** command, on lpar56 as shown in Example 4-208.

*Example 4-208 Using c1RGmove to start the HANIM resource group*

```
root@lpar55:/: c1RGmove -g NIMESIS -n lpar56 -u
Attempting to bring group NIMESIS online on node lpar56.

Waiting for the cluster to process the resource group movement request....

Waiting for the cluster to stabilize.....

Resource group movement successful.
Resource group NIMESIS is online on node lpar56.
```

Now, we will move the NIMESIS resource group with the **c1RGmove** command from lpar56 to lpar55 as shown in Example 4-209.

*Example 4-209 Using c1RGmove to move the HANIM resource group*

```
root@lpar55:/: c1RGmove -g NIMESIS -n lpar55 -m
Attempting to move resource group NIMESIS to node lpar55.

Waiting for the cluster to process the resource group movement request....

Waiting for the cluster to stabilize.....

Resource group movement successful.
```

Resource group NIMESIS is online on node lpar55.

---

Checking the movement with `clfindres` shows the process during the movement as shown in Example 4-210.

*Example 4-210 Checking HANIM resource group transfer between nodes*

---

```
root@lpar55:/: clfindres
-----
Group Name      Group State      Node
-----
NIMESIS         OFFLINE          lpar55
                 OFFLINE          lpar56
root@lpar55:/: clfindres
-----
Group Name      Group State      Node
-----
NIMESIS         ACQUIRING        lpar55
                 OFFLINE          lpar56

root@lpar55:/: clfindres
-----
Group Name      Group State      Node
-----
NIMESIS         ONLINE           lpar55
                 OFFLINE          lpar56
```

---

To examine how the HACMP cluster are performing, you can check the logfiles. For example check the `/tmp/hacmp.out` file on each cluster node, you can also use the different “cl” commands, but also the `lssrc` command and for that matter the `lssrsrc` command as well. Example 4-211 shows a few samples.

*Example 4-211 Examining the topology services daemon with `lssrc -ls topsvcs`*

---

```
root@lpar55:/: lssrc -ls topsvcs
Subsystem      Group           PID    Status
topsvcs        topsvcs         696454 active
Network Name   Indx Defd  Mbrs  St  Adapter ID      Group ID
master_net_0   [ 0] 2     2     S   10.1.1.55       10.1.1.56
master_net_0   [ 0] en0             0x44a3eff8      0x44a3f07b
HB Interval = 1.000 secs. Sensitivity = 10 missed beats
Missed HBs: Total: 0 Current group: 0
. . .
Fast Failure Detection available but off.
Dead Man Switch Enabled:
  reset interval = 1 seconds
  trip interval = 20 seconds
```

```

Configuration Instance = 20
. . .
User time 0 sec. System time 0 sec.
Number of page faults: 0. Process swapped out 0 times.
Number of nodes up: 2. Number of nodes down: 0.

```

---

Using the `lssrc` command to display information about the HACMP cluster manager shows the Dynamic Node Priority values (DNP) for node `lpar55` and `lpar56` (in our simplified example we do not use these features). See Example 4-212.

*Example 4-212 Examining the HACMP cluster manager with `lssrc -ls clstrmgrES`*

---

```

root@lpar55:/: lssrc -ls clstrmgrES
Current state: ST_STABLE
sccsid = "@(#)36 1.135.1.60 src/43haes/usr/sbin/cluster/hacmprd/main.C, hacm
p.pe, 52haes_r540, r5400622c 5/30/06 17:21:27"
i_local_nodeid 0, i_local_siteid -1, my_handle 1
ml_idx[1]=0 ml_idx[2]=1
There are 0 events on the Ibcst queue
There are 0 events on the RM Ibcst queue
CLversion: 9
local node vrmf is 5400
cluster fix level is "0"
The following timer(s) are currently active:
Current DNP values
DNP Values for NodeId - 1 NodeName - lpar55
PgSpFree = 129807 PvPctBusy = 0 PctTotalTimeIdle = 99.708770
DNP Values for NodeId - 2 NodeName - lpar56
PgSpFree = 130348 PvPctBusy = 0 PctTotalTimeIdle = 99.248557

```

---

### **HACMP start and stop scripts for HANIM**

Now let's look at our HACMP scripts as shown in Figure 4-13 on page 319. We use one script to start and one script to stop our services. Since we use HANIM, the stop part is only to stop the `nimesis` SRC demon (the NIM master daemon).

The start script will make sure that the `nimesis` SRC daemon is started and will collect some information regarding the running cluster before it issues a `nim` command ordering takeover from the alternate NIM master (just in case it was the acting master for the NIM clients).

We have created the `/local` file system on each node, and a directory called "rg" in this file system. We place the scripts in the `/local/rg` directory and make sure they are synchronized between the cluster nodes.



```

#!/bin/ksh
# start.nemesis
# Synopsis.....%M%
# Author.....The Dude!
# Created.....2006
# Version.....%Z% %M% %I% (%E% %U%) %Q%
# Description....N/A
# Input.....N/A
# Output.....Change the /etc/niminfo on all clients:
#             NIM_MASTER_HOSTNAME, NIM_MASTER_HOSTNAME_LIST and
#             NIM_HOSTS variables.
# Algorithm.....Check if nimesis are running, start it if not,
#             perform takeover regardless.
#
#-----
trap 'exit 0' EXIT

export PATH=/usr/bin:/usr/sbin:/usr/es/sbin/cluster/utilities

NIMESISPID=$(lssrc -s nimesis|awk '!/PID/{if (/active/) print $3}')

if [[ -z "$NIMESISPID" ]] ;then
    printf "The \"nimesis\" SRC will be started.\n"
    startsrc -s nimesis
    sleep 6
else
    printf "The nimesis SRC is already running \"$NIMESISPID\".\n"
fi

ALTERNATE_MASTER=$(lsnim -Z -t alternate_master|awk -F: '!/^#{print $1;exit}')

printf "The NIM alternate_master is \"$ALTERNATE_MASTER\".\n"

[[ -z "$ALTERNATE_MASTER" ]] || nim -o takeover $ALTERNATE_MASTER

```

Figure 4-13 HACMP start.nemesis script

In both scripts, we always end with a trap to ensure we return zero (0) as exit value. A non-zero return value from a HACMP application server will cause the HACMP resource group to fail and require recovery from script failure, and we do not want that regardless of the outcome of our scripts.

The stop script will make sure that the nimesis SRC daemon is stopped as shown in Figure 4-14 on page 320.

```

#!/bin/ksh
# stop.nemesis
# Synopsis.....%M%
# Author.....The Dude!
# Created.....2006
# Version.....%Z% %M% %I% (%E% %U%) %Q%
# Description....N/A
# Input.....N/A
# Output.....N/A
# Algorithm.....Try to stop the nimesis SRC normally, if that does
#                 not work, try to force it down.
#
#-----
trap 'exit 0' EXIT

export PATH=/usr/bin:/usr/sbin:/usr/es/sbin/cluster/utilities

NIMESISPID=$(lssrc -s nimesis|awk '!/PID/{if (/active/) print $3}')

if [[ ! -z "$NIMESISPID" ]] ;then
    printf "The \"nimesis\" SRC will be stopped.\n"
    stopsrc -s nimesis >/dev/null 2>&1
    sleep 3
else
    printf "The \"nimesis\" SRC is already stopped.\n"
    exit
fi

NIMESISPID=$(lssrc -s nimesis|awk '!/PID/{if (/active/) print $3}')

if [[ ! -z "$NIMESISPID" ]] ;then
    printf "The \"nimesis\" SRC will be terminated.\n"
    stopsrc -s nimesis -c >/dev/null 2>&1
fi

```

Figure 4-14 HACMP stop.nimesis script

## 4.10 NIM and Service Update Management Assistant

The Service Update Management Assistant allows the system administrators to setup the capability of automating the download of maintenance fixes onto a system and supports a comprehensive set of features:

- ▶ Automated task-based retrieval of multiple fix types (APAR, PTF, Critical, Security, Latest, Fileset, Maintenance Level, Technology Level)
- ▶ Three task actions allow for download preview, actual download of updates, or combining the download with a fix repository cleanup (utilizing **lppmgr**)
- ▶ A scheduling using **cron** allows policies to be run at various intervals in order to conform to necessary maintenance windows
- ▶ E-mail notification of update availability and task completion
- ▶ Support for HTTP, HTTPS, and FTP transfer protocols and proxy servers
- ▶ Filtering options allow comparisons against an installed software inventory or a maintenance level

**Important:** SUMA is a complement to the UNIX servers product family portion of the IBM Support Fix Central Web site located at:

<http://www.ibm.com/eserver/support/fixes/>

You can use either the **suma** command or the **smit suma** fast path. The **suma** command can be found in `/usr/sbin` and the SUMA Perl libraries are in the `/usr/suma/lib/SUMA` directory. Configuration changes are saved in indexed files in the `/var/suma/data` directory. The SUMA log files can be found under the `/var/adm/ras` directory, and the basic log file is `suma.log`.

SUMA is installed as part of the base operating system of AIX 5L V5.3. Systems at AIX 5L V5.2 or AIX 5L V5.1 may need to install additional filesets in order to enable SUMA.

**Note:** More information about the **suma** command can be found on the following IBM Web sites:

<http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.doc/cmds/aixcmds5/suma.htm>

<http://www14.software.ibm.com/webapp/set2/sas/f/suma/home.html>

<http://www.ibm.com/servers/aix/whitepapers/suma.html>

## SUMA for AIX 5L V5.2 and AIX 5L V5.1

For AIX 5L V5.2, you have to install either the 5200-05 maintenance level or the fixes and prerequisites shown in Table 4-5. For AIX 5L V5.1, you have to install the fixes and prerequisites shown in Table 4-5.

Table 4-5 SUMA for AIX 5L V5.2 and AIX 5L V5.1

Description	Fileset	AIX 5L V5.2	AIX 5L V5.1
Service Update Management Assistant (SUMA)	bos.suma (install package)	5.2.0.0	5.1.0.0
Service Update Management Assistant (SUMA)	bos.suma (update package)	5.2.0.1	5.1.0.1
Perl Library Extensions	perl.libext (install package)	2.0.58.0	2.0.56.0
Perl Version 5 Runtime Environment	perl.rte (update package)	5.8.0.10	5.6.0.10

### SUMA network connection

Remember that the system, in this case the NIM master, must have access to the Internet and use Internet Domain Name Service (DNS). SUMA needs to resolve the `ibm.com` domain and uses by default FTP (ports 20 & 21) and HTTP (port 80).

**Attention:** Check that your system can resolve the names into IP addresses for the IBM support fix servers and that you can reach them from your NIM or fix download servers.

First, use the classic ping test to the IBM support fix server, for example:

```
ping -c 1 www14.software.ibm.com
```

with one PING only. The output should look similar to Example 4-213 with “0% packet loss”.

*Example 4-213 Ping test to www14.software.ibm.com*

```
root@master:/: ping -c 1 www14.software.ibm.com
PING dispsd-42-www14.boulder.ibm.com: (9.17.252.42): 56 data bytes
64 bytes from 9.17.252.42: icmp_seq=0 ttl=241 time=184 ms
```

```
----dispsd-42-www14.boulder.ibm.com PING Statistics----
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 184/184/184 ms
```

If the test is not successful, even after increasing the number of PINGs (-c 3), then check the systems `/etc/resolv.conf` file. If it exists, it indicates the DNS servers used by the system (`ls /etc/resolv.conf` or `nameslv -s`).

If this file is empty or does not have a valid DNS name listed, add a “nameserver” entry with a valid DNS IP address to the `/etc/resolv.conf` file, either using the `vi` editor, SMIT `mknamerslv` fast path or the `namerslv -a -i #.#.#.#` command (the `#.#.#.#` should be your IP address for your DNS server with Internet resolution of hostnames).

**Note:** If this computer has a `resolv.conf` that does not allow Internet resolution of hostnames or IP addresses, contact your network administrator responsible for the network and request Internet DNS and FTP/HTTP access for the system.

The following examples show how to use the `dig`, `host` and `nslookup` commands to resolve the SUMA default `FIXSERVER_URL`, see Figure 4-219 on page 327. The `dig` command shown in Example 4-214 requests the DNS “A” record.

*Example 4-214 Checking DNS name resolution*

---

```

root@master:/: dig www14.software.ibm.com A

;; <<>> DiG 9.2.0 <<>> www14.software.ibm.com A
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42846
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www14.software.ibm.com.          IN      A

;; ANSWER SECTION:
www14.software.ibm.com. 138     IN      CNAME   dispsd-42-www14.boulder.ibm.com.
dispsd-42-www14.boulder.ibm.com. 45     IN      A       10.1.252.42

;; Query time: 3 msec
;; SERVER: 10.1.0.14#53(10.1.0.14)
;; WHEN: Tue Jun 6 12:07:01 2006
;; MSG SIZE rcvd: 94

root@master:/: host www14.software.ibm.com
dispsd-42-www14.boulder.ibm.com is 10.1.252.42, Aliases: www14.software.ibm.com

```

```
root@master:/: nslookup www14.software.ibm.com
Server: linux0.sapidi.mop.ibm.com
Address: 10.1.0.14
```

```
Non-authoritative answer:
Name:   dispsd-42-www14.boulder.ibm.com
Address: 10.1.252.42
Aliases: www14.software.ibm.com
```

---

Example 4-215 shows the `nslookup` command. There is an error message indicating that `nslookup` is not supported by the DNS, so use the `dig` or `host` commands instead.

*Example 4-215 Error message using nslookup with some DNS servers*

---

```
root@master:/: nslookup www14.software.ibm.com
*** Can't find server name for address 10.1.0.14:Non-existent host/domain
*** Can't find server name for address 10.2.0.14:Non-existent host/domain
*** Default servers are not available
```

---

## 4.10.1 How to use SUMA

Before starting to use SUMA you should know:

- ▶ *Which* systems you want to update
- ▶ *Why* you want to update these systems
- ▶ *What* software you want to update
- ▶ *When* you want to update which systems

Using SUMA should make it easier to keep systems' software updates up to date, however it is not a recommended best practice to constantly be on the latest level of the operating system, unless it is absolutely necessary.

**Attention:** If ain't broken, don't fix it.

To understand what to download with SUMA, or from the IBM support fix central Web site, and the meaning of the new IBM service terminology for AIX 5L (TL, ML, CSP or SP), see 4.10.7, "IBM AIX 5L service strategy" on page 349.

Software maintenance with SUMA allows you to automate download directly to the NIM master, but it will not automatically create new or update neither `lpp_source` nor `SPOT`. This has still to be done manually.

For those wishing to delay their adoption of a new Technology Level (TL) by a period of six to eight months, and take advantage of the latest tested group of PTF's available in a Concluding Service Pack (CSP), the "Maximum Stability" maintenance model for AIX may be appropriate. For more information on the practical handling of the model with SUMA see "Maximum stability" on page 334.

The appropriate usage is to download updates using SUMA and then update the appropriate NIM lpp\_source from the updates in the SUMA directory. The manually newly updated lpp\_source can then be used to update the associated spot.

**Important:** Do not use basic SUMA to download directly to a NIM lpp\_source directory. The SUMA FilterDir parameter assists in not having to download into the lpp\_source directly. The `niminv` command with the `fixget` option have the same risk of downloading directly to the NIM lpp\_source directory. See the "The niminv command" on page 344.

To get the latest fixes, you can use then `niminv` command for NIM machines or NIM lpp\_source objects, or the `compare_report` command for non-NIM systems.

For more detailed comparisons of the installed software, compared to what is both available from your own NIM server lpp\_source, but also from the IBM Support Fix Central Web site, refer to "Check if a system has all available updates installed" on page 342.

## 4.10.2 Settings for SUMA

You can view and configure the global configuration settings for SUMA with either the `suma` command and the "-D" flag or through the SMIT `suma_config` fastpath. Example 4-216 shows the default display.

*Example 4-216 View the SUMA global configuration settings*

---

```
root@master:/: suma -D
  DisplayName=
  Action=Download
  RqType=Security
  RqName=
  RqLevel=
  PreCoreqs=y
  Ifreqs=y
  Supersedes=n
  ResolvePE=IfAvailable
  Repeats=y
```

```
DLTarget=/usr/sys/inst.images
NotifyEmail=root
FilterDir=/usr/sys/inst.images
FilterML=
FilterSysFile=localhost
MaxDLSize=-1
Extend=y
MaxFSSize=-1
```

---

In the following examples, we assume that the `/export/lpp_suma` file system has been created<sup>3</sup>, and that different downloads are kept in separate directories under this file system mountpoint, such as the directory `/export/lpp_suma/5304` used for AIX 5L V5.3 TL04.

We want this to be reflected in the SUMA configuration, and we use the **suma command** with the `-D` flag and `-a attribute=value` to change specific attribute values as shown in Example 4-218.

Example 4-217 shows the attribute `FilterSysFile` is set to `"/dev/null"` as the new default value. The `"/dev/null"` will prevent SUMA to filter against the software inventory of the system that you are running on. The default download directory (`DLTarget`) and the default filter directory (`FilterDir`) are both set to `/export/lpp_suma`. In our scenario, this file system will only contain update level specific directories, but by setting the defaults in this way, it saves on typing when using SMIT to create or execute SUMA tasks.

*Example 4-217 Update the SUMA global configuration settings*

---

```
root@master:/: suma -D -a DLTarget='/export/lpp_suma' -a FilterDir='/export/lpp_suma'
-a FilterSysFile='/dev/null'
```

---

Example 4-217 can also be done through the SMIT `suma_config_base` fastpath. To set the value to `/export/lpp_suma/5304` in SMIT, the path is already prepended and only have to type 5304. By using `FilterDir` we can perform downloads to the same directory and if the update is already there, it will not be downloaded again.

*Example 4-218 View the SUMA global configuration settings, after change*

---

```
root@master:/: suma -D
      DisplayName=
      Action=Download
      RqType=Security
```

---

<sup>3</sup> `crfs -v jfs2 -g rootvg -a size=1G -m /export/lpp_suma -A yes -p rw -a agblksize=4096 -a logname=INLINE && mount /export/lpp_suma`



```

RqName=
RqLevel=
PreCoreqs=y
Ifreqs=y
Supersedes=n
ResolvePE=IfAvailable
Repeats=y
DLTarget=/export/lpp_suma
NotifyEmail=root
FilterDir=/export/lpp_suma
FilterML=
FilterSysFile=/dev/null
MaxDLSize=-1
Extend=y
MaxFSSize=-1

```

---

You can view and configure the general task settings for SUMA with either the **suma command** and the “-c” flag or through the SMIT `suma_task_defaults` fastpath. Example 4-219 shows the default display.

*Example 4-219 View the SUMA general task settings*

---

```

root@master:/: suma -c
FIXSERVER_PROTOCOL=http
DOWNLOAD_PROTOCOL=ftp
DL_TIMEOUT_SEC=180
DL_RETRY=1
MAX_CONCURRENT_DOWNLOADS=5
HTTP_PROXY=
HTTPS_PROXY=
FTP_PROXY=
SCREEN_VERBOSE=LVL_INFO
NOTIFY_VERBOSE=LVL_INFO
LOGFILE_VERBOSE=LVL_VERBOSE
MAXLOGSIZE_MB=1
REMOVE_CONFLICTING_UPDATES=yes
REMOVE_DUP_BASE_LEVELS=yes
REMOVE_SUPERSEDE=yes
TMPDIR=/var/suma/tmp
FIXSERVER_URL=www14.software.ibm.com/webapp/set2/fixget

```

---

To change the general task configuration, we can use the **suma** command with the “-c” and “-a attribute=value” flags to change specific attribute values.

To change the download protocol from FTP to HTTP, issue the following **suma** command shown in Example 4-220.

*Example 4-220 Update the SUMA general task settings*

---

```
root@master:/: suma -c -a DOWNLOAD_PROTOCOL='http'
```

---

This setting can be helpful if you are constrained behind a firewall that only permits outgoing and returning HTTP traffic.

### 4.10.3 SUMA tasks

Example 4-221 shows how to create and execute a SUMA task to download all updates from “5300-00” to the latest. It downloads into the /export/lpp\_suma/5304 directory and only downloading filesets that are not already there. First, we create the 5304 directory in the /export/lpp\_suma file system.

*Example 4-221 Creating a SUMA task for downloading a TL*

---

```
root@master:/: mkdir /export/lpp_suma/5304
root@master:/: suma -wx -a DisplayName='5304' -a FilterML='5300-00'
-a DLTarget='/export/lpp_suma/5304' -a FilterDir='/export/lpp_suma/5304'
```

---

**Note:** In Example 4-221, to download the Technology Level (TL) requested, the RqType needs to be set to TL, and set RqName=5300-04. If the FilterDir parameter is left blank, it defaults to the same value as the DLTarget.

Example 4-221 can also be done through the SMIT suma\_task\_new fastpath. First, we create the directory 5304 in the /export/lpp\_suma file system, if it has not been created already.

**Attention:** The directory used for any SUMA task must exist prior to scheduling or execution of the task. If not, the task will not be accepted and you will get an error message similar to the following:

```
/export/lpp_suma/5304 must be an existing absolute path.
0500-049 The task is not valid.
```

To view all saved SUMA tasks, use the ‘-l’ flag to the **suma** command as in Example 4-222 with one task defined.

*Example 4-222 View defined SUMA tasks*

---

```
root@master:/: suma -l
```

```

1:      DisplayName=5304
      Action=Download
      RqType=Security
      RqName=
      RqLevel=
      PreCoreqs=y
      Ifreqs=y
      Supersedes=n
      ResolvePE=IfAvailable
      Repeats=y
      DLTarget=/export/lpp_suma/5304
      NotifyEmail=root
      FilterDir=/export/lpp_suma/5304
      FilterML=5300-00
      FilterSysFile=/dev/null
      MaxDLSize=-1
      Extend=y
      MaxFSSize=-1

```

---

Example 4-222 can also be done through the SMIT `suma_task` fastpath and then **View All SUMA Tasks**.

To execute a saved SUMA tasks, use the '-x' flag to the `suma` command as in Example 4-223 which starts the first task from Example 4-222.

*Example 4-223 View execute a saved SUMA task*

---

```
root@master:/: suma -x 1
```

---

Example 4-223 can also be done through SMIT using the fastpath `smit suma_task_edit > Execute Now (Do Not Save Changes)`.

To unscheduled, from `cron`, a saved SUMA tasks, add the '-u' flag to the `suma` command as shown in Example 4-224 which removes the first task from Example 4-222.

*Example 4-224 Unscheduling a SUMA task*

---

```
root@master:/: suma -u 1
```

---

Example 4-224 can also be done through the SMIT `suma_task_delete` fastpath and set the "Retain task data for future use?" to "yes".

To delete a saved SUMA tasks, add the '-d' flag to the **suma** command as in Example 4-225 which removes the first task from Example 4-222

*Example 4-225 View a saved SUMA task*

---

```
root@master:/: suma -d 1
```

---

Example 4-225 on page 330 can also be done through the SMIT `suma_task_delete` fast path and set the “**Retain task data for future use?**” to “no”.

### Check monthly for a new technology level (TL)

This task checks monthly for a new specified TL. Since a TL is released approximately every 6 months, a monthly check may be appropriate. For more information on the IBM Service Strategy see “IBM AIX 5L service strategy” on page 349.

**Note:** Currently SUMA uses the previous terminology of ML (Maintenance Level) instead of TL (Technology Level). See “IBM AIX 5L service strategy” on page 349.

You may schedule a SUMA task with a command similar to the one shown Example 4-226 to check monthly (for example, on the 1st of every month at “00:30”) whether a new TL has been released. The scheduling information (-s) is in crontab format (*Minute Hour Day\_of\_month Month Weekday*) for the root user. The “Repeats” field should be set to “y” in order for the system to make monthly checks for the SP. After the SP is found, the task is deleted. If “Repeats” is set to “n”, only a single check would occur before deleting the task. An email notification will be sent with the results of the check, in this case to the root user at the localhost system. A TL or ML must be in the form xxxx-xx, for example: 5300-05 as shown in Example 4-226.

*Example 4-226 How to create and schedule a task that checks monthly for a new TL*

---

```
root@master:/: mkdir /export/lpp_suma/5305
root@master:/: suma -s “30 0 1 * **” -a Action=Preview -a RqType=ML -a RqName=5300-05
-a FilterML=5300-04 -a Repeats=y -a DLTarget='/export/lpp_suma/5305' -a
NotifyEmail="root@localhost"
```

---

This command performs a “Preview” (no download will occur) to check if TL “5300-05” has been released. The “FilterML” setting specifies that the client already has filesets in the “5300-04” level. If “5300-05” has been released, the notification email contains the list of filesets in the TL “5300-05” that would be downloaded or were downloaded.

If you only want to execute the query right away, and not save it nor schedule it, just use the “-s” flag. If “5300-05” is not yet available, the email notification will contain a message similar to “Invalid requested ML level:V530005” as shown in Example 4-227.

*Example 4-227 How to execute a task that downloads a specific new TL*

---

```
root@master:/: suma -x -a RqType=ML -a RqName=5300-05
-a FilterML=5300-04 -a DLTarget='/export/lpp_suma/5305'
```

0500-018 An internet request failed.

0500-014 The fix server responded with the following error condition:  
204|Invalid requested ML level:V530005.

---

**Note:** In Example 4-226 on page 330, and in Example 4-227, the flag RqType=TL could be used since TL is supported on 5300-05 and 5200-09.

You may also elect to automatically download the filesets in this TL by setting Action equal to “Download” instead of “Preview” as in the Example 4-226 on page 330. In this case, the filesets will only be downloaded, but no installation will occur.

### Check and download a new Concluding Service Pack (CSP)

This task check weekly for a new specified CSP – Since CSPs will be available shortly after a new TL is released and will contain all fixes for highly pervasive, critical, or security related issues for the previous TL, but it may also contain fixes from the newly released TL that fall into these categories. Therefore, a CSP will contain a very small subset of service that was just released as a part of a new TL.

An CSP must be in the form xxxx-xx-CSP, for example: “5300-04-CSP”.

Example 4-228 on page 331 shows a scheduled SUMA task check monthly (for example, on the 1st of every month at “00:30”) whether a new TL has been released. The scheduling information (-s) is in crontab format (*Minute Hour Day\_of\_month Month Weekday*) for the root user. The “Repeats” field should be set to “y” in order for the system to make monthly checks for the SP. After the SP is found, the task is deleted. If “Repeats” is set to “n”, only a single check would occur before deleting the task. An email notification will be sent with the results of the check, in this case to the root user at the localhost system. The “FilterML” setting specifies that the client already has filesets in the “5300-04” level.

*Example 4-228 Task that checks monthly for a new CSP*

---

```
root@master:/: mkdir /export/lpp_suma/530004
root@master:/: suma -s “30 0 1 * *” -a Action=Preview -a RqType=SP
```

```
-a RqName=5300-04-CSP -a FilterML=5300-04 -a Repeats=y
-a DLTarget='/export/lpp_suma/530004' -a NotifyEmail="root@localhost"
```

0500-018 An internet request failed.

0500-014 The fix server responded with the following error condition:  
204|Invalid requested ML level:5300-04-CSP.

---

If “5300-04-CSP” is not yet available, the email notification will contain a message similar to “Invalid requested ML level:V5300-04-CSP”, as in Example 4-228.

### Check and download a new Service Pack (SP)

This task check weekly for a new specified SP – Since SPs are expected to be released approximately every 4-6 weeks, between TLs that are released twice a year. An SP must be in the form xxxx-xx-xx, for example: “5300-04-03” as in Example 4-229. The “FilterML” setting specifies that the client already has filesets in the “5300-04” level. Remember that a SP is just a tested group of PTF’s, see “IBM AIX 5L service strategy” on page 349.

**Note:** Example 4-229 shows a one time execution to check and download a new service pack.

*Example 4-229 How to execute a task that downloads a specific new SP*

---

```
root@master:/: mkdir /export/lpp_suma/5304
root@master:/: suma -x -a RqType=SP -a RqName=5300-04-03
-a FilterML=5300-04 -a DLTarget='/export/lpp_suma/5304'
-a FilterDir='/export/lpp_suma/5304'

*****
Performing preview download.
*****

Download SKIPPED:   bos.64bit.5.3.0.41.bff
Download SKIPPED:   bos.adt.include.5.3.0.42.bff
Download SKIPPED:   bos.adt.prof.5.3.0.43.bff
Download SKIPPED:   bos.mp.5.3.0.43.bff
Download SKIPPED:   bos.mp64.5.3.0.43.bff
...
Download SUCCEEDED: /export/lpp_suma/5304/installp/ppc/bos.rte.aio.5.3.0.41.bff
...
Summary:
    61 downloaded
    0 failed
    9 skipped
```

---

## Downloading the latest security fixes monthly

This task create and schedule the download of the latest security fixes monthly, starting from ML “5300-00” (FilterML) using the /export/lpp\_suma/security directory. The scheduling information (-s) is in crontab format (*Minute Hour Day\_of\_month Month Weekday*) for the root user. In Example 4-230, it is scheduled for the 1st of every month at “00:30”, just after midnight, and the request type (RqType) is “Security”.

*Example 4-230 Task that downloads the latest security fixes monthly*

---

```
root@master:/: mkdir /export/lpp_suma/security
root@master:/: suma -s "30 0 1 * *" -a RqType=Security -a DisplayName="Monthly
Security Fixes" -a DLTarget='/export/lpp_suma/security' -a FilterML='5300-00'
-a FilterDir='/export/lpp_suma/security'
```

---

## Check and download a specific APAR once a week

To create and schedule a task that checks for a specific APAR once a week, and then when it is available download it and email that it has done so, you can refer to Example 4-231. The “Repeats” field should be set to “y” in order for the system to make weekly checks for an APAR. After the APAR is found, the task is deleted. If “Repeats” is set to “n”, only a single check would occur before deleting the task. An email notification will be sent with the results of the check, in this case to the root user at the localhost system.

*Example 4-231 Task that checks and downloads a specific APAR*

---

```
root@master:/: mkdir /export/lpp_suma/apar
root@master:/: suma -s "0 3 * * 4" -a RqType=APAR -a RqName=IY12345 -a
NotifyEmail="root@localhost" -a Repeats=y -a DLTarget='/export/lpp_suma/apar'
-a FilterDir='/export/lpp_suma/apar'
```

---

If you only want to execute the query right away, and not save it nor schedule it, just replace the “-s” flag with “-x”. If the APAR was not found, the “0500-035” message will be displayed. See Example 4-232.

*Example 4-232 How to execute a task that checks and downloads a specific APAR*

---

```
root@master:/: suma -x -a RqType=APAR -a RqName=IY12345
-a DLTarget='/export/lpp_suma/apar' -a FilterDir='/export/lpp_suma/apar'
```

---

0500-035 No fixes match your query.

---

## 4.10.4 Maintenance models

The following three maintenance models are recommendations based on the new IBM AIX 5L Service Strategy. For more information see “IBM AIX 5L service strategy” on page 349.

### **Maximum stability**

The “Maximum Stability” maintenance model should monitor when a Concluding Service Pack (CSP) is released, see “IBM AIX 5L service strategy” on page 349. When the CSP comes out, it should be downloaded together with the previous Technology Level (TL).

In the “Maximum Stability” maintenance model you are not moving to the latest TL when it is released, and you do not have to regularly check for the release of a TL or Concluding Service Pack (CSP). It is sufficient to check monthly for a new CSP – Since a CSP is released approximately every 6 months, a monthly check may be appropriate.

It is recommended to perform the following tasks for handling the “Maximum Stability” maintenance model:

1. Create a specific file system or directory for the CSP with corresponding TL.
2. Download the TL and then the CSP to the new file system/directory.
3. Create a new NIM lpp\_source based on the base level and adding the TL/CSP filesets.
4. Check the new NIM lpp\_source for duplicates with the `lppmgr nim` operation or standalone command, see “Removing duplicate filesets from a NIM lpp\_source directory” on page 340.
5. Create a NIM spot based on the new NIM lpp\_source.

Steps 1, 3, and 5 are covered in other sections of this book 3.1, “Setting up a basic NIM environment” on page 50), and step 4 is shown in “Removing duplicate filesets from a NIM lpp\_source directory” on page 340.

### ***Download the TL and the CSP***

To check monthly if a new CSP has been released for the next highest TL, you may schedule a SUMA task with a command similar to the one in Example 4-233 on page 335 (for example, on the 15th of every month after midnight 00:01). When it is successful an email notification will be sent to the email address specified in for the `NotifyEmail` attribute, with the results of the check, in this case `root@localhost`.



---

*Example 4-233 SUMA task to check for a coming CSP*

---

```
root@master:/: suma -s "1 0 15 * *" -a Action=Preview -a RqType=SP -a
RqName=5300-04-CSP -a FilterML=5300-04 -a Repeats=y -a NotifyEmail="root@localhost"
-a DLTarget=/export/lpp_suma/530004
```

---

Task ID 5 created.

---

The `suma` command in Example 4-233 returns a SUMA task ID (in this case #5) which can be used to perform the actual download of the task. We will perform this step after we have downloaded the corresponding TL which is shown in Example 4-234.

---

*Example 4-234 SUMA task to download a TL for a newly released CSP*

---

```
root@master:/: suma -x -a RqType=ML -a RqName=5300-04 -a FilterML=5300-03 -a
DLTarget=/export/lpp_suma/530004
Extending the /export/lpp_suma filesystem by 194690 blocks.
Download SUCCEEDED: /export/lpp_suma/530004/installp/ppc/bos.suma.5.3.0.40.bff
. . .
Summary:
    225 downloaded
     0 failed
    32 skipped
```

---

The command in Example 4-234 only downloads the TL 5300-04, but it contains all updates since the base level since each TL is cumulative. The first output line shows that the file system size was automatically increased (Extending).

After the TL is downloaded, it is time to download the CSP itself. The command in Example 4-235 could be used to download the previously scheduled task #5 for the 5300-04-CSP that had the “Preview” action defined to check for its release. The “Preview” will have to be overridden with “Download” or it will just perform another preview.

---

*Example 4-235 SUMA task to download newly released CSP*

---

```
root@master:/: suma -x -a Action=Download 5
Download SUCCEEDED: /export/lpp_suma/530004/installp/ppc/bos.mp.5.3.0.43.bff
. . .
Summary:
    76 downloaded
     0 failed
     0 skipped
```

---

Now we are ready to create a NIM lpp\_source from the TL+CSP to use for updates or rte installation.

**Note:** With the default setting for SCREEN\_VERBOSE, the messages will be the same for the “Download” and “Preview” actions, but the download will only be performed when “Download” is specified.

### **Create a NIM lpp\_source object with the base+TL+CSP**

To use a NIM lpp\_source based on GA level AIX 5L, updated with the Technology Level (TL) and its corresponding Concluding Service Pack (CSP), create a new NIM lpp\_source from a GA base level NIM lpp\_source.

Example 4-236, Example 4-237, and Example 4-238 show how the source attribute is set either to the NIM lpp\_source name of a GA base level NIM lpp\_source, or to a directory containing the Backup File Format (BFF) filesets. Example 4-236 shows how to use the NIM lpp\_source name of a GA base level as a source. In this case, the source NIM lpp\_source name is lpp5300.

#### *Example 4-236 Replicating a base level lpp\_source*

---

```
root@master:/: nim -o define -t lpp_source -a server=master -a source=lpp5300
-a location=/export/lpp_source/lpp5304 lpp5304
```

---

Example 4-237 shows how to use the GA base level BFF fileset directory as a source. In this case, the /export/lpp\_source/lpp5300/installp/ppc directory.

#### *Example 4-237 Replicating a base level lpp\_source from a directory*

---

```
root@master:/: nim -o define -t lpp_source -a server=master
-a location=/export/lpp_source/lpp5304
-a source=/export/lpp_source/lpp5300/installp/ppc lpp5304
```

---

We have downloaded AIX 5L V5.3 Technology Level (TL) 04 from IBM support fix central as shown in “Download the TL and the CSP” on page 334. To add this directory to the NIM master environment as a NIM lpp\_source resource, we do not have to declare the source since it is in the install format. See “File system hierarchy” on page 51.

#### *Example 4-238 Creating a NIM lpp\_source for a Technology Level (TL) update only*

---

```
root@master:/: nim -o define -t lpp_source -a server=master
-a location=/export/lpp_suma/530004 TL5304
```

---

Preparing to copy install images (this will take several minutes)...

Now checking for missing install images...

```
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
    bos.net
    bos.diag
    bos.sysmgt
    bos.terminfo
    bos.terminfo.all.data
    devices.graphics
    devices.scsi
    devices.tty
    x1C.rte
    bos.mp
    devices.common
    bos.64bit
```

---

The “0042-267” message in Example 4-238 warns us that the “simages” attribute were not set during creation, and therefore this NIM lpp\_source can not be used for NIM “rte” installation, but it can be used for other operations such as an update, which is what we want to accomplish.

Before using the new TL+CSP NIM lpp\_source, we check and remove duplicate filesets with the `nim` command and `lppmgr` operation as shown in Example 4-239. See also “Removing duplicate filesets from a NIM lpp\_source directory” on page 340.

*Example 4-239 Check and remove duplicate filesets after updating with TL+CSP*

---

```
root@master:/: nim -o lppmgr TL5304
lppmgr: Source table of contents location is
/export/lpp_suma/TL530004/installp/ppc/.toc

lppmgr: Building table of contents in /export/lpp_suma/TL530004/installp/ppc ..
lppmgr: Building table of contents completed.
lppmgr: Generating duplicate list..
lppmgr: Generating base level duplicate list..
```

Results:

```
===== start list =====
bos.64bit.5.3.0.40.U
bos.adt.include.5.3.0.41.U
bos.adt.prof.5.3.0.41.U
```

```

bos.adt.prof.5.3.0.42.U
bos.diag.com.5.3.0.10.U
bos.diag.rte.5.3.0.10.U
bos.diag.util.5.3.0.10.U
bos.mp.5.3.0.41.U
bos.mp.5.3.0.42.U
bos.mp64.5.3.0.41.U
bos.mp64.5.3.0.42.U
. . .
===== end list =====

```

```

lppmgr: Building table of contents in /export/lpp_source/lpp5304/installp/ppc .
lppmgr: Building table of contents completed.
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.64bit.5.3.0.40.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.adt.include.5.3.0.41.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.adt.prof.5.3.0.41.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.adt.prof.5.3.0.42.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.diag.com.5.3.0.10.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.diag.rte.5.3.0.10.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.diag.util.5.3.0.10.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.mp.5.3.0.41.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.mp.5.3.0.42.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.mp64.5.3.0.41.U
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.mp64.5.3.0.42.U
. . .

```

In Example 4-239, some filesets were found to be duplicated. By default the `nim` command `lppmgr` operation removes duplicate filesets.

Our new GA base level NIM `lpp_source` can now be updated with the TL+CSP filesets. In Example 4-240, we use the NIM `lpp_source` fileset directory of the TL+CSP as the `source`.

*Example 4-240 Updating base level `lpp_source` from a TL directory*

```

root@master:/: nim -o update -a show_progress=yes -a packages=all -a source=TL5304
lpp5304

/export/lpp_source/lpp5304/installp/ppc/sysmglib.libraries.apps.5.3.0.40.U
/export/lpp_source/lpp5304/installp/ppc/sysmglib.framework.core.5.3.0.40.U
/export/lpp_source/lpp5304/installp/ppc/sysmglib.websm.apps.5.3.0.40.U
/export/lpp_source/lpp5304/installp/ppc/sysmglib.websm.rte.5.3.0.40.U
/export/lpp_source/lpp5304/installp/ppc/sysmglib.websm.icons.5.3.0.40.U
Filesystem size changed to 11993088
Inlinelog size changed to 23 MB.

```

. . .

---

In Example 4-240, you can see that both the filesystem and inline log sizes were changed during the NIM lpp\_source update. After updating it, it is also a good idea to check for duplicate filesets with the `nim` command and `lppmgr` operation. See “Removing duplicate filesets from a NIM lpp\_source directory” on page 340.

*Example 4-241 Removing duplicates with nim lppmgr operation*

---

```
root@master:/: nim -o lppmgr TL5304
lppmgr: Source table of contents location is
/export/lpp_suma/530004/installp/ppc/.toc

lppmgr: Building table of contents in /export/lpp_suma/530004/installp/ppc ..
lppmgr: Building table of contents completed.
lppmgr: Generating duplicate list..
lppmgr: Generating base level duplicate list..
```

**Results:**

No filesets found that can be removed.

---

In Example 4-241, no filesets were found to be duplicated. The NIM lpp\_source is now ready to be used for installing and updating NIM clients using the “Maximum Stability” maintenance model.

## Yearly update

The “Yearly Update” model is for systems that are stable and which are planned to only be updated annually. This model shows an annual move to a new TL, and thereby skips the direct move to one of the two TL’s that are released during the year, instead picking up this function when the next TL is installed, as TL’s are cumulative.

In between TLs, SPs, individual PTF’s, and interim fixes (to address security and other issues) can be utilized to maintain service on a TL for up to one year.

When moving to a new TL annually, it is recommended to utilize a first half TL (for example, TL4, TL6, etc.) because it has the advantage of being a smaller release. When installing a first half TL (for example, TL6) it will contain all the new AIX 5L software enhancements that were released with the previous TL (for example, TL5), however TL5 will have already been in the field for six to eight months, allowing it to become more stable.

For examples of how to download TLs, see “Check monthly for a new technology level (TL)” on page 330 and for examples of how to download SPs, see “Check and download a new Service Pack (SP)” on page 332.

### Latest level maintenance model

The “Latest Level” model is for new systems or hardware upgraded systems that requires a new TL, or who wish to utilize new hardware or software features being introduced in a TL. This maintenance model would typically entail a move to a new TL shortly after its release.

The annual first half TL (for example, TL4, TL6, etc.) is a smaller one since it is restricted to hardware features and enablement, and software service. The annual second half TL (for example, TL5) also includes new software features, and thus will be a larger release.

In between Technology Levels (TLs) both Service Packs (SPs) and interim fixes (to address security and other issues) can be utilized in support of the current Technology Level.

For examples of how to download TLs, see “Check monthly for a new technology level (TL)” on page 330, for examples of how to download SPs, see “Check and download a new Service Pack (SP)” on page 332.

## 4.10.5 Removing duplicate filesets from a NIM lpp\_source directory

Sometimes you will end up with duplicate filesets in the same directory. This can cause problems when installing and updating SPOTs or clients. These duplicates are just a waste of space, and they should be removed.

This can be done when downloading filesets using SUMA to a specific directory, but if you move around filesets you need to take care of this manually. Before AIX 5L V5.3, this had to be done using customized scripts, but now it can be done with the `lppmgr` command or `lppmgr` operation with the `nim` command.

The SUMA global settings `REMOVE_CONFLICTING_UPDATES` and `REMOVE_DUP_BASE_LEVELS` will by default remove duplicate updates and base levels when downloading updates. See Figure 4-219 on page 327.

To make sure that there are no duplicate filesets use the `lppmgr` command/operation to cleanup. This can be done either with the standalone command `/usr/lib/inst1/lppmgr`, that can be used to check any directory containing installable filesets or using the `nim` command and `lppmgr` operation.

In Example 4-242 on page 341 and Example 4-243 on page 342, we have an `lpp_source` named `lpp5304` which contain the AIX 5L V5.3 TL04. We have after

this release downloaded updates to AIX 5L V5.3 to the directory  
/exports/lpp\_suma/5304U.

In the following **lppmgr** command in Example 4-242 on page 341, the directory  
/exports/lpp\_suma/5304U will be checked and all duplicate updates (-u) and  
base levels (-b) will be removed (-r).

```
/usr/lib/instl/lppmgr -d /exports/lpp_suma/5304U -rub
```

The same applies for the NIM **lpp\_source** object **lpp5304** using the **nim** command  
and **lppmgr** operation:

```
nim -o lppmgr -a lppmgr_flags=rub lpp5304
```

**Note:** The **lppmgr** command use the path to the directory where the filesets  
are located, and the location attribute of the NIM **lpp\_source** object is only part  
of this path.

Example 4-242 on page 341 and Example 4-243 on page 342 show the usage of  
the **lppmgr** command to remove duplicate filesets from a directory, and  
Example 4-243 shows the same operation on the same directory, but using the  
**nim** command and the NIM object name for the **lpp\_source**.

*Example 4-242 Using lppmgr command to clean up and remove duplicate filesets*

```
root@master:/: /usr/lib/instl/lppmgr -d /export/lpp_source/lpp5304/installp/ppc -rub
```

```
lppmgr: Source table of contents location is
/export/lpp_source/lpp5304/installp/ppc/.toc
lppmgr: Building table of contents in /export/lpp_source/lpp5304/installp/ppc ..
lppmgr: Building table of contents completed.
lppmgr: Generating duplicate list..
lppmgr: Generating base level duplicate list..
```

Results:

```
===== start list =====
bos.acct.5.3.0.30.bff
bos.adt.syscalls.5.3.0.40.bff
bos.clvm.enh.5.3.0.30.bff
bos.net.ewlm.rte.5.3.0.30.bff
bos.net.mobip6.rte.5.3.0.10.bff
bos.net.nfs.adt.5.3.0.30.bff
bos.net.nfs.cachefs.5.3.0.30.bff
bos.net.nfs.server.5.3.0.10.bff
bos.net.tcp.server.5.3.0.30.bff
bos.perf.diag_tool.5.3.0.40.bff
===== end list =====
```

```

rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.acct.5.3.0.30.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.adt.syscalls.5.3.0.40.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.clvm.enh.5.3.0.30.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.net.ewlm.rte.5.3.0.30.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.net.mobip6.rte.5.3.0.10.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.net.nfs.adt.5.3.0.30.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.net.nfs.cachefs.5.3.0.30.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.net.nfs.server.5.3.0.10.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.net.tcp.server.5.3.0.30.bff
rm: removing /export/lpp_source/lpp5304/installp/ppc/bos.perf.diag_tool.5.3.0.40.bff
lppmgr: Building table of contents in /export/lpp_source/lpp5304/installp/ppc ..
lppmgr: Building table of contents completed.

```

---

In the output from Example 4-242, you can see the two steps of the **lppmgr** command. The first step is to find and filter, the second step is the action on the filtered filesets, in this case remove. Example 4-243 on page 342 shows you how the report should look when all the filesets are unique in the NIM lpp\_source, and there are no duplicates.

#### *Example 4-243 Cleaning up and removing duplicate filesets*

---

```

{nimmast}:/ # nim -o lppmgr -a lppmgr_flags=rub lpp5304
lppmgr: Source table of contents location is
/export/lpp_source/lpp5304/installp/ppc/.toc
lppmgr: Building table of contents in /export/lpp_source/lpp5304/installp/ppc ..
lppmgr: Building table of contents completed.
lppmgr: Generating duplicate list..
lppmgr: Generating base level duplicate list..

```

Results:

No filesets found that can be removed.

---

### 4.10.6 Check if a system has all available updates installed

To find out if a particular system needs a Technology Level (TL), Concluding Service Pack (CSP) or Service Pack (SP), see the “IBM AIX 5L service strategy” on page 349. You will get the most details by using the new **niminv** command (AIX 5L V5.3 TL5) for NIM machines or NIM lpp\_source objects, and the **compare\_report** command for installed systems.

But lets start with the simplest steps to check a system’s software levels by using the **lslpp**, the **oslevel** and even the **instfix** command as shown in Example 4-244.



To find out if a particular system needs a Technology Level (TL), Concluding Service Pack (CSP) or Service Pack (SP) see “IBM AIX 5L service strategy” on page 349. You can start by using the `oslevel` command either with the “-s” or the “-r” flags, depending on your current system level. The “-r” option shows the recommended ML/TL, but the “-s” flag also shows the SP information.

---

*Example 4-244 Examples of using the `lspp`, `oslevel` and `uname` commands*

---

```
root@master:/: lspp -qLc bos.rte | cut -f2-3 -d:
bos.rte:5.3.0.40
```

```
root@master:/: oslevel -r
5300-04
```

```
root@master:/: oslevel -s
5300-04-03
```

```
root@master:/: oslevel -g 5.3.0.0
Fileset                               Actual Level      Maintenance Level
-----
bos.rte                                5.3.0.40         5.3.0.0
```

```
root@master:/: uname -s -v -r
AIX 3 5
```

---

In the `oslevel` command output in Example 4-244, you can see that there are some discrepancies between the “Actual Level” and what `oslevel` report are the current level. To go into details of what is missing for a higher `oslevel` report, you can use the `instfix` command to check for the installed AIX levels as shown in Example 4-245.

---

*Example 4-245 Using `instfix` to find current software level*

---

```
root@master:/: instfix -i | grep AIX
  All filesets for 5300-02_AIX_ML were found.
  All filesets for 5.3.0.0_AIX_ML were found.
  All filesets for 5300-01_AIX_ML were found.
  All filesets for 5300-03_AIX_ML were found.
  All filesets for 5300-04_AIX_ML were found.
  Not all filesets for 5300-05_AIX_ML were found.
```

---

In Example 4-246, we notice that the 5300-05 are missing some filesets, since it reports “Not all filesets for 5300-05\_AIX\_ML were found”, so we use this as our next search argument with the `instfix` command to find out what is missing.

*Example 4-246 Using instfix to find missing software level*

---

```
root@master:/: instfix -icqk 5300-05_AIX_ML|grep ":-:"
5300-05_AIX_ML:sysmgplib.framework.core:5.3.0.50:5.3.0.40:-:AIX 5300-05 Update
5300-05_AIX_ML:sysmgplib.libraries.apps:5.3.0.50:5.3.0.40:-:AIX 5300-05 Update
```

---

Two filesets were on a lower level, so we can either remove these filesets if they are unused, or update them as well.

### The niminv command

With AIX 5L V5.3 TL5, the new **niminv** command is introduced. The **niminv** command can gather, conglomerate, compare, and download fixes based on installation inventory of NIM objects.

The **niminv** command extends the functionality of the **compare\_report** command, to operate on several NIM objects such as `machines` and `lpp_sources` at the same time. The **niminv** command use the **suma** command for downloading fixes from the IBM Support Fix Central, and the **geninv** command to collect software inventory information from other systems.

**Note:** Conglomerate means that the inventories are put together without repeats of any filesets; the default action removes any higher versions of the fileset, so only the lowest version of each fileset appears in the conglomerated list.

The **geninv** command can also be used manually to collect software inventory information from other systems using IP addresses or resolvable hostnames.

You can also use the SMIT `nim_inventory` fast path to perform **niminv** operations.

Example 4-247 with the **niminv** command uses the `invget` operation and will collect an output file (**1s1pp -L** format) in the `/export/inventory` directory with the current software levels of the NIM master.

*Example 4-247 Using niminv to gather software inventory information*

---

```
root@master:/: niminv -o invget -a targets=master -a location=/export/inventory
Installation Inventory for master saved to
/export/inventory/inventory.master.060620140643.
```

Return Status = SUCCESS

---

Example 4-248 also uses the `invget` operation, but collects software levels from a group of NIM clients, called `nim_clients`, but currently only containing one NIM client (`lpar55`).

---

*Example 4-248 Software inventory information for a NIM group of clients*

---

```
root@master:/: niminv -o invget -a targets=nim_clients -a location=/export/inventory
Installation Inventory for lpar55 saved to
/export/inventory/inventory.lpar55.060620140605.
```

Return Status = SUCCESS

---

The `niminv` command with the `invcon` operation (conglomerate) compares the levels between different systems. Refer to Example 4-249.

---

*Example 4-249 Software inventory information between different NIM clients*

---

```
root@master:/: niminv -o invcon -a targets=master,nim_clients -a base=highest
-a location=/export/inventory
Conglomerate of lpar55:master save to
/export/inventory/conglomerate.1.lpar55:master.060620140607.
```

Return Status = SUCCESS

---

To display the fixes that can be downloaded based on the lowest installations in a conglomerate of the `nim_clients` group and the NIM master, use the `fixget` operation as shown in Example 4-250. The “Performing preview download” indicates that the following “Download SUCCEEDED” did not take place.

---

*Example 4-250 Using `niminv` to display available fixes*

---

```
root@master:/: niminv -o fixget -a targets=master,nim_clients
*****
Performing preview download.
*****
Download SUCCEEDED: /export/lpp_suma/installp/ppc/Java14.debug.1.4.1.0.bff
Download SUCCEEDED: /export/lpp_suma/installp/ppc/Java14.debug.1.4.1.7.bff
. . .
Summary:
    271 downloaded
     0 failed
     0 skipped
```

Return Status = SUCCESS

---

Example 4-251 shows the download of the latest fixes for the NIM master to a new `lpp_source`. The actual download will be performed with the `suma` command.

*Example 4-251 Using niminv to download available fixes*

---

```
root@master:/: niminv -o fixget -a targets=master -a download=yes -a
location=/export/lpp_source/lpp5304B -a newlppname=lpp5304B
```

```
Download SUCCEEDED: /export/lpp_source/lpp5304B/installp/ppc/
```

```
. . .
```

```
Summary:
```

```
    271 downloaded
     0 failed
     0 skipped
```

```
Return Status = SUCCESS
```

---

The `niminv` command will create both the `location` and the `newlppname` if they do not exist. If the `niminv` command is interrupted, it will perform a cleanup and remove all that was created, including the `location` and `newlppname`, giving you the following messages:

```
    niminv: Signal received. Cleaning up.
    Return Status: FAILURE
```

To download the latest fixes to an existing `lpp_source` while filtering from the same `lpp_source`, see Example 4-252.

*Example 4-252 Using niminv to download available fixes*

---

```
root@master:/: niminv -o fixget -a targets=master -a download=yes -a
lpp_source=lpp5305
```

```
Extending the /AIX5305 filesystem by 979890 blocks.
```

```
Filesystem size changed to 5242880
```

```
Inline log size changed to 10 MB.
```

```
Download SUCCEEDED: /AIX5305/installp/ppc/Java14.msg.ko_KR.1.4.2.0.bff
```

```
. . .
```

```
Summary:
```

```
    271 downloaded
     0 failed
     0 skipped
```

```
Return Status = SUCCESS
```

---

First the **niminv** command extends the NIM lpp\_source objects file system and then starts the download of new filesets.

**Note:** Any filesets already contained in the destination directory will not be downloaded again.

## The **compare\_report** command

The **compare\_report** command provides an easy way to check if a specific system are at the latest maintenance level or latest level. A comparison can be done between the filesets installed on a standalone system and the contents of an image repository or a list of fixes available from the IBM Support Fix Central Web site to determine the fixes that need to be downloaded.

The **compare\_report** command provides the following functions:

- ▶ Compares the filesets installed on a system to a list of fixes available from the IBM Support Web site or a fix repository, such as a NIM lpp\_source.
- ▶ Generates reports detailing which fixes are required for a system to be at the latest level or latest maintenance level.
- ▶ Reports may be uploaded to the “Compare report” page (select based on your desired OS level) on the IBM Support Fix Central Web site:

(<http://www.ibm.com/eserver/support/fixes/FixReleaseInfo.jsp?system=2&release=5.3>)

The fixes may then be downloaded utilizing normal Web site interfaces or using SUMA, see “Settings for SUMA” on page 325. (You need to change the value for the “release” attribute to change AIX release.)

The **compare\_report** command can be used on systems that do not have access to the Internet. This prevents systems from using SUMA which requires at least one system to have Internet access.

You can also use the SMIT `compare_report` fast path to perform **compare\_report** operations.

After downloading the LatestFixData file from the eSupport web site, this file can then be sent to a system to perform an "offline" comparison to determine which software updates are needed to bring the system or an NIM lpp\_source to the latest level or latest ML/TL. The reports produced by **compare\_report** can be uploaded to the IBM Support Fix Central Web site to request the updates to be downloaded.

To check if a specific system is back level compared to what is available from the IBM Support Fix Central Web site, manually download the fileset list from:

www-912.ibm.com

We do it here by using the Open Source **wget** command in Example 4-253.

The file (in our example, LatestFixData53) contains a list of fixes that are in the latest maintenance package as well as the latest available fixes that have been released after the latest maintenance package. The file can also be found using a browser on the IBM Support Fix Central Web site by selecting pSeries (**Server**), AIX OS (**Product or fix type**), Fix release information (**Option**), and an OS level.

*Example 4-253 Downloading LatestFixData53 from www-912.ibm.com*

---

```

root@master:/: cd /tmp
root@master:/tmp: wget
http://www-912.ibm.com/eserver/support/fixinfo/download?file=LatestFixData53
--17:27:01--
http://www-912.ibm.com/eserver/support/fixinfo/download?file=LatestFixData53
=> `download?file=LatestFixData53.1'
Resolving www-912.ibm.com... 129.42.160.32
Connecting to www-912.ibm.com[129.42.160.32]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-servicereport]

[    <=>                               ] 46,118          86.11K/s

17:27:02 (85.98 KB/s) - `download?file=LatestFixData53.1' saved [46118]

```

---

Then, compare the downloaded file from IBM with your locally installed software using the **compare\_report** command or the SMIT `instolist_compare` fast path as shown in Example 4-254.

*Example 4-254 compare\_report with LatestFixData53 and local system*

---

```

root@master:/tmp: compare_report -s -r LatestFixData53 -l -h
#(lowerthanlatest1.rpt)
#Installed Software that is at a LOWER level
#PTF:Fileset_Name:Installed_Level:Available_Level
U487072:Java14.sdk:1.4.2.20:1.4.2.75
. . .
#(higherthanmaint.rpt)
#Installed Software that is at a HIGHER level than
# the latest maintenance level
#Fileset_Name:Installed_Level:Available_Level
bos.64bit:5.3.0.41:5.3.0.40

```

. . .

The files `lowerthanlatest1.rpt` and `higherthanmaint.rpt` are saved in `/tmp` directory by default (can be changed with the `-t <directory>` flag). They each contain a list of filesets/PTFs that are either lower than or higher than the latest ML/TL.

The Fix release information section of IBM Support Fix Central Web site allows you to upload a file that has been generated by the `compare_report` command, like the file shown in the Sample Report Output - disk file section above.

The PTF numbers contained in the comparison report file will be used to provide the requested fixes. After uploading the file, you can download the requested fixes (PTF's) through the normal Fix Central interfaces, using the `suma` command or a file transfer tool such as `wget`.

### ***The wget Open Source tool***

If you do not have the `wget` command, you can use another tool or download it using a web browser or install the WGET RPM package from the IBM Open Source site (as shown in Example 4-255):

```
ftp.software.ibm.com
```

For more information on how to use the `rpm` command or the IBM Open Source FTP site, refer to the *Linux Applications on pSeries redbook*, SG24-6033-01 located at:

```
http://www.redbooks.ibm.com/abstracts/sg246033.html
```

#### ***Example 4-255 Installing Open Source wget from ftp.software.ibm.com***

```
root@master:/: rpm -iv ftp://ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/RPMS/ppc/wget/wget-1.9.1-1.aix5.1.ppc.rpm
Retrieving ftp://ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/RPMS/ppc/wget/wget-1.9.1-1.aix5.1.ppc.rpm
wget-1.9.1-1
```

```
root@master:/: wget
wget: missing URL
Usage: wget [OPTION]... [URL]...
```

```
Try `wget --help' for more options.
```

## **4.10.7 IBM AIX 5L service strategy**

To understand how to use SUMA, and for that matter all methods of downloading upgrades from IBM, it is essential to understand IBM software program update semantics.

**Important:** For detailed information about the IBM AIX 5L Service Strategy please go to the IBM Web site:

[http://www.ibm.com/servers/eserver/support/unixservers/aix\\_service\\_strategy.html](http://www.ibm.com/servers/eserver/support/unixservers/aix_service_strategy.html)

And download the IBM AIX 5L Service Strategy document in PDF format:

[http://www-03.ibm.com/servers/eserver/support/unixservers/aix\\_service\\_strategy.pdf](http://www-03.ibm.com/servers/eserver/support/unixservers/aix_service_strategy.pdf)

A Problem Management Record (PMR) is a IBM tracking record for customer-reported problems.

**PTF** A Program Temporary Fix (PTF) provides a fix to a reported defect. The fix is temporary; the fix disappears when it is incorporated into the next release of the product. PTFs might contain a single fix, but generally contain multiple fixes and are associated with a single fileset. PTFs filesets are prefixed with “U” followed by six digits.

**APAR** An Authorized Program Analysis Report (APAR) associates a fix to a PMR. You can then use the APAR number to obtain the required fix. When documenting software requirements, it's best to list the APAR number rather than the PTF or PMR number. You will always be able to determine if an APAR is installed on your system using the command `instfix -ivk APAR_NUMBER`, whereas installed PTFs are not trackable. APARs are prefixed with the characters “IY” followed by five digits.

APARs and PTFs are tightly coupled in that PTFs contain multiple APAR fixes. An APAR is a single fix that is delivered using a PTF packaging.

**Note:** To separate different distributions of the same installable software filesets, IBM AIX use four separated numbers known as the VRMF (Version.Release.Maintenance/Modification.Fix). For example, `bos.rte 5.3.0.45` is a newer fileset than `bos.rte 5.3.0.40`.

In 2006, as part of the new AIX 5L Service Strategy, what was known as a Maintenance Level (ML) were replaced by a Technology Level (TL).

**TL** A Technology Level (TL) is the new term for the twice yearly AIX 5L releases, which contain new hardware and software features and service updates. The first TL each



year will be restricted to hardware features and enablement, as well as software service. The second TL will include hardware features and enablement, software service, and new software features. A TL is supposed to be installed completely or not at all.

**Important:** Prior to installing a TL the system should be backed up.

In addition to TLs there are also the following PTF distributions to consider:

- SP A Service Pack (SP) consists of PTFs (service-only updates) that are released between TLs that are grouped together for easier identification. These fixes address highly pervasive, critical, or security-related issues. Service Packs are provided for the N and N-1 releases (for example, AIX 5L V5.3 and AIX 5L V5.2) on the latest TL for each release (for example, 5300-04 and 5200-08).
- CSP A Concluding Service Pack (CSP) is the last SP for a TL. The CSP contains fixes for highly pervasive, critical, or security-related issues just like a Service Pack, but it might also contain fixes from the newly released TL that fall into these categories. Therefore, a CSP contains a very small subset of service that was just released as a part of a new TL.

**Important:** CSPs allow for extended service on a Technology Level through the utilization of interim fixes.

The term *interim fix* is used as a replacement for "emergency fix" or "efix". While the term emergency fix is still applicable in some situations (a fix given in the middle of the night with minimal testing), the term interim fixes is more descriptive in that it implies a temporary state until an update can be applied that has been through more extensive testing.

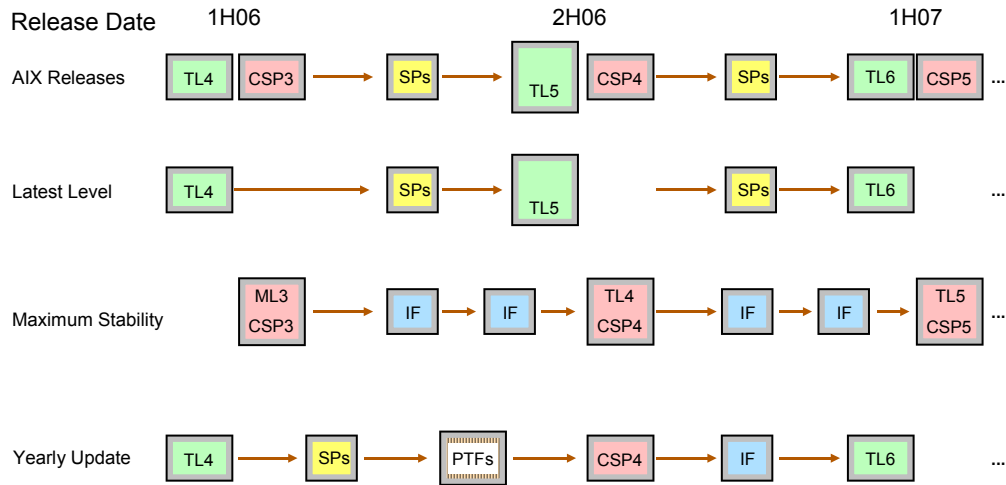


Figure 4-15 IBM AIX 5L service strategy schema

#### 4.10.8 IBM support fix central Web site

The following is a subset of a simple screen shot from the IBM Support Fix Central Web site (as shown in Figure 4-16) at:

<http://www.ibm.com/eserver/support/fixes/>

You can download TLs, SPs, CSPs, PTFs, APARs and interim fixes from this site. However, sometimes you must make a service call to IBM support to obtain specific interim fixes or PTFs.

Use the `oslevel -s` command to determine the current operating system level. **Note:** "-s" is a new and not available on all machines.

Technology Levels and Service Packs				
Type	Name	Date	Prereqs	Description
SP	<a href="#">5300-04-03</a>	May, 2006	TL 5300-04	SW fixes
SP	<a href="#">5300-04-02</a>	April, 2006	TL 5300-04	SW fixes
CSP	<a href="#">5300-03-CSP</a>	March, 2006	ML 5300-03	SW fixes
SP	<a href="#">5300-04-01</a>	March, 2006	TL 5300-04	SW fixes
TL	<a href="#">5300-04</a>	February, 2006		Preventive Maintenance and support for new hardware.

Use the `oslevel -r` command to determine the current operating system level.

Maintenance Level packages (legacy)				
Type	Name	Date	Prereqs	Description
ML	<a href="#">5300-03</a>	September, 2005		HW & SW enhancements and SW fixes
ML	<a href="#">5300-02</a>	May, 2005		HW & SW enhancements and SW fixes

Figure 4-16 IBM Support Fix Central Web site

## 4.11 Backing up clients with NIM

Backing up a system can be done in all the same ways as with any AIX/Linux server, but for a system that is also a NIM client this can also be done using NIM to create a `mksysb` image. Using NIM to create a `mksysb` image file has to be executed from the NIM master itself ("pull" operation), or you will get the following message:

```
0042-012 nim: this command may only be executed on a NIM master
```

Example 4-256, shows how the `nim` command is used for creating a `mksysb` image file named `mksysb.lpar5` in the `/export/images` directory on the NIM master and before doing so to create the `/image.data` file.

Example 4-256 Creating a `mksysb` from a NIM client

```
root@master:/: nim -o define -t mksysb -a server=master -a source=LPAR5
-a mk_image=yes -a location=/export/images/mksysb.lpar5 mksysb_lpar5
```

```
+-----+
|               System Backup Image Space Information
|               (Sizes are displayed in 1024-byte blocks.)
+-----+
```

```
Required = 868732 (848 MB)    Available = 2398052 (2342 MB)
```

Data compression will be used by the system backup utilities which create the system backup image. This may reduce the required space by up to 50 percent.

Creating information file (/image.data) for rootvg..

Creating list of files to back up.

Backing up 26669 files.....

26669 of 26669 files (100%)

0512-038 savevg: Backup Completed Successfully.

---

The command executed in Example 4-256 completed successfully, and both the specified `mksysb.lpar5` image file and the `mksysb_lpar5` NIM object are created.

**Note:** For naming NIM objects it is not supported to use dots (“.”), so we use underscore (“\_”) to enhance the readability of the name. For an AIX filename, it is supported to use one or more dots (“.”) or underscore signs (“\_”), and we could have used the NIM object name as file system name as well. Here we use separate signs to differentiate the name for the NIM `mksysb` object and the `mksysb` image filename. For additional naming examples, see 3.1.1, “Planning the NIM environment” on page 50.

The command breakdown for the example in Figure 4-256 on page 353 is:

<code>-o define</code>	Specifies the operation (defining the object).
<code>-t mksysb</code>	Specifies that the object type is <code>mksysb</code> .
<code>-a server=master</code>	Specifies that the server to hold this object (the <code>mksysb</code> image file) is the NIM master itself.
<code>-a source=LPAR5</code>	Specifies the NIM client to be used as the source the <code>mksysb</code> image, in this case, <code>LPAR5</code> .
<code>-a mk_image=yes</code>	Specifies that the <code>mksysb</code> image file should be created.
<code>-a location=/export/images/mksysb.lpar5</code>	

	Specifies the path and filename for the mksysb image file.
mksysb_lpar5	Specifies the NIM object name for this mksysb image.
Other attributes that can be used are (use <code>nim -q define -t mksysb</code> for the latest command option):	
mksysb_flags	Can be used to specify flags for the creation of the mksysb image file, see the <code>mksysb</code> command options.
exclude_files	Can be used to specify the NIM object pointing to a file containing a list of file systems to exclude from the mksysb image file, like <code>exclude.rootvg</code> file for the <code>mksysb</code> command.
size_preview	Can be used to determine the size requirement for the target mksysb image file.
comments	Can be used to specify a description of the NIM mksysb object.
verbose	Can be used to increase the message verbosity throughout the operation progress.
group	Can be used to specify a NIM group to create the mksysb resource in.

**Note:** The `nim -o define -t mksysb` command may not end immediately after the message “0512-038 savevg: Backup Completed Successfully.”, as it has not finished processing yet. To view detailed progress, use the `verbose` attribute with a value higher than zero (0). For example “-a verbose=2”.

### 4.11.1 Space requirements

One essential element when handling mksysb images from one or several servers is the space requirement. In Figure 4-256 on page 353, observe the size estimation for the mksysb image file and how much space there is available in the destination file system (`-a location`).

If there is *not* enough space available to store the mksysb image file in the directory specified by the `location` attribute (`-a location`), the `nim` command will terminate and display a message similar to the output shown in Example 4-257.

*Example 4-257 Creating a mksysb from a NIM client with insufficient storage space*

---

```
0042-001 nim: processing error encountered on "master":
0042-001 m_mkbsi: processing error encountered on "LPAR5":
0042-210 c_mkbsi: The maximum space required for the backup is greater than
the amount of free space in the target filesystem. To ignore space requirements
```

use the "-F" flag when defining the mksysb resource.

---

There are several ways to calculate how much space the mksysb image file will take, one way is to let NIM calculate it by adding the `-a size_preview=yes` attribute for the mksysb image file and object creation command. Example 4-258 shows what to expect if there is not enough space available, in which case you need to manually increase the size of the file system or free up some space to meet the requirements of the mksysb image file.

*Example 4-258 Determining backup image storage space requirement*

---

```
root@master:/: nim -o define -t mksysb -a server=master -a source=LPAR5
-a size_preview=yes -a mk_image=yes -a mksysb_flags=-i
-a location=/export/images/mksysb.lpar5 mksysb_lpar5
```

```
+-----+
|               System Backup Image Space Information               |
|               (Sizes are displayed in 1024-byte blocks.)         |
+-----+
```

```
Required = 1132200 (1106 MB)    Available = 785232 (767 MB)
```

Data compression will be used by the system backup utilities which create the system backup image. This may reduce the required space by up to 50 percent.

---

The command shown in Example 4-258 reveals that there are only 767 MB of free space available at the designated location (`/export/images`) and the NIM required size for the mksysb image file is in fact 1106 MB.

**Note:** The sizes calculated by the `nim` command are usually large since data compression is used by the system backup (which creates the system backup image), thus the final required space could be to 50 percent less than the estimation.

### 4.11.2 NFS export during mksysb image file creation

During the allocation of NIM mksysb image files, only the file is exported to the NFS client. However, during mksysb creation, the parent directory is also exported. If you use that file system to create a mksysb image of a system while another system is restoring a mksysb image from that file system, you will get NFS errors.

To avoid this problem, use the `NIM_MKSYSB_SUBDIRS` environment variable on the NIM master, as shown in Example 4-259.

When this variable is set to “yes”, subdirectories are used to separate `mksysb` image files. The subdirectories are transparent to the user, but they provide separate locations for NFS exporting. The subdirectories are named after the NIM object for which they are created.

*Example 4-259 Using NIM\_MKSYSB\_SUBDIRS environment variable*

---

```
root@master:/: export NIM_MKSYSB_SUBDIRS=yes
root@master:/: nim -o define -t mksysb -a server=master -a source=LPAR5
-a mk_image=yes -a location=/export/images/mksysb.lpar5 mksysb_lpar5
```

---

The command shown in Example 4-259 will create the `mksysb` image file in the `/export/images/LPAR5` directory and not in the `/export/images` directory.

### 4.11.3 Exclude file for `mksysb` image creation

Just like for the system `mksysb` command, an exclude file can be used to prevent certain file systems or directories to be included in the `mksysb` image file creation.

First the exclude file has to be created. The lines in this file are input to the pattern matching conventions of the `grep` command to determine which files will be excluded from the backup. For example, to exclude all the contents of the directory called `temp`, edit the exclude file to read as follows:

```
/temp/
```

For example, to exclude the contents of the directory called `/tmp`, and avoid excluding any other directories that have `/tmp` in the pathname, edit the exclude file to read as follows:

```
^./tmp/
```

All files are backed up relative to the current working directory (CWD). To exclude any file or directory for which it is important to have the search match the string at the beginning of the line, use `^` (caret character) as the first character in the search string, followed by `.` (dot character), followed by the filename or directory to be excluded.

If the filename or directory being excluded is a substring of another filename or directory, use `^.` (caret character followed by dot character) to indicate that the search should begin at the beginning of the line and/or use `$` (dollar sign character) to indicate that the search should end at the end of the line.

**Tip:** Use the vi editor, if you are uncomfortable with this editor, you can create the first line in a new file with the echo command:

```
echo "FILESYSTEMPATTERN" > excludefilename
```

To add additional lines to the one above, use the echo command again as many times as required (note the double greater-than signs):

```
echo "FILESYSTEMPATTERN" >> excludefilename
```

After the exclude file is created, a NIM object of the `exclude_files` type has to be created, as shown in Example 4-260.

*Example 4-260 Creating the exclude\_files type NIM object*

```
root@master:/: nim -o define -a verbose=2 -t exclude_files -a server=master
-a location=/export/scripts/exclude.lpar5 exclude_lpar5
```

```
m_mkres: define the exclude_lpar5 resource
  ok_to_mk_obj: server=master; location=/export/scripts/exclude.lpar5
  set_Mstate: name=master
    checking for location collisions
  stat_file: server=master; location=/export/scripts/exclude.lpar5;
  file must have: st_mode = 04000453330; st_vfstype = "0 3"
  stat-ing the file "/export/scripts/exclude.lpar5"
```

To use the exclude file (from the NIM object `exclude_lpar5`) to create a mksysb image file, use the `exclude_files` attribute as show in Example 4-261.

*Example 4-261 Using exclude\_files when creating mksysb image files*

```
root@master:/: nim -o define -t mksysb -a server=master -a source=LPAR5
-a mk_image=yes -a location=/export/images/mksysb.lpar5
-a exclude_files=exclude_lpar5 mksysb_lpar5
```

```
+-----+
          System Backup Image Space Information
          (Sizes are displayed in 1024-byte blocks.)
+-----+
```

```
Required = 869048 (849 MB)    Available = 1332332 (1301 MB)
```

Data compression will be used by the system backup utilities which create the system backup image. This may reduce the required space by up to 50 percent.



```
Creating information file (/image.data) for rootvg..
```

```
Creating list of files to back up.
```

```
Backing up 26665 files.....
```

```
26665 of 26665 files (100%)
```

```
0512-038 savevg: Backup Completed Successfully.
```

---

## 4.12 How to create bundles, BFF, and RPM packages

Starting with AIX 5L 5.1, you can install RedHat Package Manager (RPM) and InstallShield MutliPlatform (ISMP) formatted packages in addition to the native AIX installation Backup File Format (BFF - installp formatted packages).

In this section, we show how to create your own BFF and RPM packages, and how to use bundles to install these onto NIM clients.

**Note:** The procedures can also be used for standalone systems (systems not belonging to a NIM environment).

The AIX product media (CD/DVD) contains BFF and RPM packages that are installed during the base operating system (BOS) installation. The BFF packages are located by default in the following relative path under AIX 5L installation media or NIM `lpp_source` resource directories:

```
installp/ppc
```

The RPM packages are located in the following path:

```
RPMS/ppc
```

If you have media that contains ISMP packages, the packages are located in the following path:

```
ISMP/ppc
```

If you are using the **geninstall** command to install RPM or ISMP packages, use the prefix type to indicate to the **geninstall** command the type of package that you are installing. The package prefix types are the following:

I:	BFF format
R:	RPM format
J:	ISMP format

E: Interim Fix format

For example, to install the `openssl` RPM package and the `bos.perf BFF/installp` package, type the following:

```
geninstall -d /dev/cd0 R:openssl I:bos.perf
```

The **geninstall** command detects that the `openssl` package is an RPM package type and runs the **rpm** command to install the `openssl` package. The **geninstall** command then detects that `bos.perf` is an BFF/installp package type and runs the **installp** command to install the `bos.perf` package. The process for un-installation is similar to the installation process.

### 4.12.1 BFF (native installp packages)

The native AIX installation Backup File Format (BFF) is sometimes referred to as the “installp” format. The **mkinstallp** command allows you to create your own software packages for AIX in the BFF/installp format, and can be installed or removed with the **installp** command.

The **mkinstallp** command requires files that will be packaged to be in a directory structure such that the location of the file relative to the root build directory is the same as the destination of the file after installation.

After the contents of a package are located in the correct directory structure, the **mkinstallp** command prompts for basic package data. This data includes the package name, requisites, descriptions of files to be packaged, and more. The **mkinstallp** command will then generate a template file based on responses given by the user. To prevent command-line prompting, template files can be created and edited directly by the user and passed to the **mkinstallp** command with the “-T” flag.

To create a BFF/installp package you can use the following procedure:

1. Create the build structure.
2. Populate the build structure with programs and files.
3. Create the packaging template file, for use by the **mkinstallp** program.
4. Create the package file with the **mkinstallp** program.

#### Create the build structure

First we create our packaging buildroot directory. For our examples to create installation packages, we will use the `/build` file system (directory) as the top level for our packages. We choose to use a separate file system for this purpose, and we create it as an Enhanced Journaled File System (`jfs2`):

```
crfs -v jfs2 -g rootvg -a size=120M -m /build -A yes -p rw -a  
logname=INLINE
```

We mount the file system:

```
mount /build
```

And limit the access to this file system mountpoint to restrict everyone except the root user (removing read-write-execute permissions for group and others):

```
chmod go-rwx /build
```

Next we create a directory structure for the BFF package, with the root packaging directory of `/build/nodectst`, which will be the root (`/`) file system at install time. We will create a NIM client customization package, hence the “nodectst” name.

In the `/build` file system, we create our packages buildroot directories. In this example, we create the nodectst package buildroot directory:

```
mkdir -p /build/nodectst
```

Then we create subdirectories where our files will be stored. Our intention is that during the installation, specific files will be updated or replaced.

We will replace the `/etc/motd` file if it already exists, and for this we create the `/etc` directory under `/build/nodectst`:

```
mkdir -p /build/nodectst/etc
```

## Populate the build structure

All installed files should be put in their respective directories under the buildroot. In our example, we create an `motd` file under the `/build/nodectst/etc` directory:

```
banner hello world >/build/nodectst/etc/motd
```

We set the file permissions on our new version of `motd` file to the same as the original (read-read-read only for owner-group-others):

```
chmod 444 /build/nodectst/etc/motd
```

## Create the packaging template file

The `mkinstallp` template file contains specifications for the installation package. The template file can be included in the `<buildroot>` but not part of the BFF package. Refer to Figure 4-17 on page 362.

In our example, we choose to store our template file in the buildroot directory, the `/build/nodectst` directory (also referred to as `<buildroot>` in the following examples). The template file for this small package, contain the following

specification entries. We named the template file `nodecust.template` in the `<buildroot>` directory.

```

Package Name: NODECUST
Package VRMF: 1.0.0.0
Update: N
Fileset
  Fileset Name: NODECUST.rte
  Fileset VRMF: 1.0.0.0
  Fileset Description: NODECUST
  Bosboot required: N
  License agreement acceptance required: N
  Include license files in this package: N
Requisites:
ROOT Part: Y
ROOTFiles
  /etc
  /etc/motd
EOROOTFiles
EOFFileset

```

Figure 4-17 Example `mkinstallp` template file

We first declare the name of the package (Package Name). This will be used for creating the package file. The Fileset Name is the name is managed by the `installp` command, shows up in the software inventory, and it is seen by the `ls1pp` command.

### Template file Keywords

Keywords with a `*` are required, and will cause `mkinstallp` to fail if left blank or omitted in the template file. This is a list of the template file keywords:

Package Name*	Name of the package
Package VRMF*	Version, Release, Modification, and Fix level of the package
Update*	Is this an update package?
Fileset*	Start of a new Fileset
Fileset Name*	Name of the fileset
Fileset VRMF*	VRMF of the fileset
Fileset Description	Description of the fileset
Bosboot required*	Is a <b>bosboot</b> required when installing this fileset?
License agreement acceptance required*	Is license agreement acceptance required for this fileset?

Name of license agreement	Name of the license agreement (see <i>Note 1</i> )
Include license files in this package	Are the license files included in this package?
License file path	Path of the license file(s) (see <i>Note 2</i> )
Requisites	co/if/inst/pre-requisites for the fileset (see <i>Note 3</i> )
USRFiles*	Start of the USR part files section
/path/to/file	File path (see <i>Note 4</i> )
E0USRFiles*	End of the USR part files section
ROOT Part: [Y N]*	Is there a ROOT part included in this fileset?
/path/to/file	File path (see <i>Note 4</i> )
E0ROOTFiles*	End of the ROOT part files section
E0Fileset*	End of the Fileset

**Note 1:** The “Name of license agreement” is defined as LAR/path/to/license/agreement. License Agreement Requisite® (LAR). The %L tag can be used in place of a hard coded path to represent the locale of the machine that the package will be installed on. For example, if the package is installed in the en\_US locale, %L will be converted to en\_US.

**Note 2:** A “License file path” is defined as LAF/path/to/license/file. License Agreement File (LAF). A conditional License file path is defined as LAF<lc\_LC>/path/to/license/file, where lc\_LC is the locale which is associated with the license file. An example conditional License file path is LAF<en\_US>/usr/swlag/en\_US/prod.la. Either type of path may be specified or both types. Multiple license file paths are separated by semicolons.

**Note 3:** Requisites are defined as \*Type Name VRMF;. Type may be coreq, ifreq, in streq, or prereq. Multiple requisites are separated by semicolons. Requisites may also be an absolute pathname of a file specifying multiple or complex requisites.

**Note 4:** The full path name for each file in the fileset must be listed in the files section. Any custom directories should also be listed in this section. For example, to package /etc/motd, list both /etc and /etc/motd in the files section. Each entity in the final package will have the same attributes (owner/group/permissions) that it had at build time, so you must ensure that file attributes in the build root are correct prior to running **mkinstallp**.

## Creating the BFF package file

When the packaging buildroot directory structure and all package files are in their right place, and the template file is created, you can run the **mkinstallp** command to generate the package. In our example, we specify the build directory (-d) and our template file (-T) with the **mkinstallp** command as shown in Example 4-262:

```
mkinstallp -d /build/nodectust -T /build/nodectust/nodectust.template
```

---

### Example 4-262 Creating a BFF/installp package file with mkinstallp

---

```
root@master:/: mkinstallp -T /build/nodectust/nodectust.template -d /build/nodectust
Using /build/nodectust as the base package directory.
Cannot find /build/nodectust/.info. Attempting to create.
Using /build/nodectust/.info to store package control files.
Cleaning intermediate files from /build/nodectust/.info.
```

```
Using nodectust.template as the template file.
0503-880 mkinstallp: Warning: /etc/motd exists in another fileset on the system.
processing NODECUST 1.0.0.0 I package
processing NODECUST.rte 1.0.0.0 fileset
creating ./usr/lpp/NODECUST/liblpp.a
creating ./tmp/NODECUST.1.0.0.0.bff
```

---

The **mkinstallp** command creates additional directory structure under our buildroot, BFF support files, and the BFF package file itself.

The package file is created under the <buildroot>/tmp directory with the package name, VRMF, and suffixed with “.bff”. The VRMF consist of four digits, the Version, Release, Maintenance/Modification and Fix numbering. In our example, the path to the BFF file is:

```
/build/nodectust/tmp/NODECUST.1.0.0.0.bff
```

Some of the support files are found in the <buildroot>/ .info directory, the <buildroot>/lpp\_name package information file, the <buildroot>/usr directory for the <buildroot>/usr/lpp/NODECUST/liblpp.a archive file and the <buildroot>/usr/lpp/NODECUST/inst\_root directory.

For additional information on the **mkinstallp** command and BFF package file format, refer to the manual AIX 5L Verion 5.3 “*Commands Reference, Volume 3: i through m*”, SC23-4890-02, and “*General Programming Concepts: Writing and Debugging Programs*”, SC23-4896-02, section “Packaging Software for Installation”.

## Examining the BFF package file

We can examine the BFF package file with the **restore** command. In Example 4-263, we use the quiet flag (-q) which prevents user input prompting, list (-l), verbose (-v) and device file (-f), and specify the package filename.

*Example 4-263 Examining the content of mkinstallp BFF/installp package*

---

```

root@master:/build/nodectust/tmp: restore -qTvf NODECUST.1.0.0.0.bff
New volume on NODECUST.1.0.0.0.bff:
Cluster 51200 bytes (100 blocks).
  Volume number 1
  Date of backup: Mon Jul  3 11:13:46 2006
  Files backed up by name
  User root
    0 ./
    166 ./lpp_name
    0 ./usr
    0 ./usr/lpp
    0 ./usr/lpp/NODECUST
    1146 ./usr/lpp/NODECUST/liblpp.a
    0 ./usr/lpp/NODECUST/inst_root
    1004 ./usr/lpp/NODECUST/inst_root/liblpp.a
    0 ./usr/lpp/NODECUST/inst_root/etc
    480 ./usr/lpp/NODECUST/inst_root/etc/motd
    0 ./etc
    480 ./etc/motd
total size: 3276
files archived: 12

```

---

You can see that the motd file in Example 4-263 ended up under the directory `./usr/lpp/NODECUST/inst_root/etc`.

## Installing the package with installp

To install the package with the **installp** command, just specify the package filename and package name (or the "all" keyword) as shown in Example 4-264. To uninstall the package, check the proper package name with the **lspp -l**, and then use this name with the **installp -u** command.

*Example 4-264 Installing the mkinstallp created package*

---

```

root@master:/build/nodectust/tmp: installp -ad NODECUST.1.0.0.0.bff NODECUST.rte
+-----+
|                                     Pre-installation Verification...                                     |
+-----+
Verifying selections...done

```

Verifying requisites...done  
Results...

SUCSESSES

-----

Filesets listed in this section passed pre-installation verification and will be installed.

Selected Filesets

-----

NODECUST.rte 1.0.0.0 # NODECUST

<< End of Success Section >>

FILESET STATISTICS

-----

1 Selected to be installed, of which:  
1 Passed pre-installation verification

----

1 Total to be installed

+-----+

Installing Software...

+-----+

installp: APPLYING software for:  
NODECUST.rte 1.0.0.0

sysck: 3001-036 WARNING: File /etc/motd  
is also owned by fileset bos.rte.security.  
Finished processing all filesets. (Total time: 1 secs).

+-----+

Summaries:

+-----+

Installation Summary

-----

Name	Level	Part	Event	Result
NODECUST.rte	1.0.0.0	USR	APPLY	SUCCESS
NODECUST.rte	1.0.0.0	ROOT	APPLY	SUCCESS



## Installing the package with NIM

To use the `mkinstallp` created package with NIM, they need to be made available from a NIM `lpp_source` resource, just like any other BFF package. If you have several packages, you can create a separate NIM `lpp_source` specifically for them. In the following examples, we assume that all packages are AIX VRMF (Version, Release, Maintenance/Modification and Fix) independent. See Example 4-265.

Create the NIM `lpp_source` BFF/`installp` directory:

```
mkdir -p /export/pkgs/installp/ppc
```

Copy all `mkinstallp` created packages from their respective `<buildroot>/tmp` directory:

```
cp <buildroot>/tmp/*.bff /export/pkgs/installp/ppc
```

Create a Table Of Content (TOC) file (`.toc`) for the directory:

```
cd /export/pkgs/installp/ppc
inutoc .
cd -
```

Verify that the intended installation packages are accessible with the `installp` command by doing a list of the directory TOC:

```
installp -qld /export/pkgs/installp/ppc
```

*Example 4-265 installp command to list package directory for mkinstallp packages*

---

```
root@master:/: installp -qld /export/pkgs/installp/ppc
  Fileset Name                Level                I/U Q Content
  -----
  NODECUST.rte                1.0.0.0                I  N usr,root
#  NODECUST
```

---

Create a NIM `lpp_source` resource with the `nim` command and define operation as shown in Example 4-266.

*Example 4-266 nim command and define operation for mkinstallp packages*

---

```
root@master:/: nim -o define -t lpp_source -a server=master -a location=/export/pkgs pkgs
```

Preparing to copy install images (this will take several minutes)...

Now checking for missing install images...

```
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
bos.net
bos.diag
bos.sysmgt
bos.terminfo
bos.terminfo.all.data
devices.graphics
devices.scsi
devices.tty
x1C.rte
bos.mp
devices.common
bos.64bit

```

---

The warning message (0042-267) can be disregarded since we will not use this NIM lpp\_source to perform BOS (Base Operating System) installations, but only for additional software installations and updates to already installed systems. Verify the BFF packages and the NIM lpp\_source as shown in Example 4-267.

*Example 4-267 lsnim command to display the BFF pkgs NIM lpp\_source*

---

```

root@master:/: lsnim -l pkgs
pkgs:
  class      = resources
  type       = lpp_source
  arch       = power
  Rstate     = ready for use
  prev_state = verification is being performed
  location   = /export/pkgs
  alloc_count = 0
  server     = master

```

---

Verify that the intended installation packages are accessible with the **nim** command using the **lspp** operation for a list of the directory TOC as shown in Example 4-268.

*Example 4-268 nim command and lspp operation for mkinstallp packages*

---

```

root@master:/: nim -o lspp pkgs
NODECUST
  NODECUST.rte

```

---

Now the pkgs NIM lpp\_source can be allocated to NIM clients and be used to install the NODECUST.rte fileset.

## 4.12.2 RPM

The RPM Package Manager (RPM) is another popular package-based installation tool, used in many Linux distributions and also introduced in AIX for installing in AIX Toolbox for Linux packages. These packages are usually files with file names suffixed with “.rpm” and installed by the **rpm** command.

<http://www.rpm.org>

On AIX 5L V5.3, the `rpm.rte` fileset is version 3. You will have to use the **rpm** command with the `-ba` flags to build RPM packages. With Linux systems using version 4 of RPM, you must use the **rpmbuild** command instead.

To create an RPM package, you can use the following procedure:

1. Create the build structure.
2. Populate the build structure with programs and files.
3. Create the packaging SPEC file, for use by the **rpm** program.
4. Create the package file with the **rpm** program.

### Create the build structure

By default, the AIX 5L RPM environment use the `/opt/freeware/src/packages` directory as the top level for creating and building RPM packages. It is possible to change it to another top level, but for simplicity we choose to show the default method.

Under the `/opt/freeware/src/packages` directory, the subdirectories are used during different stages: during the build or the install of RPM packages:

BUILD	Source code hierarchy
SPECS	<b>rpm</b> SPEC specification file
SOURCES	gzip compressed tar archive created from the SPEC hierarchy
SRPMS	Source RPM package
RPMS/ppc	Binary RPM package for ppc architecture

We create a NIM client customization package, named “NODECUST”. The package will only replace the `/etc/motd` file.

### Populate the build structure

All files that will be installed should be put in their respective directories under the `<builddroot>`. In our example, we create an `motd` file under the `/opt/freeware/src/packages/BUILD/etc` directory:

```
banner hello world >/opt/freeware/src/packages/BUILD/etc/motd
```

We set the file permissions on our new version of `motd` file to the same as the original (read-read-read only for owner-group-others):

```
chmod 444 /opt/freeware/src/packages/BUILD/etc/motd
```

From the BUILD hierarchy, we create a source file, the `NODECUST-1.0-1.tar.gz` file that is simply created by piping the `tar` archive output to `gzip`, and redirecting the output to the source file (as shown in Example 4-269), named according to the naming convention mentioned above.

*Example 4-269 Creating a compressed archive of BUILD directory*

---

```
root@master:/: cd /opt/freeware/src/packages/BUILD
root@master:/: tar cf - . | gzip -c > /opt/freeware/src/packages/SOURCES/NODECUST-1.0-1.tar.gz
```

---

The source and RPM package use the following basic naming convention, and expect the source file to be found in the `/opt/freeware/src/packages/SOURCES` directory:

```
<name>-<version>-<release>.<arch>.rpm
```

Where:

name	Is the package Name
version	Are the software Version
release	The package Release (the number of times the package has been rebuilt using the same version of the software)
arch	The architecture the package was built under, such as i686, or ppc.

### Create the packaging SPEC specification file

The `rpm` SPEC specification file contains specifications for the installation package. The RPMs build and installation process is controlled by the SPEC file, which is the core of every RPM and SRPM (Source RPM).

By default AIX 5L `rpm` command uses the SPEC files in the `/opt/freeware/src/packages/SPECS` directory. For our examples, we use the SPEC file named `/opt/freeware/src/packages/SPECS/NODECUST.spec` as shown in Example 4-270.

*Example 4-270 Sample RPM SPEC (.spec) file*

---

```
Name: NODECUST
Summary: NODECUST perform local customization
Version: 1.0
Release: 1
Group: System Environment/Base
```

```

License: INTERNAL DUDE USE ONLY
Source: NODECUST-1.0-1.tar.gz
%description
NODECUST perform local customization
%build
%install
%files
%_builddir/etc/motd
%post
cp %_builddir/etc/motd /etc/motd

```

---

We declare our motd file using a macro, the `%_builddir`, so the file is handled and installed by the `rpm` command to the `%_builddir/etc` directory (which translates to the `/opt/freeware/src/packages/BUILD/etc` directory).

**Note:** For information on available macros on AIX 5L, look in the `/usr/opt/freeware/lib/rpm/macros` file.

We declare the Source to be the `NODECUST-1.0-1.tar.gz` file created from our BUILD hierarchy. The motd file will be extracted from the tar.gz file into the `/opt/freeware/src/packages/BUILD/etc` directory.

To put our version of the motd file in the `/etc` directory, we use a `%post` stanza. We use the `cp` command to copy from the `%_builddir` path to the file `/etc/motd`.

The next step is to actually build the binary and source packages by issuing the `rpm` command as shown in Example 4-271:

```
rpm -ba <SPEC file name>
```

#### Example 4-271 Creating a RPM

---

```

root@master:/: rpm -ba /opt/freeware/src/packages/SPECS/NODECUST.spec
Executing(%build): /bin/sh -e /var/opt/freeware/tmp/rpm-tmp.26112
+ umask 022
+ cd /opt/freeware/src/packages/BUILD
+ exit 0
Executing(%install): /bin/sh -e /var/opt/freeware/tmp/rpm-tmp.26112
+ umask 022
+ cd /opt/freeware/src/packages/BUILD
+ exit 0
Processing files: NODECUST-1.0-1
Finding Provides: (using /opt/freeware/lib/rpm/find-provides)...
Finding Requires: (using /opt/freeware/lib/rpm/find-requires)...
PreReq: /bin/sh

```

Wrote: /opt/freeware/src/packages/SRPMS/NODECUST-1.0-1.src.rpm

Wrote: /opt/freeware/src/packages/RPMS/ppc/NODECUST-1.0-1.aix5.3.ppc.rpm

---

This **rpm -ba** command above will generate the following RPMs:

/opt/freeware/src/packages/SRPMS/NODECUST-1.0-1.src.rpm

/opt/freeware/src/packages/RPMS/ppc/NODECUST-1.0-1.aix5.3.ppc.rpm

## Installing the package with RPM

The RPM package can now be installed with the **-i**, **-U**, or **-F** flags of the **rpm** command. In Example 4-272, we install the RPM package with the **-ivvv** options of the **rpm** command (install and three times verbose). The **-i** flag with **rpm** will only work if the RPM is not already installed on the system. If the RPM packages are already installed, use the **-U** or **-F** flags. To uninstall the RPM, check the proper package name with **rpm -qa**, and then use this name with the **rpm -e**.

### Example 4-272 Installing RPM with rpm

---

```
root@master:/: rpm -ivvv /opt/freeware/src/packages/RPMS/ppc/NODECUST-1.0-1.aix5.3.ppc.rpm
```

```
D: counting packages to install
```

```
D: found 1 packages
```

```
D: looking for packages to download
```

```
D: retrieved 0 packages
```

```
D: New Header signature
```

```
D: Signature size: 68
```

```
D: Signature pad : 4
```

```
D: sigsize      : 72
```

```
D: Header + Archive: 1274
```

```
D: expected size  : 1274
```

```
D: opening database mode 0x102 in /var/opt/freeware/lib/rpm
```

```
D: found 0 source and 1 binary packages
```

```
D: requires: /bin/sh satisfied by db provides.
```

```
D: installing binary packages
```

```
D: New Header signature
```

```
D: Signature size: 68
```

```
D: Signature pad : 4
```

```
D: sigsize      : 72
```

```
D: Header + Archive: 1274
```

```
D: expected size  : 1274
```

```
D: package: NODECUST-1.0-1 files test = 0
```

```
D: file: /opt/freeware/src/packages/BUILD/etc/motd action: create
```

```
D: running preinstall script (if any)
```

```
NODECUST-1.0-1
```

```
GZDIO:      7 reads,      516 total bytes in 0.000 secs
```

```
D: running postinstall scripts (if any)
```

```
+ cp /opt/freeware/src/packages/BUILD/etc/motd /etc/motd
```

---

In the output of Example 4-272, you see the creation of the `/opt/freeware/src/packages/BUILD/etc/motd` file, and the execution of our postinstall script that copies it to `/etc/motd`.

### Installing the RPM package with NIM

To use the `rpm` created package, or several such packages, with NIM, they need to be available from a NIM `lpp_source` resource, just like any other RPM package. If you have several packages, create a separate NIM `lpp_source` for these specifically. In the following examples, we assume that all packages are AIX VRMF (Version, Release, Maintenance/Modification and Fix) independent.

Create the NIM `lpp_source` RPM directory:

```
mkdir -p /export/pkgsrc/RPMS/ppc
```

Copy all `rpm` created packages from the default `<buildroot>` directory:

```
cp /opt/freeware/src/packages/RPMS/ppc/*.rpm /export/pkgsrc/RPMS/ppc
```

Create a NIM `lpp_source` resource with the `nim` command and define operation as shown in Example 4-273.

*Example 4-273 nim command and define operation for RPM packages*

---

```
root@master:/: nim -o define -t lpp_source -a server=master -a location=/export/pkgsrc/pkgsrc
```

```
Preparing to copy install images (this will take several minutes)...
```

```
Now checking for missing install images...
```

```
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
    bos.net
    bos.diag
    bos.sysmgmt
    bos.terminfo
    bos.terminfo.all.data
    devices.graphics
    devices.scsi
    devices.tty
    x1C.rte
    bos.mp
    devices.common
    bos.64bit
```

---

The warning message (0042-267) can be disregarded since we will not use this NIM lpp\_source to perform BOS (Base Operating System) installations, we only use it for additional software installations and updates. Verify and display the RPM packages and the NIM lpp\_source as shown in Example 4-274.

*Example 4-274 lsnim command to display the RPM pkgs NIM lpp\_source*

---

```
root@master:/: lsnim -l pkgs
pkgs:
  class      = resources
  type       = lpp_source
  arch       = power
  Rstate     = ready for use
  prev_state = verification is being performed
  location   = /export/pkgs
  alloc_count = 0
  server     = master
```

---

Verify that the intended installation packages are accessible with the `nim` command using the `lspp` operation for a list of the directory TOC as shown in Example 4-275.

*Example 4-275 nim command and lspp operation for mkinstallp packages*

---

```
root@master:/: nim -o lspp pkgs
NODECUST-1.0
      NODECUST-1.0
```

---

Now, the pkgs NIM lpp\_source can be allocated to NIM clients and used to install the NODECUST-1.0 RPM fileset.

If we use both the `installp/ppc` and `RPMS/ppc` directory, the `nim` command with `lspp` operation will show the packages from both packaging formats as shown in Example 4-276.

*Example 4-276 nim command and lspp operation for mkinstallp and rpm packages*

---

```
root@master:/: nim -o lspp pkgs
NODECUST
      NODECUST.rte
NODECUST-1.0
      NODECUST-1.0
```

---



## Bundles

A NIM `installp_bundle` resource represents a file that contains the names of filesets that should be managed by NIM.

During an installation or maintenance operation, NIM mounts the `installp_bundle` file on the client machine, and make it usable for the clients local **`installp` command**. NIM automatically unmounts the resource from the client when the operation has completed.

Use any editor to create bundle files, we recommend the `vi` editor, which can contain comments and fileset names. Lines beginning with the pound sign (`#`) are recognized as comments and are ignored by the installation process.

The following are examples of predefined bundles:

<code>BOS.autoi</code>	Bundle file that BOS Auto Install product options will be installed by <code>bos install</code> as part of base operating system installation on all systems.
<code>Server.bnd</code>	Bundle file that contains filesets/packages that are likely to be required by Server systems.
<code>Graphics.bnd</code>	Bundle file that contains basic X11 software packages that provides support of graphical environments. Additional packages for graphical user interfaces are <code>CDE.bnd</code> , <code>GNOME.bnd</code> and <code>KDE.bnd</code> .
<code>PerfTools.bnd</code>	Bundle file that contains additional performance tools packages.

The system bundles are located in the `/usr/sys/inst.data/sys_bundles` directory. To list the system bundles, type the following:

```
ls /usr/sys/inst.data/sys_bundles/*.bnd
```

## Using bundles with NIM

To create a bundle to be used as a NIM resource, create a file containing the filesets and packages to be installed. In this case we use the `/export/scripts` directory on the NIM master and call the file `nodecust.bundle`.

The file contains the package prefix type and then the package and fileset. The first line is the `NODECUST.rte` fileset we created in “Creating the BFF package file” on page 364 and the second line is the `NODECUST-1.0` RPM package we created in “Create the packaging SPEC specification file” on page 370. See Example 4-277.

*Example 4-277 NIM `installp_bundle` file*

---

```
I:NODECUST.rte
```

R:NODECUST-1.0

---

Next we create the NIM `installp_bundle` resource pointing to the bundle file with the `nim -o define` command, and with the `installp_bundle` type. In this case, we call our NIM resource `bundle_nodecust` as shown in Example 4-278.

```
nim -o define -a server=master -t installp_bundle -a
location=/export/scripts/nodecust.bundle bundle_nodecust
```

We can examine the NIM `installp_bundle` resource with the `lsnim` command.

*Example 4-278 The installp\_bundle NIM resource*

---

```
bundle_nodecust:
class      = resources
type      = installp_bundle
Rstate    = ready for use
prev_state = unavailable for use
location  = /export/scripts/nodecust.bundle
alloc_count = 0
server    = master
```

---

### Install NIM bundle (installp\_bundle)

To install the software specified in the bundle file, you can run the `nim` command and `cust` operation. In Example 4-279, we install the NIM `bundle_nodecust` on the NIM client LPAR55. Note that the installation process by default starts with the RPM packages.

*Example 4-279 nstallp\_bundle containing both BFF/installp and RPM packages*

---

```
root@master:/: nim -o cust -a lpp_source=pkgs -a installp_flags=agXY -a
installp_bundle=bundle_nodecust LPAR55
```

Validating RPM package selections ...

```
NODECUST      #####
+-----+
|               Pre-installation Verification...
+-----+
Verifying selections...done
Verifying requisites...done
Results...

SUCSESSES
```

```
-----
|   Filesets listed in this section passed pre-installation verification
|   and will be installed.
```

```
|   Selected Filesets
```

```
|   -----
|   NODECUST.rte 1.0.0.0                # NODECUST
```

```
|   << End of Success Section >>
```

```
| FILESET STATISTICS
```

```
|   -----
|   1 Selected to be installed, of which:
|   1 Passed pre-installation verification
```

```
|   ----
|   1 Total to be installed
```

```
+-----+
|           Installing Software...
+-----+
```

```
installp: APPLYING software for:
|           NODECUST.rte 1.0.0.0
```

```
| Finished processing all filesets. (Total time: 1 secs).
```

```
+-----+
|           Summaries:
+-----+
```

```
Installation Summary
```

```
-----
```

Name	Level	Part	Event	Result
NODECUST.rte	1.0.0.0	USR	APPLY	SUCCESS
NODECUST.rte	1.0.0.0	ROOT	APPLY	SUCCESS

```
-----
```

## 4.13 NIM and Virtual I/O (VIO)

If your environment has an IBM System p5, running in AIX 5L V5.3, and is connected to a Hardware Management Console (HMC), you can set up a VIO server. The VIO server provides Virtual SCSI that allows you to create more partitions (VIO clients) than the number available physical storage and network

adapters in the IBM System p5 CEC (Central Electronic Complex). It also enables you to share the I/O resources between the partitions.

Furthermore, the VIO server provides shared ethernet capabilities to allow multiple partitions to share ethernet adapters if those partitions need to access the external network.

If you have an environment where most of your partitions are within a POWER5 server, you can configure the NIM master in one of the VIO clients. With this, you can configure the NIM network to allow high speed inter-partition communication within the same server through the POWER Hypervisor VLAN.

### 4.13.1 Virtual SCSI

When you need to create multiple partitions but do not have enough SCSI/FC adapters, you can set up a VIO server to share the storage resources.

Virtual SCSI is based on a client/server model. The VIO server acts as a server owning the physical resources, and the partition is a client accessing the Virtual SCSI resources provided by the server.

For every VIO client, you need to create a dedicated virtual SCSI adapter. Once created, you can use a VIO server's physically attached disks in two ways: assigning LV(s) from a VG to the VIO client; or assigning one entire LUN (hdisk) to the VIO client.

You can find a brief description of the Virtual SCSI server and client adapter in the Hardware Management Console (HMC), the LV and disk configuration and how they are mapped to the virtual SCSI server adapter in 4.13.4, "Setting up the VIO server and configuring the NIM master in the VIO client" on page 379.

In Virtual I/O server version 1.2, virtual optical devices like DVD-ROM or DVD-RAM are also supported. You can create a virtual optical device from the physical optical device. With this virtual optical device, partitions are able to share the physical optical device in the VIO server.

### 4.13.2 Virtual shared ethernet

Virtual ethernet allows inter-partition communication without any physical network adapter. But if your partitions need to communicate with the external network, you need to configure a Shared Ethernet Adapter (SEA) in the VIO server. The Shared Ethernet Adapter provides the connection between the physical network with the virtual network. It enables multiple partitions to share the same physical network adapter.

### 4.13.3 Consideration when using a VIO server

There are some considerations if we are planning to configure the NIM master in one of VIO clients.

- ▶ Virtual SCSI requires about 20% of a single CPU Virtual I/O server for large block streaming or as much as 70% for the worst case small block.
- ▶ Shared Ethernet environment requires about 63% of a single CPU Virtual I/O server for simplex streaming of data or about 80% of a CPU for full duplex (bi-directional) streaming of data over a single Shared Ethernet Adapter and one Virtual Ethernet.
- ▶ Virtual SCSI runs at low priority interrupt levels, while virtual Ethernet runs on high priority interrupts due to the latency differences between disks and LAN adapters. If you have a client that has high network activity and another client having high I/O activity, the later will have performance impact.

We recommend you to allocate sufficient CPU resources to the VIO server or even use two VIO servers: one performing the Virtual SCSI and the other performing the Virtual Ethernet. It is a good idea also to have VIO server redundancy so that you can perform any VIO server maintenance, should you need to bring down the VIO server during maintenance.

Refer to Chapter 5 of the redbook “Advanced POWER Virtualization on IBM System p5, SG24-7940-01 for more details on VIO redundancy.

### 4.13.4 Setting up the VIO server and configuring the NIM master in the VIO client

This section describes the steps to setup the Virtual SCSI and Virtual Ethernet on a VIO server. We assume you already have some knowledge on creating profiles in the Hardware Management Console (HMC) and setting up the VIO server. The emphasis is on the Virtual SCSI and the Virtual Ethernet. Please refer to the Advanced POWER Virtualization on IBM System p5, SG24-7940-01 for more information.

Figure 4-18 shows the disk setup of the VIO server with two VIO clients. One of the clients is configured as the NIM master.

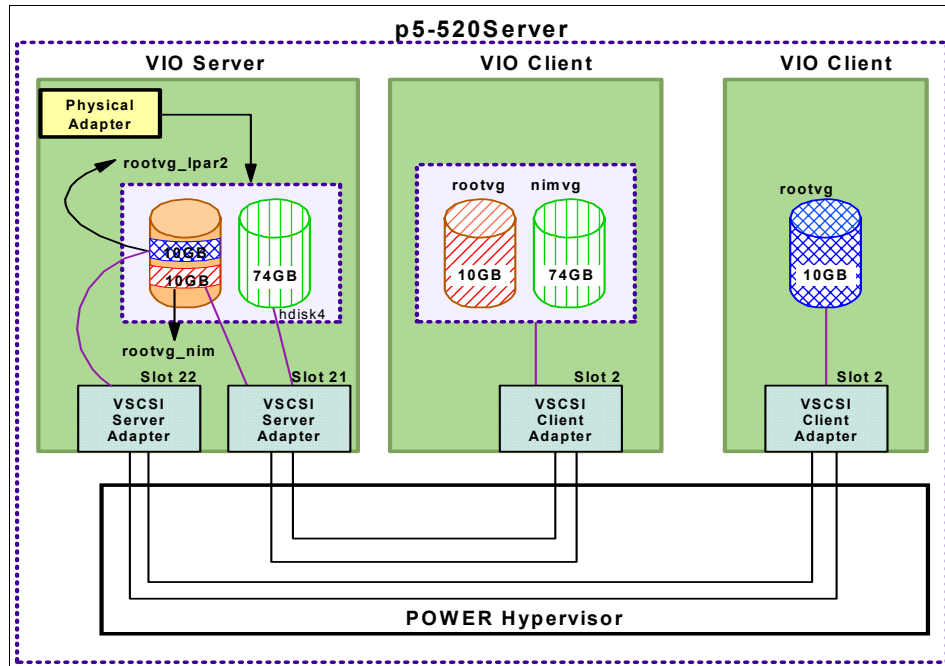


Figure 4-18 Virtual SCSI setup

Figure 4-19 shows the VLAN configuration for the same environment (one VIO server with two VIO clients)

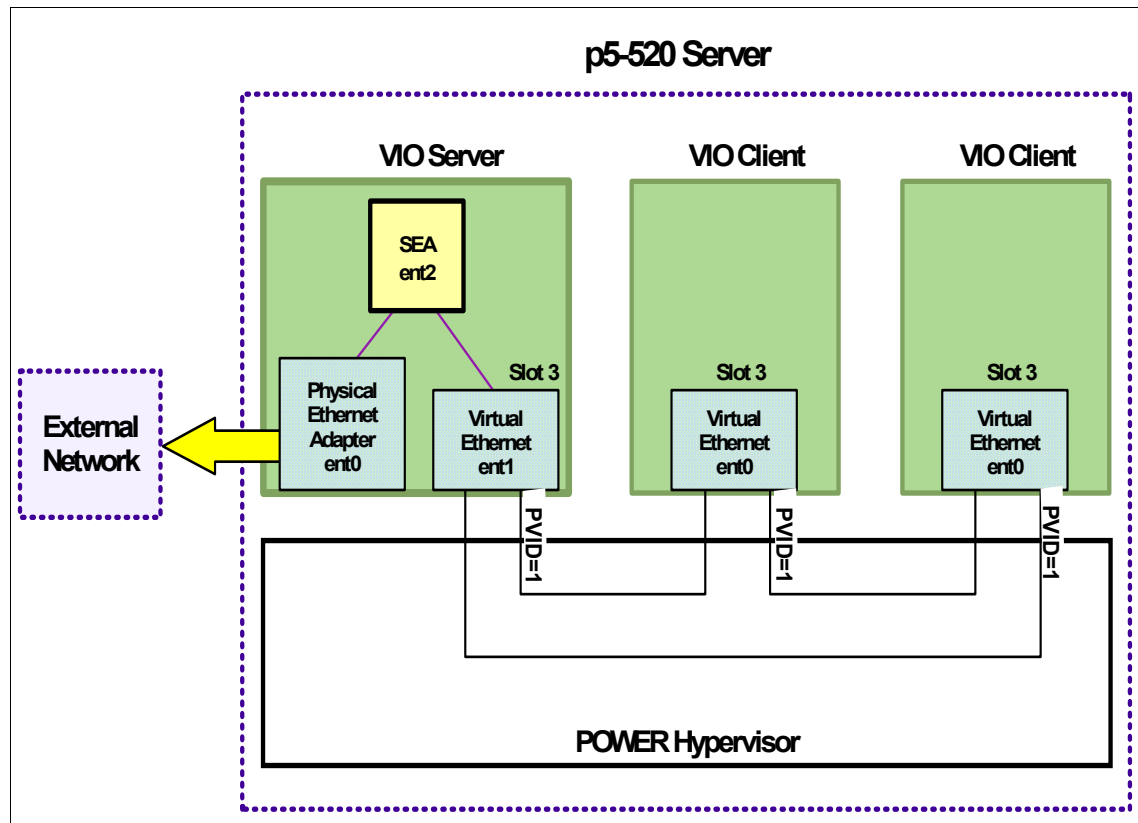


Figure 4-19 Virtual Ethernet setup

### Setting up the VIO server's and clients' profile

There are a few examples taken while setting up the Virtual SCSI and the Virtual Ethernet.

1. Open up the VIO server's profile. Perform the Virtual SCSI and Ethernet creation as shown in Figure 4-20 on page 382.

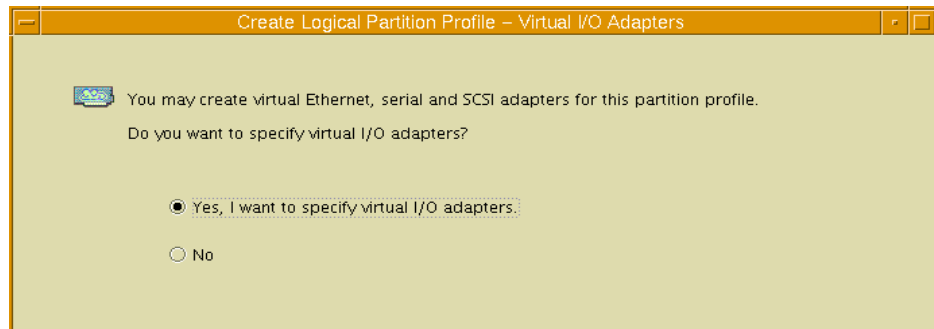


Figure 4-20 Create Virtual SCSI and Ethernet

2. Create the Virtual SCSI in the VIO server's profile. Click on the "Create server adapter" icon as shown in Figure 4-21.

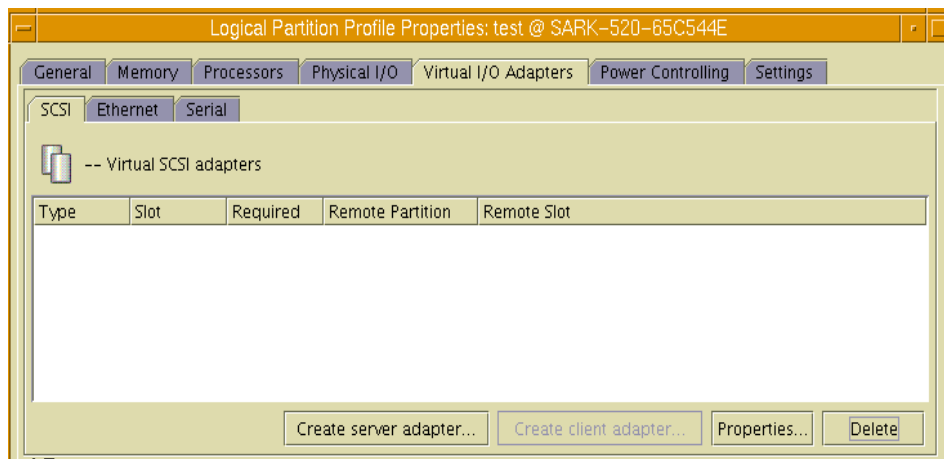


Figure 4-21 Click on "Create server adapter" icon

3. Specify the server's SCSI slot ID. If you have already created an LPAR, you can key in the client partition name and its slot ID. If not, check on the "Any client can connect". Remember to come back to check on the correct Client partition after the client partition is created. See Figure 4-22 on page 383.



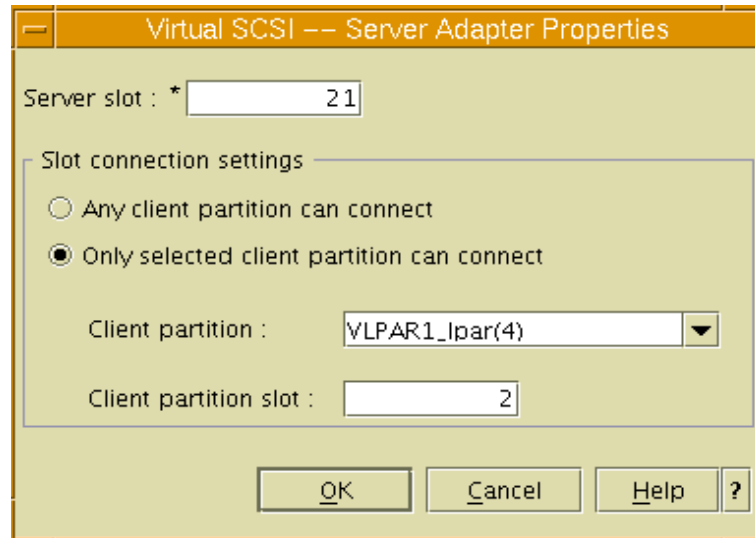


Figure 4-22 Key in the server and client virtual SCSI ID

4. Next, create the Virtual Ethernet on the VIO server. Click on the “Ethernet Tab” on the Top Left and then click on “Create Adapter” icon as shown in Figure 4-23.

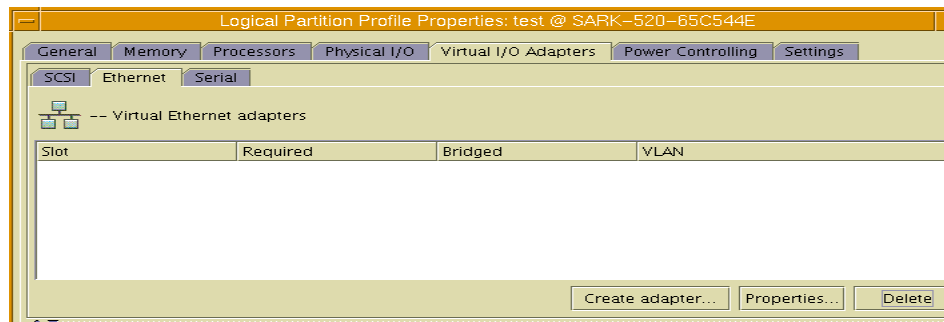


Figure 4-23 Create Virtual Ethernet on VIO server

5. Enter the Virtual SCSI slot ID and the VLAN ID. If you need to create a Shared Ethernet adapter to allow the partition to access the external network, you need to check on the “Access external network”. See Figure 4-24 on page 384.

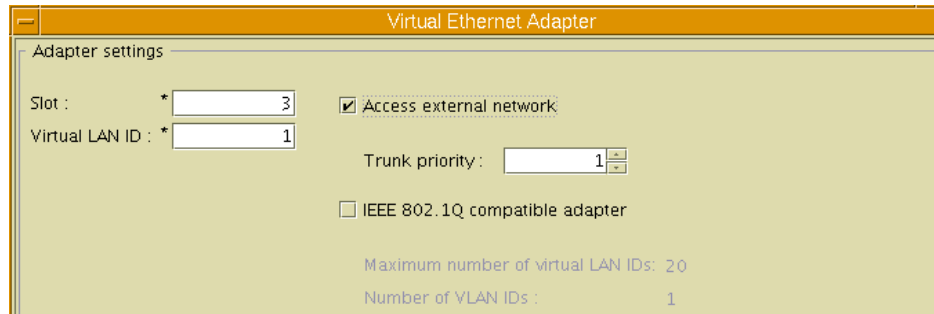


Figure 4-24 Key in the Virtual Ethernet Slot ID and the VLAN ID

- Once you have completed the VIO server's profile, proceed to the VIO client's profile. The steps involved are very similar. You click on the "Create client adapter" button instead. Figure 4-25 illustrates the Virtual SCSI on the client.

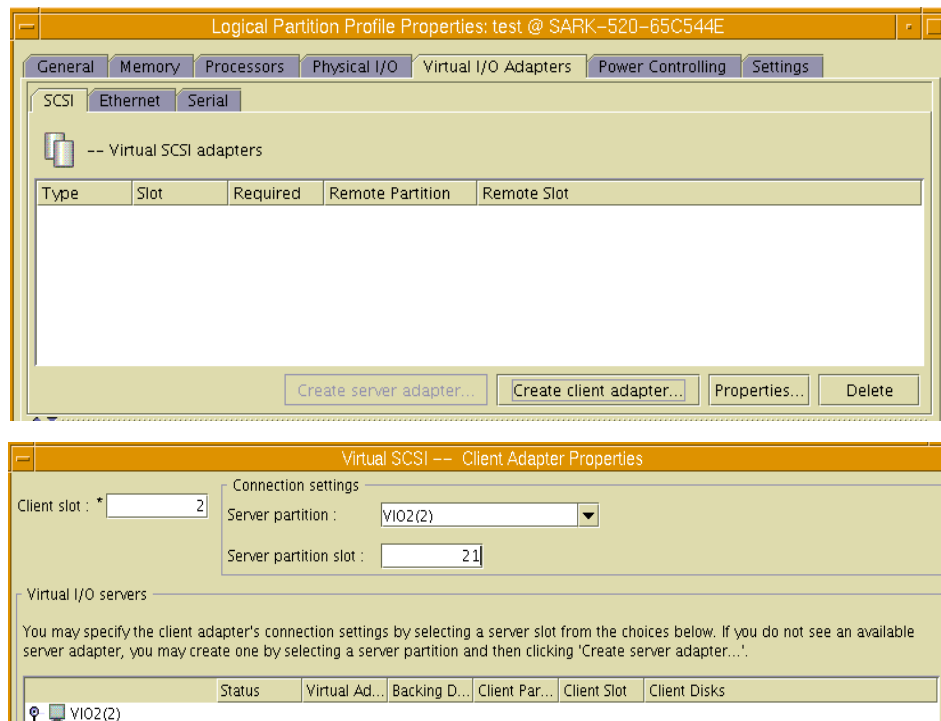


Figure 4-25 Create Virtual SCSI on VIO client

- When configuring the Virtual Ethernet on the VIO client, Key in the Slot ID and the VLAN ID. You do not need to check on the “Access external network” as shown in Figure 4-26.

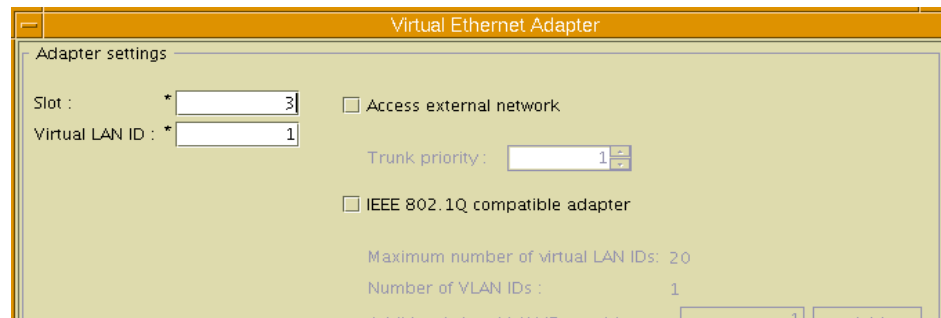


Figure 4-26 Key in Virtual Ethernet slot ID and VLAN ID

## Define the Volume Group (VG) and the Logical Volumes (LV)

Once we have defined the Virtual SCSI and Virtual Ethernet in the Hardware Management Console (HMC)'s profile, we can continue to create the VG and LV for the clients' usage in the VIO server.

As the NIM master requires a considerable amount of disk space, we assigned an entire hdisk and created a LV to the VIO client which we used for our NIM master:

- Create a VG to be partitioned into LVs for the usage of the VIO clients' rootvg. For example:

```
$ mkvg -f -vg rootvg_clients hdisk2
```

If you need to extend the rootvg\_clients VG, use the following command:

```
$ extendvg -f rootvg_clients hdisk3
```

- Create LVs for usage by the VIO clients.

We create a LV on the rootvg\_client VG for NIM master, named rootvg\_nim and another LV for a NIM client, named rootvg\_lpar2. For example:

```
$ mklv -lv rootvg_nim rootvg_clients 10G hdisk2
$ mklv -lv rootvg_lpar2 rootvg_clients 10G hdisk2
```

Example 4-280 show the details of the Volume Group configuration.

Example 4-280 VG's information

```
$ lspv
NAME                PVID                VG                STATUS
hdisk0              0000c83603e569ef   rootvg           active
```

hdisk1	none	None	
hdisk2	0000c836a42eeeb1	rootvg_clients	active
hdisk3	0000c836a42f5b75	rootvg_clients	active
hdisk4	0000c836cc576c89	None	

---

## Configure the Virtual SCSI device

Before we can create the Virtual target device, we need to identify the virtual host devices, vhost, on the VIO server. These vhosts are the Virtual SCSI server adapters. They are configured once you have define all the Virtual SCSI in the VIO server's profile. You need to run **cfgdev command in the VIO server** to get these vhosts configured.

1. Verify the virtual host devices

Example 4-281 on page 386 shows the command listing all the vhosts. You can see the slot ID that is assigned when you specify it in the VIO server's profile. For example, vhost0 is having the Virtual SCSI slot ID of 21.

*Example 4-281 List the vhosts info*

---

```
$ lsdev -virtual
name          status      description
vhost0        Available   Virtual SCSI Server Adapter
vhost1        Available   Virtual SCSI Server Adapter

$ lsdev -vpd |grep vhost
vhost1 U9131.52A.650E4DG-V2-C22  Virtual SCSI Server Adapter
vhost0 U9131.52A.650E4DG-V2-C21  Virtual SCSI Server Adapter
```

---

2. Create Virtual Target Device mapping (VTD). See the mapping in Example 4-282.

Once the hdisk, LV, and the vhosts are in place, we can create the Virtual Target devices.

For the NIM master partition using vhost0 enter:

```
$ mkvdev -vdev rootvg_nim -vadapter vhost0 -dev vrootvg_nim
$ mkvdev -vdev hdisk4 -vadapter vhost0 -dev vnimvg_nim
```

For the NIM client lpar2 using vhost1 enter:

```
$ mkvdev -vdev rootvg_lpar2 -vadapter vhost1 -dev vrootvg_lpar2
```

*Example 4-282 Virtual target device mapping*

---

```
$ lsmap -all
SVSA          Physloc          Client Partition ID
vhost0        U9131.52A.650E4DG-V2-C21  0x00000003
```

```

VTD          vrootvg_nim
LUN          0x8100000000000000
Backing device rootvg_nim
Physloc

VTD          vnimvg_lpar1
LUN          0x8200000000000000
Backing device hdisk4
Physloc      U787F.001.DPMD01L-P1-T11-L5-L0

```

```

SVSA          Physloc          Client Partition ID
vhost1       U9131.52A.650E4DG-V2-C22  0x00000004

```

```

VTD          vrootvg_lpar2
LUN          0x8100000000000000
Backing device rootvg_lpar2
Physloc

```

## Configuring the Virtual optical device

The steps to configure the optical device as a Virtual SCSI device is similar to configuring the LV or disk as a Virtual SCSI device. First, check that the physical optical device is configured in the VIO server as shown in Example 4-283.

*Example 4-283 List physical optical device*

```

$ lsdev -vpd |grep cd
cd0U787F.001.DPMD01L-P4-D2IDE DVD-RAM Drive

```

Once the physical device is available, create the Virtual optical SCSI target device. In Figure 4-284, the Virtual optical device is assigned to the vhost0.

```

$ mkvdev -vdev cd0 -vadapter vhost0

```

*Example 4-284 List virtual optical SCSI target device*

```

$ lsmap -vadapter vhost0
SVSA          Physloc          Client Partition ID
vhost0       U9131.52A.650E4DG-V2-C21  0x00000003
VTD          vrootvg_nim
LUN          0x8100000000000000
Backing device rootvg_nim
Physloc

VTD          vnimvg_lpar1
LUN          0x8200000000000000

```

```
Backing device      hdisk4
Physloc             U787F.001.DPMOD1L-P1-T11-L5-L0

VTD                vtopt0
LUN                0x8300000000000000
Backing device      cd0
Physloc             U787F.001.DPMOD1L-P4-D2
```

```
$ lsdev -vpd |grep vtopt
vtopt0          U9131.52A.650E4DG-V2-C21-L3    Virtual Target Device -
Optical Media
```

---

**Note:** Only one Virtual target device can be created for a particular physical optical device.

You need to perform the following steps if you need to assign the optical device to another partition:

1. Remove the existing virtual optical device on the client partition that is having the virtual optical device:

From VIO client,

```
# rmdev -d1 cd0
```

2. Remove the virtual optical SCSI target device on the VIO server:

From VIO server,

```
$ rmvdev -vdev cd0
```

3. Create the Virtual optical SCSI target device with the desired client's vhost adapter:

From VIO server,

```
$ mkvdev -vdev cd0 -vadapter vhost1
```

4. Run `cfgmgr` command on the new client to configure the `cd0` device.

## Configuring the Shared Virtual Ethernet

If the partitions in the IBM System p5 need to access the external network, you need to configure the Shared Ethernet adapter (SEA). Follow the steps to configure the SEA adapter.

1. Check the existing network adapter

From Example 4-285 on page 389, we have a physical network adapter, `ent0` and the Virtual Ethernet defined, `ent1`. The `ent1` is having a slot ID 3 (C3) which is defined in the VIO server's profile.

*Example 4-285 List network adapter*


---

```

$ lsdev -type adapter
name          status          description
ent0          Available      10/100/1000 Base-TX PCI-X
Adapter(14106902)
ent1          Available      Virtual I/O Ethernet Adapter (1-lan)

$ lsdev -vpd |grep ent
Model Implementation: Multiple Processor, PCI bus
ent1  U9131.52A.650E4DG-V2-C3-T1    Virtual I/O Ethernet Adapter
(1-lan)

ent0  U787F.001.DPMOD1L-P1-T5    10/100/1000 Base-TX PCI-X
Adapter(14106902)

```

---

## 2. Create the Shared Ethernet Adapter

Now, we can create the Shared Ethernet Adapter using the physical Ethernet adapter, ent0 and the Virtual Ethernet, ent1. See Example 4-286.

```
$ mkvdev -sea ent0 -vadapter ent1 -default ent1 -default 1
```

*Example 4-286 List the Shared Ethernet Adapter*


---

```

$ lsdev -virtual |grep ent
ent1          Available      Virtual I/O Ethernet Adapter (1-lan)
ent2          Available      Shared Ethernet Adapter

$ lsmmap -all -net
SVEA  Physloc
-----
ent1  U9131.52A.650E4DG-V2-C3-T1

SEA          ent2
Backing device  ent0
Physloc       U787F.001.DPMOD1L-P1-T5

```

---

## 3. Configure the IP address of the Shared Ethernet adapter

We configure the IP address on the Shared Ethernet adapter, ent2.

```
$ mktcpip -hostname vio1 inetaddr 10.1.1.51 -interface ent2 -start \
-netmask 255.255.255.0
```

## Installing AIX 5L in the VIO clients' partition

After you have assigned the disk and optical device like DVD-Rom to the client partition, you can bring up the partition and install the AIX operating system (OS).

Once the AIX OS installation is completed, configure the NIM master as explained in Chapter 3, "Basic configuration of a Network Installation Manager (NIM) environment" on page 49. After the NIM master and NIM client is set up, you can start to install the NIM client through the NIM master.

Example 4-287 shows the Virtual devices in the client partition, in this case, the NIM master.

*Example 4-287 List virtual SCSI disk, optical and Ethernet devices in VIO client*

---

Virtual SCSI disk Information

```
# lscfg -v1 vscsi0
```

```
vscsi0    U9131.52A.650E4DG-V3-C2-T1  Virtual SCSI Client Adapter
```

```
Device Specific.(YL).....U9131.52A.650E4DG-V3-C2-T1
```

```
# lspv
```

```
hdisk0    0000c836a446e179    rootvg    active
```

```
hdisk1    0000c836cc576c89    nimvg     active
```

```
# lscfg -v1 hdisk0
```

```
hdisk0    U9131.52A.650E4DG-V3-C2-T1-L810000000000 Virtual SCSI Disk
Drive
```

```
# lscfg -v1 hdisk1
```

```
hdisk1    U9131.52A.650E4DG-V3-C2-T1-L820000000000 Virtual SCSI Disk
Drive
```

Virtual SCSI Optical device information

```
# lsdev -C |grep cd
```

```
cd0       Available          Virtual SCSI Optical Served by VIO Server
```

```
# lscfg -v1 cd0
```

```
cd0       U9131.52A.650E4DG-V3-C2-T1-L820000000000 Virtual SCSI
Optical Served by VIO Server
```

Virtual Ethernet Information

```
# lsdev -C |grep ent0
```



```

ent0      Available      Virtual I/O Ethernet Adapter (1-lan)

# lspcfg -vl ent0
ent0      U9131.52A.650E4DG-V5-C3-T1  Virtual I/O Ethernet Adapter
(1)

Network Address.....1A8CF000501E
Displayable Message.....Virtual I/O Ethernet Adapter
(1-lan)
Device Specific.(YL).....U9131.52A.650E4DG-V5-C3-T1

```

---

## 4.14 Backup and restore of the VIO Server using NIM

There are several ways to perform backup and restore of VIO server. You can perform this task by tape, DVD-RAM or file system. In this section, we explain how we can perform the backup and restore through the NIM server.

**Note:** As of the writing of this Redbook, the `installios` command on AIX only supports using VIO base media. It is not yet supported on VIO backup images.

### 4.14.1 Backup of VIO server

We export the NIM master server's `mksysb` directory to the VIO server, and then perform a NFS mount of the directory, as shown in Example 4-288.

On the VIO server:

1. Create a temporary NFS mount point

```
$ mkdir viobackup
```

2. mount the NIM master, `v1par1`, server's `mksysb` directory

```
$ mount v1par1:/export/images /home/padmin/viobackup
```

*Example 4-288 List nfs mount directory on VIO server*

```

$ mount
node   mounted  mounted over  vfs   date       options
-----
      /dev/hd4  /           jfs2  Jun 13 09:42  rw,log=/dev/hd8
      /dev/hd2  /usr        jfs2  Jun 13 09:42  rw,log=/dev/hd8
      /dev/hd9var /var       jfs2  Jun 13 09:43  rw,log=/dev/hd8
      /dev/hd3  /tmp       jfs2  Jun 13 09:43  rw,log=/dev/hd8

```

```

/dev/hd1    /home    jfs2  Jun 13 09:43  rw,log=/dev/hd8
/proc      /proc    procfs Jun 13 09:43  rw
/dev/hd10opt /opt     jfs2  Jun 13 09:43  rw,log=/dev/hd8
v1par1 /export/images /home/padmin/viobackup nfs3   Jun 19 17:04

```

---

3. Perform the VIO backup using the `backupios` command as shown in Example 4-289. The `VIO.mksysb` is the `mksysb` filename.

```
$ backupios -file /home/padmin/viobackup/VIO.mksysb -mksysb
```

*Example 4-289 Backup VIO mksysb using backupios command*

```
$ backupios -file /home/padmin/viobackup/VIO.mksysb -mksysb
```

Backup in progress. This command can take a considerable amount of time to complete, please be patient...

Creating information file (/image.data) for rootvg....

```

Creating list of files to back up.
Backing up 29181 files.....
26047 of 29181 files (89%)
29181 of 29181 files (100%)
0512-038 savevg: Backup Completed Successfully.

```

**Note:** You need to specify the `-mksysb` option in the `backupios` command. This creates the `mksysb` image for the use of the NIM master server.

Use the `backupios` command without the `-mksysb` option only when you want install the VIO through the HMC.

We recommend you backup the following VIO configuration information. You can use these information to configure back the VIO should the `mksysb` restore does not perform successfully.

The information to backup is the following:

- ▶ All physical volume groups and logical volumes devices  
Commands: `lsvg` ; `lsvg -lv <VGname>`; `lspv`; `lspv -lv hdisk#`
- ▶ All physical disk and virtual Logical Volumes  
Command: `lsdev -type disk`
- ▶ All physical and virtual adapters  
Command: `lsdev -type adapter`
- ▶ Mapping between physical devices / logical volume and virtual devices

Commands: `lsmap -all ; lsmap -vadapter vhost# ; lsmap -all -net`

- ▶ Network information

Commands: `netstat -state ; netstat -num -state ;`

`netstat -routinfo`

## 4.14.2 Restoring the VIO server

You can perform the VIO server restoration using similar steps as to restore an AIX NIM client. You need to create a NIM mksysb resource from the image created by the `backupios` command in section 4.14.1, “Backup of VIO server” on page 391. Next, you create the SPOT using this mksysb resource. Once the mksysb and the SPOT resource are in place, we can carry out the pull BOS installation to restore the VIO server. Follow these steps to restore the VIO server:

1. Define a mksysb resource

Define the mksysb resource from the `/export/images/VIO.mksysb` image as shown in Example 4-290.

```
# smitty nim_mkres
  Select “mksysb = a mksysb image”
```

*Example 4-290 Define a mksysb resource*

---

```

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

[TOP]                                [Entry Fields]
* Resource Name                       [MK-VIO]
* Resource Type                       mksysb
* Server of Resource                  [master] +
* Location of Resource                [/export/images/VIO.mksysb> /
Comments                              []

Source for Replication                 [] +
-OR-
System Backup Image Creation Options:
  CREATE system backup image?         no +
  NIM CLIENT to backup                 [] +
  PREVIEW only?                       no +
  IGNORE space requirements?          no +
[MORE...10]
```

---

## 2. Define a SPOT resource

Define the SPOT from the mkysb resource defined in the previous step. See Example 4-291.

```
# smitty nim_mkres
  Select " spot = Shared Product Object Tree - equivalent to /usr
  file"
```

### *Example 4-291 Define SPOT resource using mkysb resource*

---

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

	[Entry Fields]
* Resource Name	[SPOT-VIO]
* Resource Type	spot
* Server of Resource	[master] +
* Source of Install	
Images	[MK-VIO] +
* Location of Resource	[/export/spot] /
Expand file systems if space needed?	yes +
Comments	[]
installp Flags	
COMMIT software updates?	no +
SAVE replaced files?	yes +
AUTOMATICALLY install requisite software?	yes +
OVERWRITE same or newer versions?	no +
VERIFY install and check file sizes?	no +

---

## 3. Perform the BOS installation (pull mode)

Once the SPOT and the mkysb resources are in place, you can allocate both resources and perform a BOS installation through the NIM pull installation. Refer to 3.1, "Setting up a basic NIM environment" on page 50 on how to create SPOT, mkysb resources and perform BOS installation.

You get a warning message stating that the SPOT level is older than the mkysb level as shown in Example 4-292, but that is fine. You are still able to use this SPOT to perform the BOS installation.

### *Example 4-292 Warning message when perform BOS installation*

---

```
warning: 0042-360 m_bos_inst: The SPOT level is older than the mkysb
level. Therefore,
  the BOS installation may encounter problems.
  Update the SPOT to match the mkysb level or create a
```

---

new SPOT that has the same level.

---

#### 4. Reset the NIM BOS installation operation

After the restoration is completed, you have to reset the NIM BOS installation operation in the NIM master as shown in Example 4-293 on page 395. The following steps are followed when using the SMIT menus:

```
# smitty nim
  Perform NIM Administration Tasks
    Manage Machines
      Perform Operations on Machines
        Select the Target Machine
          Select "reset = reset an object's NIM state"
```

Or using SMIT Fast Path, **smitty nim\_mac\_op**

---

#### *Example 4-293 Reset NIM client BOS installation Operation*

---

Reset the NIM State of a Machine

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

	[EntryFields]
Target Name	VI02
Deallocate All Resources? (removes all root data for diskless/dataless clients)	yes +
Force	yes+

---

In the VIO server, you need to import all the other Volume Groups. The `bosinst.data` of the VIO mksysb image, the `IMPORT_USER_VGS` is set to `no`. You need to create the `bosinst_data` resource if you want to import the user VG after the NIM restoration

Next, check on all the VIO configuration:

- ▶ All physical and Logical volumes created for the VIO clients are in open/syncd
  - `lsvg -lv <VGname>`
- ▶ All the virtual devices:
  - `lsmap -all`
- ▶ All the network configuration:
  - `lsmap -all -net`
  - `netstat -state ; netstat -num -state ; netstat -routinfo`

Once everything is in place, you can activate each individual VIO clients.

## 4.15 Using RSCT PeerDomain with HANIM

When using HANIM ( see 4.1, “High Availability NIM (HA NIM)” on page 124), the NIM master and the alternate NIM master are not aware of each others state. The takeover of the role as acting NIM master, are based on whether NIM clients wish to use a specific system as their NIM master or not. The NIM master for a NIM client is specified in the clients `/etc/niminfo` file.

In this section, we will show how RSCT PeerDomain (RPD) can be used to automate HANIM takeover operations. The purpose is only to execute the appropriate `nim` command to perform a takeover of NIM clients (change their `/etc/niminfo` file). Refer to Figure 4-27.

An RSCT peer domain is a cluster of nodes configured for high availability, and it uses:

- ▶ RSCT core cluster security services for authentication.
- ▶ RSCT core Topology Services subsystem for node/network failure detection.
- ▶ RSCT core Group Services subsystem for cross node/process coordination.
- ▶ Resource Monitoring and Control (RMC) subsystem for coordination between the various RSCT subsystems.

In this section, we will use the following three POWER5 logical partitions:

lpar55	will be the first and original NIM master (RPD node one).
lpar56	will be the first and alternate NIM master (RPD node two).
lpar6	will be the first and only NIM client (already running as a NIM client to another NIM master environment).

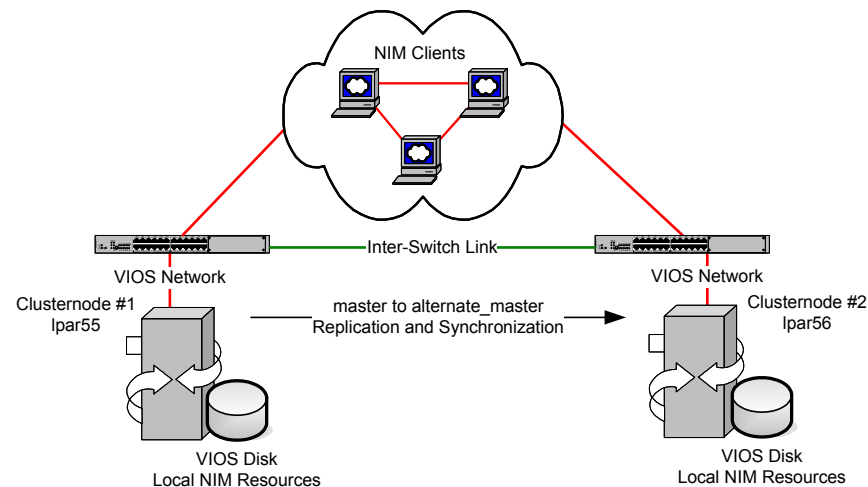


Figure 4-27 Graphic layout of our RSCT PeerDomain cluster setup for HANIM

All the LPARs are configured in the same way, using VIO servers for disk and network resources, and CPU micro-partitioning as shown in Figure 4-294.

*Example 4-294 Basic LPAR config for RSCT PeerDomain HANIM examples*

---

```
root@lpar56:/: lscfg -vl hdisk0
hdisk0 U9111.520.65C544E-V6-C2-T1-L810000000000 Virtual SCSI Disk Drive

root@lpar56:/: lscfg -vl ent0
ent0 U9111.520.65C544E-V6-C3-T1 Virtual I/O Ethernet Adapter (1-lan)

Network Address.....223380006003
Displayable Message.....Virtual I/O Ethernet Adapter (1-lan)
Device Specific.(YL).....U9111.520.65C544E-V6-C3-T1

root@lpar56:/: lparstat -h
System configuration: type=Shared mode=Capped smt=0n lcpu=2 mem=512 psize=2 ent=0.10
. . .
```

---

The nodes configuration will only use the ent0 network virtual network adapter on the cluster nodes, and all NIM database and resource file systems will be local to each of the two cluster nodes. HANIM functionality will be used to keep the database synchronized and resources replicated.

**Note:** In this scenario, we assume that one NIM master contain all server resources that are needed to service the network.

The RSCT PeerDomain (RPD) configuration is made using these assumptions and requirements regarding availability:

- ▶ The systems management personnel shall not have to restore and reconfigure the NIM server in case of hardware failure.
- ▶ The longest downtime for the NIM server shall be two hours.
- ▶ Hardware failures might occur during the NIM servers lifetime, but is unlikely.
- ▶ Planned downtime will occur during the NIM servers lifetime to install new firmware or system updates.
- ▶ In case of system failure of the acting NIM master, the RSCT peer will perform takeover of the NIM clients.

Based on the assumptions and requirements, we used the following: AIX 5L V5.3 TL5 and RSCT V2.4.5 (use SUMA to download updates to AIX 5L and RSCT respectively, see 4.10, “NIM and Service Update Management Assistant” on page 321).



After the NIM master has performed the BOS “rte” install of the LPARs, they are customized with the following NIM commands for the NIM clients.

The NIM client name in the following examples is LPAR55 as shown in Example 4-295. First it is defined in the current NIM master (hostname “master”), then a AIX 5.3 TL5 SPOT (cosi5305) and NIM lpp\_source (lpp5305) are allocated, and finally the bos\_inst operation are performed to reinstall the NIM client.

After this, some additional bos filesets are installed: updates to the rsct.core, together with the bos.sysmgt.nim.master fileset for NIM master functionality itself. However, first we want to have two RPM packages installed: the openssl and openssh.

**Note:** RPM packages should be in the NIM lpp\_source/RPMS/ppc directory (in this example the full path to this directory on the NIM master is /export/lpp\_source/lpp5305/RPMS/ppc).

The same are done for the NIM client LPAR56.

*Example 4-295 NIM master additional software updates of RSCT PeerDomain nodes*

---

```
root@master:/: nim -o define -t standalone -a if1="net_10_1_1 lpar55 0" LPAR55
root@master:/: nim -o allocate -a spot=cosi5305 -a lpp_source=lpp5305 LPAR55
root@master:/: nim -o bos_inst -a source=rte LPAR55
root@master:/: nim -o cust -a lpp_source=lpp5305 -a installp_flags=agXY -a
filesets="openssl-0.9.7d-2 openssl.base bos.adt.libm bos.adt.syscalls rsct.core
bos.sysmgt.nim.master" LPAR55
```

---

Now we are ready to configure the RSCT PeerDomain, and activate it. First we make sure that /etc/hosts on each node contain the IP-address of both peer nodes and of the client, lpar6. For our purposes we use the following IP-addresses:

```
10.1.1.55 lpar55
10.1.1.56 lpar56
10.1.1.6 lpar6
```

We also adds the NSORDER variable to /etc/environment to ensure that we use the local systems /etc/hosts file to resolve hostnames to IP-addresses, and log in again (we could also use the /etc/netsvc.conf or /etc/irs.conf files for the same purpose, see “Setting up IP name resolution” on page 62):

```
NSORDER=local,bind
```

## RSCT PeerDomain Configuration

Run the **preprnode** command on both nodes. In Example 4-296 we use the **-V** flag only with the **preprnode** command. To get more detailed output use the **-T** flag also.

*Example 4-296 Using preprnode for the RSCT PeerDomain*

---

```
root@lpar55:/: preprnode -V lpar55 lpar56
Begining to prepare nodes to add to the peer domain.
Completed preparing nodes to add to the peer domain.
```

```
root@lpar56:/: preprnode -V lpar55 lpar56
Begining to prepare nodes to add to the peer domain.
Completed preparing nodes to add to the peer domain.
```

---

Now, we create the actual RSCT PeerDomain Cluster with the **mkrpdomain** command as shown in Example 4-297. The peer domain name is **NIMESIS** and consisting of two (2) nodes: **lpar55** and **lpar56**.

*Example 4-297 Create the RSCT PeerDomain with mkrpdomain*

---

```
root@lpar55:/: mkrpdomain -V NIMESIS lpar55 lpar56
Making the peer domain "NIMESIS".
Completed making the peer domain "NIMESIS".
```

---

Check the status of the **NIMESIS** cluster with the **lsrpdomain** command as shown in Example 4-298.

*Example 4-298 Checking status of RSCT PeerDomain after mkrpdomain command*

---

```
root@lpar55:/: lsrpdomain
Name      OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
NIMESIS  Offline  2.4.5.2             No              12347   12348
```

---

Example 4-298 shows us that the **NIMESIS** cluster is **Offline**, so we should start it with the **starttrpdomain** command as shown in Example 4-299.

*Example 4-299 Starting the RSCT PeerDomain after mkrpdomain command*

---

```
root@lpar55:/: starttrpdomain -V NIMESIS
Starting the peer domain "NIMESIS".
Completed starting the peer domain "NIMESIS".
```

---

Check again, check the status of the **NIMESIS** cluster with the **lsrpdomain** command as shown in Example 4-300.

*Example 4-300 Checking status of RSCT PeerDomain after startdomain command*

```

root@lpar55:/: lsrdomain
Name OpState RSCTActiveVersion MixedVersions TSPort GSPort
NIMESIS Pending online 2.4.5.2 No 12347 12348

root@lpar55:/: lsrdomain
Name OpState RSCTActiveVersion MixedVersions TSPort GSPort
NIMESIS Online 2.4.5.2 No 12347 12348

```

Example 4-300 shows us that the NIMESIS cluster is Pending online. When we run the **lsrdomain** command a second time it has become Online and is ready for use. After starting the RSCT PeerDomain, it will take about a minute before it is Online.

If you get the following error message, when using **lsrdomain** command, just wait a second and try again:

```

/usr/sbin/rsct/bin/lrsrsrc-api: 2612-022 A session could not be
established with the RMC daemon on "local_node".

```

Now we use the **lsrpnod** command to check the individual status of our NIMESIS RSCT PeerDomain cluster nodes as shown in Example 4-301.

*Example 4-301 Checking status of RSCT PeerNodes after startdomain command*

```

root@lpar55:/: lsrpnod
Name OpState RSCTVersion
lpar55 Online 2.4.5.2
lpar56 Online 2.4.5.2

```

Both are Online as expected, with the `rsct.core` fileset level of 2.4.5.2. You can verify this with the following **lslpp** command on each node:

```

lslpp -L rsct.core.rmc

```

**Note:** To summarize: to create an RSCT PeerDomain cluster with two (2) nodes, run the following commands:

- ▶ `preprnod node1 node2` # on node1
- ▶ `preprnod node1 node2` # on node2
- ▶ `mkrpdomain RPD node2 node2`
- ▶ `startdomain RPD`

and it is done.

## Check the RSCT PeerDomain configuration

To check the RSCT PeerDomain configuration, we use the `lsrsrc` command to access specific RSCT resource managers (RM).

A resource manager is a daemon process that provides the interface between Resource Monitoring and Control subsystem (RMC) and actual physical or logical entities. Although RMC provides the basic abstractions (resource classes, resources, and attributes) for representing physical or logical entities, it does not itself represent any actual entities. A resource manager maps actual entities to RMCs abstractions. The RSCT RMC subsystem (`ctrmc`) can be monitored with the `lsrsrc` command. Example 4-302 shows the registered resource managers.

*Example 4-302 Listing dynamic information from the ctrmc with the lssrc (not lsrsrc)*

---

```
root@lpar55:/: lssrc -ls ctrmc
. . .
Resource Manager Information
```

Name	ClassKey	ID	FD	SHMID
IBM.WLMRM	1	0	-1	-1
IBM.DRM	1	1	26	3145730
IBM.ConfigRM	1	2	23	-1
IBM.ERRM	1	3	24	-1
IBM.AuditRM	1	4	20	-1
IBM.FSRM	1	5	-1	-1
IBM.HostRM	1	6	19	3145731
IBM.SensorRM	1	7	-1	-1
IBM.LPRM	1	8	29	-1
IBM.ServiceRM	1	9	27	3145729
IBM.CSMAgentRM	1	10	21	-1
IBM.HacmpRgRm	1	11	-1	-1

---

Each resource manager represents a specific set of administrative tasks or system features. The resource manager identifies the key physical or logical entity types related to that set of administrative tasks or system features, and defines resource classes to represent those entity types.

Example 4-303 shows the current registered resource managers on one of our cluster nodes, and in the following examples we will examine the `IBM.PeerDomain` and `IBM.PeerNode`, resource managers in more detail.

*Example 4-303 Listing resource managers with lsrsrc (not lssrc)*

---

```
root@lpar55:/: lsrsrc | pr -t -2
class_name                "IBM.PeerNode"
"IBM.Association"         "IBM.RSCTParameters"
```

"IBM.ATMDevice"	"IBM.NetworkInterface"
"IBM.AuditLog"	"IBM.CommunicationGroup"
"IBM.AuditLogTemplate"	"IBM.HostPublic"
"IBM.Condition"	"IBM.DRM"
"IBM.EthernetDevice"	"IBM.WLM"
"IBM.EventResponse"	"IBM.TieBreaker"
"IBM.FDDIDevice"	"IBM.LPAR"
"IBM.Host"	"IBM.LPCCommands"
"IBM.FileSystem"	"IBM.HacmpEquivalency"
"IBM.PagingDevice"	"IBM.HacmpManagedRelationship"
"IBM.PhysicalVolume"	"IBM.HacmpManagedResource"
"IBM.Processor"	"IBM.HacmpResourceGroup"
"IBM.Program"	"IBM.HacmpCluster"
"IBM.TokenRingDevice"	"IBM.HacmpNode"
"IBM.Sensor"	"IBM.HacmpNetwork"
"IBM.Sfp"	"IBM.HacmpServiceIP"
"IBM.ServiceEvent"	"IBM.HacmpApplication"
"IBM.ManagementServer"	"IBM.HacmpAppMonitor"
"IBM.PeerDomain"	"IBM.HacmpManagedStorage"

---

Using the `lsrsrc` command, we can examine both the persistent and dynamic attributes of the RMs.

The `IBM.PeerDomain` RM in Example 4-304 shows the persistent Name and its `ActivePeerDomain` called NIMESIS.

*Example 4-304 IBM.PeerDomain persistent attributes*

---

```
root@lpar55:/: lsrsrc IBM.PeerDomain
Resource Persistent Attributes for IBM.PeerDomain
resource 1:
    Name = "NIMESIS"
    RSCTActiveVersion = "2.4.5.2"
    MixedVersions = 0
    TSPort = 12347
    GSPort = 12348
    RMCPort = 657
    ResourceClasses = {}
    QuorumType = 0
    ActivePeerDomain = "NIMESIS"
```

---

Example 4-305 shows that the `IBM.PeerDomain` dynamic `OpState` is one (1), and one is good since it means that the peer domain is online, two (2) means offline.

*Example 4-305 IBM.PeerDomain dynamic attributes*


---

```

root@lpar55:/: lsrsrc -A d IBM.PeerDomain
Resource Dynamic Attributes for IBM.PeerDomain
resource 1:
    OpState          = 1
    ConfigChanged    = 0
    OpQuorumState    = 0

```

---

The IBM.PeerNode RM in Example 4-306 shows the known nodes and that they belong to the ActivePeerDomain NIMESIS. The NodeList number is the sequence number for the node in the domain, and it is the same sequence on all nodes.

*Example 4-306 IBM.PeerNode persistent attributes*


---

```

root@lpar55:/: lsrsrc IBM.PeerNode
Resource Persistent Attributes for IBM.PeerNode
resource 1:
    Name              = "lpar55"
    NodeList           = {1}
    RSCTVersion        = "2.4.5.2"
    ClassVersions      = {}
    CritRsrcProtMethod = 0
    ActivePeerDomain   = "NIMESIS"
    NodeNameList       = {"lpar55"}
resource 2:
    Name              = "lpar56"
    NodeList           = {2}
    RSCTVersion        = "2.4.5.2"
    ClassVersions      = {}
    CritRsrcProtMethod = 0
    ActivePeerDomain   = "NIMESIS"
    NodeNameList       = {"Unknown_Node_Name"}

root@lpar56:/: lsrsrc IBM.PeerNode
Resource Persistent Attributes for IBM.PeerNode
resource 1:
    Name              = "lpar55"
    NodeList           = {1}
    RSCTVersion        = "2.4.5.2"
    ClassVersions      = {}
    CritRsrcProtMethod = 0
    ActivePeerDomain   = "NIMESIS"
    NodeNameList       = {"Unknown_Node_Name"}
resource 2:
    Name              = "lpar56"

```

```

NodeList          = {2}
RSCTVersion       = "2.4.5.2"
ClassVersions     = {}
CritRsrcProtMethod = 0
ActivePeerDomain  = "NIMESIS"
NodeNameList     = {"1par56"}

```

---

Example 4-307 shows that the IBM.PeerNode dynamic OpState is one (1), and one is good since it means that the peer domain is online, two (2) means offline.

*Example 4-307 IBM.PeerNode dynamic attributes*

---

```

root@1par55:/: lsrsrc -Ad IBM.PeerNode
Resource Dynamic Attributes for IBM.PeerNode
resource 1:
    OpState          = 1
    ConfigChanged    = 0
    CritRsrcActive   = 0
resource 2:
    OpState          = 1
    ConfigChanged    = 0
    CritRsrcActive   = 0

```

---

Communication groups control how RSCT core topology services' "heartbeats" are performed between the communication resources within the peer domain. Each communication group corresponds to a RSCT core topology services' heartbeat ring, and identifies the attributes that control the checks between the nodes.

Now, we will use the environment variable CT\_MANAGEMENT\_SCOPE set to two (2), this will enable RSCT PeerDomain display when using the `lsrsrc` command as shown in Example 4-308. If it is unset it will only show the local node information.

*Example 4-308 Using CT\_MANAGEMENT\_SCOPE=2 variable*

---

```

root@1par55:/: CT_MANAGEMENT_SCOPE=2 lsrsrc -Ad IBM.PeerNode
Resource Dynamic Attributes for IBM.PeerNode
resource 1:
    OpState          = 1
    ConfigChanged    = 0
    CritRsrcActive   = 0
resource 2:
    OpState          = 1
    ConfigChanged    = 0
    CritRsrcActive   = 0

```

---

To examine the communication groups in our RSCT PeerDomain, use the **lscmg** command. In Example 4-309, there is only one communication group, CG1.

*Example 4-309 Checking the communication groups with lscmg*

---

```
root@lpar55:/: lscmg
Name Sensitivity Period Priority Broadcast SourceRouting NIMPathName NIMParameters
CG1 4 1 1 Yes Yes
```

---

To find out the interfaces used to communicate in this communication group, use the **-i** flag and specify the communication group with the **lscmg** command. In Example 4-310, CG1 communication group have two members, lpar55 and lpar56.

*Example 4-310 Checking the interfaces in a communication group*

---

```
root@lpar55:/: lscmg -i CG1
Name NodeName IPAddress Subnet SubnetMask
en0 lpar56 10.1.1.56 10.1.1.0 255.255.255.0
en0 lpar55 10.1.1.55 10.1.1.0 255.255.255.0
```

---

We can examine the communications interfaces on our cluster nodes using the **lsrsrc** command with the **IBM.NetworkInterface**, and setting the environment variable **CT\_MANAGEMENT\_SCOPE** to two (2). This will enable RSCT PeerDomain display when using the command. If it is unset it will only show the local node information. The attribute **OpState** should be one (1), which is online. See Example 4-311.

*Example 4-311 Listing interfaces in a peer domain*

---

```
root@lpar55:/: CT_MANAGEMENT_SCOPE=2 lsrsrc -A d IBM.NetworkInterface
Resource Dynamic Attributes for IBM.NetworkInterface
resource 1:
    OpState = 1
    ConfigChanged = 0
resource 2:
    OpState = 1
    ConfigChanged = 0
```

---

The RSCT core topology services subsystem (cthats) can be monitored with the **lssrc** command, and controlled with the **cthatsctrl**, **thatstune**, and **cthactrl** commands (in the **/usr/sbin/rsct/bin** directory). In Example 4-312 on page 407 you can see the communications group, CG1, has two (2) members (Mbrs) and the status (St) are stable (S). The connection are between 10.1.1.55 and 10.1.1.56, and it uses the local en0 TCP/IP interface for communication.



*Example 4-312 Checking cthats (topology services)*


---

```

root@lpar55:/: lssrc -ls cthats
Subsystem          Group          PID      Status
  cthats           cthats         286832   active
Network Name      Indx Defd  Mbrs  St  Adapter ID      Group ID
CG1                [ 0] 2      2    S   10.1.1.55       10.1.1.56
CG1                [ 0] en0          0x44a4d38b    0x44a4d397
HB Interval = 1.000 secs. Sensitivity = 4 missed beats
Missed HBs: Total: 0 Current group: 0
Packets sent      : 83 ICMP 0 Errors: 0 No mbuf: 0
Packets received: 130 ICMP 0 Dropped: 0
NIM's PID: 671826
  2 locally connected Clients with PIDs:
  rmcd(565364) hagsd(544840)
  Fast Failure Detection available but off.
  Configuration Instance = 1151652855
  Daemon employs no security
  Segments pinned: Text Data.
  Text segment size: 785 KB. Static data segment size: 1526 KB.
  Dynamic data segment size: 3009. Number of outstanding malloc: 100
  User time 0 sec. System time 0 sec.
  Number of page faults: 0. Process swapped out 0 times.
  Daemon is in a refresh grace period.
  Number of nodes up: 2. Number of nodes down: 0.

```

---

The RSCT core group services subsystem (cthags) can be monitored with the **lssrc** command, and controlled with the **cthagsctrl**, **cthagstune**, and **cthactrl** commands (in the `/usr/sbin/rsct/bin` directory). In Example 4-313, you can see that the `rmc_peers` group is registered.

*Example 4-313 Checking cthags (group services)*


---

```

root@lpar55:/: lssrc -ls cthags
Subsystem          Group          PID      Status
  cthags           cthags         544840   active
2 locally-connected clients. Their PIDs:
225376(IBM.ConfigRMd) 565364(rmcd)
HA Group Services domain information:
Domain established by node 1
Number of groups known locally: 2

```

Group name	Number of providers	Number of local providers/subscribers
rmc_peers	2	1 0
IBM.ConfigRM	2	1 0

---

To check the authentication and trusted host list created with the **preprnode** command, you can use the **ctsth1** command. Example 4-314 shows our two peer nodes, lpar55 and lpar56 with their respective IP-addresses.

*Example 4-314 Verifying ctcas trusted host list keys with ctsth1*

---

```

root@lpar55:/: ctsth1 -l
ctsth1: Contents of trusted host list file:
-----
Host Identity:          10.1.1.56
Identifier Generation Method: rsa512
Identifier Value:
120200d004f8e71a20d92484dda354a8ddc76986768fa82ff74000cf42e694f516a75c9dba14564b
0d9b8ec50e05b21460128bb9f05a9f461267fbb7212d7e2ddea0430103
-----
Host Identity:          lpar56
Identifier Generation Method: rsa512
Identifier Value:
120200d004f8e71a20d92484dda354a8ddc76986768fa82ff74000cf42e694f516a75c9dba14564b
0d9b8ec50e05b21460128bb9f05a9f461267fbb7212d7e2ddea0430103
-----
Host Identity:          10.1.1.55
Identifier Generation Method: rsa512
Identifier Value:
120200cd29169553c272c4e02dc042ba33b80430cfb761eb4034ec70f8f4f1461a42c425c37e7a81
2b1ad9e9e930d2ef659d060ee263007906ddccc0f8e8678d94f3eb0103
-----
Host Identity:          lpar55
Identifier Generation Method: rsa512
Identifier Value:
120200cd29169553c272c4e02dc042ba33b80430cfb761eb4034ec70f8f4f1461a42c425c37e7a81
2b1ad9e9e930d2ef659d060ee263007906ddccc0f8e8678d94f3eb0103
. . .

```

---

Must of the RSCT dynamic configuration is stored under the /var/ct directory. The /var/ct/cfg/ctrmc.acls file contains the access control lists (ACL) also added by the **preprnode** command as shown in Example 4-315.

*Example 4-315 RPD ctrmc.acls file (configured by preprnode command)*

---

```

none:root * rw      // give root on any node access to all
IBM.PeerDomain     // Peer Domain class
root@10.1.1.55 * rw // cluster node
root@10.1.1.56 * rw // cluster node
root@lpar55 * rw   // cluster node
root@lpar56 * rw   // cluster node

```

```
none:any_root * rw // root on any node of any cluster that this node is defined to
none:root * rw // root on any node of active cluster
```

---

## HANIM takeover operation

If a NIM client `/etc/niminfo` file looks like Example 4-316 (first part `lpar6`, after the takeover operation is performed (from `lpar55`), it changes to look like Example 4-316 (second part `lpar55`).

*Example 4-316 Changes to a NIM clients `/etc/niminfo` file by `nim -o takeover`*

---

```
{lpar6}:/ # cat /etc/niminfo
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=lpar6
export NIM_HOSTNAME=lpar6
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=lpar56
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_MASTER_HOSTNAME_LIST="lpar56 lpar55"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.6:lpar6 10.1.1.56:lpar56 "
```

```
root@lpar55:/: nim -o takeover lpar56
```

```
{lpar6}:/ # cat /etc/niminfo
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=lpar6
export NIM_HOSTNAME=lpar6
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=lpar55
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_MASTER_HOSTNAME_LIST="lpar55 lpar56"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.6:lpar6 10.1.1.55:lpar55 "
```

```
export NIM_MOUNTS=""
```

---

The `nim` command with the takeover operation are the core of what we use in our takeover script shown in Figure 4-28 on page 415.

### Creating a RSCT PeerDomain event detection method

Now that we have a working RSCT PeerDomain, and we know how to handle the NIM takeover operation, we can perform the steps to allow RSCT RMC event handling to perform the action for us. It involves the following steps and commands:

1. Create a program to perform actions when the intended conditions are met.
2. `mkresponse` - Define a program to execute
3. `mkcondition` - Define a condition to monitor
4. `mkcondresp` - Connect the condition with the response program
5. `startcondresp` - Start monitoring the metrics in the condition

**Attention:** The type of condition and response we are using in this scenario with RSCT is a one sided condition-response, and all commands must be performed on both peer nodes. However, the `mkcondition` command will be different between the nodes (explained in Example 4-317).

First we create a script, `/local/rg/takeover.nimesis`, that will execute the `nim` command with the takeover operation. See Figure 4-28 on page 415. We recommend using the `vi` editor, both for script creation and for command line editing.

Then we define the response with the `/local/rg/takover.nimesis` script, naming the response `START_NIMESIS`, and the action `TAKEOVER`. Additional actions can be added to the response with the `chresponse` command, but for our purposes we do not need to use this feature.

#### *Example 4-317 Creating a response with mkresponse*

---

```
root@lpar55:/: mkresponse -n TAKEOVER -s /local/rg/takeover.nimesis START_NIMESIS
```

```
root@lpar55:/: lsresponse START_NIMESIS
```

```
Displaying response information:
```

```
ResponseName = "START_NIMESIS"
Node         = "lpar55"
Action       = "TAKEOVER"
DaysOfWeek   = 1-7
```

```

TimeOfDay      = 0000-2400
ActionScript   = "/local/rg/takover.nimesis"
ReturnCode     = 0
CheckReturnCode = "n"
EventType      = "a"
StandardOut    = "n"
EnvironmentVars = ""
UndefRes       = "n"

```

---

Now we create the condition to evaluate. This part is where you have to find the appropriate attribute (metric) from the appropriate resource managers resource class. Use the `lsrsrc` command to find what you need.

When you know what you want to monitor, you just specify the expression to the `mkcondition` command, with the `-e` flag and the `rearm` expression with the `-E` flag. In our case we will monitor the `OpState` attribute in the `IBM.PeerNode` resource class for the remote peer. When it is set to one (1), the peer is online, and when it is *not* set to one (1) it is *not* online, it can be unconfigured, pending or in some error state. However, it has to be online and nothing but online.

When run the `mkcondition` command on our peer node `lpar55`, we use `“-n lpar56”` but when we run the command on node `lpar56`, we use `“-n lpar55”`. The `-n` flag specifies where the condition will be monitored as shown in Example 4-318.

*Example 4-318 Creating a condition with mkcondition*

---

```

root@lpar55:/: mkcondition -mp -Sc -n "lpar56" -r IBM.PeerNode -e 'OpState <> 1'
-E 'OpState == 1' PEER_DOWN

```

```

root@lpar55:/: lsccondition PEER_DOWN
Displaying condition information:

```

```

condition 1:
Name           = "PEER_DOWN"
Node           = "lpar55"
MonitorStatus  = "Not monitored"
ResourceClass  = "IBM.PeerNode"
EventExpression = "OpState <> 1"
EventDescription = ""
RearmExpression = "OpState == 1"
RearmDescription = ""
SelectionString = ""
Severity       = "c"
NodeNames      = {"lpar56"}

```

MgtScope = "p"

---

Now we have a response and we have a condition. To get the response to be performed when the condition becomes true, we need to create an association between the response and condition. This is done with the **mkcondresp** command. But do not worry, the condition will not be monitored just by creating an association as shown Example 4-319.

*Example 4-319 Associating a condition with a response*

---

```
root@lpar55:/: mkcondresp PEER_DOWN START_NIMESIS
```

```
root@lpar55:/: lscondresp PEER_DOWN
Displaying condition with response information:
Condition  Response      Node      State
"PEER_DOWN" "START_NIMESIS" "lpar55" "Not active"
```

---

In order for the condition to be monitored, we start it with the **startcondresp** command as shown in Example 4-320. Now when the condition becomes true, the associated response is activated.

*Example 4-320 Starting monitoring a condition with startcondresp*

---

```
root@lpar55:/: startcondresp PEER_DOWN
```

```
root@lpar55:/: lscondresp PEER_DOWN
Displaying condition with response information:
```

```
condition-response link 1:
    Condition = "PEER_DOWN"
    Response  = "START_NIMESIS"
    Node      = "lpar55"
    State     = "Active"
```

```
root@lpar55:/: lsrsrc -Ad IBM.PeerNode
Resource Dynamic Attributes for IBM.PeerNode
resource 1:
```

```
    OpState      = 1
    ConfigChanged = 0
    CritRsrcActive = 0
```

```
resource 2:
```

```
    OpState      = 1
    ConfigChanged = 0
    CritRsrcActive = 0
```

---

Now we have a metric that is being monitored, and when our condition is met and evaluates to true, our response is activated. This is our `takover.nimesis` script as shown in Figure 4-28 on page 415.

To verify it, we can use the `stoprnode` command to bring the remote node offline from our RSCD domain as shown in Example 4-321. This will cause the `OpState` attribute in `IBM.PeerNode`, to change from online (1), to offline (2).

*Example 4-321 Stopping a RSCD PeerDomain node with stoprnode*

---

```
root@lpar55:/: stoprnode -V lpar56
Stopping the node(s).
Stopping these node(s) in the peer domain. lpar56
Completed stopping the node(s).
```

---

Now we check the status with the `lsrc` command as shown in Example 4-322. The first time `OpState` has changed from one (1) to six (6), then shortly after it has changed to two (2) and are offline. We can verify that it is offline with the `lsrc` command, but our PeerDomain should still be active which is seen from the output of the `lsrcdomain` command.

*Example 4-322 Checking the progress of stoprnode of the peer node*

---

```
root@lpar55:/: lsrc -Ad IBM.PeerNode
Resource Dynamic Attributes for IBM.PeerNode
resource 1:
    OpState          = 6
    ConfigChanged    = 0
    CritRsrcActive   = 0
resource 2:
    OpState          = 1
    ConfigChanged    = 0
    CritRsrcActive   = 0

root@lpar55:/: lsrc -Ad IBM.PeerNode
Resource Dynamic Attributes for IBM.PeerNode
resource 1:
    OpState          = 1
    ConfigChanged    = 0
    CritRsrcActive   = 0
resource 2:
    OpState          = 2
    ConfigChanged    = 0
    CritRsrcActive   = 0

root@lpar55:/: lsrnode
```

```
Name OpState RSCTVersion
lpar55 Online 2.4.5.2
lpar56 Offline 2.4.5.2
```

```
root@lpar55:/: lsrpdomain
```

```
Name OpState RSCTActiveVersion MixedVersions TSPort GSPort
NIMESIS Online 2.4.5.2 No 12347 12348
```

---

When we restart the peer node, the condition will be rearmed, and ready to go again.

```
starttrpnode lpar56
```

## T takeover script for HANIM

Now lets look at our RPD script. We use one script to start the NIM services, or rather to change the /etc/nimfo file on the NIM clients defined in the NIM database.

The takeover script will make sure that the `nimesis` SRC daemon is started and will collect some information regarding the running cluster, before it will issue a `nim` command ordering takeover from the alternate NIM master (just in case it was the current master for the NIM clients).

We have created the /local file system on each node, and a directory called "rg" in this file system. We place the scripts in the /local/rg directory and make sure they are synchronized between the cluster nodes.

When activated by the Event Response Resource Manager (ERRM), the script will be supplied with several environment variables, with names starting with "ERRM\_". We are mostly interested in the following of those:

The `ERRM_RSRC_CLASS_NAME`, `ERRM_COND_NAME` and `ERRM_ER_NAME` to identify the RSRC RM class set by the condition, the condition name, and the event name. These variables can be used to identify, in the program, what condition activated the it, in case the same program are used to handle several conditions. In our case, the variables will have the following values when the script is activated as shown in Example 4-323.

*Example 4-323 ERRM environment variables*

---

```
ERRM_RSRC_CLASS_NAME=PeerNode
ERRM_COND_NAME=PEER_DOWN
ERRM_ER_NAME=START_NIMESIS
ERRM_EXPR=OpState <> 1
ERRM_VALUE=6
```

---



The `ERRM_EXPR` variable contain the condition expression that became true and activated the response event to start our program. The actual value of the `OpState` variable, at the time of setting of the trigger, is supplied in the `ERRM_VALUE` variable, and in our case the six (6) means a transitory state from online to offline.

```
#!/bin/ksh
# takeover.nemesis
# Synopsis.....%M%
# Author.....The Dude!
# Created.....2006
# Version.....%Z% %M% %I% (%E% %U%) %Q%
# Description....N/A
# Input.....N/A
# Output.....Change the /etc/niminfo on all clients:
#               NIM_MASTER_HOSTNAME, NIM_MASTER_HOSTNAME_LIST and
#               NIM_HOSTS variables.
# Algorithm.....Check if nimesis are running, start it if not,
#               perform takeover regardless.
#
#-----
trap 'exit 0' EXIT

export PATH=/usr/bin:/usr/sbin:/usr/es/sbin/cluster/utilities

NIMESISPID=$(lssrc -s nimesis|awk '!/PID/{if (/active/) print $3}')

if [[ -z "$NIMESISPID" ]];then
    printf "The \"nimesis\" SRC will be started.\n"
    startsrc -s nimesis
    sleep 6
else
    printf "The nimesis SRC is already running \"$NIMESISPID\".\n"
fi

ALTERNATE_MASTER=$(lsnim -Z -t alternate_master|awk -F: '!/^#/{print $1;exit}')

printf "The NIM alternate_master is \"$ALTERNATE_MASTER\".\n"

[[ -z "$ALTERNATE_MASTER" ]] || nim -o takeover $ALTERNATE_MASTER
```

Figure 4-28 RSCT PeerDomain takeover.nimesis script

In the script, we end with a trap to ensure we return zero (0) as exit value.





# Network Installation Manager (NIM) best practices

This chapter provides some examples of how to configure and use NIM based on best practices and experiences of the redbook authors and contributors:

- ▶ Updating the NIM environment (lpp\_source, SPOT)
- ▶ Secondary adapter
- ▶ NIMSH, OpenSSL and firewall considerations
- ▶ “Cloning” NIM clients using mksysb
- ▶ Using NIM to migrate systems to new hardware
- ▶ Using SAN zoning for “cloning” LPARs
- ▶ System maintenance for NIM clients
- ▶ Automatic scripts
- ▶ Implementing a standard operating environment for AIX 5L V5.3 systems with NIM
- ▶ How to backup and re-install a NIM master

## 5.1 Updating the NIM environment (lpp\_source, SPOT)

Whenever there is a need to update the technology level (TL) or Service Pack (SP) in any NIM client, the first thing you need to do is to update your NIM master server. After the NIM master server is updated, you will need to update the lpp\_source follow by the SPOT.

**Note:** The NIM server must be at the same or higher level than the NIM clients. You need to upgrade your NIM server before upgrading your NIM client.

If your NIM environment has a mixture of TL levels, in order to have some NIM clients at the original level, you need to keep a copy of the original lpp\_source and SPOT level. These clients will still need the original level of lpp\_source and SPOT.

In the next section, we describe the steps to perform the lpp\_source update. In this example, we have an original lpp\_source and SPOT level at TL04, and we are creating a new TL05 lpp\_source and SPOT.

1. Create a new AIX 5L V5.3 TL05's lpp\_source directory, for example, /export/lppsource/lpp-aix5305, and copy the AIX 5L V5.3 TL04's lpp\_source directory into it.

```
# cp -rp /export/lppsource/lpp-aix5304/* \
/export/lppsource/lpp-aix5305
```

**Attention:** Make sure that the files in /export/lppsource/lpp-aix5304/install/ppc have the "Read" permission for everybody. If not, use the **chmod 644 \*** command inside the lpp\_source directory. Also make sure the ownership of the files are root:system.

2. Create an AIX 5L V5.3 TL05's lpp\_source resource from the AIX 5L V5.3 TL05's lpp\_source directory as shown in Example 5-1 on page 419. The following steps are followed when using the SMIT menus:

```
# smitty nim
  Perform NIM Administration Tasks
    Manage Resources
      Define a Resource
        Select "lpp_source = source device for optional product
        images"
```

Or using SMIT Fast Path, **smitty nim\_mkres**

*Example 5-1 Creating an AIX 5L V5.3 TL05 lpp\_source*


---

Define a Resource

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

	[EntryFields]
* Resource Name	[LPP-AIX5305]
* Resource Type	lpp_source
* Server of Resource	[master] +
* Location of Resource	[/export/lppsource/lpp-aix5305 /
Architecture of Resource	[ ] +
Source of Install Images	[ ] +/
Names of Option Packages	[ ]
Show Progress	[yes] +
Comments	[ ]

---

You can also use the command line as follows:

```
# nim -o define -t lpp_source -a server=master -l \  
/export/lppsource/lpp-aix5305 LPP-AIX5305
```

**Note:** NIM updates the .toc file in <lppsource>/install/ppc automatically. Make sure the .toc file in /export/lppsource/lpp-aix5305/install/ppc is updated.

- Update the TL05 to the LPP-AIX5305 resource as shown in Example 5-2. You use the NIM update operation to perform the task. This operation allow you to update or remove any packages from the lppsource resource. The next steps are followed when using the SMIT menus:

```
# smitty nim
Perform NIM Administration Tasks
  Manage Resources
    Perform Operations on Resources
      Select LPP-AIX5305
        Select "update = add or remove software to or from an
          lpp_source"
          Select "Add" and enter the Software media source
```

Or using SMIT Fast Path, **smitty nim\_update**

*Example 5-2 Adding software to an lpp\_source*


---

Add Software to an lpp\_source

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

	[EntryFields]
TARGET lpp_source	LPP-AIX5305
SOURCE of Software to Add	cd0
SOFTWARE Packages to Ad	[all]+
-OR-	
INSTALLP BUNDLE containing packages to add	[] +
gencopy Flags	
DIRECTORY for temporary storage during copying	[/tmp]
EXTEND filesystems if space needed?	yes +
Process multiple volumes?	no +

---

You can also use the command line as follows:

```
# nim -o update -a packages=all -a source=/dev/cd0 LPP-AIX5305
```

After you have updated the TL05 lpp\_source, there might be some duplicate updates. You can perform the NIM `lppmgr` operation to remove them. This operation enables you remove any duplicate software, superseded updates, any unnecessary languages and any non-simage software.

We recommend to remove the duplicate software and any superseded updates, but not remove any language software and non simage filesets unless you are sure that they are not needed for future use. We also recommend you to perform a preview to check what are the software that will be removed before the actual removal.

4. Remove any duplicate updates in lpp\_source resource as shown in Example 5-3.

The following steps are followed when using the SMIT menus:

```
# smitty nim
Perform NIM Administration Tasks
  Manage Resources
    Perform Operations on Resources
      Select LPP_AIX5305
        Select "lppmgr = eliminate unnecessary software images
          in an lpp_source"
```

Or using SMIT Fast Path, `smitty nim_lppmgr`

*Example 5-3 Removing duplicate updates in the lpp\_source*

---

Eliminate Unnecessary Software Images in an lpp\_source

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

	[EntryFields]
TARGET lpp_source	LPP-AIX5305
PREVIEW only?	no +
REMOVE DUPLICATE software	yes +
REMOVE SUPERSEDED updates	yes +
REMOVE LANGUAGE software	no +
PRESERVE language	[en_US]
REMOVE NON-SIMAGES software	no +
SAVE removed files	no +
DIRECTORY for storing saved files	[]
EXTEND filesystems if space needed?	yes +

---

You can also use the command line as follows:

```
# nim -o lppmgr -a lppmgr_flags="-bu -x -r -e" LPP-AIX5305
```

**Attention:** Please run `lsnim -l LPP-AIX5305` to check the Rstate and simages. Rstate should show “Ready for use” and simages = yes.

Next, we need to create the AIX 5L V5.3 TL05’s SPOT. We will create the SPOT using the latest LPP-AIX5305 lpp\_source.

5. Create the AIX 5L V5.3 TL05’s SPOT as shown in Example 5-4. The following steps are followed when using the SMIT menus:

```
# smitty nim_mkres
Select “spot = Shared Product Object Tree - equivalent to /usr
file”
```

#### *Example 5-4 Creating a SPOT*

---

##### Define a Resource

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

	[Entry Fields]
* Resource Name	[SPOT-AIX5305]
* Resource Type	spot
* Server of Resource	[master] +
* Source of Install Images	[LPP-AIX5305] +
* Location of Resource	[/export/spot]/
EXPAND file systems if space needed?	yes +
Comments	[]

installp Flags	
COMMIT software updates?	no +
SAVE replaced files?	yes +
AUTOMATICALLY install requisite software?	yes +
OVERWRITE same or newer versions?	no +
VERIFY install and check file sizes?	no +

---

Once the updated lpp\_source and SPOT are created, you can perform a NIM check operation to check the usability of these resources. When you perform the NIM check operation on the lpp\_source resource, it rebuilds the Table of Contents (.toc) file in the lpp\_source directory and check whether all the necessary filesets in the directory to qualify the lpp\_source for the simage attribute.

6. Check the usability of the lpp\_source and SPOT as shown in Example 5-5. The following steps are followed when using the SMIT menus:

```
# smitty nim_res_op
  Select SPOT-AIX5305 Or LPP-AIX5305
  Select "check = check the status of a NIM object"
```

*Example 5-5 Checking the SPOT*

---

Check the Status of a SPOT

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

	[EntryFields]
* Resource Name	SPOT-AIX5305
Build Debug Boot Images?	no +
Force	no +

---

After the lppsource and SPOT resources are updated and ready for use, you can perform software maintenance on NIM clients. Please refer to "Client initiated software maintenance tasks ("pull")" on page 505 for details.



## 5.2 Secondary adapter

The following section describes the secondary adapter support for the Network Installation Manager (NIM).

### 5.2.1 Secondary adapter support

Secondary adapter support is available for AIX 5L V5.2 or later. Prior to AIX 5L V5.2, during a Network Installation Manager (NIM) BOS rte installation operation, only the network adapter and interface used during BOS installation was configured. Using the NIM secondary adapter definitions, you can have additional network adapters and interfaces configured during a BOS installation or a customized installation.

The **nimadapters** command parses a secondary adapter stanza file to build the files required to add NIM secondary adapter definitions to the NIM environment as part of an `adapter_def` resource. The **nimadapters** command does not configure secondary adapters. The configuration takes place during a **nim -o bos\_inst** operation or a **nim -o cust** operation that references the `adapter_def` resource.

The secondary adapter stanza file is processed by the **nimadapters** command and turned into a file that contains one stanza for each secondary adapter or interface on the NIM client. During a BOS installation, NIM processes this information and configures the secondary adapters. If a secondary adapter is already configured in the requested manner, NIM does not reconfigure the secondary adapter.

**Note:** NIM uses the `/usr/lpp/bos.sysmgt/nim/methods/c_cfgadptrs` client method to configure secondary adapters.

Table 5-1 on page 424 shows the flags accepted by the **nimadapters** command, and the purpose of each flag.

Table 5-1 *nimadapters* command options

Flag	Purpose
-a	<p>Assigns the following attribute=value pairs:</p> <ul style="list-style-type: none"> <li>▶ client=nim_client_name</li> </ul> <p>Specifies the NIM client that will have a secondary adapter definition added or removed. This option allows you to define one secondary adapter for a client. To define multiple secondary adapters, use a stanza file.</p> <ul style="list-style-type: none"> <li>▶ info=AttributeList</li> </ul> <p>When previewing or defining a secondary adapter, the info attribute must be used when the client attribute is specified. AttributeList is a list of attributes separated by commas.</p>
-d	<p>Defines secondary adapters. A Client.adapter file is created in the adapter_def location for each valid secondary adapter definition. If the <b>nimadapters</b> command encounters existing secondary adapter definitions for a NIM client, the existing definitions are replaced.</p>
-f	<p>SecondaryAdapterFileName Specifies the name of the secondary adapter file.</p>
-p	<p>Displays a preview operation to identify any errors. This flag processes the secondary adapter file or info attribute but does not add adapter definitions to the NIM environment.</p>
-r	<p>Removes the secondary adapter definitions of a specific client or all the clients listed in a secondary adapter stanza file. If the client attribute or secondary adapter stanza file are not specified, then all the secondary adapter definitions in the adapter_def resource will be removed.</p>

**Note:** Before using the **nimadapters** command, you must configure the NIM master.

## 5.2.2 Working with secondary adapter file rules

The format of the secondary adapter file must comply with the following rules:

- ▶ After the header stanza, the attribute lines format is : Attribute = Value.
- ▶ If the value of an attribute is defined multiple times within the same stanza, only the last definition is used.
- ▶ If an invalid attribute value or name is used, that attribute definition is ignored.
- ▶ Each line of the file can contain only one header or attribute definition.
- ▶ More than one stanza can exist in a definition file for each machine (host name).

- ▶ Each stanza for a machine (host name) represents a secondary adapter definition on that NIM client. No two secondary adapter definitions for the same machine host name can have the same location or interface\_name. There should be only one definition per adapter or interface on a given NIM client.
- ▶ If the stanza header entry is the default keyword, this specifies to use that stanza for the purpose of defining default values.
- ▶ A default value for any secondary adapter attribute can be specified. However, the netaddr and secondary\_hostname attributes must be unique. Also, the location and interface\_name attributes must be unique on a NIM client.
- ▶ If an attribute for a secondary adapter is not specified but default value exists, the default value is used.
- ▶ Default values at any location in the definition file can be specified and changed. After a default value is set, it applies to all definitions that follow.
- ▶ To turn off a default value for all following machine definitions, the attribute value in a default stanza must not be set.
- ▶ To turn off a default value for a single machine definition, the attribute value in the machine stanza must not be set.
- ▶ Comments in a client definition file can be included. Comments begin with the number sign (#).
- ▶ When parsing the definition file for header and attribute keywords and values, tab characters and spaces are ignored.

**Note:** During a `nim -o bos_inst` or `nim -o cust` operation, if NIM examines the configuration data on the client and determines that a secondary adapter is already configured with precisely the attributes requested in the adapter\_def resource, this secondary adapter is not re configured.

### 5.2.3 Secondary adapter files

This section describes secondary adapter files.

#### **Using secondary adapter file keywords**

The secondary adapter file uses keywords (Table 5-2) to specify machine attributes.

#### ***Using required adapter attributes***

Table 5-2 shows the required attributes used in configuring adapters.

Table 5-2 Required adapter attributes

Attribute	Purpose
machine_type = secondary   etherchannel   install	Specifying the machine_type attribute as secondary clearly distinguishes the nimadapters input from nimdef input. If a secondary adapter's file is mistakenly passed to the nimdef command, the error can be detected. The etherchannel option is only supported on clients running AIX 5L V5.3 or later. Stanzas with a machine_type of install are ignored.
netaddr	Specifies the network address for the secondary adapter.
interface_type = en   et   sn   ml   vi	Specifies the type of network interface. The network interface can be en (Ethernet interface), et (Ethernet interface), sn (switch network interface), ml (multi-link interface), or vi (virtual interface). This attribute is only supported on clients running AIX 5L V5.3 or later, and it replaces the deprecated network_type attribute.
subnet_mask	Specifies the subnet mask used by the secondary adapter.

### Using optional attributes

Table 5-3 on page 426 shows the optional secondary adapter attributes.

Table 5-3 Optional adapter attributes

Attribute	Purpose
adapter_attributes	Blank-separated list of physical adapter attributes and values. For example, Attribute1=Value1 Attribute2=Value2. To see the list of attributes that can be set for the requested adapter, run the command lsattr -E -l AdapterName. This attribute is only supported on clients running AIX 5L V5.3 or later.
interface_attributes	Blank-separated list of interface attributes and values. For example, Attribute1=Value1 Attribute2=Value2. To see the list of attributes that can be set for the requested interface, run the command lsattr -E -l InterfaceName. This attribute is only supported on clients running AIX 5L V5.3 or later, and it replaces the deprecated attributes attribute.
cable_type	Specifies the cable type (optional if network_type is en or et).
comments	Specifies a comment to include in the secondary adapter definition. Enclose the comment string in quotation marks.

Attribute	Purpose
interface_name	Specifies the name of the network interface for the secondary adapter (for example, en1, sn0, ml0). Do not specify both location and interface_name.
location	Specifies the physical location of the adapter corresponding to this network interface. Do not specify both the location and the interface_name attributes.
multiple_physloc	Specifies the physical adapters to associate with an interface when you use an etherchannel or VIPA stanza.
media_speed	Specifies the media speed (optional if the network_type attribute's value is either en or et).
secondary_hostname	Host name to save in the /etc/hosts file with the netaddr attribute. This host name is not set using the hostname command or the uname -S command.

### **Working with deprecated attributes**

Table 5-4 shows attributes that were deprecated in AIX 5L V5.3, but are still available on clients running AIX 5L V5.2 or later.

**Note:** If you have a NIM secondary adapter configuration with an AIX 5L V5.2 client, then you must use network\_type and attributes because the new attributes are not supported.

Table 5-4 *Deprecated attributes*

Attribute	Purpose
network_type	Replaced by interface_type.
attributes	Replaced by interface_attribute.

**Note:** Configuring a secondary adapter sharing the same subnet with another adapter does not provide failover. Packets alternate between adapters when they are configured on the same subnet. If one of the adapters fails, the other adapter will not take over the failed adapter's workload, and the subnet will have connectivity problems. Commands, such as **mount**, might fail if this occurs.

### **Sample secondary adapter file**

Example 5-6 shows a sample file for the customization of a secondary adapter.

*Example 5-6 Secondary adapter file sample*

---

```
# Set default values.
default:
    machine_type = secondary
    subnet_mask  = 255.255.240.0
    network_type = en
    media_speed  = 100_Full_Duplex
# Define the machine "lab1"
# Take all defaults and specify 2 additional attributes.
# Unlike the case of the client definitions that are input to the
# nimdef command, the secondary adapter definition includes at least
# one required field that cannot be defaulted.
lab1:
    netaddr = 9.53.153.233
    location = P2-I1/E1
# Change the default "media_speed" attribute.
default:
    media_speed = 100_Half_Duplex
# define the machine "test1"
# Take all defaults and include a comment.
test1:
    comments = "This machine is a test machine."
# define a machine with a VIPA interface that uses interfaces en2 and
# en3.
lab2:
    machine_type      = secondary
    interface_type    = vi
    interface_name    = vi0
    netaddr           = 9.53.153.235
    subnet_mask       = 255.255.255.0
    secondary_hostname = lab3
    interface_attributes = "interface_names=en2,en3"
# define a machine with an etherchannel adapter that uses the adapters
# at the following location codes P1-I4/E1 and P1/E1
lab4:
    machine_type      = etherchannel
    interface_type    = en
    interface_name    = en2
    netaddr           = 9.53.153.237
    subnet_mask       = 255.255.255.0
    multiple_physloc  = P1-I4/E1,P1/E1
# define a machine with an etherchannel adapter that uses the
# ent2 and ent3 adapters and uses mode 8023ad.
lab6:
```

```

machine_type      = etherchannel
interface_type    = en
interface_name    = en2
netaddr           = 9.53.153.239
subnet_mask       = 255.255.255.0
adapter_attributes = "adapter_names=ent2,ent3 mode=8023ad"

```

---

## Working with secondary adapter definitions

The following steps are used to accomplish common tasks working with secondary adapter definitions:

- ▶ To preview the secondary\_adapters.defs client definition file, type:
 

```
nimadapters -p -f secondary_adapters.defs adapter_def
```
- ▶ To add the NIM secondary adapters described in the secondary\_adapters.defs secondary adapters definition file, type:
 

```
nimadapters -d -f secondary_adapters.defs adapter_def
```
- ▶ To define the NIM secondary adapters for the pilsner client, type:
 

```
nimadapters -d \
-a info="en,P2-I1/E1,N/A,100_Full_Duplex,9.53.53.15,255.255.254.0" \
-a client=pilsner adapter_def
```
- ▶ To remove the NIM secondary adapter definitions for a client called pilsner from the my\_adapter\_def resource, type:
 

```
nimadapters -r -a client=pilsner my_adapter_def
```
- ▶ To remove the NIM secondary adapter definitions for clients defined in the file secondary\_adapters.defs, type:
 

```
nimadapters -r -f secondary_adapters.defs my_adapter_def
```
- ▶ To remove all the NIM secondary adapter definitions from the my\_adapter\_def resource, type:
 

```
nimadapters -r my_adapter_def
```

### 5.2.4 Configuring a secondary adapter

This section shows a practical example of how to configure a secondary adapter. This environment comprises two machines. Machine lpar2 which is the NIM master and vpar3 which is the client.

## Defining adapter\_def resource

The following procedure shows the steps executed for the creation of an adapter\_def resource:

- ▶ Type **smitty nim\_res** in the NIM master.
- ▶ Select **Define a Resource** from the Manage Resources SMIT panel.
- ▶ Select **adapter\_def** from the list that shows up.
- ▶ Type the Resource Name (in this example we choose SEC\_ADAPTR)
- ▶ Type the Location of Resource (in this example /other\_res/SEC\_ADAPTR)

### *Example 5-7 Adapter\_def resource creation*

---

* Resource Name	[SEC_ADAPTR]
* Resource Type	adapter_def
* Server of Resource	[master] +
* Location of Resource	[/other_res/SEC_ADAPTR] /
Comments	[]

---

Example 5-7 shows the SMIT panel used to create the adapter\_def resource.

## Creating a secondary adapter definition file

The following procedure shows the steps executed for the creation of an adapter definition file:

- ▶ Type **smitty nim\_adapt\_def** in the NIM master.
- ▶ Select **Define Secondary Definition File** from the Manage Secondary Adapter Definition Files SMIT panel.
- ▶ Select the Adapter Definition Resource for the adapter.
- ▶ Select the Client Name.
- ▶ Select the Network Address.
- ▶ Select the Interface Type
- ▶ Select the Subnet Mask

The rest of the fields are blank for the purposes of this example.

Example 5-8 shows the SMIT panel used to create the adapter definition file.

### *Example 5-8 Adapter definition file creation*

---

* Adapter Definition Resource	[SEC_ADAPTR] +
* Client Name	[VLPAR3] +



Machine Type	secondary
* Network Address	[10.1.1.75]
* Interface Type	[en] +
* Subnet Mask	[255.255.255.0]
Interface Attributes	[]
Cable Type	[] +
Interface Name	[]
Location	[]
Media Speed	[] +
Secondary Hostname	[]
Adapter Attributes	[]
Multiple Location Codes	[]
Comments	[]

---

SMIT uses the **nimadapters** command to generate the NIM secondary adapter definitions to the NIM environment as part of the adapter\_def resource. This command can also be executed using the command line instead of SMIT, either giving the **nimadapters** command a secondary adapter stanza file or all the necessary options.

Example 5-9 shows the output of the command execution for the creation of the adapter definition file.

*Example 5-9 Adapter definition file creation output*

---

1 secondary adapter definition is complete. The following secondary adapter will be added to the NIM environment:

```
VLPAR3:
  hostname=VLPAR3
  machine_type=secondary
  interface_type=en
  hostaddr=10.1.1.63
  netaddr=10.1.1.75      v1par3_en1
  subnet_mask=255.255.255.0
  adapter_attributes=secondary
```

Summary

1 Secondary adapter will be added to the NIM environment.

Summary

1 Secondary adapter will be added to the NIM environment.

---

The *VLPAR3.adapters* file is created in the /other\_res/SEC\_ADAPTR directory. Example 5-10 shows the content of the adapter definition file.

*Example 5-10 Adapter definition file content*

---

```
VLPAR3:
```

```

hostname=VLPAR3
machine_type=secondary
interface_type=en
hostaddr=10.1.1.63
netaddr=10.1.1.75      v1par3_en1
subnet_mask=255.255.255.0
adapter_attributes=secondary

```

---

NIM customization must be performed now in order to apply these definitions. Example 5-11 shows the execution of the customization to enable the secondary adapter in the client.

*Example 5-11 Client customization*

---

```

(root@lpar2):/ # nim -o cust -a adapter_def=SEC_ADAPTR VLPAR3
nim_name = VLPAR3
machine_type = secondary
interface_type = en
network_type =
logical_name =
location =
multiple_physloc =
secondary_hostname =
netaddr = 10.1.1.75
subnet_mask = 255.255.255.0
cable_type =
media_speed =
attributes =
interface_attributes =
adapter_attributes = secondary
bos_preconfig =
cust_preconfig =
route =
Could not get the current value of "secondary" for ent1. Will
  attempt to set "secondary".

Network device busy. Changing "ent1". Will take effect on the next
reboot.

en1
en1 changed

```

---

At this point, the client v1par3 is configured with two interfaces, en0 and en1. The interface en1 has been configured through NIM customization.

## 5.3 NIMSH, OpenSSL and firewall considerations

Up to the actual release, by default, the Network Installation Manager (NIM) makes use of the remote shell server (rshd) when it performs remote command execution on clients. The server provides remote execution facilities with authentication based on privileged port numbers from trusted hosts. However, starting with AIX 5L V5.2 ML 07 and AIX 5L V5.3, a new feature called the NIM Service Handler (NIMSH) is available. This eliminates the need for classic “r” commands during NIM client communication. For environments where the standard rsh protocols are not considered secure enough, nimsh should be implemented. The NIM client daemon (NIMSH) uses reserved ports 3901 and 3902, and it installs as part of the bos.sysmgt.nim.client fileset.

NIMSH provides a “wrapper” for classic “r” commands. Only the commands registered with NIMSH (residing in /usr/lpp/bos.sysmgt/nim/methods directory) are executed as root, anything else is denied execution.

NIMSH allows you to query network machines by hostname. NIMSH processes query requests and returns NIM client configuration parameters used for defining hosts within a NIM environment. Using NIMSH, you can define NIM clients without knowing any system or network-specific information.

While NIMSH eliminates the need for rsh, in the default configuration it does not provide trusted authentication based on key encryption. To use cryptographic authentication with NIMSH, you can configure NIMSH to use OpenSSL in the NIM environment. When you install OpenSSL on a NIM client, SSL socket connections are established during NIMSH service authentication. Enabling OpenSSL provides SSL key generation and includes all cipher suites supported in SSL version 3.

There are two ports involved in NIMSH communication. These ports are referred to as the primary (port 3901) and secondary port (port 3902). The primary port listens for service requests. When a request is accepted, the primary port is used for stdin(0) and stdout(1), while stderr(2) is redirected to the secondary port. This implementation allows the NIM master connection to stay consistent with current support of client connections through rsh. Using a reserved secondary port in NIMSH allows firewall administrators to write firewall rules for accepting incoming connections on privileged ports from the secondary port. This rules can have the requirement that the originating socket address (hostname : secondary port) comes from a trusted source.

NIMSH is registered with the System Resource Controller (SRC). The SRC group name is nimclient and the subsystem defined is NIMSH. The client daemon is started by SRC when the configuration routine is run using the **nimclient** command.

It is possible to have an heterogeneous environment of clients using the remote shell server (rshd) and NIM Service Handler (NIMSH) although it is not possible to perform operations for which information is provided using NIMSH in those clients which are not using NIMSH. For example, the **nimquery** command is not able to retrieve information from those clients which are not using NIMSH.

**Note:** The NIM client daemon logs data, used only for debug purposes, in the `/var/adm/ras/nimsh.log` file.

### 5.3.1 Authentication process

NIMSH handles service requests similar to the way they are handled in `rcmd()`. The NIM master builds packets with the following data for authentication:

- ▶ Host name of NIM client.
- ▶ CPUID of NIM client.
- ▶ CPUID of NIM master.
- ▶ Return port for secondary (stderr) connection.
- ▶ Query flag.

If the Query flag is set to “1”, the **nimsh** daemon treats the incoming request as a client discovery for information. The following data is returned:

- ▶ Default host name obtained from `inet0`.
- ▶ Default route obtained from `inet0`.
- ▶ Network address obtained from host name.
- ▶ Network interface obtained from host name.

If the query flag is not set, then a request for service (NIM operation) is pushed by the NIM master. The `nimshd` daemon validates the method request as follows:

- ▶ Verify the host name of the NIM master is the recognized master host name to the client.
- ▶ Check the client CPUID passed in the authentication data. It should match the client’s machine ID.
- ▶ Check the master CPUID passed in the authentication data. It should match the master’s machine ID stored in the memory. It is read into the memory from the `/etc/niminfo` file and the `mktcpip -S primary_nim_interface` command outputs.
- ▶ Verify that the operation passed in the method is a method residing in the `/usr/lpp/bos.sysmgmt/nim/methods` directory (path).

- ▶ Check for cryptographic authentication settings.

Figure 5-1 shows the connections involved in performing a NIM push operation when the client is configured to use NIMSH.

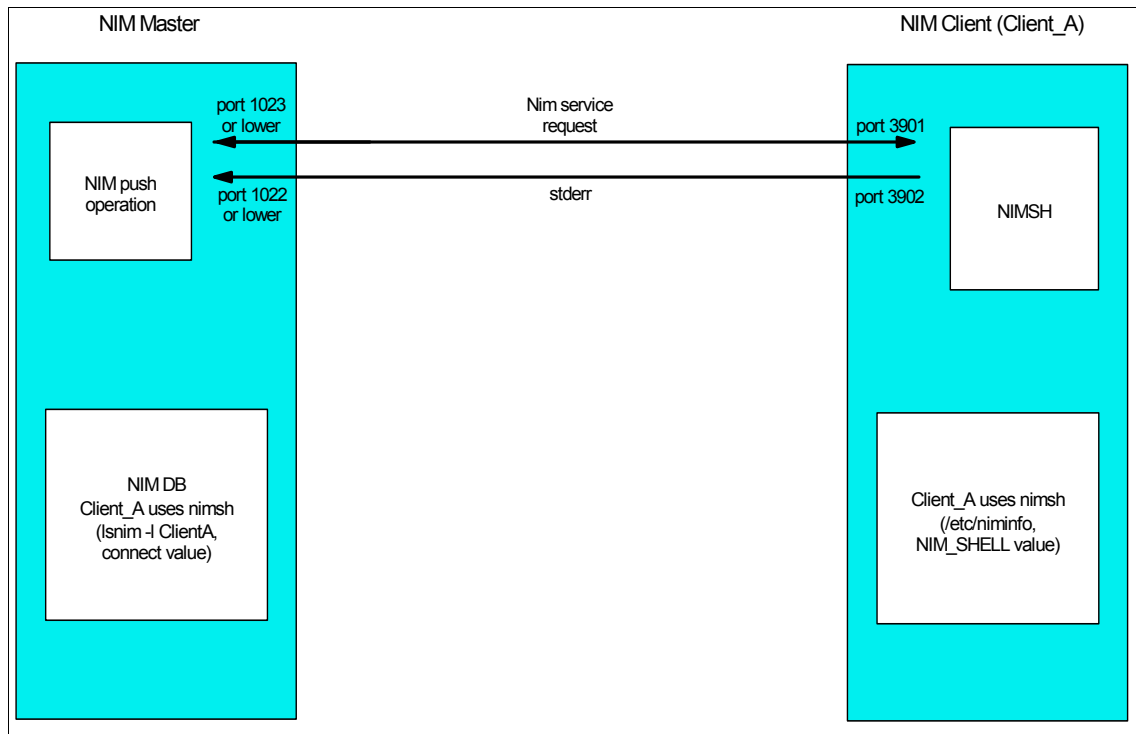


Figure 5-1 Nim operations through NIMSH

### 5.3.2 Setting up NIMSH

The following procedure should be used to configure existing standalone clients with NIMSH.

#### **Using SMIT**

Use the following SMIT menus:

```
# smitty nim_config_services
Select nimsh
```

#### **Or, from the command line**

The following step is done via the command line.

- ▶ Type the following command on the NIM client:

```
# nimclient -C
```

**Notes:**

- ▶ In order to set up NIMSH, the NIM client must already be configured and both, client and master, must have AIX 5L V5.3 or later installed.
- ▶ NIMSH must be configured only in the client.

### 5.3.3 Verifying NIMSH startup

Run this command to verify that the NIMSH daemon is enabled on the client:

```
# lssrc -s nimsh
```

The output should look similar to the one shown in Example 5-12.

*Example 5-12 Verifying NIMSH startup*

---

```
# lssrc -s nimsh
```

Subsystem	Group	PID	Status
nimsh	nimclient	245944	active

---

**Note:** Once NIMSH is configured on the client you no longer need the entry for the NIM master on the `~/rhosts` file on that client since NIM no longer uses classical “r” commands to perform operations in the client.

### 5.3.4 Enabling cryptographic authentication

You can configure existing standalone clients to use the NIMSH communication with SSL enabled.

NIM supports OpenSSL versions 0.9.6e and higher. When OpenSSL is installed, NIMSH uses SSL-encrypted certificates for authenticating the connecting NIM master.

You can install and configure the OpenSSL cryptographic software using the NIM command options. Scripts are provided for configuring OpenSSL in the NIM environment, and you can use these without any modifications. The scripts are installed as part of the `bos.sysmgmt.nim.client` fileset and located in the `/usr/samples/nim/ssl` directory. The scripts are used to define SSL keys and certificates for NIM SSL usage.

Because NIM masters can support a large system environment, it is necessary to impose a hierarchy on SSL certificate and key storage structure. During NIM setup, the directory structure described in Table 5-5 is created.

Table 5-5 Directory hierarchy on SSL certificate and key storage

Directory	Usage
/ssl_nimsh	SSL parent directory for NIM
/ssl_nimsh/configs	Contains scripts used to configure SSL in NIM
/ssl_nimsh/certs	Contains SSL certificates used during host authentication
/ssl_nims/keys	Contains SSL keys used during SSL protocol communication

The NIM SSL directory structure is considered static and you should not modify it. To change SSL certificate options, you can modify the scripts described in Table 5-6.

Table 5-6 SSL authentication configuration scripts

Script name	Usage
SSL_root.cnf	Generates Certificate Authority key for signing certificates
SSL_server.cnf	Generates the NIM master's certificate for distributing to clients
SSL_client.cnf	Generates the NIM master's local certificate for authenticating

## Installing OpenSSL on the NIM master

OpenSSL must be installed on the NIM master in order to be able to configure SSL authentication on the NIM master.

To install the OpenSSL on the NIM master follow these steps:

- ▶ Type the smitty nim\_task\_inst fastpath on the NIM master.
- ▶ Select **Install Software** from the SMIT panel.
- ▶ Select the name of your NIM master from the list that appears.
- ▶ Select the LPP\_SOURCE containing the install images for OpenSSL.
- ▶ Select the package **openssl-0.9.7g** on the field Software to Install.
- ▶ Select the values for the rest of the options which best fit your requirements.

The package version may be different. The openssl-0.9.7g package can be found on the Linux Toolbox for AIX distribution media.

Example 5-13 shows the SMIT panel used for the OpenSSL installation on the NIM master.

*Example 5-13 OpenSSL installation on the NIM master*

---

* Installation Target	master	
* LPP_SOURCE	LPP_53_ML4	
* Software to Install	[openssl-0.9.7g	> +
Customization SCRIPT to run after installation (not applicable to SPOTs)	[]	+
Force	no	+
installp Flags		
PREVIEW only?	[no]	+
COMMIT software updates?	[yes]	+
SAVE replaced files?	[no]	+
AUTOMATICALLY install requisite software?	[yes]	+
EXTEND filesystems if space needed?	[yes]	+
OVERWRITE same or newer versions?	[no]	+
VERIFY install and check file sizes?	[no]	+
ACCEPT new license agreements? (AIX V5 and higher machines and resources)	[yes]	+
Preview new LICENSE agreements?	[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)	[]	+#

---

### Installing OpenSSL on the NIM client

OpenSSL must be installed on the NIM client in order to be able to configure SSL authentication on the NIM client.

Follow these steps to configure OpenSSL on the nim client:

- ▶ Type the **smitty nim\_client\_inst** fast path on the NIM client.
- ▶ Select **Install Software** from the SMIT menu.



- ▶ Select the LPP\_SOURCE containing the install images for OpenSSL
- ▶ Select the package **openssl-0.9.7g** on the field Software to Install.
- ▶ Select the values for the rest of the options which best fit your requirements.

Example 5-14 shows the SMIT panel used for the OpenSSL installation on the NIM master.

*Example 5-14 OpenSSL installation on the NIM client*

---

* LPP_SOURCE	LPP_53_ML4	
* Software to Install	[openssl-0.9.7g	> +
Customization SCRIPT to run after installation (not applicable to SPOTs)	[ ]	+
installp Flags		
PREVIEW only?	[no]	+
Preview new LICENSE agreements?	[no]	+
ACCEPT new license agreements?	[yes]	+
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]	+

---

### **SSL authentication configuration on the NIM master**

The NIM master must already be configured for SSL authenticating within the NIM environment since it is the public key server for the clients. The following procedure configures the NIM master for SSL authentication within the NIM environment.

The following sections shows how to configure SSL authentication on the NIM master using either SMIT or the command line.

#### ***Using SMIT***

Follow these steps to configure SSL authentication on the NIM master:

- ▶ Type the smitty nim\_ssl fast path on the NIM master.
- ▶ Set the value for “Enable Cryptographic Authentication for client communication?” to *enable*.

The output shown in Example 5-15 is the output obtained when enabling cryptographic authentication for client communication on the NIM master using SMIT.

*Example 5-15 SSL authentication configuration on the NIM master using smit*

---

```
x - /opt/freeware/lib/libssl.so.0
x - /opt/freeware/lib/libcrypto.so.0
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to '/ssl_nimsh/keys/rootkey.pem'
-----
Signature ok
subject=/C=US/ST=Texas/L=Austin/O=ibm.com/CN=Root CA
Getting Private key
Generating a 1024 bit RSA private key
.....+++++
.+++++
writing new private key to '/ssl_nimsh/keys/clientkey.pem'
-----
Signature ok
subject=/C=US/ST=Texas/L=Austin/O=ibm.com
Getting CA Private Key
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to '/ssl_nimsh/keys/serverkey.pem'
-----
Signature ok
subject=/C=US/ST=Texas/L=Austin/O=ibm.com
Getting CA Private Key
Target "all" is up to date.
```

---

***From the command line***

The following command configures SSL authentication on the NIM master:

```
# nimconfig -c
```

The output should look similar to the one shown in Example 5-16.

*Example 5-16 SSL authentication configuration on the NIM master*

---

```
# nimconfig -c
NIM_MASTER_HOSTNAME=lpar2
x - /opt/freeware/lib/libssl.so.0
```

```

x - /opt/freeware/lib/libcrypto.so.0
Target "all" is up to date.
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to '/ssl_nimsh/keys/rootkey.pem'
-----
Signature ok
subject=/C=US/ST=Texas/L=Austin/O=ibm.com/CN=Root CA
Getting Private key
Generating a 1024 bit RSA private key
....+++++
.....
.....+++++
writing new private key to '/ssl_nimsh/keys/clientkey.pem'
-----
Signature ok
subject=/C=US/ST=Texas/L=Austin/O=ibm.com
Getting CA Private Key
Generating a 1024 bit RSA private key
...+++++
.....+++++
writing new private key to '/ssl_nimsh/keys/serverkey.pem'
-----
Signature ok
subject=/C=US/ST=Texas/L=Austin/O=ibm.com
Getting CA Private Key

```

**Note:** When cryptographic authentication is enabled, output regarding to authentication is sent to the syslogd daemon.

## Enabling cryptographic authentication on the client

The following procedure describes the steps to configure existing standalone clients to use NIMSH communication protocol with SSL enabled.

### Using SMIT

The following are the steps while using the SMIT menus:

- ▶ Type the smitty nim\_config\_services fast path on the NIM client.
- ▶ Select nimsh as the Communication Protocol used by client.
- ▶ Set the value for “Enable Cryptographic Authentication” to *enable*.

- ▶ Select **yes** as the option for Installing Secure Socket Layer Software, if OpenSSL is not installed on the client.
- ▶ Specify the absolute path for the RPM package or select the lpp\_source resource that contains the OpenSSL RPM package.

### ***From the command line***

The following are the steps while using the command line:

- ▶ If OpenSSL is installed on the NIM client, type the following command:  

```
# nimclient -c
```
- ▶ If OpenSSL is not installed on the NIM client, complete the following steps:
  - Locate the Toolbox for Linux Applications CD.
  - Install OpenSSL RPM package.
  - Type the following command on the NIM client after OpenSSL is installed:  

```
# nimclient -c
```

Example 5-17 shows the output for the command issued to enable cryptographic authentication.

#### *Example 5-17 Enabling cryptographic authentication via command line*

```
# nimclient -c
Received 2718 Bytes in 0.0 Seconds
0513-044 The nimsh Subsystem was requested to stop.
0513-059 The nimsh Subsystem has been started. Subsystem PID is 319594.
```

**Note:** In order to enable cryptographic authentication the NIM master must already be configured for SSL authentication and both client and master must have AIX 5L V5.3 or later installed.

**Attention:** Any communication initiated from the NIM client (pull operation) reaches the NIM master on the request for services and registration ports (1058 and 1059 respectively). This communication is not encrypted. Communications initiated from the NIM master (push operations), the NIM master communicates with the NIM client using the NIMSH daemon. This allows encrypted handshake dialog during authentication. However, data packets are not encrypted.

### 5.3.5 Enabling a secondary port for NIMSH communication

By default, NIMSH uses a reserved port for returning stderr output during command execution. The default setting allows administrators to specify a specific port for opening through a firewall, but it can cause performance issues when several connections are attempted in a short amount of time.

When TCP connections are closed, the closing sockets enter TIME\_WAIT state. The length of time for this state may last up to 240 seconds depending on system settings. The secondary port option allows you to specify any specific range of ports to cycle through during NIMSH operation.

For firewalls, administrators might want to open a specific port range through the firewall, and then for each machine on the internal network, ensure that the port range on the machine coincides with the open range through the firewall. When changing the NIMSH secondary port, you should choose a range of ports outside of the range used for system services. We recommend using ports 49152 through 65535.

#### Using SMIT to enable a secondary port

Complete these steps to configure existing standalone clients to use the NIMSH communication protocol with a secondary port range.

- ▶ Type the `smitty nim_config_services` fast path on the NIM client.
- ▶ Select `nimsh` as the Communication Protocol used by client.
- ▶ Specify a start value for the secondary port number.
- ▶ Specify an increment value for the secondary port range.

#### Using command line to enable a secondary port

Complete these steps to configure existing standalone clients to use the NIMSH communication protocol with a secondary port range from the command line.

- ▶ Edit the `/etc/environment` file.
- ▶ Add the variable `NIM_SECONDARY_PORT=60000:5`, to use ports 60000 - 60005 within NIMSH.
- ▶ Use the desired `nimclient` command option to restart the NIMSH daemon.

**Note:** In order to enable a secondary port the NIM client must already be configured and both, master and client must have AIX 5L V5.3 or later.

### 5.3.6 Disabling push operations using NIMSH

NIM clients can prohibit the NIM master from allocating resources or initiating operations by disabling push operations. When push disablement is set, NIMSH does not process any NIM operation controlled by the NIM master.

Although master control is disabled, the client can still control the allocation of NIM resources and the initiation of NIM operations. The NIM master must have push permissions to perform push operations on the NIM clients.

To configure existing standalone clients to use NIMSH communication protocol with NIM master control disabled use the following procedures.

#### Using Web-Based System Management

This section describes how to use the Web-Based System Management.

- ▶ From the main Web-Based System Management container, select the **Software** icon.
- ▶ From the Software menu, select **NIM Client** → **Permissions**.
- ▶ Select whether to grant or deny permission for the NIM master to initiate push installations.

#### Using SMIT

You can use the `smit nim_perms` fast path to disable the master push permissions.

To disable the master's push permissions, enter the `smit nim_perms` fast path from the client machine and select whether to grant or deny permission for the NIM master to initiate push installations.

#### Using command line

You can disable and re-enable the master push permissions from the command line.

To set control on the client to `push_off`, enter the following on the client machine:

```
nimclient -P
```

To re-enable push permission on the client, enter the following on the client machine:

```
nimclient -p
```

### 5.3.7 Disabling pull operations

You can disable pull operations (operations initiated from the NIM client) by shutting down the nimesis subsystem from SRC on the NIM master as shown in Example 5-18. Disabling pull operations this way applies to all the NIM clients. If you need to disable pull operations for a particular NIM client you can enable CPU validation on the NIM master and provide a non-matching CPU id for the CPU id on the NIM client object definition. For information regarding how to configure CPU validation refer to 4.4, “Using NIM to perform AIX migrations of the NIM master and clients” on page 153.

*Example 5-18 Shutting down the nimesis subsystem*

---

```
(root@lpar2):/ # stopsrc -s nimesis
0513-044 The nimesis Subsystem was requested to stop.
```

---

### 5.3.8 NIM communication within a firewall environment

NIM is unaware of any network security when it attempts to perform either a push or pull operation. If there is a firewall between the master/server and the client it has to be configured to allow traffic for all the protocols involved in NIM operations. When a network install is performed, there are several protocols involved. The following part should provide adequate steps to help firewall administrators to configure the firewalls to provide access for the clients to the different resources.

#### Install overview

NIM performs network install by using a client/server model based on the bootp/tftp protocols for receiving a network boot image. Once the boot image is obtained, the client requests (tftp) a configuration file (niminfo) to determine which server(s) contain the install image and other necessary install resources. The install image and resources are nfs mounted using nfsd/mountd services. Once all mounts succeed, the install begins and subsequent information is sent to the NIM master via nimclient calls to the nimesis daemon (NIM).

Upon completion of install, the client sends state information to the master via nimclient calls to the nimesis daemon. The NIM master then deallocates all install resources from the client object which has completed installing. The deallocation process consists of:

- ▶ Removing files from designated tftp directory
  - Remove niminfo file
  - Remove link to boot image
  - Remove file entry in /etc/bootptab

- ▶ Unexporting nfs resources from client
  - Remove entries from /etc/exports
  - Export remaining entries from /etc/exports
- ▶ Updating client object information in the NIM database
  - Machine state (running / not running)
  - Command result (success or failure) client state (ready for an operation)

### **Install process**

The Install Overview gives a short summary of the install process. In this section, we describe the client-server communication during a NIM install. A NIM master push is presented with specifics into the install process.

### **Master initiated install**

When a network install is initiated from the master's end, the NIM master prepares resources for install (create script files, NFS export resources, create file entry in /etc/bootptab) and then executes an RSH command on the client to set the bootlist for install over the network interface. The client resets and attempts to boot over the network using bootp and tftp services. Figure 5-2 shows the ports and protocols involved in NIM communications.



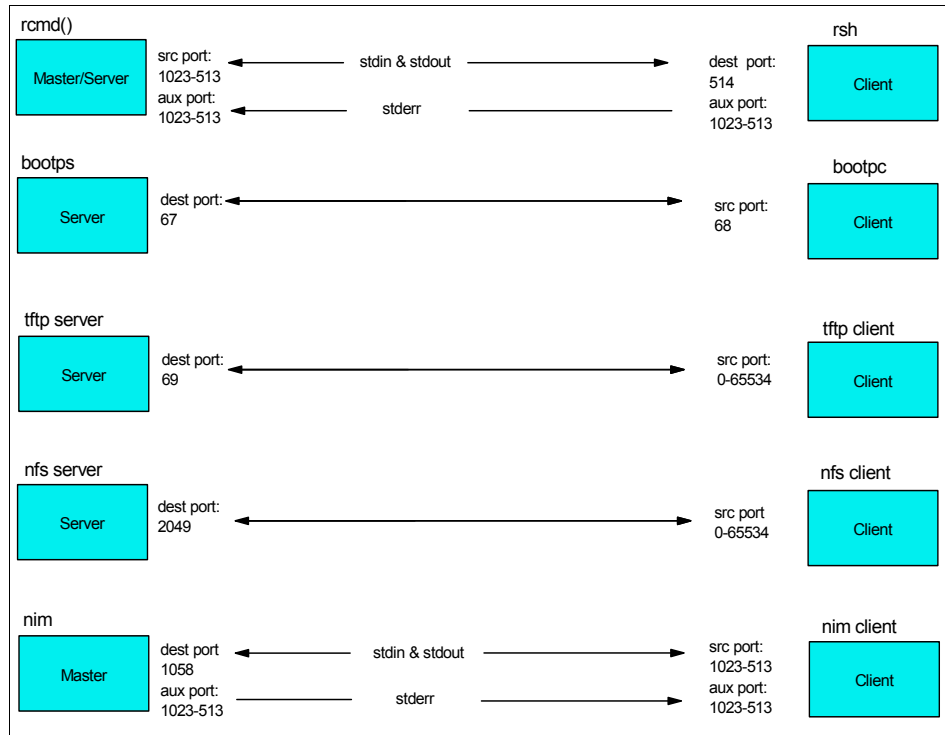


Figure 5-2 NIM protocols

**Note:** Firewall administrators must allow traffic for the protocols involved in a NIM operation.

## Protocols involved in a NIM operation

This section describes the protocols involved in a NIM operation.

### Remote Shell (RSH)

Remote shell requires clients to connect using source ports obtained from the reserved port range of 1023-513. Since NIM clients do not have a client service, the clients communicate by calling `rcmd()`, which in turn, calls `reservport()` to create a TCP socket and binds a port from the privileged port range of 1023-513. The port is determined by initializing the starting port at 1023 and attempting to bind it; if this fail, the port number is decremented and the bind to the port re-attempted. This process continues until an unused port is found or port 513 has been reached

Upon successful binding of the source port, `rcmd()` allows the option of binding a secondary (auxiliary) port for any `stderr`. When set, `rresvport()` is called, but this time the starting port is based on the source port that was obtained in the previous step (`source - 1`). Once initialized, the process for binding is repeated and upon success, this port keeps open for any `stderr` received from the destination service.

NIM makes use of this option and passes return code status (in addition to error messages) over the secondary port.

### ***Boot Protocol (BOOTP)***

The BOOTP protocol uses two reserved port numbers, 'BOOTP client' (68) and 'BOOTP server' (67). The client sends requests using 'BOOTP server' as the destination port; this is usually a broadcast. The server sends replies using 'BOOTP client' as the destination port; depending on the kernel or driver facilities in the server, this may or may not be a broadcast. The reason two reserved ports are used, is to avoid 'waking up' and scheduling the BOOTP server daemons, when a boot reply must be broadcasted to a client.

### ***Trivial File Transfer Protocol (TFTP)***

The TFTP protocol uses transfer identifiers (TIDs) as ports for communication; therefore they must be between 0 and 65,535. In order to create a connection, each end of the connection chooses a TID for itself, to be used for the duration of that connection. The TIDs chosen for a connection should be randomly chosen, so that the probability that the same number is chosen twice in immediate succession is very low. Every packet has associated with it the two TID's of the ends of the connection, the source TID and the destination TID.

These TIDs are handed to the supporting UDP (or other datagram protocol) as the source and destination ports. A requesting host chooses its source TID as described above, and sends its initial request to the known TID 69 decimal (105 octal) on the serving host. The response to the request, under normal operation, uses a TID chosen by the server as its source TID and the TID chosen for the previous message by the requestor as its destination TID. The two chosen TID's are then used for the remainder of the transfer.

### ***Network File System (NFS)***

The NFS protocol currently uses the UDP port number 2049. This is not an officially assigned port, so later versions of the protocol use the port mapping facility of RPC (Remote Procedure Call). The port mapper program maps RPC program and version numbers to transport-specific port numbers. This program makes dynamic binding of remote programs possible.

This is desirable because the range of reserved port numbers is very small and the number of potential remote programs is very large. By running only the port

mapper on a reserved port (111), the port numbers of other remote programs can be ascertained by querying the port mapper.

The port mapper also aids in broadcast RPC. A given RPC program will usually have different port number bindings on different machines, so there is no way to directly broadcast to all of these programs. The port mapper, however, does have a fixed port number. So, to broadcast to a given program, the client actually sends its message to the port mapper located at the broadcast address. Each port mapper that picks up the broadcast then calls the local service specified by the client. When the port mapper gets the reply from the local service, it sends the reply on back to the client.

### ***Network Installation Manager (NIM)***

Clients communicate to the NIM master using TCP ports 1058 (nim) and 1059 (nimreg). During the install process, the NIM clients send status information. The information contains details specific to the install progress. This information is updated in the NIM database and actions on the master's end are handled accordingly; when necessary, resources are deallocated (unexported) and boot images are removed.

The clients do not have a registered client service, so they use `rcmd()` to obtain sockets based on the rules mentioned above in the RSH section of this document. The API is passed the service port of 1058 for establishing a connection to the NIM master. The nimesis daemon runs on the NIM master and listens on the NIM service port. When a request for service is received, the nimesis daemon accepts the connection, verifies the originator, and sends an ACK signal in a similar fashion as expected by `rcmd`. Upon a successful connection, state information is passed and commands are placed on the client using the secondary port which has a file descriptor associated with the socket.

Since NIM clients do not have access to the NIM database, all NIM commands are interpreted on the NIM master's end and subsequent operations are placed on the client for shell execution. This detail is important to understand since clients are allowed the option of requesting NIM operations. Since clients have no knowledge of which commands must execute per NIM operation, the requests are always sent to the NIM master (1058) and the master responds by pushing (`rsh`) the necessary commands on the client machines.

The registration port (1059) is used when clients attempt to add themselves to a current NIM environment. The clients use `rcmd()` to obtain sockets and pass the service port 1059 for establishing a connection to the NIM master. When connected, clients pass machine configuration information to the NIM master. The NIM database is updated with the newly defined client object and `rsh` permissions are given to the NIM master.

### 5.3.9 NFS reserved ports

Usage of NFS reserved ports can be enabled within a NIM environment starting with AIX 5L V5.3 TL 05 and later. Enabling usage of NFS reserved ports forces NIM clients to use reserved ports (ports below 1024) for NFS communications to the NIM server.

Enabling usage of NFS reserved ports adds the attribute `nfs_reserved_ports` to the NIM master object. The value for this attribute is set to `yes`. All installing clients configure the `nfso` option `nfs_use_reserved_ports` setting this value to 1 within the install environment.

Any clients installed this way are also be able to configure the `nfso` client option prior to NIM client operations. Clients participating in a NIM environment where `nfs_reserved_ports` is enabled are able to set the NFS client option prior to any NFS service requests.

When usage of NFS reserved ports is not enabled (which is the value by default) NIM clients use non reserved ports to perform NFS service requests to the NIM Server. In Example 5-19 we show the NFS traffic between a NIM client and a NIM server. It can be seen that the ports used for NFS communications are not the reserved ports. In this example our NIM server uses the IP 10.1.1.22 and the NIM client uses the IP 10.1.1.70. The `netstat` command is executed on the NIM server while the NIM client is performing a NIM operation involving NFS service requests.

*Example 5-19 Non reserved ports usage*

---

```
(root@lpar2):/ # netstat -na | grep 10.1.1.70
tcp4      0 101040 10.1.1.22.2049      10.1.1.70.32786      ESTABLISHED
.....
```

---

The NIM clients already installed before enabling usage of NIM reserved ports should change their settings following the procedures shown in “Enabling NFS reserved ports on a NIM client already installed”.

Once usage of NFS reserved ports is enabled NIM clients use reserved ports for NFS traffic as shown in Example 5-20.

*Example 5-20 Reserved ports usage*

---

```
(root@lpar2):/ # netstat -na | grep 10.1.1.70
tcp4      0      0 10.1.1.22.2049      10.1.1.70.1021      ESTABLISHED
.....
```

---

## Enabling NFS reserved ports on the NIM master

This section enables NFS reserved ports on the NIM master.

### *Within a NIM environment*

Through this procedure you change the NIM master database to set that any client installed with this option sets the `nfs_reserved_ports` parameter to 1 and uses NFS reserved ports.

Follow these steps to configure NFS reserved ports usage within a NIM environment:

- ▶ Enter the smitty `nim_global_nfs` fastpath in the NIM master.
- ▶ Select **enable** as the value for the Enable/Disable Global Usage of NFS Reserved Ports? option.

Example 5-21 Shows the SMIT panel used to enable usage of NFS reserved ports.

#### *Example 5-21 Enabling usage of NFS reserved ports*

---

Manage NFS Client Communication

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

```

                                     [Entry Fields]
* Enable/Disable Global Usage of NFS Reserved Ports? [enable]      +
* Allow NIM to enable port-checking on NIM master?   [no]         +

```

```

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

```

---

In Example 5-22, we see the attribute `nfs_reserved_port` set to `yes` to enable usage of NIM reserved port.

#### *Example 5-22 NIM master attributes*

---

```

(root@lpar2):/ # lsnim -l master
master:
    class                = machines

```

```

type                = master
max_nimesis_threads = 20
if_defined          = chrp.mp.ent
comments           = machine which controls the NIM environment
platform           = chrp
netboot_kernel     = mp
if1                = ent-Network1 lpar2 00096B4EAD9C
cable_type1        = N/A
Cstate             = ready for a NIM operation
prev_state         = ready for a NIM operation
Mstate            = currently running
serves            = LPP_5305
serves            = SEC_ADAPTR
serves            = SPOT_5305
serves            = boot
serves            = nim_script
master_port        = 1058
registration_port  = 1059
reserved          = yes
ssl_support        = yes
nfs_reserved_port = yes

```

---

### For a single NIM client

Use the following steps to configure the usage of NIM reserved ports between the NIM master and a specific client:

- ▶ Enter the smitty nim\_chmac fastpath in the NIM master.
- ▶ Select the NIM client in which you want to configure the usage of NIM reserved ports.
- ▶ Select **yes** for “NFS Client Reserved Ports”.

In Example 5-23 we see the SMIT panel used to set NFS client reserved ports values.

*Example 5-23 Setting NFS client reserved ports values*

---

```

Machine Name                [VLPAR3_p5]
* Hardware Platform Type    [chrp]                +
* Kernel to use for Network Boot [mp]                +
  Machine Type              standalone
  Network Install Machine State currently running
  Network Install Control State ready for a NIM opera>
  Primary Network Install Interface
    Network Name            ent-Network2

```

Host Name	[v1par3_p5]	
Network Adapter Hardware Address	[0]	
Network Adapter Logical Device Name	[ent]	
Cable Type	bnc	+
Network Speed Setting	[]	+
Network Duplex Setting	[]	+
IPL ROM Emulation Device	[]	+/
CPU Id	[00CC544E4C00]	
Communication Protocol used by client	[shell]	+
<b>NFS Client Reserved Ports</b>	<b>[yes]</b>	+
Comments	[]	
Force	no	+

Use the `lsnim` command to check if the `nfs_reserved_port` value is set to `yes` for a NIM client in the NIM master database as shown in Example 5-24.

*Example 5-24 Checking NFS reserved ports values*

```
(root@lpar2):/ # lsnim -l VLPAR3_p5
VLPAR3_p5:
  class           = machines
  type            = standalone
  connect         = shell
  platform        = chrp
  netboot_kernel = mp
  if1             = ent-Network2 v1par3_p5 0
  cable_type1    = bnc
  Cstate         = ready for a NIM operation
  prev_state     = ready for a NIM operation
  Mstate         = currently running
  cpuid          = 00CC544E4C00
  nfs_reserved_port = yes
```

**Note:** It is possible to use various combinations for NFS reserved ports. For example, you can have NFS reserved ports enabled for your NIM environment and select NIM clients using non-reserved ports. Also you can have NFS reserved ports disabled for the NIM environment and select clients using reserved ports for the NFS communication.

### Enabling NFS reserved ports on a NIM client already installed

If the NIM client is installed prior to setting usage of NFS reserved ports you have to do some changes in order to the client start using reserved ports for the NFS

operations. The `/etc/niminfo` file must be recreated to reflect the changes on the NIM master regarding usage of NFS reserved ports.

To recreate the `/etc/niminfo` file you can proceed on two different ways. You can rebuild the NIM client `/etc/niminfo` file from the NIM master or from the NIM client.

### ***Rebuilding the NIM client /etc/niminfo file from the NIM master***

To rebuild the NIM client `/etc/niminfo` file from the NIM master follow the next steps.

- ▶ Type the smitty `nim_switch_master` fastpath on the NIM master
- ▶ Select the Machine Name for the NIM client in which you want to rebuild the `/etc/niminfo` file.
- ▶ Select the **Host Name of Network Install Master** using the same host name than the actual one.

In Example 5-25 we show the SMIT panel used to rebuild the `/etc/niminfo` file on the NIM client named `VLPAR3_p5` from the NIM master `lpar2` introducing the Host Name `lpar2` for the value of Host Name of Network Install Master

*Example 5-25 Rebuilding the NIM client /etc/niminfo file from the NIM master*

---

Machine Name	[VLPAR3_p5]
<b>Host Name of Network Install Master</b>	<b>[lpar2]</b>
Force	no +

---

In Example 5-26, we can see that the `/etc/niminfo` file on the NIM client has been updated with the changes regarding usage of NFS reserved ports. A new variable is added to the file. The variable is called `NFS_RESERVED_PORT` and the value for this variable is `yes` as usage of NFS reserved ports is enabled on the NIM master.

*Example 5-26 New /etc/niminfo file*

---

```
# cat /etc/niminfo
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=VLPAR3_p5
export NIM_HOSTNAME=v1par3_p5
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=lpar2
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
```



```

export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.70:v1par3_p5 10.1.1.22:lpar2 "
export NIM_MOUNTS=""
export NFS_RESERVED_PORT=yes
export ROUTES=" default:0:10.1.1.1 "

```

---

### ***Rebuilding the NIM client /etc/niminfo file from the NIM client***

To rebuild the NIM client /etc/niminfo file from the NIM client proceed as follows:

- ▶ Rename the /etc/niminfo file to other name (you also can delete it but in this way you can keep a copy just in case you may need it)
- ▶ Enter the smitty nimit fastpath on the NIM client
- ▶ Select the **Machine Name**. This is the name of the client on the NIM master. This name is already defined on the NIM master since this client is already registered on the NIM master
- ▶ Select the **Primary Network Install Interface** for the NIM client
- ▶ Select the **Host Name of Network Install Master**. This is the NIM master that you are already using.

Example 5-27 shows the SMIT panel used to rebuild the /etc/niminfo on the NIM client in the same scenario as above. The /etc/niminfo file is rebuilt on the NIM client which is named VLPAR3\_p5 from the NIM master which is named lpar2.

*Example 5-27 Rebuilding the NIM client /etc/niminfo file from the NIM client*

---

* <b>Machine Name</b>	<b>[VLPAR3_p5]</b>	
* Primary Network Install Interface	[en0]	+
* <b>Host Name of Network Install Master</b>	<b>[lpar2]</b>	
Hardware Platform Type	chrp	
Kernel to use for Network Boot	[mp]	+
Communication Protocol used by client	[]	+
Ethernet Interface Options		
Network Speed Setting	[]	+
Network Duplex Setting	[]	+
Comments	[]	
Alternate Port Numbers for Network Communications (reserved values will be used if left blank)		
Client Registration	[]	#
Client Communications	[]	#

---

**Note:** The changes performed regarding usage of NFS reserved ports, for example enabling or disabling usage of NFS reserved ports must be impacted to the NIM client through rebuilding the `/etc/niminfo` file.

## Disabling the usage of NFS reserved ports

The following sections show how to disable the usage of NFS reserved ports.

### *For the NIM environment*

Follow these steps to disable the usage of NFS reserved ports for the entire NIM environment.

- ▶ On the NIM master execute the `nim` command using the change operation to change the `nfs_reserved_port` attribute to `no` for the NIM machine object corresponding to the NIM master as shown in Example 5-28. In the example master is the name of the NIM machine object for the NIM master.
- ▶ Rebuild the `/etc/niminfo` file on the NIM clients. For information about how to rebuild the `/etc/niminfo` file on the NIM client refer to “Rebuilding the NIM client `/etc/niminfo` file from the NIM master” or “Rebuilding the NIM client `/etc/niminfo` file from the NIM client”

*Example 5-28 Disabling usage of NFS reserved ports on the NIM environment*

---

```
(root@lpar2):/ # nim -o change -a nfs_reserved_port=no master
(root@lpar2):/ #
```

---

### *For a particular NIM client*

Follow these steps to disable the usage of NFS reserved ports for a particular NIM client.

- ▶ On the NIM master execute the `nim` command using the change operation to change the `nfs_reserved_port` attribute to `no` for the NIM machine object corresponding to the NIM client on which you are disabling the usage of NFS reserved ports as shown in . In the example `VLPAR3_p5` is the name of the NIM machine object for the NIM client.
- ▶ Rebuild the `/etc/niminfo` file on the NIM clients. For information about how to rebuild the `/etc/niminfo` file on the NIM client refer to “Rebuilding the NIM client `/etc/niminfo` file from the NIM master” or “Rebuilding the NIM client `/etc/niminfo` file from the NIM client”

*Example 5-29 Disabling usage of NFS reserved ports for a particular NIM client*

---

```
(root@lpar2):/ # nim -o change -a nfs_reserved_port=no VLPAR3_p5
(root@lpar2):/ #
```

---

**Note:** If the `nfso` server `portcheck` option is enabled and after disabling the usage of NFS reserved ports the `portcheck` option is not required any more in your environment you can disable it using the `nfso` command as shown in “Disabling the NFS server `portcheck` option”

### Enabling the NFS server `portcheck` option (with `nfso`)

This option provides a convenient way for users to enable the port-checking that is done by the NFS server. When `nfso` `portcheck` option is enabled the NFS server checks whether an NFS request is originated from a privileged port, that is a reserved port (below 1024). The default value of 0 disables port checking by the NFS server. A value of 1 directs the NFS server to do port checking on the incoming NFS requests.

This option does not have to be configured through NIM if resources are being served from a resource server separate from the NIM master. Use the `nfso` command as shown in Example 5-30 to enable this value on the resource server if the resources are being served from a resource server separate from the NIM master.

*Example 5-30 Setting `portcheck` through the `nfso` command.*

---

```
(root@lpar2):/ # nfso -o portcheck=1
Setting portcheck to 1
```

---

Allowing port-check enablement adds the `nfso` server option `portcheck` with a value of 1 to the NIM master. This can be checked through the `nfso` command as shown in Example 5-31.

*Example 5-31 Checkint `portcheck` enablement throug the `nfso` command*

---

```
(root@lpar2):/ # nfso -o portcheck
portcheck = 1
```

---

To enable the use of the `nfso` server `portcheck` option follow the next procedure:

- ▶ Enter the `smitty nim_global_nfs fastpath` in the NIM master.
- ▶ Select **yes** as the value for the Allow NIM to enable port-checking on NIM master? option.

In Example 5-32 we see the SMIT panel used to enable the `nfso` server `portcheck` option.

*Example 5-32 Enabling `nfso` server `portcheck` option*

---

Manage NFS Client Communication

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

	[Entry Fields]	
* Enable/Disable Global Usage of NFS Reserved Ports?	[enable]	+
* Allow NIM to enable port-checking on NIM master?	[no]	+

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

**Note:** If you enable the `nfs` server `portcheck` option the NIM clients configured not to use NFS reserved ports fail to perform NIM operations involving NFS traffic. The error displayed is similar to the one shown in Figure 5-3.

```

                                Software to Install

Move cursor to desired item and press F7. Use arrow keys to scroll.
ONE OR MORE items can be selected.
Press Enter AFTER making all selections.

■ warning: 0042-175 c_showres: An unexpected result was returned by the
"/usr/sbin/mount" command:

mount: giving up on:
      lpar2:/exports/lpp
vmount: Permission denied
rc=175
0042-175 c_showres: An unexpected result was returned by the "/usr/sbi
mount: giving up on:
      lpar2:/exports/lpp
vmount: Permission denied

```

Figure 5-3 NFS operation error

### Disabling the NFS server portcheck option

Use the `nfs` command on the NIM master to disable the `portcheck` option. Example 5-33 shows how to use the `nfs` command to disable the `portcheck` option.

*Example 5-33 Disabling the nfsd portcheck option*

---

```
(root@lpar2):/ # nfsd -o portcheck=0  
Setting portcheck to 0
```

---

### 5.3.10 Firewall considerations

NIM makes use of several protocols which are generally considered risky services on firewall machines. It is recommended that users who desire firewall protection within their NIM environment follow a few rules described next.

- ▶ The NFS program usually runs at port 2049 which is outside of the privileged port space. Normally, access to portmapper (port 111) is needed to find which port this service runs on, but since most installations run NFS on this port, hackers can bypass NFS and try this port directly. NFS was designed as a LAN service and contains numerous security vulnerabilities when used over the Internet. NFS services should not be run on firewall machines; if a NIM master resides on a firewall machine, then resources should reside on another client as clients may also be used as resource servers in a NIM environment (Refer to 4.3.1, “Configuring a resource server” on page 145)
- ▶ If possible, TFTP servers should not be placed on firewall machines since no authentication is needed when requesting service. The TFTP protocol does allow for denying access based on entries contained in `/etc/tftpaccess.ctl`. NIM manages access to files in `/tftpboot` only; so all other directory locations should be off limits. When managed properly, TFTP access can be viewed as acceptable in the NIM environment.
- ▶ Since rsh is the standard method of client control, clients participating in the NIM environment must allow shell service (514), enable NIMSH for client communication, or enable Kerberos in the NIM environment per client. In order to reduce the amount of open ports in the NIM environment, the following rules may be applied:
  - For every NIM communication using rsh, leave five (5) ports open starting at 1023 and decremented from there. So if a client is communicating in the NIM environment, the client should leave open ports (1023-1019) and the master should leave open ports (1023-1019). This is an estimate and may not work in all environments since other services may call `rreservport()` prior to, or during, NIM operations. When monitored, this approach should work fine in small environments since access to ports in the privileged space are restricted to super-user access only.
  - Users may also add secondary interfaces for each client participating in the NIM environment. The additional interfaces should be packet filtered

When NIM clients no longer participate in the NIM environment, or are temporarily removed from the NIM environment, users should disable rsh services on client machines by removing `~/rhosts` and/or removing rshd service.

## 5.4 “Cloning” NIM clients using mksysb

This section describes how to clone a client using an existing mksysb image and Network Installation Manager (NIM) procedures.

### 5.4.1 Using a mksysb image to clone NIM Client

An mksysb image is a backup of the operating system (that is, the root volume group). This backup can be used to reinstall a system to its original state it needed. It also can be used to clone the machine from which the mksysb image has been taken to another machine. Another usage of this image may be to customize your environment, creating an installable image of AIX 5L or AIX 4.3 containing the customizations and products that best fit your requirements and using it to install the clients.

The target systems might not contain the same hardware devices or adapters, require the same kernel (uniprocessor or multiprocessor).

This procedures are intended to be used to clone NIM clients, for information on how to backup and restore a NIM master refer to 5.10, “How to backup and re-install a NIM master” on page 547.

Because NIM configures TCP/IP at the end of an installation, it is recommended that a `bosinst_data` resource be allocated for cloning mksysb installations with the `RECOVER_DEVICES` field set to “no”. This will prevent the BOS installation process from attempting to re-configure the devices as they were on the source mksysb image.

**Note:** Starting in AIX 5L V5.2, devices are not recovered if the mksysb image that is being installed was not created on the same system.

**Attention:** If the system you have cloned is using OpenGL or graPHIGS, there may be some device filesets from these LPPs that must be installed after a clone. OpenGL and graPHIGS have graphics adapter-specific filesets, so if you cloned onto a system with a different graphics adapter, you will need to create a bundle as follows:

```
echo OpenGL.OpenGL_X.dev > \  
/usr/sys/inst.data/user_bundles/graphic_dev.bnd
```

```
echo PEX_PHIGS.dev >> /usr/sys/inst.data/user_bundles/graphic_dev.bnd
```

You can allocate this bundle when you install the mksysb, and the device filesets will be installed automatically if OpenGL and graPHIGS are in your lpp\_source.

## 5.4.2 Prerequisites

In order to restore an mksysb image using NIM the following prerequisites must be met:

- ▶ The NIM master must be configured, and SPOT and mksysb resources must be defined.
- ▶ The NIM client to be installed must already exist in the NIM environment.
- ▶ The mksysb must be available on the hard disk of the NIM master or a running NIM client, or the mksysb image is created during this procedure from either the NIM master or a running NIM client.
- ▶ 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. A sparse file is one with empty space, or gaps, left open for future addition of data. If the empty spaces are filled with the ASCII null character and the spaces are large enough, the file will be sparse, and disk blocks will not be allocated to it. This situation creates an exposure in that a large file will be created, but the disk blocks will not be allocated. As data is then added to the file, the disk blocks will be allocated, but there may not be enough free disk blocks in the file system.  
The file system can become full, and writes to any file in the file system will fail. It is recommended that you either have no sparse files on your system or that you ensure you have enough free space in the file system for future allocation of the blocks.

- ▶ If you are cloning machines that have different hardware you need to have a LPP\_SOURCE to be able to install the additional filesets to support the new hardware which support may not be contained on the mksysb.

### 5.4.3 mksysb image installation on a NIM Client

There are three ways to perform the tasks required to install a mksysb image on a NIM client. This ways are:

- ▶ Web-Based System Management
- ▶ SMIT
- ▶ Command line

In this section, we use SMIT to perform the tasks. For information about how to perform this tasks using Web-Based System Management or the command line refer to AIX 5L V5.3 Installation and Migration, SC23-4887-03.

It has been assumed that the NIM master is configured, a SPOT resource is defined and the NIM client to be installed already exists in the NIM environment. For this example we are using a NIM master (and resource server) named “master”, and a client named VLPAR3\_p5. The mksysb image does not exist as a resource on the NIM master, so it is created at the moment of defining the mksysb resource.

### 5.4.4 Mksysb resource creation

The first step is to create a mksysb resource. The following procedure lead us to accomplish the task:

- ▶ Enter the `smitty nim_mkres` fastpath.
- ▶ Select **mksysb** from the list of resource types that can be defined.
- ▶ 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. The option **CREATE system backup image?** has been set to yes in order to create a system image. Example 5-34 shows the input values for the creation of the mksysb resource in our environment.

*Example 5-34 Mksysb resource creation*

---

* Resource Name	[MKSYSB_VLPAR5_p5]
* Resource Type	mksysb
* Server of Resource	[master] +
* Location of Resource	[/mksysb/VLPAR5] /
Comments	[]



```

Source for Replication          [] +
      -OR-
System Backup Image Creation Options:
  CREATE system backup image?    yes +
  NIM CLIENT to backup           [VLPAR3_p5] +
  PREVIEW only?                  no +
  IGNORE space requirements?     no +
  EXPAND /tmp if needed?         yes +
  Create MAP files?              no +
  Backup extended attributes?    yes +
  Number of BLOCKS to write in a single output [] #
  (leave blank to use system default)
  Use local EXCLUDE file?        no +
  (specify no to include all files in backup)
      -OR-
  EXCLUDE_FILES resource         [] +
  (leave blank to include all files in backup)

```

---

**Tip:** You should select **yes** for the **PREVIEW only?** option to perform a preview of the operation.

**Note:** If the **mksysb** image already exists as a file on the hard disk of the NIM master or client, no additional information is needed to define your **mksysb** resource.

Example 5-35 shows the output for the creation of the **mksysb** resource. In this case, the output includes the output for the creation of the **mksysb** because we choose to create the **mksysb** at the moment of creation of the **mksysb** resource.

*Example 5-35 Output of the mksysb resource creation*

```

-----+
                System Backup Image Space Information
                (Sizes are displayed in 1024-byte blocks.)
+-----+

Required = 730240 (713 MB)    Available = 2096504 (2047 MB)

```

Data compression will be used by the system backup utilities which create the system backup image. This may reduce the required space

by up to 50 percent.

```
Creating information file (/image.data) for rootvg...
```

```
Creating list of files to back up .
Backing up 23634 files.....
16016 of 23634 files backed up (67%).....
23634 of 23634 files backed up (100%)
0512-038 savevg: Backup Completed Successfully.
```

---

### 5.4.5 Correlating mksysb and SPOT resources

A SPOT resource must be allocated to the client to restore the mksysb. Mksysb and SPOT resources must be at the same level. The level of the SPOT NIM resource can be checked using the `nim` command giving the SPOT resource name to the command with the options shown in Example 5-36.

*Example 5-36 Checking SPOT level*

---

```
{nimmast}:/etc # nim -o fix_query SPOT_53_ML4 | grep ML
  All filesets for 5300-02_AIX_ML were found.
  All filesets for 5.3.0.0_AIX_ML were found.
  All filesets for 5300-01_AIX_ML were found.
  All filesets for 5300-03_AIX_ML were found.
  All filesets for 5300-04_AIX_ML were found.
```

---

The `lsnim` command can also be used to check the SPOT level as shown in Example 5-37. The `oslevel_r` attribute shows the SPOT level.

*Example 5-37 Checking SPOT level using the lsnim command*

---

```
{nimmast}:/ # lsnim -l SPOT_53_ML4
SPOT_53_ML4:
  class      = resources
  type       = spot
  plat_defined = chrp
  arch       = power
  bos_license = yes
  Rstate     = ready for use
  prev_state = ready for use
  location   = /AIX53ML4/SPOT_53_ML4/usr/SPOT_53_ML4/usr
  version    = 5
  release    = 3
  mod        = 0
```

```

oslevel_r      = 5300-04
alloc_count    = 0
server         = master
if_supported   = chrp.mp ent
Rstate_result  = success

```

---

The level of the mksysb can be checked using the `nim` command with the options shown in Example 5-38 on the value of `oslevel_r`.

*Example 5-38 Checking the mksysb level*

---

```

{nimmast}:/etc # lsnim -l MKSYB_VLPAR5_p5
MKSYB_VLPAR5_p5:
class          = resources
type           = mksysb
arch           = power
Rstate         = ready for use
prev_state     = unavailable for use
location       = /mksysb/VLPAR5
version        = 5
release        = 3
mod            = 0
oslevel_r      = 5300-04
alloc_count    = 0
server         = master
extracted_spot = SPOT_PRUEBA

```

---

## 5.4.6 SPOT creation from an existing mksysb

There are several reasons which may lead to differences between a SPOT and a mksysb resource. For instance the mksysb can be taken from other machine which is not even in our NIM environment or the machine from where the mksysb is taken is updated without updating the SPOT.

In this cases a SPOT needs to be created from an existing mksysb.

Follow these steps to create a SPOT starting from an existing mksysb:

- ▶ Enter `smitty nim_mkres` on the NIM master
- ▶ Select SPOT as the Resource Type
- ▶ Enter the values required to define the resource. For the value Source of Install Images, select the mksysb resource which you want to use to create the SPOT.

Example 5-39 shows the smitty options used to create a SPOT using a mksysb NIM resource as the source for the creation.

*Example 5-39 SPOT creation from a mksysb*

---

* Resource Name	[SPOT_VLPAR_p5]
* Resource Type	spot
* Server of Resource	[master] +
* Source of Install Images	[MKSYSB_VLPAR5] +
* Location of Resource	[/export/spot] /
Expand file systems if space needed?	yes +
Comments	[]
installp Flags	
COMMIT software updates?	no +
SAVE replaced files?	yes +
AUTOMATICALLY install requisite software?	yes +
OVERWRITE same or newer versions?	no +
VERIFY install and check file sizes?	no +

---

### 5.4.7 Mksysb installation

Once the mksysb resource has been created and a mksysb image has been taken you can follow with the next step which is to restore the mksysb in another (or the same) machine. The following steps guide you through the steps to restore the mksysb:

- ▶ Enter the **smit nim\_bosinst** fast path.
- ▶ Select a TARGET for the operation. This is the partition where the restoring of the mksysb takes place.
- ▶ Select **mksysb** as the installation TYPE.
- ▶ Select the MKSYSB to use for the installation.
- ▶ Select the SPOT to use for the installation.
- ▶ In the displayed dialog fields, supply the correct values for the installation options or accept the default values. Use the help information or the LIST option to help you.
- ▶ Run the SMIT dialog to install the NIM client.

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 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`. If the client was not rebooted automatically from SMIT, initiate a

network boot from the client to install it. Example 5-40 shows the SMIT panel and values to restore the mksysb.

*Example 5-40 Restoring the mksysb*

---

```

* Installation Target                VLPAR3_p5
* Installation TYPE                  mksysb
* SPOT                               SPOT_53_ML4
LPP_SOURCE                          [] +
MKSYSB                               MKSYSB_VLPAR5_p5

BOSINST_DATA to use during installation [] +
IMAGE_DATA to use during installation  [] +
RESOLV_CONF to use for network configuration [] +
Customization SCRIPT to run after installation [] +
Customization FB Script to run at first reboot [] +
  ACCEPT new license agreements?      [no] +
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] +
  ACCEPT new license agreements?      [no] +
  (AIX V5 and higher machines and resources)
  Preview new LICENSE agreements?     [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)	[55] +#

---

In our scenario, the client where we are restoring the mksysb resets and the restore begins.

**Important:** If the mksysb that you are restoring comes from a different machine than the one on which you are restoring it, there may be some additional software which needs to be installed (for example device drivers). In that case you need to allocate a LPP\_SOURCE resource to supply the required filesets.

## 5.4.8 Monitoring the mksysb installation

You can monitor the restore process either opening a console to the machine or through NIM commands used on the master. The `lsnim` command provides information about the status of the client. Example 4-7 shows the output of the `lsnim` command used with different parameters to monitor the restore process.

### Example 5-41 Monitoring mksysb restore

```
{nimmast}:/ # lsnim -l VLPAR3_p5
VLPAR3_p5:
  class      = machines
  type       = standalone
  connect    = nimsh
  platform   = chrp
  netboot_kernel = mp
  if1        = NET_EN1 VLPAR3_p5 0
  cable_type1 = N/A
  Cstate     = Base Operating System installation is being performed
  prev_state = BOS installation has been enabled
  Mstate     = in the process of booting
  info       = BOS install 16% complete : 12% of mksysb data restored.
  boot       = boot
  mksysb     = MKSYSB_VLPAR5_p5
  nim_script = nim_script
```

```
spot          = SPOT_53_ML4
cpuid         = 00CC544E4C00
control       = master
Cstate_result = success
```

---

**Tip:** The `lsnim` command flag `-a` can be used to display a particular attribute. This can be used to display only the attribute info as shown in Example 5-42. Using this into a loop you can automatically monitor the process.

*Example 5-42 Displaying the info attribute*

---

```
{nimmast}:/ # lsnim -a info VLPAR3_p5
VLPAR3_p5:
info = BOS install 16% complete : 12% of mksysb data restored.
```

---

When the restore process completes the `lsnim` command displays an output similar to the one in Example 5-43. The value for the attribute `Cstate` must be ready for a NIM operation.

*Example 5-43 Restore process completed*

---

```
{nimmast}:/ # lsnim -l VLPAR3_p5
VLPAR3_p5:
class          = machines
type           = standalone
connect        = nimsh
platform       = chrp
netboot_kernel = mp
if1            = NET_EN1 VLPAR3_p5 0
cable_type1    = N/A
Cstate         = ready for a NIM operation
prev_state     = not running
Mstate         = currently running
cpuid          = 00CC544E4C00
Cstate_result  = success
```

---

Also check that there is no link in the `/tftpboot` directory for any of the resources previously assigned to restore the `mksysb`, and that there is no entry in the file `/etc/bootptab` for the machine where the `mksysb` is restored.

## 5.5 Using NIM to migrate systems to new hardware

In this section, we briefly discuss how to use the Network Installation Manager (NIM) to move systems to new hardware whether that hardware resides in the same or a remote site. By new hardware, we mean both moving to similar hardware (for example, from POWER4™ to POWER4) and also moving to a different platform (for example, POWER4 to POWER5).

The topics covered will be:

- ▶ Migrating an LPAR to another managed system using NIM.
- ▶ Relocating an LPAR to a remote location via NIM.
- ▶ Migrating to new hardware utilizing custom bosinst.data and image.data files with NIM.

### 5.5.1 Migrating the image of an LPAR to another system using NIM

Migrating the image of an LPAR from one managed system to another is achieved by using a mksysb image to clone a system (see 5.4, “Cloning” NIM clients using mksysb” on page 460). An example of this would be if your current managed system is a System p 650 (POWER4) and you want to “move” an LPAR from this system to a different LPAR on an IBM System p 595 (POWER5). Using mksysb cloning makes this task fairly simple. However, you need to consider the following before proceeding:

- ▶ What version of AIX is the source system running? This is very important as not all versions of AIX will run on newer IBM System p hardware. For example, AIX 5L V5.1 systems will not run on POWER5 so the system must migrate to AIX 5L V5.3 or be migrated using the new **mksysb migration** procedure (see 4.5, “NIM mksysb migration and nim\_move\_up POWER5 tools” on page 206). Check that your AIX version and target hardware platform are compatible. This information is found in the AIX release notes.

[http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304\\_base\\_relnotes.pdf](http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304_base_relnotes.pdf)

- ▶ If the hardware type and configuration of the source and target systems are very different, then you will need to ensure you have all the correct device support in an lpp\_source with an updated SPOT. For example migrating a system from a p660 (POWER3™ RS64) to a p595 (POWER5) will require additional device support for the POWER5 hardware.
- ▶ You should create a new NIM client machine definition for the target LPAR. This will be used as the target of the mksysb cloning.



- ▶ Verify that the new hardware is ready to accept the source system. Before attempting to clone the source image to the new LPAR, you should first create the new LPAR definition on the target system with all the necessary disk, CPU, memory and adapters (Fibre Channel, SCSI, Ethernet, etc.). Any cabling required for network or storage (FC) connections should also be completed and verified. An IP address should also be assigned for the target LPAR (unless the source system is going to be decommissioned after the move and the target system is going to reuse its IP address). Once the new LPAR definition is ready you should then perform an NIM rte installation of AIX for this LPAR. Performing this step will allow you to check and verify that the LPAR configuration is valid and that all the necessary adapters, processors, memory, network/storage connectivity and other devices are valid before attempting the move of the source system. Running `cfgmgr` at this point is also recommended so that if there are any devices that cannot be configured because of missing device filesets, `cfgmgr` will display a message outlining which filesets are required. Section 5.6, “Using SAN zoning for “cloning” LPARs” on page 485 also presents information on moving LPARS via SAN zoning.
- ▶ It is also a good idea to make sure that the microcode on the target system’s adapters, managed system and HMC are up-to-date before attempting a hardware migration. Ensure that all necessary device support is available.
- ▶ When moving an LPAR to a managed system that is connected to a different HMC, the `mksysb` cloning operation will recreate the RMC (part of RSCT) access control lists (ACLs, `/var/ct/cfg/ctrmc.acls`). This is done to allow the target LPAR’s HMC to identify and manage the new LPAR. Section 5.6, “Using SAN zoning for “cloning” LPARs” on page 485 presents details on recreating the RMC ACLs.  
Another related issue with DLPAR and cloning via `alt_disk_install` may arise: after cloning, the `/etc/ct_node_id` file has the same contents as the source LPAR so RSCT does not detect the LPAR to manage. As a workaround, we advise to run the following commands after the first boot of the new LPAR followed by a reboot:

```
# /usr/sbin/rsct/install/bin/uncfgct -n  
# /usr/sbin/rsct/install/bin/cfgct
```

**Note:** It is important to note that NIM rte installations avoid this problem by creating a unique `/etc/ct_node_id` file.

- ▶ Your NIM master must be configured with the latest level of AIX code and must include all the necessary device filesets to support the target system. The `lpp_source` must contain the filesets and the corresponding SPOT must be updated. See also 5.1, “Updating the NIM environment (`lpp_source`, SPOT)” on page 418.

- For example if you are moving an LPAR from a System p650 to a System p595 it is most likely that the adapter types will be different between the two systems. As an example, on your System p650 you may have 6228 Fibre Channel (FC) adapters but the target system has 5716 FC adapters, requiring a different device driver. For a smooth hardware migration you will need that all device filesets are available in your `lpp_source` to support the 5716 adapter. Otherwise you may find that some devices will be in an unconfigured state after the move. Or if your system is booting from the SAN then you may not be able to perform the cloning operation as it may not be able to detect the SAN attached disk or it may fail when trying to access the disks. Starting with AIX 5L V5.2, an `lpp_source` is no longer a prerequisite for a `mksysb` installation i.e. when installing AIX 5L V5.2, by default, all device support files are also installed, so installing a `mksysb` from such a system would not require an `lpp_source`. However, when moving a system image to a different hardware platform we recommend that you allocate an `lpp_source`.
- Also, if the correct disk device support is not available during the `mksysb` cloning, the install may appear to work fine but when the client boots for the first time it may fail with LED code 554 (The boot device could not be opened or read). This situation often occurs when moving a system that boots of a SCSI disk to one that boots from a SAN attached disk, and the necessary device filesets are not available (or the administrator mistakenly forgets to allocate an `lpp_source` for the `mksysb` installation).
- Another example of trouble you may encounter (if you do not have all the required device support in your `lpp_source` and an up-to-date SPOT) is network booting the client. If the required device support is not on your NIM master for the target systems network adapter (from which you are trying to network boot) then you will be unable to start the cloning operation as the client will be unable to configure the network boot adapter. You will see LED code 605 (605=Configuration of physical network boot device failed).
- ▶ When moving a system via NIM `mksysb` cloning, it is good idea to check that the source system's devices are not recovered during the cloning operation. Letting a `mksysb` restore to recreate the source system's devices could lead to duplicate IP addresses on your network. Unless the source system is on a different network, you should choose "no" to recovering devices. How you do this depends on how you are moving, for example manually booting the client and entering details in the BOS installation menu, in which case you can select no to "Recover Devices", or automatically, by using a custom `bosinst.data` file (with `RECOVER_DEVICES = no`).

**Note:** When a mksysb image is created, the CPU ID is saved. If you are reinstalling the same system, then the device information is recovered, for example, RECOVER\_DEVICES is automatically set to *yes*. If the mksysb image is used to install another system, device information is not recovered from the mksysb image, for example, RECOVER\_DEVICES = Default (no).

## 5.5.2 Relocating the image of an LPAR to a remote location

If you have a requirement to move a NIM client to a remote location, it is possible to use NIM to assist in this task. Using a mksysb image of the remote client will allow you to “move” it to your local site without the need for tapes or to relocate the hardware. For this approach to be successful there must be a high-speed network link between both sites. Then all that is required is to export your NIM masters mksysb directory (for example, /export/images, with enough free space for the image). NFS mount the export from the client, perform a mksysb to the mount and then install the mksysb image onto the new (target) system.

For example:

- ▶ We need to relocate an AIX 5L V5.2 LPAR running on a System p650 (lpar1) from Paris to Montpellier. This is our remote (source) system.
- ▶ The target system is an LPAR on a System p595 (lpar1\_p5). This is our local (target) system. The NIM master (montpnim) is also located at the local site in Montpellier.
- ▶ A high-speed network link exists between the two sites.
- ▶ First, we create a mksysb image of the remote system which will be written to our local NIM masters mksysb directory via NFS:

```
<remote_server># mount <montpnim>:/export/images /mnt
<remote_server># mksysb -i /mnt/lpar1-mksysb.5205.14Jun2006
```

- ▶ We also perform a **savevg** of the remote systems data volume group to the mksysb directory. Depending on the size of your data volume group this may not be possible. For example, a 1TB volume group backed up over a network via NFS is not the best approach. We need to check that we have enough free space on NIM master for this backup. Our data volume group was only 5 gigabytes in size. We shutdown all applications on the system before performing the savevg.

```
<remote_server># savevg -f /mnt/lpar1-datavg.savevg.14Jun2006 datavg
<remote_server># cd / ; umount /mnt
```

- ▶ Now we have a mksysb image and savevg image(s) of our remote system. Now we can install the mksysb image onto the new p595 LPAR (lpar1\_p5). We restore with RECOVER\_DEVICES set to “no”. Refer to 5.4, “Cloning”

NIM clients using mksysb” on page 460 for details on cloning via a mksysb image. Once this is complete, we can configure lpar1\_p5 with new IP details (gateways, DNS servers, etc.) for the local site and reconfigure devices such disk, adapters and so on. The new site also uses Tivoli Storage Manager for backup so we install the TSM client and configure it so the new system can be backed up. We also check the AIX level is correct (**oslevel**) and that all the filesets are consistent (**lppchk**). The data volume group is then restored (**restvg**) and verified. The system is now ready for application testing and if OK the relocation is complete. Further (site specific) customization may be required to suite the new environment (for example, mksysb/savevg scripts, hosts file, print queues, security settings etc.).

If you need to relocate a system but a network link does not exist between the sites, you can try the following alternative (this procedure assumes there are tapes drives, of similar type, available at both sites):

- ▶ Perform a mksysb of the remote system to tape. Send the tape to the destination site.
- ▶ Using a suitable tape drive attached to your NIM master, extract the mksysb image. You will need enough free space in a file system to restore the image.

```
# tctl -l /dev/rmt0 rewind
# tctl -f /dev/rmt0.1 fsf 3
# cd /migrate/restore
# dd if=/dev/rmt0.1 of=/migrate/restore/lpar1-dd-mksysb-image-file
```

- ▶ We now have a mksysb image that we can use with NIM to install onto the target LPAR at the local site. More information on extracting a mksysb from tape is discussed in the next section.

### 5.5.3 Migrating to new hardware utilizing custom bosinst.data and image.data files

In this section, we investigate a specific example of using a custom bosinst.data and image.data file when migrating to new hardware. For example, the mksysb image comes from a mirrored rootvg and the target server has only one disk allocated for rootvg; or the mksysb image is from the tape media instead of an image file. This section describes how we use the NIM environment to resolve these issues. To perform this, we need to have access to the mksysb image, the bosinst.data and image.data files. This section explains how to extract the mksysb image from the tape media, customize the bosinst.data and image.data file to suit the new server (target hardware), then perform a BOS installation.

## Extracting mksysb image file from a tape media

If your mksysb image is on a tape media, we first need to extract the mksysb image from this tape media.

1. Determine the tape blocksize

```
# tcopy /dev/rmt0
```

The tcopy command gives a list of all the files found on the media with their byte count and the block size used. See Example 5-44 shown in bold. This command may take a while.

*Example 5-44 tcopy command to find out tape blocksize*

---

```
# tcopy /dev/rmt0
tcopy : Tape File: 1; Records: 1 to 7179 ; size:512
tcopy : Tape File: 1; End of file after :7179 records; 3675648 bytes
tcopy : Tape File: 2; Records: 1 to 2900 ; size:512
tcopy : Tape File: 2; End of file after 2900 records, 76890 bytes
...
```

---

2. Change the blocksize of the tape drive

```
# chdev -l rmt0 -a block_size=512
```

3. Rewind the tape media, /dev/rmt0, to the beginning

```
# tctl -f /dev/rmt0 rewind
```

4. Move the tape forward to the beginning of the forth tape marker

```
# tctl -f /dev/rmt0.1 fsf 3
```

5. Copy the mksysb image to a directory, eg), copy mksysb of node1 to /export/images directory

```
# dd if=/dev/rmt0.1 of=/export/images/Node1.mksysb ibs=1024 \
obs=1024 conv=sync
```

Once the mksysb image is extracted out from the tape, we can create the NIM mksysb resource (see Example 5-45 on page 475).

```
# smitty nim_mkres
Select "mksysb = a mksysb image"
```

*Example 5-45 Define a NIM mksysb resource*

---

Define a Resource

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

```

[EntryFields]
* Resource Name      [MK-NODE1]
* Resource Type     mksysb
* Server of Resource [master] +
* Location of Resource  [/export/images/Node1.mksysb> /
Comment            []
Source for Replication [] +
                    -OR-
System Backup Image Creation Options:
  CREATE system backup image?      no +
  NIM CLIENT to backup              [] +
  PREVIEW only?                    no +
  IGNORE space requirements?        no +
[MORE...10]

```

---

## bosinst.data file

The bosinst.data file contains information for the BOS installation program. For example, the BOS installation program looks for information in the bosinst.data file to find out which target disk to install the rootvg; whether to install and updates any devices in the target server, and so on.

The BOS installation program looks for the information from the /bosinst.data in the mksysb image. If the BOS installation program could not get all the information that it requires, it will prompt the user to manually specify through the console.

The bosinst.data file is usually used in a NIM environment when you need a no-prompt installation. This can be done by prior specifying the bosinst\_data resource. The bosinst.data file should contains all the necessary information needed by the BOS installation program.

There are many parameters in the bosinst.data file. You can find the detail information in the /usr/lpp/bosinst/bosinst.template.README file. To install a mksysb image to another server (different hardware), the following list presents some important parameters that you need to specify.

### ► EXISTING\_SYSTEM\_OVERWRITE

Specifies which disks can be overwritten during the BOS install. This variable is applicable only for a non-prompted overwrite installation.

If the target\_disk\_data stanza has no field values, then the EXISTING\_SYSTEM\_OVERWRITE field is used to determine which disks to install on.

If EXISTING\_SYSTEM\_OVERWRITE is set to 'any' any disk can be used for the install.

If EXISTING\_SYSTEM\_OVERWRITE is set to 'no' only disks that contain no volume group can be used during the install.

If EXISTING\_SYSTEM\_OVERWRITE is set to 'yes' disks that contain the existing root volume group will be used first, and if additional disks are needed those containing no volume group can be used.

► **RECOVER\_DEVICES**

If set to Default, the install process will determine if you are installing to the same machine or not, based on the CPU ID and LPAR nodeid. If you are reinstalling a system backup to the same machine RECOVER\_DEVICES will be set to yes, and if not, it is set to no. This field may also set directly to yes or no.

► **INSTALL\_DEVICES\_AND\_UPDATES**

When installing a mksysb image to a system with a different hardware configuration, one boots from product media, in order to get any missing device drivers installed, if all devices and kernels were not installed prior to creating the mksysb image. In our case, it is the lpp\_source rather than the product media. In addition, if the product media is a later level of AIX than the mksysb, software in the mksysb image will be updated.

To prevent either of these additional installs from occurring, set this field to 'no'. The default is 'yes'

Note: If none of the new devices are required for disk/boot support, we recommend to set INSTALL\_DEVICES\_AND\_UPDATES to No. You will then manually update the system (after the mksysb is installed) using the correct level if CD media / lpp\_source. But if the new devices are required by disk/boot support, you need to set INSTALL\_DEVICES\_AND\_UPDATES to yes. Be prepared that the devices will be updated if the lpp\_source used during the mksysb installation is of higher level.

ALL\_DEVICES\_KERNELS parameter is not required in this case. Specify ALL\_DEVICES\_KERNELS only if you are performing new system installs (overwrite, preservation, or migration), NOT for mksysb install.

► **ENABLE\_64BIT\_KERNEL and CREATE\_JFS2\_FS fields**

If the system is model IBM eServer p5 or BladeCenter® JS20 or later, the default for ENABLE\_64BIT\_KERNEL and for CREATE\_JFS2\_FS will be yes, otherwise it will be no. If either of these fields are set to a value other than Default, that value will be used. If blank, the value Default will be assumed. On any system JFS2 file systems can be selected with either the 64bit or 32bit kernel.

## ► TARGET DISK DATA

There can be multiple `target_disk_data` stanzas. They define which disk(s) will contain the root volume group. Only one field (PVID, PHYSICAL\_LOCATION, SAN\_DISKID, CONNECTION, LOCATION, SIZE\_MB, HDISKNAME) must be non-null for BOS install to choose a disk. The order of precedence is as shown below. Please refer to the `/usr/lpp/bosinst/bosinst.template.README` for more detail on the precedence information.

- PVID (Physical Volume ID), then
- PHYSICAL\_LOCATION, then
- SAN\_DISKID, then
- CONNECTION(parent attribute//connwhere attribute), then
- LOCATION,
- then SIZE\_MB, then
- HDISKNAME.

If the BOS installation is no-prompt and you leave the `target_disk_stanza` empty, the BOS installation program will check on the **EXISTING\_SYSTEM\_OVERWRITE** field

Now that we have a good understanding of the parameters in the `bosinst.data` file, we can edit this file to suit the target server's environment.

There are two approaches to perform this task: With source server running or without the source server running.

If the source server is running, you can edit the `/bosinst.data` file directly from the server. As of AIX 4.3.3 and later versions, `mksysb` command always updates the `target_disk_data` stanzas in `/bosinst.data` to match the disks currently in the root volume group of the system where the `mksysb` is running, we need to create a `/save_bosinst.data_file` file in order not to overwrite the edited/customized `/bosinst.data` file during the `mksysb` backup. You can just touch a file with the file name as `save_bosinst.data_file`.

Just make sure that the edited/customized `/bosinst.data` file is not updated after the `mksysb` command is run.

If the source server is not running, we can extract out the `bosinst.data` file from the `mksysb` image.

In our scenario, we do not have the source server running, only its `mksysb` image. We extract `bosinst.data` file from the `mksysb` image and create a NIM `bosinst_data` resource, ready for our NIM operation.

1. Extract the `bosinst.data` file from `mksysb` image file



Assume we are extracting the bosinst.data file from the /export/images/Node1.mkysyb and place it in /other\_res directory in the NIM master.

```
# cd /other_res
# restore -xvqf /export/images/Node1.mkysyb ./bosinst.data
```

You can rename this bosinst.data file to bosinst.data.Node1 to denote that this file is used for the Node1. Change the above parameter fields in the bosinst.data.Node1 file and then create a NIM bosinst\_data resource.

```
# mv bosinst.data bosinst.data.Node1
```

## 2. Create a NIM bosinst\_data resource

```
# smitty nim_mkres
  Select "bosinst_data = config file used during base system
  installation"
```

### *Example 5-46 Define NIM bosinst\_data resource*

---

```

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

* Resource Name                    [Entry Fields]
* Resource Type                    [BID_NODE1]
* Server of Resource               bosinst_data
* Location of Resource              [master] +
                                   [/other_res/bosinst.data.Node1>/
Comments                            []
Source for Replication              [] +

```

---

## **image.data file**

The image.data file contains information required by the BOS installation program. This file describes how physical disks, volumes groups, logical volumes, file systems and paging space should be configured in the root volume group during installation. This file is generated whenever you run a **mkysyb -i** or the **mkzfile** command.

In our scenario, we are restoring the mkysyb image with a mirrored rootvg to another server which has only one disk for the rootvg. We need to edit the image.data file to change the **LV\_SOURCE\_DISK\_LIST**, **COPIES** and the **PP** parameters in order to reflect one copy of rootvg. See Example 5-47 on page 480. Note that we change the **LV\_SOURCE\_DISK\_LIST**, **COPIES** and the **PP** parameters as marked with arrow and bold.

**Note:** The Example 5-47 on page 480 shows only the changes on hd2. You need to perform the changes for all the other LVs in the rootvg

There are two approaches to work on the image.data file: With source server running or without the source server running.

If the source server is still running, you can edit the image.data file in the server, as shown in Example 5-47 on page 480, then perform a mksysb without generating a new image.data (mksysb without -i option). Just make sure that the edited/customized image.data file is not overwritten after a mksysb command.

If the source server is already down and you only have its mksysb image, we can extract the image.data file from the mksysb image.

In our scenario, we do not have the source server running, only its mksysb image. We describe the steps to extract the image.data file from the mksysb image, edit the necessary parameters, create a NIM image\_data resource, ready for NIM operation.

1. Extract the image.data file from the mksysb image, /export/images/Node1.mksysb, and edit the file.

```
# cd /other_res
# restore -xvqf /export/images/node1.mksysb ./image.data
```

You can rename this image.data file to image.data.Node1 to denote that this file is for target Node1.

```
# mv image.data image.data.Node1
```

Edit image.data.Node1 file to enable us to restore with one copy of rootvg on the target server, as shown in Example 5-47.

*Example 5-47 Edit image.data.Node1 file to install one copy of rootvg*

---

```
lv_data:
    VOLUME_GROUP= rootvg
===> LV_SOURCE_DISK_LIST= hdisk0 hdisk1 <-- change this
    LV_IDENTIFIER= 00cc544e00004c000000010bec51e388.5
    LOGICAL_VOLUME= hd2
    VG_STAT= active/complete
    TYPE= jfs2
    MAX_LPS= 32512
===> COPIES= 2 <-- change this
    LPS= 70
    STALE_PPs= 0
    INTER_POLICY= minimum
    INTRA_POLICY= center
```



```

====>  PP= 70
        BB_POLICY= relocatable
        RELOCATABLE= yes
        UPPER_BOUND= 32
        LABEL= /usr
        MAPFILE=
        LV_MIN_LPS= 83
        STRIPE_WIDTH=
        STRIPE_SIZE=
        SERIALIZE_IO= no
        FS_TAG=

```

---

## 2. Create a NIM image\_data resource

After we have changed the parameters in the image.data.Node1 file, we can create a NIM image\_data resource as shown in Example 5-49.

```

# smitty nim_mkres
  Select "image_data = config file used during base system
  installation"

```

### Example 5-49 Define NIM image\_data resource

---

Define a Resource

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

	[Entry Fields]
* Resource Name	[ID_NODE1]
* Resource Type	image_data
* Server of Resource	[master] +
* Location of Resource	/other_res/image.data.Node1>/
Comments	[]
Source for Replication	[] +

---

## Performing BOS installation

Once we have the mksysb, bosinst\_data and image\_data resources defined in our NIM master, we need to define a NIM client machine to our NIM master. With all these in place, we can start the BOS installation on the target server.

There are certain things to take care when you want to execute a “push” installation with no-prompt option.

If the target server is not already a NIM client defined to the NIM master, we need to assign the /etc/niminfo file to the target server so that this server knows who the NIM master is, Or we need to perform a push installation with a Force option.

**Note:** To perform either options, the target server must be contactable through remote shell i.e., include NIM master hostname in the target server's `~/.rhosts` file.

### 1. Assign `/etc/niminfo` file to the target server

Once the NIM client machine has been defined in the NIM master, we assign the `/etc/niminfo` file to the target server by specifying a new master to the client machine. Follow the smit steps.

```
# smitty nim_mac
Specify New Master for Client Machine
Select the NIM client machine
```

Or use the SMIT `smitty nim_switch_master` fast path, as shown in Example 5-50.

#### *Example 5-50 Specify a NIM master for a client machine*

---

Specify New Master for Client Machine

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

	[Entry Fields]
Machine Name	[NODE1]
Host Name of Network Install Master	[nimmast]
Force	no

---

### 2. Forcing a NIM push installation

After you have allocated the necessary NIM resources (`smitty nim_mac_res`), perform the SMIT steps to enable a NIM push installation with a Force option. This operation will update the `/etc/niminfo` file to the target server.

```
# smitty nim_mac_op
Select the NIM client
Select "bos_inst = perform a BOS installation"
Change the "Source for BOS runtimes file" to mkysyb;
"Force Unattended Installation Enablement?" to yes
"ACCEPT new license agreements?" to yes
```

See Example 5-51:

#### *Example 5-51 Perform NIM PUSH installation with a Force option*

---

Perform a Network Install

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

	[EntryFields]
Target Name	NODE1
<b>Source for BOS Runtime Files</b>	<b>mksysb +</b>
installp Flags	[-agX]
Fileset Names	[]
Remain NIM client after install?	yes +
<b>Initiate Boot Operation on Client?</b>	<b>yes +</b>
Set Boot List if Boot not Initiated on Client?	no +
<b>Force Unattended Installation Enablement?</b>	<b>yes +</b>
<b>ACCEPT new license agreements?</b>	<b>[yes] +</b>

---

Or you can use the command, **smitty nim\_bosinst** to perform the BOS installation. Select the target machine, the mksysb, and the SPOT. Remember to set "FORCE PUSH the installation?" to yes and "ACCEPT new license agreements?" to yes.

## 5.6 Using SAN zoning for “cloning” LPARs

This procedure is useful if you want to deploy and install AIX onto LPARs in a production environment but you do not want to perform a NIM installation over the production network. In this case, you have the choice of deploying the OS image via NIM on LPARs in a test environment, then simply move the LUNs from the test environment to the production environment and activate the LPARs in the production environment. This can be achieved via SAN zoning and LUN masking, and with minimal changes when you activate the LPARs in the production environment.

To move an LPAR from one physical machine to another, when all disks (incl. rootvg) are available via SAN, the supported methods are:

- Reinstall the LPAR from scratch
- Create an mksysb image from the running LPAR and reinstall it on the target LPAR.

These are also the supported procedures for LPARs running in a Virtual I/O Server (VIOS) environment (even though you can use other means, such as the `dd` command to copy virtual disks between logical volumes in the VIOS).

It is also possible to use SAN zoning to move SAN based rootvg disks (Logical Units) from one LPAR to another LPAR in another physical machine, assuming both LPARs are connected to the same SAN fabric, and that the OS level is supported by the destination physical machine. Both LPARs should be configured with the same type of physical machine, with the same number and type of adapters, and are not restricted to VIOS resources.

The general steps for SAN zoning of an LPARs rootvg LUNs are:

1. Document
2. Plan
3. Verify
4. Execute

### Document

LPAR environment such as, but not limited to, LPAR profiles, HMCs involved, NIM servers, IP addresses, gateways, DNS servers, NTP servers and LDAP servers.

### Plan

When can the zoning be implemented, who needs to be involved, the exact steps to verify the SAN, LAN and AIX, and a roll back plan.

## Verify

Zone a new rootvg disk to the target LPAR and perform a basic rte installation from NIM to the target LPAR, to make sure all adapters are properly allocated and working with the new IP settings.

## Execute

Open the SAN zone for the originating LPAR's SAN disks, to the new destination LPAR, stop the running LPAR, start the target LPAR, change IP settings and change the HMC and RSCT settings for the LPAR. After functional verification, remove the originating LPAR from the SAN zone.

**Important:** Depending on your particular changes after a SAN zoning, such as changes to the hostname, node id or IP addresses, you should consult the IBM documentation and verify the appropriate command sequence for your specific level of AIX and HMC software. See the manual “*Hardware Management Console for pSeries Installation and Operations Guide*”, SA38-0590.

If you change the HMC that manages the Physical Machine during the SAN zoning, you need also to change the `/var/ct/cfg/ctrmc.ac1s` to allow the target LPAR's HMC to manage the new LPAR. When moving an LPAR using mksysb cloning, the HMC connections are recreated.

To determine if you have this problem, use the `lsrsrc` command as shown in Example 5-52.

*Example 5-52 Correct output from the lsrsrc IBM.ManagementServer command*

---

```
root@lpar55:/: lsrsrc IBM.ManagementServer
Resource Persistent Attributes for IBM.ManagementServer
resource 1:
    Name                = "10.1.1.100"
    Hostname             = "10.1.1.100"
    ManagerType         = "HMC"
    LocalHostname       = "lpar5"
    ClusterTM           = "9078-160"
    ClusterSNum         = ""
    ActivePeerDomain    = ""
    NodeNameList        = {"lpar55"}
```

---

Verify also the output from the `ctsth1` command on the originating LPAR as shown in Example 5-53. In this case, verify the output from the default cluster security services trusted host list for the node (`/var/ct/cfg/ct_has.th1`).



*Example 5-53 Output from the ctsthl command*


---

```

root@lpar55:/: ctsthl -l
ctsthl: Contents of trusted host list file:
-----
Host Identity:                10.1.1.100
Identifier Generation Method: rsa512
Identifier Value:
1202009bedb21d54f57dcc2217f46f3b1352a30317e87dac7cb2d593582159e0cab3bc5a3e211b606e2fd
94a0552f0c81d36e33e346bc567810e3dbd8d8e44327ae3110103
-----
Host Identity:                loopback
Identifier Generation Method: rsa512
Identifier Value:
120200ca6e9e199d8177b0ede67ec30fb4abaaacb56779decea348bcc7a7ae904485e5f0cc284e2017e3f
295a75acb8038b6332477a8f530ff9240a0fa3f5a9e917c0b0103
-----
Host Identity:                127.0.0.1
Identifier Generation Method: rsa512
Identifier Value:
120200ca6e9e199d8177b0ede67ec30fb4abaaacb56779decea348bcc7a7ae904485e5f0cc284e2017e3f
295a75acb8038b6332477a8f530ff9240a0fa3f5a9e917c0b0103
-----
Host Identity:                10.1.1.55
Identifier Generation Method: rsa512
Identifier Value:
120200ca6e9e199d8177b0ede67ec30fb4abaaacb56779decea348bcc7a7ae904485e5f0cc284e2017e3f
295a75acb8038b6332477a8f530ff9240a0fa3f5a9e917c0b0103
-----
Host Identity:                lpar55
Identifier Generation Method: rsa512
Identifier Value:
120200ca6e9e199d8177b0ede67ec30fb4abaaacb56779decea348bcc7a7ae904485e5f0cc284e2017e3f
295a75acb8038b6332477a8f530ff9240a0fa3f5a9e917c0b010

```

---

After moving the LPAR, the new HMC's IP address in our examples is 10.1.2.100. On the LPAR and on the HMC the respective node identifier is stored in the `/etc/ct_node_id` file. This identifier is used to grant access from the HMC to the LPAR, specified in the `/var/ct/cfg/ctrmc.ac1s` file. The `ctrmc.ac1s` file contains a node's resource monitoring and control (RMC) access control list (ACL). Example 5-54 shows the `/etc/ct_node_id` file from the current and the destination HMC.

*Example 5-54 ct\_node\_id from HMCs*


---

```

root@lpar55:/: ssh hscroot@10.1.1.100 head -1 /etc/ct_node_id

```

```
9b4fc2ff2c7b3fb7
```

```
root@lpar55:/: ssh hscroot@10.1.2.100 head -1 /etc/ct_node_id  
c75d71c5f7b9279a
```

---

The **ctskeygen** command is used to update the RMC security files `ct_has.pkf` (public key file), `ct_has.qkf` (private key file), `ct_has.th1` (trusted host list file). Example 5-55 displays the current public key value for the local system.

*Example 5-55 Output from the ctskeygen command*

---

```
root@lpar55:/: /usr/sbin/rsct/bin/ctskeygen -d  
120200b6b9979e2d01cb5d97189db870e47af464f9ebb9cbe57f1e1045ed07f83c72e51c1ebbb93a  
c642d3c37d5212c8T6hfe83Dbufd3e5f600b65c12f3726ef7c03896107f9f0103  
(generation method: rsa512)
```

---

## 5.7 System maintenance for NIM clients

Pro-active system maintenance for NIM clients is important if we wish to keep our systems stable and up-to-date. System maintenance includes installing software fixes or updates, verifying installed software, reviewing installed software and operating system levels and performing hardware maintenance and diagnostics. We can use NIM to perform all of these maintenance activities in a very fast and efficient way. We can perform maintenance on a single system or a group of systems thus reducing the time and effort required to keep our systems updated.

In this section, we will discuss the following topics:

- ▶ Software maintenance approach.
- ▶ Creating a mksysb backup of a NIM client.
- ▶ Performing software maintenance on NIM clients.
- ▶ Booting a NIM client in maintenance and diagnostic mode.

### 5.7.1 Software maintenance approach

Software maintenance is an important task for any AIX or NIM administrator. By software maintenance we mean applying AIX PTFs, APARs, Service Packs, Concluding Service Packs (CSPs) and Technology Levels (TLs) on a NIM client. It is generally a good idea to keep your NIM environment running at the latest most stable release of AIX code. This will ensure that your systems can take advantage of the latest features and enhancements for AIX. You should develop a maintenance strategy for your systems that is most appropriate for your

environment. IBM introduced a new service strategy for AIX 5L in early 2006. This strategy offers several maintenance models but the one that will meet the needs of most sites is the “Maximum Stability” model.

### ***Maximum stability maintenance model***

The maximum stability maintenance strategy is designed for sites that would like to adopt a new TL every six to eight months and take advantage of the latest group of PTFs available in a Concluding Service Pack. By this we mean that a system that is currently running TL4 would wait for TL5 and its CSP to be released before applying the software. In between TLs, the site could also install IFs (Interim Fixes) to address security and other issues that may be required before the latest TL and CSP is available. Refer to the “[The AIX 5L Service Strategy and Best Practices: AIX 5L Service Scenarios and Tools](#)” document for more information located at the following Web site:

[http://www-03.ibm.com/servers/eserver/support/unixservers/aix\\_service\\_strategy.pdf](http://www-03.ibm.com/servers/eserver/support/unixservers/aix_service_strategy.pdf)

### ***Software maintenance on the NIM master***

Before you can apply maintenance to a NIM client you must first make sure that your NIM master has been updated with the highest level of code first. But before you can do that you must first obtain the software in some form. The latest TLs, SPs and CSPs can be downloaded from the IBM fix central Web site:

<http://www.ibm.com/eserver/support/fixes/fcgui.jsp>

However, we recommend that you use the Service Update Management Assistant (SUMA) in conjunction with NIM to download and manage the latest fixes. You should create a new file system to which SUMA can download the TL and CSP filesets. A new `lpp_source` should also be created for the new TL/CSP which initially only contains the base level code. This new `lpp_source` will be updated with the fixes downloaded via SUMA (for example, Base+TL+CSP). Create a new SPOT from the new `lpp_source` and update it each time the `lpp_source` is updated. Section 4.10, “NIM and Service Update Management Assistant” on page 321 has additional information on using NIM and SUMA. Section 5.1, “Updating the NIM environment (`lpp_source`, SPOT)” on page 418 also has more information on updating software in a NIM environment.

### ***Using NIM to perform software maintenance on NIM clients***

Before applying the latest TL on a NIM client, it is a good idea to test the code on a non-critical system. This will allow you to verify the code and refine your installation procedures before applying it to any production systems.

The current level of your client can be determined with the `oslevel` command. The `-s` flag shows the current Service Pack level and `-r` displays the current Technology Level as shown in Example 5-56.

*Example 5-56 oslevel command*

---

```
{nimmast}:/ # rsh lpar4 oslevel -r
5300-04
{nimmast}:/ # rsh lpar4 oslevel -s
5300-04-03
```

---

You should always create a `mksysb` backup of the client before performing any type of software maintenance task. You can use NIM to backup the client from the master (or script a that runs on the client to backup to the NIM master), as shown in Example 5-57.

*Example 5-57 Creating a mksysb of a NIM client from the NIM master*

---

```
{nimmast}:/ # nim -o define -t mksysb -a server=master -a source=LPAR4 -a
mk_image=yes -a location=/export/images/mksysb.lpar4.5303.Fri lpa
r4-mksysb
+-----+
                System Backup Image Space Information
                (Sizes are displayed in 1024-byte blocks.)
+-----+

Required = 1072580 (1047 MB)    Available = 1691148 (1652 MB)
```

Data compression will be used by the system backup utilities which create the system backup image. This may reduce the required space by up to 50 percent.

Creating information file (/image.data) for rootvg..

Creating list of files to back up.  
 Backing up 26669 files.....  
 26669 of 26669 files (100%)  
 0512-038 savevg: Backup Completed Successfully.

```
{nimmast}:/ # lsnim -t mksysb
lpar1nim-mksysb      resources      mksysb
MK_LPAR6_AIX530403  resources      mksysb
MKSYSB_VI012_53_ML3  resources      mksysb
MK_AIX5104           resources      mksysb
mksysb_lpar5c        resources      mksysb
mksysb_lpar5d        resources      mksysb
```

mksysb_vlpar1	resources	mksysb
MK_VI01	resources	mksysb
VLPAR3_phys_mksysb	resources	mksysb
MK_MST_53ML4	resources	mksysb
VLPAR3_p5_mksysb	resources	mksysb
MK_MST_15JUN06	resources	mksysb
MKSYSB_VLPAR5_p5	resources	mksysb
<b>lpar4-mksysb</b>	<b>resources</b>	<b>mksysb</b>

```
{nimmast}:/ # lsnim -l lpar4-mksysb
lpar4-mksysb:
  class      = resources
  type       = mksysb
  arch       = power
  Rstate     = ready for use
  prev_state = unavailable for use
  location  = /export/images/mksysb.lpar4.5300.Fri
  version    = 5
  release    = 3
  mod        = 0
  oslevel_r = 5300-03
  alloc_count = 0
  server     = master
```

---

You can perform software maintenance tasks on one or more NIM clients at once. If you have defined a NIM machine group you can choose to perform an operation on the machine group and update all the group members (clients) at the same time. Depending on the size and type of your environment you may only need to update on a per client basis but the flexibility to manage multiple clients exists.

In the following examples, we are performing software maintenance on a NIM client via the NIM master. Our task is to apply the latest maintenance to our client. Once completed, our client, which was running AIX 5L V5.3 ML3 (5300-03), will be updated to AIX 5L V5.3 TL4 SP3 (5300-04-03).

We will perform the following tasks:

- ▶ First, we will commit any software in an applied state. There may be software in an applied state from a previous maintenance task.
- ▶ Software will be applied via a NIM (cust) update\_all operation.
- ▶ LPAR4 is the NIM client.
- ▶ The lpp\_source name is LPP\_53\_ML4.
- ▶ COMMIT software updates? No.
- ▶ SAVE replaced file? Yes.

- ▶ Accept new license agreements? Yes.
- ▶ We verify that the `oslevel` and `instfix` commands display the correct TL and SP information. We also verify the filesets installed from the NIM master via the NIM `lppchk` and `showlog` operation. We also perform a `fix_query` NIM operation on the NIM client.

We have chosen not to commit the software updates and to save the replaced files. This will allow us to reject a fileset or filesets if there are issues after the update. This is called applying software updates. We can choose to commit the updates later which will remove the saved files from the system.

**Attention:** It is not recommended to reject a whole Technology Level once it has been applied to an AIX 5L V5.3 system. TLs are usually quite large in terms of the amount of filesets updated. It is faster and less risky to fall back to the previous level using other methods such as `alt_disk_install`, `multibos`, or a `mksysb restore` via NIM.

Note, as we are applying a new TL, we perform a PREVIEW first so that we can review what will actually be installed and check for any missing prerequisites. In most cases when applying a new TL you will be asked to update the `bos.rte.install` fileset first. All other filesets are deferred until it is updated. Once updated, you will be able to apply the remaining filesets.

Ensure that you have taken a backup of any other data volume groups (non-rootvg volume groups) also. It is recommended that all your applications be shutdown and all users logged off the system before you apply any AIX operating system maintenance. Also check the AIX error report before and after to see if there are any issues (hardware or other) with the system. It may also be worth cleaning up your old SMIT log files beforehand. This will make reviewing the logs much less confusing after the maintenance has been applied. See Example 5-58.

*Example 5-58 bos.rte.install must be updated first when applying a TL*

---

```
{nimmast}:/ # smit nim_task_maint
Commit Applied Software Updates (Remove Saved Files)
LPAR4          machines          standalone

                Commit Applied Software Updates (Remove Saved Files)

* Target                LPAR4
* Software Names        [all] +

Force                  no +

PREVIEW only?          [no] +
```

```

COMMIT requisites?                                [yes] +

nimmast}:/ # smit nim_task_inst

Update Installed Software to Latest Level (Update All)
LPAR4          machines          standalone
LPP_53_ML4     resources          lpp_source

Update Installed Software to Latest Level (Update All)

* Installation Target                            LPAR4
* LPP_SOURCE                                     LPP_53_ML4
Software to Install                             update_all

Customization SCRIPT to run after installation    [] +
(not applicable to SPOTs)

Force                                             no +

installp Flags
PREVIEW only?                                  [yes] +
COMMIT software updates?                       [no] +
SAVE replaced files?                           [yes] +
AUTOMATICALLY install requisite software?     [yes] +
EXTEND filesystems if space needed?            [yes] +
OVERWRITE same or newer versions?             [no] +
VERIFY install and check file sizes?          [no] +
ACCEPT new license agreements?                 [yes] +
(AIX V5 and higher machines and resources)
Preview new LICENSE agreements?                [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)                                [] +#

COMMAND STATUS

Command: OK          stdout: yes          stderr: no

Before command completion, additional instructions may appear below.

```





```

Software to Install                                update_all

Customization SCRIPT to run after installation    [] +
(not applicable to SPOTs)

Force                                             no +

installp Flags
PREVIEW only?                                   [no] +
COMMIT software updates?                       [no] +
SAVE replaced files?                           [yes] +
AUTOMATICALLY install requisite software?     [yes]+
EXTEND filesystems if space needed?           [yes]+
OVERWRITE same or newer versions?             [no] +
VERIFY install and check file sizes?         [no] +
ACCEPT new license agreements?               [yes]+
(AIX V5 and higher machines and resources)
Preview new LICENSE agreements?               [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)                                 [] +#

```

---

We verify the installed filesets by running an `lppchk` operation on the NIM client from the NIM master. We also review the `installp` log with the `showlog` operation. See Example 5-60.

*Example 5-60 Running an `lppchk` and `showlog` operation after the TL is applied*

---

```

{nimmast}:/ # nim -o lppchk -a lppchk_flags="-c -m3" LPAR4
/usr/bin/lppchk: 0504-230 8367 files have been checked.
/usr/bin/lppchk: 0504-230 13186 files have been checked.
/usr/bin/lppchk: 0504-230 14798 files have been checked.
/usr/bin/lppchk: 0504-230 561 files have been checked.
/usr/bin/lppchk: 0504-230 377 files have been checked.

```

```

{nimmast}:/ # nim -o showlog -a log_type=niminst LPAR4
BEGIN:Tue Jul 4 21:48:44 2006:070411484406
Command line is:
/usr/sbin/installp -acgXY -e /var/adm/ras/nim.installp -f \
/tmp/.genlib.installp.list.548880-d /tmp/_nim_dir_557166/mnt0

```

```

+-----+
|                                     |
|                                     |
+-----+
done
.....
SUCSESSES
-----

Filesets listed in this section passed pre-installation verification
and will be installed.

Selected Filesets
-----
Java14.sdk 1.4.2.75                # Java SDK 32-bit
X11.adt.include 5.3.0.50           # AIXwindows Application Devel...
X11.apps.aixterm 5.3.0.50         # AIXwindows aixterm Application
X11.apps.config 5.3.0.50          # AIXwindows Configuration App...
X11.apps.xdm 5.3.0.50             # AIXwindows xdm Application
X11.base.lib 5.3.0.50             # AIXwindows Runtime Libraries
X11.base.rte 5.3.0.50             # AIXwindows Runtime Environment
.....
bos.rte.libpthread 5.3.0.50       USR      COMMIT   SUCCESS
bos.rte.serv_aid 5.3.0.50         USR      COMMIT   SUCCESS
bos.rte.serv_aid 5.3.0.50         ROOT     COMMIT   SUCCESS
rsct.core.utils 2.4.5.2           USR      COMMIT   SUCCESS
rsct.core.utils 2.4.5.2           ROOT     COMMIT   SUCCESS
rsct.core.sec 2.4.5.1             USR      COMMIT   SUCCESS
rsct.core.sec 2.4.5.1             ROOT     COMMIT   SUCCESS
rsct.core.rmc 2.4.5.2             USR      COMMIT   SUCCESS
rsct.core.rmc 2.4.5.2             ROOT     COMMIT   SUCCESS
rsct.core.errm 2.4.5.1            USR      COMMIT   SUCCESS
rsct.core.errm 2.4.5.1            ROOT     COMMIT   SUCCESS
rsct.core.hostrm 2.4.5.1          USR      COMMIT   SUCCESS
rsct.core.hostrm 2.4.5.1          ROOT     COMMIT   SUCCESS
rsct.core.gui 2.4.5.1             USR      COMMIT   SUCCESS

installp: * * * A T T E N T I O N ! ! !
Software changes processed during this session require this system
and any of its diskless/dataless clients to be rebooted in order
for the changes to be made effective.

END:Tue Jul 4 22:08:05 2006:070412080506

```

Using rsh from the NIM master we run the **oslevel** and **instfix** commands on the NIM client to check that the correct AIX level is shown and that all filesets have been updated correctly. We then run a **fix\_query** operation from the master, which is the same as running the **instfix** command on the client, as shown in Example 5-61.

*Example 5-61 oslevel, instfix commands and fix\_query operation*


---

```
{nimmast}:/ # rsh lpar4 instfix -icqk 5300-04_AIX_ML | grep ":-:"
{nimmast}:/ # rsh lpar4 instfix | grep AIX
    All filesets for 5300-02_AIX_ML were found.
    All filesets for 5.3.0.0_AIX_ML were found.
    All filesets for 5300-01_AIX_ML were found.
    All filesets for 5300-03_AIX_ML were found.
    All filesets for 5300-04_AIX_ML were found.
{nimmast}:/ # rsh lpar4 oslevel -r
5300-04
{nimmast}:/ # rsh lpar4 oslevel -s
5300-04-03
{nimmast}:/ # nim -o fix_query LPAR4 | grep AIX
    All filesets for 5300-02_AIX_ML were found.
    All filesets for 5.3.0.0_AIX_ML were found.
    All filesets for 5300-01_AIX_ML were found.
    All filesets for 5300-03_AIX_ML were found.
    All filesets for 5300-04_AIX_ML were found.
```

---

In Example 5-62, we perform software maintenance on a NIM machine group. We install the `bos.alt_disk_install` filesets on all members of the group `NIM_MAC_GRP_1`.

*Example 5-62 Performing software maintenance on a NIM machine group*


---

```
{nimmast}:/ # lsnim -t mac_group
LPAR123456      groups      mac_group
nimgrp         groups      mac_group
NIM_MAC_GRP_1 groups    mac_group

{nimmast}:/ # lsnim -l NIM_MAC_GRP_1
NIM_MAC_GRP_1:
  class      = groups
  type       = mac_group
  comments   = NIM Machine Group One.
  member1   = LPAR4
  member2   = LPAR5

{nimmast}:/ # smit nim_task_inst
Install and Update from ALL Available Software
NIM_MAC_GRP_1  groups      mac_group
LPP_53_ML4     resources  lpp_source
```

Install and Update from ALL Available Software

```
* Installation Target          NIM_MAC_GRP_1
* LPP_SOURCE                   LPP_53_ML4
* Software to Install          [bos.alt_disk_install] +
```

```

Customization SCRIPT to run after installation      [] +
(not applicable to SPOTs)

Force                                              no +

installp Flags
PREVIEW only?                                    [no] +
COMMIT software updates?                        [yes] +
SAVE replaced files?                            [no] +
AUTOMATICALLY install requisite software?      [yes] +
EXTEND filesystems if space needed?            [yes] +
OVERWRITE same or newer versions?              [no] +
VERIFY install and check file sizes?          [no] +
ACCEPT new license agreements?                 [no] +
(AIX V5 and higher machines and resources)
Preview new LICENSE agreements?                 [no] +

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)                                  [] +#

```

## COMMAND STATUS

```
Command: OK          stdout: yes          stderr: no
```

Before command completion, additional instructions may appear below.

```

+-----+
Initiating "cust" Operation
+-----+
Allocating resources ...

Initiating the cust operation on machine 1 of 2: LPAR4 ...

Initiating the cust operation on machine 2 of 2: LPAR5 ...

+-----+
"cust" Operation Summary
+-----+
Target          Result
-----
LPAR4           INITIATED

```

LPAR5 INITIATED

Note: Use the `lsnim` command to monitor progress of "INITIATED" targets by viewing their NIM database definition.

---

Operations performed on machine groups are by default performed asynchronously on members of the group, as shown in Example 5-62. NIM does not wait for an operation to complete on one group member before initiating the operation on the next member. When performing operations asynchronously, it is not possible for NIM to display all the output as it occurs on each client. Therefore, you should use the `lsnim` command to check the states of the group members to determine how far, and how successfully, the operations have executed (see Example 5-64). If errors do occur, the log files on client machines can be viewed using the NIM `showlog` operation. To change the behavior of NIM group operations from asynchronous to synchronous, use the `async=no` attribute when running the `nim` command.

The number of machines permitted in a machine group is not explicitly limited by NIM. However, the following factors limit the number for practical reasons:

- ▶ Operation being Performed. Operations that are not resource-intensive (such as the `maint` or `showlog` operations) may be performed on a group containing any number of machines. Operations that are resource-intensive (such as `cust` or `bos_inst`) are limited by the throughput of the network, the disk access throughput of the installation servers, and the platform type of servers.
- ▶ NFS Export Limitations. The maximum number of hosts to which a file or directory may be exported with root permissions is limited by NFS to 256. Also, the length of a line in an exports file has an upper limit which could determine the maximum number of machines permitted in a group. For information on how to increase the number of machines to which a resource can be allocated, refer to the section "Exporting NIM resources globally" in the manual AIX 5L V5.3 Installation and Migration, SC23-4887-03.

*Example 5-63 Displaying the installation log for a NIM machine group*

---

```
{nimmast}:/ # smit nim_grp_op
NIM_MAC_GRP_1      groups      mac_group
showlog            = display a log in the NIM environment
```

Display Installation Log

```
* Target Name          NIM_MAC_GRP_1
Log Type              [niminst] +
Only Show Last Entry in Log?  yes +
```

COMMAND STATUS

```
Command: OK          stdout: yes          stderr: no
```

Before command completion, additional instructions may appear below.

```
[TOP]
```

```
+-----+
|                                     |
|               Performing "showlog" Operation               |
|-----+
| Performing showlog operation on machine 1 of 2: LPAR4 ...
```

```
BEGIN:Fri Jun 16 15:02:48 2006:061613024806
```

```
Command line is:
```

```
/usr/sbin/installp -acNgX -e /var/adm/ras/nim.installp -f \
/tmp/.genlib.installp.list.323610-d /tmp/_nim_dir_307446/mnt0
```

```
+-----+
|                                     |
|               Pre-installation Verification...             |
|-----+
| done
```

```
SUCSESSES
-----
```

```
Filesets listed in this section passed pre-installation verification
and will be installed.
```

```
Selected Filesets
```

```
-----
bos.alt_disk_install.boot_images 5.3.0.30 # Alternate Disk Installation ...
bos.alt_disk_install.boot_images 5.3.0.40 # Alternate Disk Installation ...
bos.alt_disk_install.rte 5.3.0.30        # Alternate Disk Installation ...
bos.alt_disk_install.rte 5.3.0.40        # Alternate Disk Installation ...
```

```
<< End of Success Section >>
```

```
FILESET STATISTICS
```

```
-----
      4 Selected to be installed, of which:
        4 Passed pre-installation verification
-----
```

```
[MORE...164]
```

---

If there are any issues with a software maintenance task, you will see that the Cstate and prev\_state information will reflect the status of the operations that are running or have run. In Example 5-64 we can see that the prev\_state for LPAR4 was “*lppchk operation being performed*” and the prev\_state for LPAR5 was “*customization is being performed*”. Their current state is “*ready for a NIM operation*” so no problems were encountered in any of the previous NIM operations on these clients.

*Example 5-64 Displaying NIM client Cstate and prev\_state information*


---

```
{nimmast}:/ # lsnim -l LPAR4 LPAR5
LPAR4:
  class      = machines
  type       = standalone
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = NET_EN1 lpar4 0
  cable_type1 = tp
  Cstate    = ready for a NIM operation
  prev_state = lppchk operation is being performed
  Mstate     = currently running
  cpuid      = 001A85D24C00
  Cstate_result = success
  current_master = nimmast
  sync_required = yes
LPAR5:
  class      = machines
  type       = standalone
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = NET_EN1 LPAR5 0
  cable_type1 = N/A
  Cstate    = ready for a NIM operation
  prev_state = customization is being performed
  Mstate     = currently running
  cpuid      = 00CC544E4C00
  Cstate_result = success
```

---

The output from the `installp` process is also logged on the NIM client to `/var/adm/ras/nim.installp`.

You can also perform other software maintenance related tasks with NIM (via SMIT), for example listing installed software, checking for installed fixes and committing applied filesets as shown in Example 5-65 and Example 5-66.

*Example 5-65 NIM software maintenance tasks*


---

```
{nimmast}:/ # smit nim_task_maint
      Software Maintenance and Utilities

Commit Applied Software Updates (Remove Saved Files)
Reject Applied Software Updates (Use Previous Version)
Remove Installed Software
```

Copy Software to Hard Disk for Future Installation  
 Add Software to an lpp\_source  
 Remove Software from an lpp\_source  
 Eliminate Unnecessary Software Images in an lpp\_source

Check Software File Sizes After Installation  
 Verify Software Installation and Requisites

Clean Up After Failed or Interrupted Installation

---

#### *Example 5-66 NIM software listing and related tasks*

---

```
{nimmast}:/ # smit nim_task_maint
          List Software and Related Information
```

```
List Installed Software and Related Information
List Software on Media and Related Information
```

---

The nim command can also be used from the NIM master to perform several software maintenance tasks:

- ▶ Querying installed software (**lslpp**) on a client (or client's) and a machine group:
 

```
{nimmast}:/ # nim -o lslpp LPAR4
{nimmast}:/ # nim -o lslpp LPAR4 LPAR5
{nimmast}:/ # nim -o lslpp NIM_MAC_GRP_1
```
- ▶ Querying installed fixes on a client:
 

```
{nimmast}:/ # nim -o fix_query LPAR4 | grep AIX
```
- ▶ Previewing the installation of the bos.alt\_disk\_install filesets on a NIM client:
 

```
{nimmast}:/ # nim -o cust -a lpp_source=LPP_53_ML4 -a \
installp_flags="-pacgXY" -a filesets=bos.alt_disk_install LPAR4
```
- ▶ Performing an installation of the bos.alt\_disk\_install filesets on a NIM client:
 

```
{nimmast}:/ # nim -o cust -a lpp_source=LPP_53_ML4 -a \
installp_flags="-acgXY" -a filesets=bos.alt_disk_install LPAR4
```
- ▶ Previewing the removal of the bos.alt\_disk\_install filesets on a NIM client:
 

```
{nimmast}:/ # nim -o cust -a lpp_source=LPP_53_ML4 -a \
installp_flags="-pug" -a filesets=bos.alt_disk_install LPAR4
```
- ▶ Performing a removal of the bos.alt\_disk\_install filesets on a NIM client:
 

```
{nimmast}:/ # nim -o cust -a lpp_source=LPP_53_ML4 -a \
installp_flags="-ug" -a filesets=bos.alt_disk_install LPAR4
```
- ▶ Displaying the installp log for a NIM client:



```
{nimmast}:/ # nim -o showlog -a log_type=niminst LPAR4
```

- ▶ Previewing an update\_all on a NIM client:

```
{nimmast}:/ # nim -o cust -a lpp_source=LPP_53_ML4 -a \
installp_flags="-pacgXY" -a fixes=update_all LPAR4
```

- ▶ Performing an update\_all on a NIM client (installing the bos.rte.install fileset first):

```
{nimmast}:/ # nim -o cust -a lpp_source=LPP_53_ML4 -a \
installp_flags="-acgXY" -a filesets=bos.rte.install LPAR4
```

```
{nimmast}:/ # nim -o cust -a lpp_source=LPP_53_ML4 -a \
fixes=update_all LPAR4
```

- ▶ Performing an update\_all on a NIM machine group as shown in Example 5-67.

---

*Example 5-67 Performing an update\_all on a NIM machine group*

---

```
{nimmast}:/ # nim -o cust -a lpp_source=LPP_53_ML4 -a fixes=update_all
NIM_MAC_GRP_1
```

```
+-----+
```

```
Initiating "cust" Operation
```

```
+-----+
```

```
Allocating resources ...
```

```
Initiating the cust operation on machine 1 of 2: LPAR4 ...
```

```
Initiating the cust operation on machine 2 of 2: LPAR5 ...
```

```
+-----+
```

```
"cust" Operation Summary
```

```
+-----+
```

Target	Result
-----	-----
LPAR4	INITIATED
LPAR5	INITIATED

---

**Note:** Use the `lsnim` command to monitor progress of "INITIATED" targets by viewing their NIM database definition.

Performing an installation of fileset's and RPMs on a NIM client. In Example 5-68, we install the OpenSSH and OpenSSL software. Note that both software packages are located in our lpp\_source (/AIX53ML4/LPP\_53\_ML4/) but that the

RPMs (OpenSSL) are located in the RPMS/ppc directory. The geninstall command will find and install the RPMs from that location. The OpenSSH filesets are located in install/ppc.

*Example 5-68 Installing the OpenSSH and OpenSSL software*

```
{nimmast}:/ #nim -o cust -a lpp_source=lpp5305 -a installp_flags=agXY -a
filesets="openssl-0.9.7d-2 openssh.base" LPAR4

Validating RPM package selections ...

openssl
#####

+-----+
                                     Pre-installation Verification...
+-----+

Verifying selections...done
Verifying requisites...done
Results...

SUCSESSES
-----
Filesets listed in this section passed pre-installation verification
and will be installed.

Selected Filesets
-----
openssh.base.client 4.1.0.5301          # Open Secure Shell Commands
openssh.base.server 4.1.0.5301        # Open Secure Shell Server

<< End of Success Section >>

FILESET STATISTICS
-----
  2 Selected to be installed, of which:
    2 Passed pre-installation verification
  ----
  2 Total to be installed

+-----+
                                     Installing Software...
+-----+
```

```
installp: APPLYING software for:
    openssl.base.client 4.1.0.5301
    openssl.base.server 4.1.0.5301
```

. . .

#### Installation Summary

Name	Level	Part	Event	Result
openssl.base.client	4.1.0.5301	USR	APPLY	SUCCESS
openssl.base.server	4.1.0.5301	USR	APPLY	SUCCESS
openssl.base.client	4.1.0.5301	ROOT	APPLY	SUCCESS
openssl.base.server	4.1.0.5301	ROOT	APPLY	SUCCESS

**Tip:** It is possible to install RPMs using FTP via the NIM master. For example, the following command will download the openssl rpm from the NIM master and install it on lpar4.

```
{lpar4}:/ # rpm -i ftp://root@nimmast/AIX53ML4/LPP_53_ML4/RPMS/ppc/\
openssl-0.9.7g-1.aix5.1.ppc.rpm
Password for root@nimmast:
{lpar4}:/ # rpm -qa
cdrecord-1.9-4
mkisofs-1.13-4
AIX-rpm-5.2.0.50-1
openssl-0.9.7g-1
{lpar4}:/ #
```

### ***Client initiated software maintenance tasks (“pull”)***

There are several NIM operations related to software maintenance that the client can initiate via the `nimclient` command. The client can perform operations upon itself only.

- ▶ Listing all available resources for the client:
 

```
{lpar4}:/ # nimclient -l -L LPAR4
```
- ▶ Allocating an lpp\_source to the client:
 

```
{lpar4}:/ # nimclient -o allocate -a lpp_source=lpp5305
```
- ▶ Deallocate a resource from the client:
 

```
{lpar4}:/ # nimclient -o deallocate -a lpp_source=lpp5305
```
- ▶ Show allocated resources for the client:
 

```
{lpar4}:/ # nimclient -l -c resources LPAR4
```

```
lpp5305      lpp_source
```

- ▶ Starting a BOS installation from the client (lpp\_source and SPOT allocated first. **accept\_licenses** set to yes.):

```
{lpar4}:/ # nimclient -o allocate -a lpp_source=LPP_53_ML4 -a spot=SPOT_53_ML4
{lpar4}:/ # nimclient -o bos_inst -a accept_licenses=yes
```

```
Broadcast message from root@lpar4 (tty) at 10:21:57 ...
```

```
*****
*****
*****
```

```
NIM has initiated a bos installation operation on this machine.
Automatic reboot and reinstallation will follow shortly...
```

```
*****
*****
*****
```

- ▶ Performing an update\_all (cust) operation from the client:

```
{lpar4}:/ # nimclient -o cust -a lpp_source=lpp5305 -a fixes=update_all
BEGIN:Mon Jun 26 10:24:38 2006:062608243806
```

```
Command line is:
```

```
/usr/sbin/installp -agQX -e /var/adm/ras/nim.installp -f \
/tmp/.genlib.installp.list.270544-d /tmp/_nim_dir_327922/mnt0
```

```
+-----+
|                                     |
|                               Pre-installation Verification...          |
|                                     |
+-----+
```

```
.....Output removed.....
```

- ▶ Resetting the NIM client state:

```
{LPAR5}:/ # nimclient -Fo reset
```

- ▶ Setting the date and time to that of the NIM master:

```
{LPAR5}:/ # nimclient -d
```

## 5.7.2 Booting in maint\_boot and diag modes

This section provides information on booting a NIM client in maintenance mode. If you need to perform maintenance on a NIM client, you can enter maintenance mode directly by enabling the **maint\_boot** operation for a NIM standalone machine.

## Booting in maintenance mode

Follow this procedure for initiating the `maint_boot` operation from the NIM master (see Example 5-69 on page 507):

### Example 5-69 Enabling maintenance boot for a NIM client

---

```
{nimmast}:/ # smit nim_mac_op
LPAR5      machines      standalone
maint_boot      = enable a machine to boot in maintenance mode
```

Enable Maintenance Boot

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

	[Entry Fields]
Target Name	LPAR5
* SPOT	[SPOT_53_ML4] +

```
{nimmast}:/ #lsnim -l VLPAR_p5
LPAR5:
class          = machines
type           = standalone
connect        = shell
platform       = chrp
netboot_kernel = mp
if1            = NET_EN1 LPAR5 0
cable_type1    = N/A
Cstate       = maintenance boot has been enabled
prev_state     = not running
Mstate        = in the process of booting
info           = setting_console
boot           = boot
spot           = SPOT_53_ML4
cpuid          = 00CC544E4C00
control        = master
```

---

1. Enter the **smit nim\_mac\_op fast** path
2. Select the client's machine object
3. Select the **maint\_boot** operation
4. Select the SPOT to be used for the operation
5. Press Enter to enable the client for maintenance boot
6. Verify the NIM client state is now set to "*maintenance boot has been enabled*"
7. Network boot the client. It will boot from the SPOT in maintenance mode

8. You are presented with a menu to perform various maintenance activities as shown in Example 5-70 on page 508. In our scenario, we access a root volume group. We may want to do this if we are troubleshooting a system which does not boot normally. For example, because of file system or JFS log corruption.
9. We are presented with a maintenance shell
10. When we are finished we can halt the machine and restart it
11. We reset the NIM client. Its Cstate is now set to “*ready for a NIM operation state*”.

*Example 5-70 Accessing rootvg in maintenance boot mode*

---

Maintenance

Type the number of your choice and press Enter.

```
>>> 1 Access a Root Volume Group
      2 Copy a System Dump to Removable Media
      3 Access Advanced Maintenance Functions
      4 Erase Disks
```

```
      88 Help ?
      99 Previous Menu
```

```
>>> Choice [1]:
```

Warning:

If you choose to access a root volume group, you will NOT be able to return to the Base Operating System Installation menus without rebooting.

Type the number of your choice and press Enter.

```
      0 Continue
```

```
      88 Help ?
>>> 99 Previous Menu
```

```
>>> Choice [99]:0
```

## Access a Root Volume Group

Type the number for a volume group to display the logical volume information and press Enter.

```
1)Volume Group 00cc544e00004c000000010bcc5a1e11 contains these disks:
      hdisk0 10240          vscsi
```

Choice:1

Volume Group Information

```
-----
Volume Group ID 00cc544e00004c000000010bcc5a1e11 includes the following
logical volumes:
```

```
hd5 hd6 hd8 hd4 hd2 hd9var hd3 hd1 hd10opt
-----
```

Type the number of your choice and press Enter.

```
1) Access this Volume Group and start a shell
2) Access this Volume Group and start a shell before mounting filesystems
```

99) Previous Menu

Choice [99]:1

```
Importing Volume Group...
rootvg
Checking the / filesystem.
```

```
log redo processing for /dev/rhd4
syncpt record at 114d1ac
end of log 117924c
syncpt record at 114d1ac
syncpt address 114ba2c
number of log records = 1551
number of do blocks = 94
number of nodo blocks = 3
/dev/rhd4 (/): ** Unmounted cleanly - Check suppressed
Checking the /usr filesystem.
```

```
/dev/rhd2 (/usr): ** Unmounted cleanly - Check suppressed
Saving special files and device configuration information.
Checking and mounting the /tmp filesystem.
```

```
/dev/rhd3 (/tmp): ** Unmounted cleanly - Check suppressed
Checking and mounting the /var filesystem.
```

```
/dev/rhd9var (/var): ** Unmounted cleanly - Check suppressed
Filesystems mounted for maintenance work.
#
```

---

Resetting the NIM client after a maintenance boot is shown in Example 5-71.

*Example 5-71 Resetting a NIM client after a maintenance boot*

---

```
{nimmast}:/ # smit nim_mac_op
```

```
LPAR5      machines      standalone
reset      = reset an object's NIM state
```

Reset the NIM State of a Machine

```
Target Name                LPAR5
Deallocate All Resources?  yes +
    (removes all root data for
    diskless/dataless clients)
Force                       yes +
```

```
{nimmast}:/ # lsnim -l LPAR5
```

```
LPAR5:
  class      = machines
  type       = standalone
  connect    = shell
  platform   = chrp
  netboot_kernel = mp
  if1        = NET_EN1 LPAR5 0
  cable_type1 = N/A
  Cstate     = ready for a NIM operation
  prev_state = maintenance boot has been enabled
  Mstate     = currently running
  Cstate_result = reset
```

---

It is also possible for the client to configure itself for a maintenance mode boot:

```
{LPAR5}:/ # nimclient -o maint_boot -a spot=SPOT_53_ML4
```

### ***Performing boot diagnostics on NIM clients***

This section provides information on performing hardware diagnostics on a NIM client.



Hardware diagnostics can be performed on all NIM clients using a diagnostic boot image from the NIM master, rather than booting from a diagnostic tape or CD-ROM. This is useful for standalone clients because the diagnostics do not have to be installed on the local disk. Diagnostic support comes from a SPOT resource.

The following procedure initiates the **diag** operation from the master:

1. Enter the **smit nim\_mac\_op** fastpath as shown in Example 5-72 on page 511.
2. Select the machine object.
3. Select the **diag** operation from the list of operations.
4. Verify the NIM client state is now set to “*diagnostic boot has been enabled*”
5. Network boot the client in service mode. It will boot from the SPOT in diagnostic mode as shown in Example 5-73 on page 512.
6. The diagnostic menu appears. Various hardware diagnostic activities can be performed.
7. When we are finished we can halt the machine and restart it.
8. We reset the NIM client. Its Cstate is now set to “*ready for a NIM operation state*”. See Example 5-71 on page 510.

*Example 5-72 Enabling diagnostics boot for a NIM client*

```
{nimmast}:/ # smit nim_mac_op
LPAR5      machines      standalone
diag              = enable a machine to boot a diagnostic image
```

Enable Diagnostic Boot

```
Target Name          LPAR5
* SPOT                [SPOT_53_ML4] +
```

```
{nimmast}:/ # lsrim -l LPAR5
LPAR5:
  class          = machines
  type           = standalone
  connect        = shell
  platform       = chrp
  netboot_kernel = mp
  if1            = NET_EN1 LPAR5 0
  cable_type1    = N/A
  Cstate       = diagnostic boot has been enabled
  prev_state     = ready for a NIM operation
  Mstate        = currently running
  boot           = boot
```

```
spot          = SPOT_53_ML4
control       = master
Cstate_result = reset
```

---

You will be prompted to define the system console when booting in diagnostic mode.

*Example 5-73 Booting the client in diag mode*

---

\*\*\*\*\* Please define the System Console. \*\*\*\*\*

Type a 1 and press Enter to use this terminal as the system console.

Pour definir ce terminal comme console systeme, appuyez sur 1 puis sur Entree.

Taste 1 und anschliessend die Eingabetaste druecken, um diese Datenstation als Systemkonsole zu verwenden.

Premere il tasto 1 ed Invio per usare questo terminal come console.

Escriba 1 y pulse Intro para utilizar esta terminal como consola del sistema.

Escriviu 1 i premeu Intro per utilitzar aquest terminal com a consola del sistema.

Digite um 1 e pressione Enter para utilizar este terminal como console do sistema.

Starting errdemon  
Starting Diagnostics

DIAGNOSTIC OPERATING INSTRUCTIONS VERSION 5.3.0.40

LICENSED MATERIAL and LICENSED INTERNAL CODE - PROPERTY OF IBM  
(C) COPYRIGHTS BY IBM AND BY OTHERS 1982, 2004.  
ALL RIGHTS RESERVED.

These programs contain diagnostics, service aids, and tasks for the system. These procedures should be used whenever problems with the system occur which have not been corrected by any software application procedures available.

In general, the procedures will run automatically. However, sometimes you will be required to select options, inform the system when to continue, and do simple tasks.

Several keys are used to control the procedures:

- The Enter key continues the procedure or performs an action.
- The Backspace key allows keying errors to be corrected.

- The cursor keys are used to select an option.

To continue, press Enter.

#### FUNCTION SELECTION

##### 1 Diagnostic Routines

This selection will test the machine hardware. Wrap plugs and other advanced functions will not be used.

##### 2 Advanced Diagnostics Routines

This selection will test the machine hardware. Wrap plugs and other advanced functions will be used.

##### 3 Task Selection (Diagnostics, Advanced Diagnostics, Service Aids, etc.)

This selection will list the tasks supported by these procedures. Once a task is selected, a resource menu may be presented showing all resources supported by the task.

##### 4 Resource Selection

This selection will list the resources in the system that are supported by these procedures. Once a resource is selected, a task menu will be presented showing all tasks that can be run on the resource(s).

##### 99 Exit Diagnostics

NOTE: The terminal is not properly initialized. You will be prompted to initialize the terminal after selecting one of the above options.

To make a selection, type the number and press Enter. [1 ]

To make a selection, type the number and press Enter. [99]

#### SHUTDOWN

A system shutdown is about to occur.

Type 99 again and press <Enter> to continue the shutdown process.

--or--

Press the <Enter> key to abort the shutdown and return to Diagnostics.

99

....Halt completed....

---

It is also possible for the client to configure itself for a diagnostic boot:

```
{LPAR5}:/ # nimclient -o diag -a spot=SPOT_53_ML4
```

## 5.8 Automatic scripts

This appendix describes how to use automatic scripts to perform system customization after a Basic Operating System (BOS) mksysb installations.

This appendix describes the following:

- ▶ Customizing using NIM script resource
- ▶ Customizing using NIM bosinst\_data resource
- ▶ Customizing using NIM fb\_script resource

### 5.8.1 Install time customization

When installing an AIX LPAR or physical machine, with either a Basic Operating System (BOS) install or a mksysb image file, there are several ways to provide install time customization.

Customization for BOS installations (NIM bos\_inst operations using source=rte or source=spot), can be performed in one or more of the following ways:

- ▶ Using NIM script resources to provide command line and script customization - during installation and NIM cust operations
- ▶ Using NIM bosinst\_data resources to provide command line and script customization - during installation
- ▶ Using SPOT modified /usr file system files - during installation
- ▶ Using installable filesets and scripts customization - anytime after the first boot
- ▶ Using automated remote commands (**rsh/ssh**) from scripts run from another system - anytime after the first boot
- ▶ Using manually executed commands - anytime after the first boot
- ▶ Using a NIM fb\_script resource script to be executed - at the end of the first boot

Customization of mksysb installations (NIM mksysb), in addition to the BOS installation above, can also be performed in the following way:

- ▶ Modifying files on the source system before creating a NIM mksysb image file

### 5.8.2 Customization for BOS installations

We will describe how to create and use:

- ▶ NIM script resources

- ▶ NIM bosinst\_data CUSTOMIZATION\_FILE attribute script
- ▶ NIM fb\_script resource

Creating installable filesets and customization scripts is described in 4.12, “How to create bundles, BFF, and RPM packages” on page 359. To run remote commands from another node, such as the NIM master, you can use commands such as `rsh` or `ssh`.

## Using NIM script resources

A NIM script resource represents a file that is an user-created shell script and can be allocated to a NIM machine, after it is defined.

The NIM script resources are always run by NIM *after* software installation are performed during `bos_inst` operations, or during `cust` operations. Multiple script resources can be allocated to be executed, but the order in which the scripts will be run is not predictable.

The order to use the NIM script resources are:

1. Create a shell script
2. Define the NIM script resource pointing to the shell script
3. Allocate the NIM script resource for a `cust` or `bos_inst` operation, to one or more NIM clients or NIM groups
4. Execute the `cust` or `bos_inst` operation, on one or more NIM clients or NIM groups

Example 5-74 shows a simple shell script that performs some customization on a NIM client. First it acquires the `/etc/motd` file from the NIM master, then it customize the login herald, sets the timezone and specifies the resolve order for hostname and IP-address.

*Example 5-74 Example NIM script, node\_cust, for cust of bos\_inst operation*

---

```
#!/bin/ksh
# node_cust
# Synopsis.....%M%
# Author.....The Dude!
# Created.....2006
# Version.....%Z% %M% %I% (%E% %U%) %Q%
# Description....N/A
# Input.....N/A
# Output.....Perform Node Customization.
# Algorithm.....N/A
#-----
#
```

```

# Redirect STDOUT and STDERR to a logfile
#
exec 1>/tmp/${0##*/}.log
exec 2>&1
#
# Acquiring the /etc/motd file from the NIM master...
#
if [[ "$NIM_SHELL" = "shell" ]];then
    if [[ "$NIM_MASTER_HOSTNAME" != "" ]];then
        ping -c 1 -w 1 $NIM_MASTER_HOSTNAME >/dev/null 2>&1
        if [[ $? -eq 0 ]];then
            cp /etc/motd /etc/motd~$(date +%j)
            rcp $NIM_MASTER_HOSTNAME:/etc/motd /etc
        fi
    fi
fi
#
# Customize the login herald, for telnet login screens etc...
#
/usr/bin/chsec -f /etc/security/login.cfg -s default -a 'herald=\n\r\n\rThis computer
is the property of Our Company.\n\rUnauthorized access is prohibited. If you are not
authorized \r\nto have access to this computer, leave now.\r\nAll sessions on this
computer can and may be monitored.\n\r \r\nlogin: '
#
# Timezone
# Universal Time Coordinated
# + one hour during winter or
# + two hours during summertime
# Changing the last sunday in March (3) and October (10)
#
/usr/bin/chtz 'UTC-1UTC-2,M3.5.0/02:00:00,M10.5.0/03:00:00'
#
# Specify resolv order for hostname and IP-address...
#
if ! grep NSORDER /etc/environment; then
    echo "NSORDER=local,bind" >> /etc/environment
fi

```

---

We check the script for syntax errors by running it without execution. The script name in our examples is `node_cust`:

```
ksh -xvn node_cust
```

When this step has completed successfully, you should run the script manually on a test system to verify that it works as desired.

After we are convinced the script works properly, it can be defined as a NIM script resource as shown in Example 5-75. In this case we use the `/export/scripts` directory.

*Example 5-75 Creating a NIM script resource*

---

```
root@master:/: nim -o define -t script -a server=master
-a location=/export/scripts/node_cust node_cust
```

```
root@master:/: lsnim -l node_cust
node_cust:
  class      = resources
  type       = script
  Rstate     = ready for use
  prev_state = unavailable for use
  location   = /export/scripts/node_cust
  alloc_count = 0
  server     = master
```

---

Now when the NIM script has been defined, you can use the `nim` command to view the content of the script file itself:

```
nim -o showres node_cust
```

The `node_cust` NIM script resource is now ready for use, and can be allocated and executed during `cust` or `bos_inst` operations. You can allocate the script to a NIM standalone machine or NIM group as shown in Example 5-76. In this case the NIM machine is `LPAR55`.

*Example 5-76 Allocate a NIM script to a NIM standalone machine or NIM group*

---

```
root@master:/: nim -o allocate -a script=node_cust LPAR55
```

---

To execute the script on a running NIM client, use the `cust` operation as shown in Example 5-77.

*Example 5-77 Run a NIM script on a running NIM client or NIM group*

---

```
root@master:/: nim -o cust LPAR55
```

```
+-----+
|               Initiating "cust" Operation               |
+-----+
Allocating resources ...

Initiating the cust operation on machine 1 of 1: LPAR55 ...
```

```
+-----+
          "cust" Operation Summary
+-----+
Target          Result
-----
LPAR55         INITIATED
```

Note: Use the `lsnim` command to monitor progress of "INITIATED" targets by viewing their NIM database definition.

---

Each NIM script will be run with the name "script" on the NIM client, after being copied to a separate and temporary directory in the /tmp file system. Thus our debug output redirection, in the beginning of the script, will create a log file in the /tmp directory named "script.log".

```
exec 1>/tmp/${0##*/}.log
exec 2>&1
```

To monitor the progress of the NIM `cust` operation, use the `lsnim` command:

```
lsnim -l <NIM machine name>
```

In our case, the command is:

```
lsnim -l LPAR55
```

When the installation (`bos_inst`) or customization (`cust`) are finished, the actions specified in the NIM script customization script have been performed. In the even of problems, check the output of the NIM script execution as shown in Example 5-78.

*Example 5-78 Debug output of NIM script execution during cust operation*

---

```
+ [[ shell = shell ]]
+ [[ nimmast !=  ]]
+ ping -c 1 -w 1 nimmast 1> /dev/null 2>& 1
+ [[ 0 -eq 0 ]]
+ date +%j
+ cp /etc/motd /etc/motd~172
+ rcp nimmast:/etc/motd /etc
+ /usr/bin/chsec -f /etc/security/login.cfg -s default -a herald=\n\r\n\rThis co
mputer is the property of Our Company.\n\rUnauthorized access is prohibited. If
you are not authorized \r\nto have access to this computer, leave now.\r\nAll se
ssions on this computer can and may be monitored.\n\r \r\nlogin:
+ /usr/bin/chtz UTC-1UTC-2,M3.5.0/02:00:00,M10.5.0/03:00:00
+ grep NSORDER /etc/environment
+ echo NSORDER=local,bind >> /etc/environment
```

---



**Note:** The script resources must not point to files that reside in the /export/nim/scripts directory. This directory is used for the nim\_script resource that is managed by NIM. NFS restrictions prevent defining multiple resources in the same location.

## Using NIM bosinst\_data resources

When using a NIM bosinst\_data resource during a bos\_inst operation on a NIM client, you can specify the path to a script for the CUSTOMIZATION\_FILE attribute, in the file pointed to by the bosinst\_data resource. This script is started at the end of the main installation program, by the /usr/lpp/bosinst/bi\_main program. For spot installations it will occur after approximately 90 % of the total installation process, and for rte installations after approximately 95 %.

**Note:** This method has been available to use with **mksysb** since the early 1990s with AIX 3.1. With NIM you can use the NIM script resources instead, which is usually more efficient. However, tape, DVD or similar **mksysb** media can still make use of this method.

Example 5-79 is a simple shell script that performs some customization on a NIM client, in this case overwrites the /etc/motd (message of the day) file.

*Example 5-79 Example CUSTOMIZATION\_FILE script, node\_bosinst*

```
#!/bin/ksh
# node_bosinst
# Synopsis.....%M%
# Author.....The Dude!
# Created.....2006
# Version.....%Z% %M% %I% (%E% %U%) %Q%
# Description....N/A
# Input.....N/A
# Output.....Perform Node Customization during bosinst.
# Algorithm.....N/A
#-----
#
# Redirect STDOUT and STDERR to a logfile
#
exec 1>/usr/local/bosinst/node_bosinst.log
exec 2>&1

cat <<EOF > /etc/motd
```

This computer is the property of Our Company.

Unauthorized access is prohibited. If you are not authorized to have access to this computer, leave now.  
All sessions on this computer can and may be monitored.  
Violations will be punished to the full extent of the LAW.

EOF

---

We check the script for syntax errors by running it without execution, the script name is `node_bosinst`:

```
ksh -xvn node_bosinst
```

When this step has completed successfully, you should run the script manually on a test system to verify that it works as desired.

After we are convinced the script works properly, it can be copied to a directory on the SPOT that will be used during installation. In our case, we create and use the `/usr/local/bosinst` directory in the SPOT for our script, since the SPOT will be mounted over the RAM-/usr file system.

The full path to the customization script on our NIM master will then be `/export/spot/spot5305/usr/local/bosinst/node_bosinst`.

**Note:** The path name specified for the `CUSTOMIZATION_FILE` attribute, must be the path as known to the installation process during installation.

Example 5-80 is a simple `bosinst_data` resource specification file for automated installation. It should not require any user input unless something goes wrong during the installation process.

*Example 5-80 Example bosinst\_data resource specification file*

---

```
control_flow:  
  CONSOLE = Default  
  INSTALL_METHOD = overwrite  
  PROMPT = no  
  EXISTING_SYSTEM_OVERWRITE = yes  
  INSTALL_X_IF_ADAPTER = no  
  RUN_STARTUP = no  
  CUSTOMIZATION_FILE = /usr/local/bosinst/node_bosinst  
  ACCEPT_LICENSES = yes  
  DESKTOP = NONE  
  INSTALL_DEVICES_AND_UPDATES = yes  
  ENABLE_64BIT_KERNEL = yes  
  CREATE_JFS2_FS = yes
```

```
ALL_DEVICES_KERNELS = no
GRAPHICS_BUNDLE = no
MOZILLA_BUNDLE = no
KERBEROS_5_BUNDLE = no
SERVER_BUNDLE = no
REMOVE_JAVA_118 = no
HARDWARE_DUMP = yes
ADD_CDE = no
ADD_GNOME = no
ADD_KDE = no
```

```
target_disk_data:
  LOCATION =
  SIZE_MB =
  HDISKNAME =
```

```
locale:
  BOSINST_LANG =
  CULTURAL_CONVENTION =
  MESSAGES =
  KEYBOARD =
```

---

To check the contents of your customized `bosinst_data` resource file, use the **bicheck** command as follows:

```
/usr/lpp/bosinst/bicheck <filename>
```

In our case:

```
/usr/lpp/bosinst/bicheck bosinst_group55
```

The **bicheck** command will verify the existence of the `control_flow`, `target_disk_data`, and `locale` stanzas as needed.

The value for each field (if given) will be confirmed to match an allowable value, or other limitations, if they exist. If a non-prompted install is specified, the existence of values for required fields will be confirmed.

The command will not stop after the first error, but continue as far as possible to indicate all problems with the `bosinst.data` file. If the return code is 1 (failed), an error message to standard error will name the stanza(s) and field(s) which have incorrect values.



Initiating the bos\_inst operation on machine 1 of 1: LPAR55 ...

```

+-----+
|                                     |
|                               "bos_inst" Operation Summary                               |
|                                     |
+-----+
| Target           | Result           |
|-----|-----|
| LPAR55          | INITIATED       |

```

Note: Use the `lsnim` command to monitor progress of "INITIATED" targets by viewing their NIM database definition.

The installation process screen will show "SPOT data" being copied as shown in Figure 5-4.

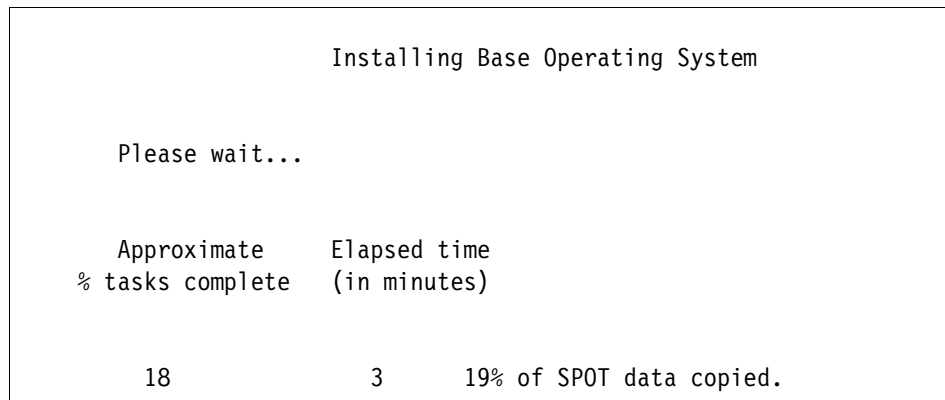


Figure 5-4 Point in time screenshot of SPOT cloning

### Using the CUSTOMIZATION\_FILE with mksysb image files

Instead of updating the SPOT file tree, copy the customization file to the system to be cloned. In our example, we put it under the `/usr/local/bosinst` directory. Perform cleanup and additional customization needed on the system prior to creating the `mksysb` image file for cloning purposes as shown in Example 5-85. For more information on cloning using the `mksysb` command, see 5.4, “Cloning” NIM clients using `mksysb`” on page 460.

*Example 5-85 Creating a mksysb with the CUSTOMIZATION\_FILE script*

```

root@master:/: nim -o define -t mksysb -a server=master -a mk_image=yes -a
source=LPAR55 -a mksysb_flags=i -a location=/export/images/mksysb_lpar55
mksysb_lpar55

```

```

+-----+
|           System Backup Image Space Information
|           (Sizes are displayed in 1024-byte blocks.)
+-----+

```

```

Required = 734753 (718 MB)    Available = 811576 (793 MB)

```

```

| Creating information file (/image.data) for rootvg..

```

```

Creating list of files to back up .
Backing up 20141 files.....
20141 of 20141 files backed up (100%)
| 0512-038 savevg: Backup Completed Successfully.

```

---

Then activate the installation of the newly created mksysb. In this case, we install it on the NIM client LPAR66 as shown in Example 5-86.

*Example 5-86 Installing allocated mksysb*

---

```

root@master:/: nim -o bos_inst -a source=mksysb LPAR66

```

```

+-----+
|           Initiating "bos_inst" Operation
+-----+
| Allocating resources ...
|
| Initiating the bos_inst operation on machine 1 of 1: LPAR66 ...
+-----+
|           "bos_inst" Operation Summary
+-----+
| Target          Result
| -----
| LPAR66          INITIATED

```

---

Note: Use the `lsnim` command to monitor progress of "INITIATED" targets by viewing their NIM database definition.

---

The installation process screen will show "mksysb data" being restored as shown in Figure 5-5.

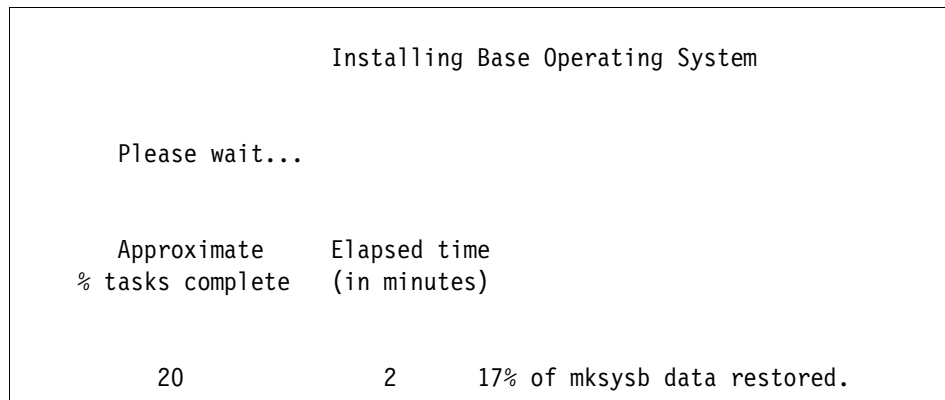


Figure 5-5 Point in time screen shot of mkysyb restore

### Using NIM fb\_script script resources

A NIM fb\_script resource represents a file that is a user-created shell script and can be allocated to a NIM machine, after it is defined. The BOS installation process creates a script in the /etc directory named firstboot, containing the content of the allocated NIM fb\_script resource. When a client is booted, the /usr/sbin/fbcheck script command checks for the existence of the /etc/firstboot file. If it exists it renames it to a script in the /etc directory prefixed with “fb\_”. This script is then executed.

The fbcheck command is run by init master process according to the entry in /etc/inittab. This allows the fb\_script to perform configuration processing on the client after all the software is installed, but before it has finished the first startup in multi-user mode.

The order to use NIM fb\_script resources are:

1. Create a shell script
2. Define the NIM fb\_script resource pointing to the shell script
3. Allocate the NIM fb\_script resource for a bos\_inst operation, to one or more NIM clients or NIM groups
4. Execute the bos\_inst operation, on one or more NIM clients or NIM groups

Example 5-87 is a simple shell script that performs some customization on a NIM client. It creates a couple of file systems.

*Example 5-87 Example NIM fb\_script, node\_firstboot, for bos\_inst operation*

```
#
# node_firstboot
```

```

# Synopsis.....%M%
# Author.....The Dude!
# Created.....2006
# Version.....%Z% %M% %I% (%E% %U%) %Q%
# Description....N/A
# Input.....N/A
# Output.....Perform First Boot Node Customization.
# Algorithm.....N/A
#-----
#
# Redirect STDOUT and STDERR to a logfile
#
exec 1>/tmp/${0##*/}.log
exec 2>&1
#
# Create /local JFS2 filesystem with 1PP size, mount it, create a directory and
# add the directory path to the root users KornShell .profile
#
crfs -v jfs2 -g rootvg -a size=1 -m /local -A yes -p rw -a logname=INLINE && mount
/local && mkdir /local/pbin && echo 'export PATH=$PATH:/local/pbin:' >>/.profile
#
# Create /usr/local JFS2 filesystem with 1PP size and mount it
#
crfs -v jfs2 -g rootvg -a size=1 -m /usr/local -A yes -p rw -a logname=INLINE &&
mount /usr/local

```

We check the script for syntax errors by running it without execution, the script name is `node_firstboot`:

```
ksh -xvn node_firstboot
```

When this step has completed successfully, you should run the script manually on a test system to verify that it works as desired.

After we are convinced the script works properly, it can be define as a NIM `fb_script` resource as shown in Example 5-88. In this case we use the `/export/scripts` directory.

---

#### *Example 5-88 Creating a NIM `fb_script` resource*

---

```
root@master:/: nim -o define -t fb_script -a server=master
-a location=/export/scripts/node_firstboot node_firstboot
```

```
root@master:/: lsnim -l node_cust
node_firstboot:
  class          = resources
```



```

type      = fb_script
Rstate    = ready for use
prev_state = unavailable for use
location  = /export/scripts/node_firstboot
alloc_count = 1
server    = master

```

---

Now when the NIM `fb_script` has been defined, you can use the `nim` command to view the content of the script file itself:

```
nim -o showres node_firstboot
```

The `node_firstboot` NIM `fb_script` resource is now ready for use, and can be allocated and used for `bos_inst` operations. Allocate the script to a NIM standalone machine or NIM group. In this case the NIM machine is LPAR55 as shown in Example 5-89.

*Example 5-89 fb\_script for a standalone or a group of machines*

---

```

root@master:/: nim -o allocate -a spot=spot5305 -a lpp_source=lpp5305 -a
bosinst_data=bosinst_group55 -a fb_script=node_firstboot LPAR55

```

---

Now we can initiate the actual installation process, in this case we specify the installation source to be the allocated NIM `lpp_source` resource `lpp5305`, using the source attribute keyword `"rte"` as shown in Example 5-90.

*Example 5-90 Activating the installation with firstboot fb\_script customization*

---

```

root@master:/: nim -o bos_inst -a source=rte -a installp_flags=agX group55

```

```

+-----+
|                                     |
|               Initiating "bos_inst" Operation               |
|                                     |
+-----+
| Allocating resources ...                                         |
|                                     |
| Initiating the bos_inst operation on machine 1 of 1: LPAR55 ... |
|                                     |
+-----+
|               "bos_inst" Operation Summary                       |
+-----+
| Target           Result                                         |
|-----|-----|
| LPAR55          INITIATED                                       |

```

Note: Use the `lsnim` command to monitor progress of "INITIATED"

targets by viewing their NIM database definition.

---

The installation process screen will show “Installing additional software”, and then software installation using the **installp** command will commence as shown in Figure 5-6.

```

                                Installing Base Operating System

                                Please wait...

                                Approximate      Elapsed time
                                % tasks complete  (in minutes)

                                18                3      Installing additional software.
```

*Figure 5-6 Point in time screen shot of lpp\_source rte installation*

When the installation is finished, the actions specified in the NIM fb\_script customization script have been performed.

## 5.9 Implementing a standard operating environment for AIX 5L V5.3 systems with NIM

This section describe practices for implementing a standard operating environment (SOE) for AIX systems with NIM. It is intended as an example based guide on how to install AIX systems in a repeatable and consistent way, with the objective to reduce the associated system management and administration overhead.

The practices described here may not be suitable for all environments, so use your best judgement and take only what you consider can be applied to your site in keeping with the overall recommendations when possible. The list of recommendations is not exhaustive, is rather a starting point for developing an SOE for AIX in a NIM environment.

We will discuss the following topics:

- ▶ Do you need an SOE for your AIX systems?
- ▶ What is an SOE for AIX systems?
- ▶ What is the purpose of the SOE image?
- ▶ Using NIM to install and manage the SOE environment.

### 5.9.1 Do you need an SOE for your AIX systems?

If you manage one or more AIX systems an SOE maybe useful. If you have a NIM master and you need to manage 5 or more AIX system images in a consistent and standard way, an SOE can offer many advantages.

### 5.9.2 What is an SOE for AIX systems?

An SOE for AIX systems is a set of best practices, procedures, documents, standards, conventions, tools and utilities with which you install and administer an AIX server. Administration tasks within the SOE are performed via NIM. All this information would be outlined in an SOE document. This document would include (but is not limited to) the following:

1. Guidelines for installing new hardware within your data centre, for example, hardware delivery, cabling for network/storage, how and where to record hardware serial numbers and model types, location of pSeries hardware environment diagrams, etc.
2. List of standard filesets included (or that should be included) for installation in the AIX SOE image. This includes AIX filesets, storage device drivers/software (for example, SDD, SDDPCM), etc.

3. AIX installation and management procedures via NIM. A standard image will be used to install all new NIM clients. This image would reside in `/export/images` on the NIM master and would be named so as to identify its purpose. For example, `soe_mkysyb - /export/images/AIX5304_64bit_SOE_V1-mkysyb`. The latest Technology Level would be applied to this image regularly.
4. NIM standards, for example, naming conventions for NIM clients, SPOTs, `lpp_source` resources and standard locations for NIM resources. Refer also to section 2.3, “Planning for your NIM environment” on page 44. For example:
  - A NIM client machine name consists of the hostname with “nim” appended to it. For example, `uataix01nim`.
  - NIM client `mksysb` backups are performed from the client via a script from root’s crontab. For example, such a script could be executed from root’s crontab twice a week.
  - NIM client `mksysb` resource names format. For example, `hostname-mkysyb.AIXV.Day`. (where *hostname* is the name of the system, *AIXV* is the AIX version and *Day* is the day of the week that the backup was created) e.g. `uataix01nim-mkysyb.530401.Mon`.
  - NIM client `mksysb` images are stored in `/export/images`.
  - NIM `lpp_source` resource names and locations will be `lpp_sourceaix VERSION`, `/export/lpp_source/lpp_sourceaix VERSION` (where *VERSION* is the AIX version) e.g. `lpp_sourceaix5304 - /export/lpp_source/lpp_sourceaix5304`.
  - NIM SPOT resource names and location will be `spotaix VERSION` and `/export/spot/spotaix VERSION` (where *VERSION* is the AIX version). For example, `spotaix5304 - /export/spot/spotaix5304`.
  - `/tftpboot` is created as a separate file system. For example, not in `/` (root). This avoids the root file system from filling in the event that many boot images are created.
  - Location of other software that may need to be installed on the system. For example, a separate file system is created which will hold other software (e.g. DB2, Oracle®, TSM, controlm, installp bundles and RPMs, etc.). Refer to section 4.12, “How to create bundles, BFF, and RPM packages” on page 359 for use with NIM.
  - Location of SOE materials. For example, a file system (`/software/soe`), would contain SOE only related software such as SOE scripts, tools, utilities, installp bundles, RPMs, etc.
  - Location of custom/firstboot NIM scripts. For example, the `/export/scripts` file system would contain firstboot and customization scripts that could be used as NIM resources during an installation.

- Location of other custom NIM resource files. For example, /export/etc would contain custom NIM resource configuration files like bosinst.data and nimclient.defs.
5. Network configuration. For example, a separate VLAN for administration exists within the data center network. This VLAN is used only for administration and/or backup tasks. For example, you create a new VLAN that will be used exclusively by NIM (and/or CSM and/or TSM) network traffic that is completely isolated from any production network. Each host has at least 2 network adapters, with at least one interface connected to the administration/management VLAN.
  6. LPAR configuration standards, for example profile names, LPAR names, recommended LPAR profile settings (CPU, memory min/des/max and adapter layout), special settings (such as “*Small Real Mode Address Region*”, “*Allowed Shared Processor Pool Utilization Authority*”, “*Enable Connection Monitoring*”, etc.). For example:
    - Profile name = normal
    - The partition name will be the same as the hostname.
    - The partition minimum and desired settings will be the same.
    - The partitions should be configured for Dynamic LPAR, for example CPU min=1, Des=1, Max=4.
    - Where possible, spread adapters over different I/O drawers and halves of the drawers, for redundancy.
    - All partitions have a minimum of two Ethernet adapters and two Fibre Channel adapters.
  7. Hostname standards and naming conventions. For example, uataix01, u=Unix, a=AIX, t=Test (p=Production, d=Development, etc.), aix is a three letter label indicating the systems purpose (for example, sap, db2, ora, etc.) and 01 (e.g. 02, 03, 04, etc.) indicates the unique number of this type of server.
  8. Volume group, logical volume and file system naming standards/conventions For example, all VG's and LV's will have vg/lv appended like datavg and usrlocal/lv. All file systems within rootvg should be of type *JFS*. All file systems within non-rootvg file systems should be of type *JFS2*.
  9. Paging space and dump space considerations and sizing. For example, true paging space sizing will depend on application requirements. But as a starting point set paging space to at least the same (or twice the) size of real memory (on small memory systems). A separate LV for system dump space should be created, for example, hd7. Use the **dumpcheck/sysdumpdev -e** commands to ensure it is large enough.

10. AIX customization, such as TCB, kernel type, syslog configuration, TZ, LANG, AIX licenses, tuning recommendations, sendmail configuration, NTP settings, /etc/motd, etc. For example:
  - TCB should be enabled
  - 64bit Kernel should be installed
  - LANG=en\_US
  - AIX licenses set to 1000 (not an issue with AIX 5L V5.3)
  - sys0 attributes, e.g. maxuproc, iostat, cpuguard and fullcore
  - Syslog configuration. For example:
 

```
mail.debug /var/log/maillog      rotate time 7d files 4 compress
*.debug;mail.none /var/log/syslog rotate time 7d files 4 compress
```
11. Disk configuration such as bootlist, mirrored rootvg, SDD, SDDPCM. System boot type will also be defined: boot from SAN or boot from SCSI disk. For example, rootvg should be mirrored when using SCSI disks or if using SDDPCM and DS8300 disk (SAN boot MPIO) then rootvg will not be mirrored.
12. Network settings. For example, hardcode speed and duplex (where appropriate), use DNS (/etc/resolv.conf, /etc/netscv.conf). The resolv.conf is contained within the image so we do not need to use a resolv\_conf NIM resource).
13. Security settings (site specific), for example, user attributes, password attributes, disabling unnecessary users and groups, disabling unnecessary services (telnet, ftp, sendmail, etc.).
14. Standard/required users and groups. Login herald.
15. Root account settings and restrictions. For example, no direct login only via console (*rlogin=false*), use of sudo, allow only system administrators that are part of the system group to *su* to root.
16. Common set of system services and tools on all systems, for example: *OpenSSH, OpenSSL, lsof, nmon, topas, sudo, pstree, tcsh, vim, logrotate, websm, nimsh, TSM client*, etc.
17. Backup strategy for all AIX systems. For example **sysbtonim** and **skelvg** scripts to perform mksysb to NIM master and backup data volume group structure, TSM client used to backup the entire system.
18. Local custom scripts for administration tasks e.g. backup, nmon data collection, etc. These scripts would typically reside in /usr/local/bin (a separate file system, /usr/local, would be created for this purpose).
19. Standard crontab entries for root user.

Additional materials - scripts - to be added on the web....

20. Standard for /etc/rc.local. Any local startup scripts should be placed in rc.local and an entry created in inittab for rlocal.
21. AIX error report and SMIT log management. For example, to retain any information in the AIX error report and the SMIT logs, you could run some scripts to redirect or copy the information to a file for later reference. The following simple scripts are examples only as shown in Example 5-91.

*Example 5-91 Sample AIX error report and SMIT log cleanup scripts*

---

```

clean_errpt.ksh:
thedata=`date +%d%b%Y`
varerr=/var/log/errpt
errnm=errpt
errpt >> $varerr/$errnm.$thedata
errpt -a >> $varerr/$errnm.$thedata
echo "Written $varerr/$errnm.$thedata"
errclear 0
echo "Cleared Error Report."

clean_smit.ksh:
thedata=`date +%d%b%Y`
varsmi=/var/log/smit
cp -p /smit.log $varsmi/smit.log.$thedata
cp -p /smit.script $varsmi/smit.script.$thedata
cp -p /smit.transaction $varsmi/smit.transaction.$thedata
echo "Written $varsmi/smit.log.$thedata"
echo "Written $varsmi/smit.script.$thedata"
echo "Written $varsmi/smit.transaction.$thedata"
>/smit.log
>/smit.script
>/smit.transaction
echo "Cleared SMIT log."

# /var/log/errpt/clean_errpt.ksh
Written /var/log/errpt/errpt.27Jun2006
Cleared Error Report.

# /var/log/smit/clean_smit.ksh
Written /var/log/smit/smit.log.27Jun2006
Written /var/log/smit/smit.script.27Jun2006
Written /var/log/smit/smit.transaction.27Jun2006
Cleared SMIT log.

```

---

22. Setup for AIX systems within a DMZ. For example special network tuning options required to secure AIX network connections. If the system is within a DMZ then there may be VLAN restrictions imposed upon the system that may impact it's ability to be administered via NIM. Refer to section 5.3, "NIMSH, OpenSSL and firewall considerations" on page 433. If you have a requirement

to only administer a DMZ system infrequently then you can consider using a 'floating network adapter' for installs and temporary administration work. For example, assuming the system is an LPAR, you could configure and cable a network adapter on the managed system that is connected to your administration VLAN. When you need to perform a task on the DMZ LPAR you can use DLPAR to assign the adapter, configure it and then perform the task. When the task is finished you could then unconfigure it and unassign it from the DMZ system. Another consideration is how to create a mksysb backup of the client to the NIM master from the DMZ. One way is via scp. You can write a small script to perform a mksysb to a local file system and then use scp (secure copy) to copy the image to the NIM master. Sample script in Additional Materials - **sysbtofs** script.

23. Steps to verify and handover an AIX system. For example, SOE verification and system peer review/QA would be performed prior to application installation. Custom scripts can be used to perform this task, which will be discussed later.
24. Documenting the system configuration. For example, a system support pack would be created that outlined how the system was configured and why.
25. Administration tasks. Where possible all systems should be administered from the NIM master. Tasks such as software maintenance can be performed with NIM quickly and efficiently (Refer to section 5.7, "System maintenance for NIM clients" on page 488). Also commands such as **dsh/dcp** (CSM tools) also make administration easier as they will allow you to run commands on many systems at once. CSM is capable of much more in terms of administering your systems, for details refer to chapter 6.1, "NIM and CSM" on page 570. It is possible to install the csm.dsh fileset and run the **dsh/dcp** commands but it is not supported. If you do not have CSM within in your environment but would like a dsh like command, one alternative is pdsh (Public Domain Korn Shell):

<http://www.llnl.gov/linux/pdsh/pdsh.html>

### **What is the purpose of the SOE image?**

In order to maintain a consistent image across all your AIX systems an SOE mksysb image is created. This mksysb image contains the following:

1. A standard set of AIX filesets such as bos.net\*, devices.fcp\* and much more.
2. A standard set of AIX tools and utilities. These may be open source tools or those provided by IBM such as topas, lparstat, mpstat, nmon, lsof and others.
3. Preferred kernel and device support such as 64 bit, jfs file systems for rootvg, etc.
4. Standard user accounts, for example, for system administrators.
5. Scripts for system administration.



## 6. Backup tools, for example, TSM client.

To keep the SOE image as clean and as small as possible, rootvg only contains AIX operating system related software and tools. Applications, databases and other software are never installed into rootvg. A separate volume group is created for any additional (non-standard) software outside of the SOE. This will keep rootvg and the mksysb small and helps reduce the time required to install/restore the image and manage the root volume group.

### **How is an SOE image created?**

The image is created by a base install of AIX which is then customized and a mksysb image created. If you wish to create an SOE image you should first install AIX 5L V5.3 (via NIM rte install) onto a test system. When performing the install ensure that you have selected the desired configuration for your SOE image i.e. kernel type, TCB on or off, all kernel/device support yes or no, language, other AIX OS filesets. All devices on this system should be unconfigured, for example, no network settings. Once AIX has been installed you should then install all of the tools and utilities you would like to include in your SOE image, create any file systems you need (for example, /usr/local), add scripts and configuration files required, etc.

When the system is fully configured to your SOE design you should perform a mksysb of the system to a local file system. For example, you create a /mksysb file system within rootvg and perform a mksysb to it (`# mksysb -i /mksysb/soe_image_mksysb`). Now that you have a 'gold' image of your SOE you can now configure IP on this system and transfer it to your NIM master's mksysb directory (/export/images) ready for installation.

To update your SOE in the future, if you need to add/change/remove any of it's components you can use the same procedure as above but instead of installing base AIX you would install your SOE image onto an unconfigured test system, perform the updates and then create a new mksysb image as before.

An alternative way to update your SOE is by assigning a LUN as your SOE disk. This disk can be assigned to an LPAR. You install the SOE onto this disk and then perform your updates. This LUN could be permanently assigned to a small micro-partition "SOE server" using virtual I/O (VIO) devices. You would install the VIO client using your SOE image and then perform the updates.

Another way of easily deploying your SOE is via `alt_disk_install`. If you have a spare disk within your pSeries environment, either an internal SCSI disk or a small LUN on your SAN disk device (for example, ESS800, DS8300) you can use this disk to update your SOE quickly. For example, you could install your SOE image onto a system that boots from SAN disk. This SOE LUN could be

assigned temporarily to any test system that you will use for the SOE update. You then perform any updates required to your SOE image.

Once the system is up and running you could assign another LUN (Clone LUN) and use `alt_disk_install` to clone the disk. The LUN could then be unassigned from the SOE server and then assigned to the new LPAR you are attempting to install. The new LPAR can then boot from this disk and is now running the SOE image. Refer to section 5.6, “Using SAN zoning for “cloning” LPARs” on page 485.

It is important to keep your `lpp_source` and SPOT up-to-date and to have all the available device and kernel support within these resources. If you have several types of System p hardware (p660, p650, p690, p570, p595, etc.) within your environment, on which you intend to install your SOE, it is necessary to ensure that all required device (and other) filesets be in the `lpp_source` for the SOE to be installed on many types of pSeries hardware. Refer to section 5.5, “Using NIM to migrate systems to new hardware” on page 470.

### Using NIM to install and manage the SOE environment

Once the SOE image is created it can be used to install new systems with standard NIM cloning techniques (refer to section 5.4, ““Cloning” NIM clients using `mksysb`” on page 460). However, a standard set of procedures for creating and installing the client will exist in an SOE environment. Hostnames and `mksysb` images will be named according to this standards. In larger AIX environments that need to install multiple clients at once, the `nimdefs` command will be used to pre-define all the NIM clients that need to be created. Once created, the clients are then installed at the same time via a NIM `bos_int` operation on a NIM machine group.

**Note:** An example of a `nimdef` stanza file is located in `/usr/samples/nim/client.defs`.

What follows is a short example of using the `nimdefs` command to quickly define six new NIM clients to our NIM environment. Having an SOE in place makes this process simple as we already know what the hostnames will be and what network (VLAN) to use.

The `client.defs` file begins with some default values for our environment such as the machine type, subnet mask, gateway, platform (`chrp`), kernel type (`mp`) and the machine group for the new clients.

The NIM client definitions follow next and contain the network type, cable type and a comment. We could have included just the comment as the network type

and cable type are already defined in the default stanza. We are defining six clients, uataix01nim through to uataix06nim.

The NIM master's /etc/hosts file is updated with the IP addresses of the new NIM clients:

```
10.1.1.91      uataix01nim
10.1.1.92      uataix02nim
10.1.1.93      uataix03nim
10.1.1.94      uataix04nim
10.1.1.95      uataix05nim
10.1.1.96      uataix06nim
```

We update our /export/etc/client.defs file with the new client definitions as shown in Example 5-92.

*Example 5-92 Update the /export/etc/client.defs file*

---

```
# cat /export/etc/client.defs
# set default values
default:
    machine_type = standalone
    subnet_mask  = 255.255.255.0
    gateway      = 10.1.1.1
    network_type = ent
    cable_type   = tp
    platform     = chrp
    netboot_kernel = mp
    machine_group = nimgrp1

# Define uataix01, uataix02, uataix03, uataix04, uataix05, uataix06
uataix01nim:
    network_type = ent
    cable_type = tp
    comment = "Test AIX LPAR number 1"

uataix02nim:
    network_type = ent
    cable_type = tp
    comment = "Test AIX LPAR number 2"

uataix03nim:
    network_type = ent
    cable_type = tp
    comment = "Test AIX LPAR number 3"

uataix04nim:
    network_type = ent
    cable_type = tp
```

```

        comment = "Test AIX LPAR number 4"

uataix05nim:
    network_type = ent
    cable_type = tp
    comment = "Test AIX LPAR number 5"

uataix06nim:
    network_type = ent
    cable_type = tp
    comment = "Test AIX LPAR number 6"

```

---

Creating the client.defs file can be also be streamlined by writing a small script to generate a client.defs based on a list of new NIM client names. For example, the script in Example 5-93 will create a new client.defs file in /export/etc/client.defs based on the list of NIM client names in the file /export/etc/newclients.list.

*Example 5-93 Automating the creation of the client.defs file*

---

```

# cat /export/scripts/cr_cldef
#!/usr/bin/ksh
#
# Script to create a new client.def file based on newclients.list file.
#

client=$(</export/etc/newclients.list)

if [ "$client" = "" ] ; then
    echo "Please add new clients to /export/etc/newclients.list e.g.
        uataix01nim
        uataix02nim
        etc.....
        "
    exit 1
fi

echo '
# set default values
default:
    machine_type = standalone
    subnet_mask  = 255.255.255.0
    gateway      = 10.1.1.1
    network_type = ent
    cable_type   = tp
    platform     = chrp
    netboot_kernel = mp
    machine_group = nimgrp1
' > /export/etc/client.defs

```

```

for i in $client
do
echo "
$i:
    comment = 'NIM client $i'
" >> /export/etc/client.defs
done

# cat /export/etc/newclients.list
uataix07nim
uataix08nim

# /export/scripts/cr_cldef

# cat /export/etc/client.defs
# set default values
default:
    machine_type = standalone
    subnet_mask  = 255.255.255.0
    gateway      = 10.1.1.1
    network_type = ent
    cable_type   = tp
    platform     = chrp
    netboot_kernel = mp
    machine_group = nimgrp1

uataix07nim:
    comment = 'NIM client uataix07nim'

uataix08nim:
    comment = 'NIM client uataix08nim'

```

Next, we run the **nimdef** command to preview the client definition file, and also to create a script that we can run to perform the actual creation of the NIM client definitions as shown in Example 5-94.

---

*Example 5-94 Running the nimdef command*

---

```

# nimdef -p -f client.defs > /export/scripts/mkcldef
# cat /export/scripts/mkcldef
6 machine definitions are complete. The following machines will be
added to the NIM environment:

uataix01nim:
  hostname=uataix01nim
  gateway=10.1.1.1

```

```
subnet_mask=255.255.255.0
network_type=ent
cable_type=tp
machine_type=standalone
platform=chrp
hostaddr=10.1.1.91
netaddr=10.1.1.0
netboot_kernel=mp
groupname=nimgrp1
uataix02nim:
hostname=uataix02nim
gateway=10.1.1.1
subnet_mask=255.255.255.0
network_type=ent
cable_type=tp
machine_type=standalone
platform=chrp
hostaddr=10.1.1.92
netaddr=10.1.1.0
netboot_kernel=mp
groupname=nimgrp1
uataix03nim:
hostname=uataix03nim
gateway=10.1.1.1
subnet_mask=255.255.255.0
network_type=ent
cable_type=tp
machine_type=standalone
platform=chrp
hostaddr=10.1.1.93
netaddr=10.1.1.0
netboot_kernel=mp
groupname=nimgrp1
uataix04nim:
hostname=uataix04nim
gateway=10.1.1.1
subnet_mask=255.255.255.0
network_type=ent
cable_type=tp
machine_type=standalone
platform=chrp
hostaddr=10.1.1.94
netaddr=10.1.1.0
netboot_kernel=mp
groupname=nimgrp1
uataix05nim:
hostname=uataix05nim
gateway=10.1.1.1
subnet_mask=255.255.255.0
```

```

network_type=ent
cable_type=tp
machine_type=standalone
platform=chrp
hostaddr=10.1.1.95
netaddr=10.1.1.0
netboot_kernel=mp
groupname=nimgrp1
uataix06nim:
hostname=uataix06nim
gateway=10.1.1.1
subnet_mask=255.255.255.0
network_type=ent
cable_type=tp
machine_type=standalone
platform=chrp
hostaddr=10.1.1.96
netaddr=10.1.1.0
netboot_kernel=mp
groupname=nimgrp1

```

1 machine group will be created with new members.

```

nimgrp1:
machine_type=standalone
member=uataix01nim
member=uataix02nim
member=uataix03nim
member=uataix04nim
member=uataix05nim
member=uataix06nim

```

1 network in the NIM environment will have new machine interfaces added.

```

NET1:
network_type=ent
address=10.1.1.0
subnet=255.255.255.0
hostname=uataix01nim
hostname=uataix02nim
hostname=uataix03nim
hostname=uataix04nim
hostname=uataix05nim
hostname=uataix06nim

```

Summary

6 Machines will be added to the NIM environment.

- 1 Machine group will be created with new members.
- 1 Network will have new NIM machine interfaces added.

```
#####
#
# Commands to define new machines in the NIM environment.
#
#####
nim -o define -t standalone -a if1="find_net uataix01nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1.1 " -a
netboot_kernel=mp -a platform=chrp uataix01nim

nim -o define -t standalone -a if1="find_net uataix02nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1.1 " -a
netboot_kernel=mp -a platform=chrp uataix02nim

nim -o define -t standalone -a if1="find_net uataix03nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1.1 " -a
netboot_kernel=mp -a platform=chrp uataix03nim

nim -o define -t standalone -a if1="find_net uataix04nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1.1 " -a
netboot_kernel=mp -a platform=chrp uataix04nim

nim -o define -t standalone -a if1="find_net uataix05nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1.1 " -a
netboot_kernel=mp -a platform=chrp uataix05nim

nim -o define -t standalone -a if1="find_net uataix06nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1.1 " -a
netboot_kernel=mp -a platform=chrp uataix06nim

#####
#
# Commands to define new groups and members in the NIM environment.
#
#####
nim -o define -t mac_group -a add_member=uataix01nim -a add_member=uataix02nim
-a add_member=uataix03nim -a add_member=uataix04nim -a add_member=uataix05nim
-a add_member=uataix06nim nimgrp1
```

We can now run the `mkcldef` script to define the clients. We remove everything but the lines that contain the commands to define the clients and machine groups as shown in Example 5-95.



*Example 5-95 Defining the machines in the new environment*


---

```
#####
#
# Commands to define new machines in the NIM environment.
#
#####
nim -o define -t standalone -a if1="find_net uataix01nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1
.1 " -a netboot_kernel=mp -a platform=chrp uataix01nim

nim -o define -t standalone -a if1="find_net uataix02nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1
.1 " -a netboot_kernel=mp -a platform=chrp uataix02nim

nim -o define -t standalone -a if1="find_net uataix03nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1
.1 " -a netboot_kernel=mp -a platform=chrp uataix03nim

nim -o define -t standalone -a if1="find_net uataix04nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1
.1 " -a netboot_kernel=mp -a platform=chrp uataix04nim

nim -o define -t standalone -a if1="find_net uataix05nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1
.1 " -a netboot_kernel=mp -a platform=chrp uataix05nim

nim -o define -t standalone -a if1="find_net uataix06nim 0 ent" -a
cable_type1=tp -a net_definition="ent 255.255.255.0 10.1.1
.1 " -a netboot_kernel=mp -a platform=chrp uataix06nim

#####
#
# Commands to add new members to existing groups in the NIM environment.
#
#####
nim -o change -a add_member=uataix01nim -a add_member=uataix02nim -a
add_member=uataix03nim -a add_member=uataix04nim -a add_
member=uataix05nim -a add_member=uataix06nim nimgrp1

# ksh /export/scripts/mkcldef
```

---

Our six new NIM clients are now defined:

```
# lsnim -t standalone
uataix01nim      machines      standalone
uataix02nim      machines      standalone
uataix03nim      machines      standalone
uataix04nim      machines      standalone
```

```

uataix05nim    machines    standalone
uataix06nim    machines    standalone

```

To automate the SOE installation process, a custom bosinst.data and first boot script resource is created to support an unattended installation and automatic customization of the system.

**Note:** A template for the bosinst.data file is located in the /usr/lpp/bosinst/bosinst.template file. More information about the stanzas in this file can be found in /usr/lpp/bosinst/bosinst.template.README

A NIM bosinst\_data resource is created for the SOE bosinst.data:

```

# lsnim -t bosinst_data
soe_bosinst_data    resources    bosinst_data

# lsnim -l soe_bosinst_data
soe_bosinst_data:
  class      = resources
  type      = bosinst_data
  Rstate    = ready for use
  prev_state = unavailable for use
  location  = /export/etc/soe.bosinst.data
  alloc_count = 1
  server    = master

```

In particular, the following entries will be changed in the bosinst.data file as shown in Example 5-96.

*Example 5-96 Example SOE bosinst.data customizations*

---

```

INSTALL_METHOD = overwrite
PROMPT = no
EXISTING_SYSTEM_OVERWRITE = yes
TCB = yes
RECOVER_DEVICES = no
ACCEPT_LICENSES = yes
ALL_DEVICES_KERNELS = no
ENABLE_64BIT_KERNEL = yes
ALT_DISK_INSTALL_BUNDLE = yes

```

```

locale:
BOSINST_LANG = en_US
CULTURAL_CONVENTION = en_US
MESSAGES = en_US
KEYBOARD = en_US

```

```

target_disk_data:

```

HDISKNAME = hdisk0

---

If you need to install the SOE image onto several different LPARs on several different managed systems, you may find it more convenient to use a single NIM client definition for the mksysb install. For example, you may create a new NIM client definition named `utaixinst`, which is used only for installing the SOE on several LPARs.

**Note:** If you plan to use a single NIM client to install many systems, you should consider disabling CPU ID validation on the NIM master. Refer to sections 4.5 on page 206 and 4.6 on page 261 for more information on Managing client CPU ID validation.

A first boot script is also used to perform any additional customization required upon the clients first boot e.g. setting the hostname, configuring additional adapters, changing network settings, modifying `.profile/.kshrc` files, etc. This is generally not required as the SOE is already heavily customized for all of the above, but there may be circumstances where it is required when the SOE must change to meet a new requirement. Refer to section 5.8, “Automatic scripts” on page 514 for more information on scripting NIM automation. The BOS installation process adds the content of the `fb_script` resource to the `/etc/firstboot` file, which is run the first time that a client is booted (`fbcheck` from `inittab`). See Example 5-97 for a sample first boot script (`fbscript`) NIM resource:

*Example 5-97 Sample first boot script*

---

```
# lsnim -l soe_fbscript
soe_fbscript:
  class      = resources
  type       = fb_script
  Rstate     = ready for use
  prev_state = unavailable for use
  location   = /export/scripts/soe_fbscript.ksh
  alloc_count = 1
  server     = master
```

---

When installing the SOE onto a new LPAR, we use the following NIM resources:

- ▶ An AIX 5L V5.3 `lpp_source`: `LPP_53_ML4`.
- ▶ An AIX 5L V5.3 `SPOT` resource: `SPOT_53_ML4`.
- ▶ An SOE first boot script resource: `soe_fbscript`.
- ▶ The SOE `mksysb` image resource: `soe_mksysb`
- ▶ The SOE `bosinst_data` resource: `soe_bosinst_data`

To install the SOE image onto all clients in the *nimgrp1* machine group, we would perform the following:

```
# nim -o bos_inst -a source=mksysb -a spot=SP0T_53_ML4 -a \  
lpp_source=LPP_53_ML4 -a mksysb=soe_mksysb -a \  
bosinst_data=soe_bosinst_data -a fb_script=soe_fbscript -a \  
accept_licenses=yes -a installp_flags=-acNgX -a no_client_boot=yes \  
preserve_res=yes nimgrp1
```

Once the SOE mksysb installation is complete, there is little left for us to do (in terms of AIX installation and configuration), except check that the install completed OK and that the system conforms to our SOE design. At the end of the firstboot script we ran a script on the system after the installation. This script checks the system against our predefined SOE configuration and reports on anything that may be different to what we expect to find in our SOE image.

Before installing any applications on the system it is recommended that the build be reviewed by another technical resource to check for any oversights in the systems configuration. If everything is OK then the systems current configuration should be documented. The new AIX system is now ready for use.

In summary, using NIM and our SOE image allows us to ensure that:

- ▶ All our AIX installations are consistent and adhere to standards set out for our environment.
- ▶ That the time consuming and tedious post OS installation tasks are reduced significantly if not removed altogether.

NIM (and the SOE image) reduced the administration and management overhead associated with installing and then customizing an AIX system.

## 5.10 How to backup and re-install a NIM master

In this section, we present three methods to backup and restore a NIM master with as little work and as automatically as possible.

Keeping an up to date image of your NIM master machine or NIM master machines is one important task a NIM administrator should perform. The main reason is to be protected in case of a failure (HW or SW). Another reason is when you are using several complex NIM environments with several NIM master machines. Even if this is not the majority of the cases, this architecture becomes more and more prevalent specially in complex clustering environments.

For two of the three methods, we use a mksysb tape already created while for the third one, we use NIM.

**Note:** For detailed information on how to create and install system backups, refer to the manual “AIX 5L Version 5.3: Installation Guide and Reference”, SC23-4887-02.

### 5.10.1 Scenario 1: Backup and restore a NIM master on the same machine using a tape

As a starting point, we assume we have a mksysb tape of our NIM master. The file `bosinst.data` which is included into the mksysb image must have the “`RECOVER_DEVICES = yes`”, as shown in Example 5-98. This parameter is used to specify whether the devices will be recreated as they were on the source system. Practically, this avoids to reconfigure the network adapter.

*Example 5-98 bosinst.data file with RECOVER\_DEVICES value*

---

```
control_flow:
  CONSOLE = /dev/lft0
  INSTALL_METHOD = overwrite
  PROMPT = no
  EXISTING_SYSTEM_OVERWRITE = yes
  SWITCH_TO_PRODUCT_TAPE =
  RECOVER_DEVICES = yes
  BOSINST_DEBUG = no
  ACCEPT_LICENSES = yes
  .....

target_disk_data:
  PVID = 00554d4ab6bb3366
  PHYSICAL_LOCATION = U0.1-P2/Z1-A8
  CONNECTION = scsi0//8,0
```

```
LOCATION = 1S-08-00-8,0
SIZE_MB = 17357
HDISKNAME = hdisk0
```

locale:

```
BOSINST_LANG = en_US
CULTURAL_CONVENTION = en_US
MESSAGES = en_US
KEYBOARD = en_US
```

---

There is no particular difficulty to do this task: to initiate mksysb restore we must go into SMS mode and select the appropriate boot device (in this case, the tape).

### 5.10.2 Scenario 2: Backup and restore a NIM master on a different machine using a tape

The starting point is the same than for the Scenario 1: we assume we have a mksysb tape of our NIM master. This image contains the same bosinst.data file as shown in Example 5-98.

The target machine is different (hardware configuration) than the source machine, thus some additional work must be done after the installation itself:

- ▶ Change network definition
- ▶ Change the hostname of the new master stored into the NIM database and the related hardware address.

**Note:** If the machine we want to reinstall has not the same hardware then the one which served to create the mksysb image, we need to use CDs to boot properly.

There is no specific additional work compared to the restore of a standard mksysb . We need to follow the basic steps:

- ▶ Go to SMS mode
- ▶ Select the right device you want to boot from (either a tape or a CD) depending if the target machine has the same hardware or not than the source machine.
- ▶ Follow the menus to restore from tape

Once the machine has rebooted, two actions must be taken: change the network definition and master hostname, and its related hardware address saved into the NIM database.

To do the first step (Network definition change), run the following command:

```
/usr/sbin/mktcpip -h'cev223' -a'129.1.2.23' -m'255.255.255.0'
-i'en0' -A'no' -t'N/A'
```

where:

- cev223 is the new hostname
- 129.1.2.23 is the new IP address
- 255.255.255.0 is the new mask
- en0 is the interface name

A best practice is to check the result of the work. We suggest to run the following command:

```
netstat -in
```

Example 5-99 gives the output of the **netstat** command. This output gives also the hardware address (0002554F2693 in our case) used in the second step.

*Example 5-99 Output of netstat -in command*

---

```
{cev223}:/ #netstat -in
Name Mtu Network Address Ipkts Ierrs Opkts Oerrs Coll
en0 1500 link#2 0.2.55.4f.26.93 884 0 562 0 0
en0 1500 129.1.2 129.1.2.23 884 0 562 0 0
lo0 16896 link#1 4148 0 4149 0 0
lo0 16896 127 127.0.0.1 4148 0 4149 0 0
lo0 16896 ::1 4148 0 4149 0 0
{cev223}:/ #
```

---

To do the second step (NIM master hostname and hardware address changes) we use SMIT by running the command:

```
smitty nim
```

and select the following sub panels in order:

- ▶ Perform NIM Administration Tasks
- ▶ Manage Machines
- ▶ Change/Show Characteristics of a Machine
- ▶ Chose master

After this choice is done, the SMIT menu shown in Example 5-100 on page 550 is launched. You only need to change the HostName value and the Network Adapter Hardware Address by the appropriate values and press Enter for execution.

*Example 5-100 SMIT panel to change master hostname and hardware address*

## Change/Show Characteristics of a Machine

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

```

[TOP]                                     [Entry Fields]
  Machine Name                            [master]
* Hardware Platform Type                   [chrp]          +
* Kernel to use for Network Boot           [mp]            +
  Machine Type                             master
  Network Install Machine State             currently running
  Network Install Control State             ready for a NIM opera>
  Primary Network Install Interface
    Network Name                           NET_ENO
    Host Name                             [cev223]
    Network Adapter Hardware Address       [0002554F2693]
    Network Adapter Logical Device Name     [ent]
    Cable Type                              N/A            +
    Network Speed Setting                   []              +
[MORE...7]

```

A best practice is to check the result of the work by running the following command:

```
lsnim -l master
```

Example 5-101 gives the output of the **lsnim** command.

*Example 5-101 Output of lsnim -l mster command*

```

{cev223}:/ #lsnim -l master
master:
  class           = machines
  type            = master
  max_nimesis_threads = 20
  comments        = machine which controls the NIM environment
  platform        = chrp
  netboot_kernel  = mp
  if1            = NET_ENO cev223 0002554F2693
  cable_type1     = N/A
  Cstate          = ready for a NIM operation
  prev_state      = ready for a NIM operation
  Mstate          = currently running
  serves          = BID_NP_HD0
  serves          = LPP_53_ML3
  serves          = SPOT_53_ML3

```



```

serves                = boot
serves                = nim_script
master_port           = 1058
registration_port     = 1059
reserved              = yes
{cev223}:/ #

```

---

### 5.10.3 Scenario 3: Backup and restore a NIM master using another NIM master.

Basically, even if a mksysb image of one of NIM master is available, when restoring it using NIM, this machine becomes a simple client. In effect, several specific and important NIM scripts are used in the network boot process. One of them is

```
/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. This variable tells NIM master if the currently installed machine is going to remain (or not) a NIM client to that master after install.

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`.

In the most cases, we want this machine be part of our NIM environment after the installation. Thus, the value of `no_nim_client` must be to `no`. This is done when you start an operation on the client. Example 5-102. shows the adequate SMIT panel.

*Example 5-102 Start an operation on a NIM client*

---

```
Install the Base Operating System on Standalone Clients
```

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

```
[MORE...1]
```

```
* Installation TYPE
```

```
[Entry Fields]
```

```
mksysb
```

* SPOT	SPOT_53_ML4	
LPP_SOURCE	[]	+
MKSYSB	MK_MST_53ML4	
BOSINST_DATA to use during installation	[]	+
IMAGE_DATA to use during installation	[]	+
RESOLV_CONF to use for network configuration	[]	+
Customization SCRIPT to run after installation	[]	+
Customization FB Script to run at first reboot	[]	+
ACCEPT new license agreements?	[no]	+
<b>Remain NIM client after install?</b>	<b>[yes]</b>	+
PRESERVE NIM definitions for resources on	[yes]	+
[MORE...35]		

---

Thus, after the installation is finished, this machine has all the information needed to be reconfigured as a NIM master but this is not used.

What are the three things that are missing to have this machine back as a NIM master?

- Installation of nim.master fileset, as this fileset has been uninstalled through the script:  
     **/SPOT/usr/lpp/bos.sysmgt/nim/methods/c\_mk\_nimclient**
- NIM database restoration
- Modification of the hostname and the Hardware address

To revert this machine into a NIM master, we use a fb\_script resource to run the needed commands as shown in Example 5-103.

*Example 5-103 Script to reconfigure the machine as a NIM master*

---

```
{nimmast}:/JMB/scripts # cat nimmast.script
#!/bin/ksh
# JM BERAIL June, 15th 2006
#
# PHASE 1: Installation of nim.master fileset from the lpp_source
# included into the mksysb image
#

/usr/lib/install/sm_inst installp_cmd -a -Q -d '/AIX53ML4/LPP_53_ML4' -f '+
5.3.0.40 Network Install Manager - Master Tools
@@S:bos.sysmgt.nim.master 5.3.0.40' '-c' '-N' '-g' '-X' '-G'

#
# PHASE 2: NIM database restoration
#
```

```
#           A NIM database backup has been made before the mkysyb creation
#           and it has been included onto the mkysyb image.
#           In our case, the NIM database backup file name is
#           nimdb.backup.15jun06
#           and it is located into the /tmp directory
#
```

```
/usr/lpp/bos.sysmgmt/nim/methods/m_restore_db /tmp/nimdb.backup.15jun06
```

```
#
# PHASE 3:  Modification of the hostname related to the master.
#           Because the mkysyb restauration can occur on a different
machine
#           than the one that has been used for the mkysyb creation,
#           we need to be sure that the hostname of the master machine
stored
#           onto the NIM database is the same than the hostname of the
#           machine used for the restauration. To fix this problem,
#           we need to run the nim -o change command with the appropriate
#           parameters.
#
```

```
typeset -R12 MAC
MAC=$(lscfg -v1 $(netstat -i | grep $(uname -n) | awk '{print $1}' | sed
-e 's/en/ent/') | grep 'Network Address')
```

```
nim -o change -a if1=NET_ENO $(uname -n) $MAC" -a platform=chrp \
-a netboot_kernel=mp master
```

```
#
# PHASE 4:  Reboot the machine
#
```

```
/usr/bin/sleep 30
/usr/sbin/shutdown -Fr
```

---

**Note:** The bold values like the `lpp_source` name directory, the backup file name of the NIM database and the network class name must be adapted to your environment.

## 5.10.4 How to change the NIM master hostname

In order to change the hostname for the NIM master the following steps are presented. They are shown in more detail through the section.

- ▶ Update name resolution (for the NIM master and the NIM clients) keeping the old names.
- ▶ Change the hostname on the NIM database.
- ▶ Remove old name from name resolution.

### Scenario

For this section we are using two machines configured one as a NIM master and the other one as a NIM client. The NIM master name is changed and the change is reflected to the NIM client.

### Updating name resolution

Name resolution must be updated for the NIM master and the NIM client. The old name must be kept in order to be able to communicate to the NIM clients to update the `/etc/niminfo` files with the new name.

**Note:** If `/etc/hosts` is being used for the name resolution the new name for the NIM master on the NIM client has to be added at the end of the corresponding entry. Different order may lead the update from the NIM master to fail.

In our scenario the NIM master name is `lpar2`. Example 5-104 shows the output for the `lsnim` command. The value for `if1` shows that the hostname on the NIM database is `lpar2`.

*Example 5-104 Host name on the NIM database*

---

```
(root@lpar2):/ # lsnim -l master
master:
  class           = machines
  type            = master
  max_nimesis_threads = 20
  if_defined      = chrp.mp.ent
  comments        = machine which controls the NIM environment
  platform        = chrp
  netboot_kernel  = mp
  if1             = NET_EN1 lpar2 00096B4EAD9C
  cable_type1     = N/A
  Cstate          = ready for a NIM operation
  prev_state      = ready for a NIM operation
  Mstate          = currently running
```

```

serves          = LPP_PRUEBA
serves          = SEC_ADAPTR
serves          = boot
serves          = nim_script
master_port     = 1058
registration_port = 1059
reserved        = yes
ssl_support     = yes

```

---

The new name for the NIM master is nimmast. Example 5-105 shows how the name resolution is updated for the NIM master.

*Example 5-105 /etc/hosts on NIM master*

---

```

(root@lpar2):/ # cat /etc/hosts
.....
10.1.1.22      lpar2 nimmast
.....

```

---

In Example 5-106 is shown how the name resolution is updated for the NIM client.

*Example 5-106 /etc/hosts on NIM client*

---

```

# more /etc/hosts
.....
10.1.1.22      lpar2 nimmast
.....

```

---

Example 5-107 shows the contents of the /etc/niminfo file for the NIM client. Check that the value for NIM\_MASTER\_HOSTNAME is lpar2.

*Example 5-107 /etc/niminfo on the NIM client*

---

```

# cat /etc/niminfo
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=VLPAR3_p5
export NIM_HOSTNAME=v1par3_p5
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=lpar2
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos

```

```
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.70:v1par3_p5 10.1.1.22:lpar2 "
export NIM_MOUNTS=""
export ROUTES=" default:0:10.1.1.1 "
```

---

### Change the hostname in the NIM database

This step changes the hostname on the NIM database and reflect the change on the NIM clients configuration.

To perform the change follow these steps:

- ▶ Type the smitty nim\_ch\_if1 fastpath on the NIM master.
- ▶ Select the proper values for the smit panel. If there is an existing NIM network referring to the hostname that is being changed, choose it in order to reflect the changes.

Example 5-108 shows the panel and values used to change the hostname for the NIM master on the database.

#### *Example 5-108 Changing the hostname using smit*

---

New Host Information (Optional)

New Host Name	[nimmast]	
New Cable Type	[]	+
New Network Information		
Existing NIM Network Name	[NET_EN1]	+
-OR-		
New NIM Network Type	[]	+
New NIM Network Name	[]	
New NIM Network Address	[]	
New NIM Network Subnet Mask	[]	
New NIM Network Default Gateway	[]	

---

The output for the operation is shown in Example 5-109. It can be seen on the output that the NIM clients are updated.

#### *Example 5-109 Hostname change output.*

---

```
Switching master on NIM clients
  changed master for v1par3_p5
Finished switching master on NIM clients
```

---

Example 5-110 shows the output for the `lsnim` command on the NIM server. This output shows that `master` is the new hostname on the NIM server and that the new hostname is propagated to the NIM client which is `v1par3_p5`.

*Example 5-110* `lsnim` command output

---

```
(root@lpar2):/ # lsnim -l master
master:
  class           = machines
  type            = master
  max_nimesis_threads = 20
  if_defined      = chrp.mp.ent
  comments        = machine which controls the NIM environment
  platform        = chrp
  netboot_kernel  = mp
  if1             = NET_EN1 nimmast 00096B4EAD9C
  cable_type1     = N/A
  Cstate          = ready for a NIM operation
  prev_state      = ready for a NIM operation
  Mstate          = currently running
  serves          = LPP_PRUEBA
  serves          = SEC_ADAPTR
  serves          = boot
  serves          = nim_script
  master_port     = 1058
  registration_port = 1059
  reserved        = yes
  ssl_support     = yes
```

---

On the NIM client the `/etc/niminfo` file is updated with the new hostname as shown in Example 5-111.

*Example 5-111* `/etc/niminfo` updates

---

```
# cat /etc/niminfo
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#         and should only be updated by NIM
export NIM_NAME=VLPAR3_p5
export NIM_HOSTNAME=v1par3_p5
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=nimmast
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="shell"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
```

```
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.70:vlpar3_p5 10.1.1.22:nimmast "
export NIM_MOUNTS=""
export ROUTES=" default:0:10.1.1.1 "
```

---

## Removing old name from name resolution

At this point the old names can be removed from the name resolution.

Example 5-112 shows the content of the `/etc/hosts` file on the NIM master after removing the old name.

*Example 5-112 /etc/hosts on the NIM master*

---

```
(root@lpar2):/ # cat /etc/hosts
.....
10.1.1.22      nimmast
.....
```

---

Example 5-113 shows the content of the `/etc/hosts` file on the NIM client after removing the old name.

*Example 5-113 /etc/hosts on the NIM client*

---

```
# cat /etc/hosts
.....
10.1.1.22      nimmast
.....
```

---

## 5.10.5 How to change the NIM master IP address

**Important:** Perform a NIM DB backup before changing NIM master object attributes.

In order to change the IP address for the NIM master the following steps are presented. They are shown in more detail through the section.

- ▶ Change the IP address for the NIM master.
- ▶ Update name resolution (for the NIM master and the NIM clients).
- ▶ Update the NIM networks object.
  - Remove all client NIM objects.
  - Recover the NIM master.



- Remove old NIM networks object and rename the new one.
- Check for new NIM networks.
- Re-create the NIM clients.

### **Scenario**

For this section we are using three machines configured one as a NIM master and the other one as a NIM client. The third machine is configured as a gateway routing network traffic through different network subnet. Initially the NIM master and the NIM client are on the same subnet. In this example we are moving the NIM master to other subnet and showing how to reconfigure the NIM environment to reflect the new environment.

Figure 5-7 on page 560 shows the initial and the final stages of the environment when changing the IP address for the NIM master to a new IP address on a different subnet to the one of the NIM client.

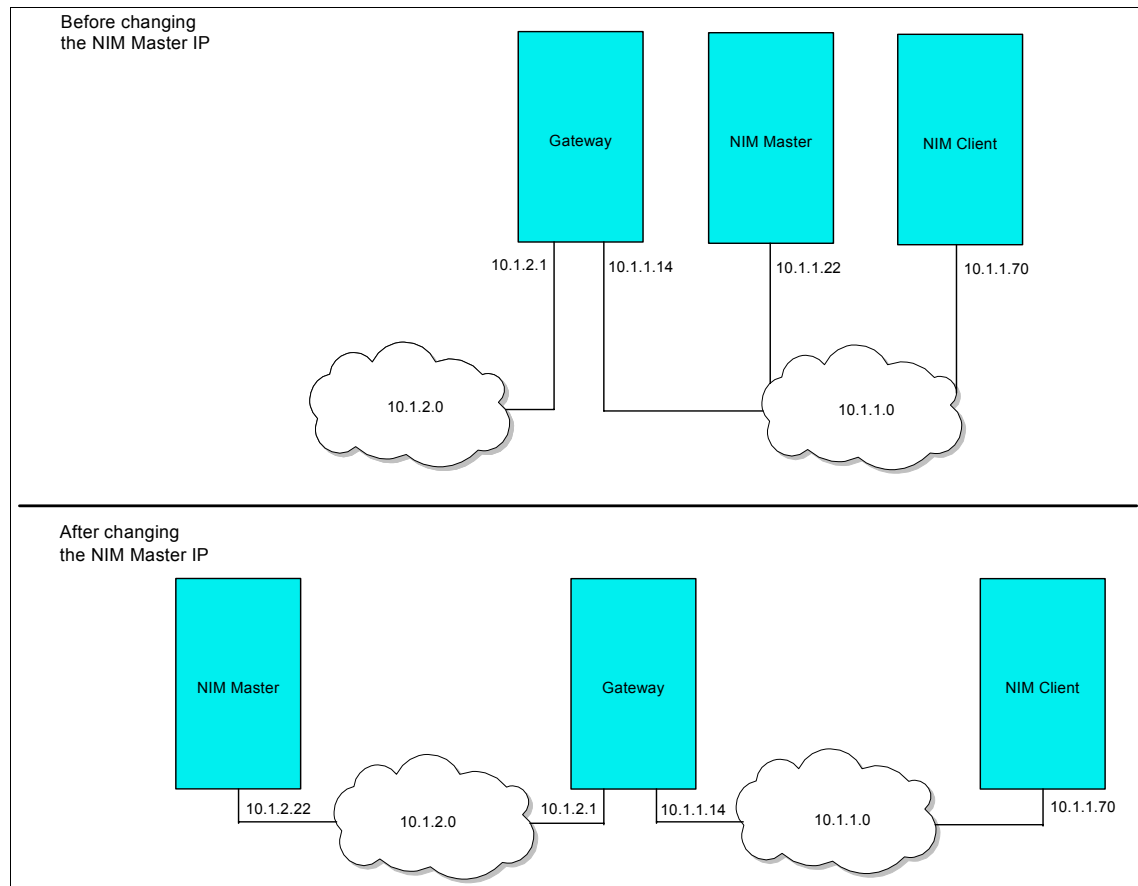


Figure 5-7 NIM master IP change scenario

### Change the IP address for the NIM master

At this stage the NIM master ip is changed. The actual IP is 10.1.1.22 using 255.255.255.0 for the network mask. We change the IP to 10.1.2.22 for the NIM master. The NIM client remains on the 10.1.1 subnet using the 10.1.1.70 IP address. For information on how to change the IP address refer to the manual “AIX 5L Version 5.3 System Management Guide: Networks and Communications Management”, SC23-5203-00.

### Updating name resolution

The name resolution must be updated for the NIM server and the NIM client. There is no need to keep the old names for this procedure.

## Updating the NIM networks object

We need to reflect the IP address change to the NIM database. To reflect the change we need to remove all the NIM clients objects and recover the NIM master using a NIM method. This method creates a new NIM network object. The old NIM network object must be removed and the new one is renamed to the old NIM network object.

### **Removing all client NIM objects**

In this example we only have one client which is named VLPAR3\_p5 as seen on Example 5-114. The master machine shown in Example 5-114 is the NIM master itself.

#### *Example 5-114 List of clients*

---

```
(root@lpar2):/ # lsnim -c machines
master      machines      master
VLPAR3_p5   machines      standalone
```

---

Now we proceed to delete the client as shown in Example 5-115.

#### *Example 5-115 Client removal*

---

```
(root@lpar2):/ # nim -o remove VLPAR3_p5
```

---

Example 5-116 shows that the client is removed.

#### *Example 5-116 List of NIM machines*

---

```
(root@lpar2):/ # lsnim -c machines
master      machines      master
```

---

In Example 5-117 we can see the NIM attributes for the master machine.

#### *Example 5-117 Master properties*

---

```
(root@lpar2):/ # lsnim -l master
master:
  class           = machines
  type            = master
  max_nimesis_threads = 20
  if_defined      = chrp.mp.ent
  comments        = machine which controls the NIM environment
  platform        = chrp
  netboot_kernel  = mp
  if1              = NET_EN1 lpar2 00096B4EAD9C
  cable_type1     = N/A
```

```

Cstate          = ready for a NIM operation
prev_state      = ready for a NIM operation
Mstate         = currently running
serves         = LPP_PRUEBA
serves         = SEC_ADAPTR
serves         = boot
serves         = nim_script
master_port     = 1058
registration_port = 1059
reserved       = yes
ssl_support     = yes

```

---

Example 5-118 shows the properties for the NIM network object named NET\_EN1.

*Example 5-118* NIM network object attributes

---

```

(root@lpar2):/ # lsnim -l NET_EN1
NET_EN1:
  class      = networks
  type       = ent
  comments   = Generated during definition of lpar2
  Nstate     = ready for use
  prev_state = ready for use
  net_addr   = 10.1.1.0
  snm        = 255.255.255.0
  routing1  = default 10.1.1.1

```

---

There is only one NIM network object for this NIM environment. The next stage is to recover the NIM master

***Recovering the NIM master***

To recover the NIM master we use a NIM method. NIM methods are located under `/usr/lpp/bos.sysmgmt/nim/methods`. The method that we use to recover the NIM master is the **nim\_master\_recover**. The network interface is given to the method as a parameter. This method creates a new NIM network object named `ent-Network1` with the new network address. Example 5-119 shows the execution of the method.

*Example 5-119* NIM master recovery

---

```

(root@lpar2):/ # /usr/lpp/bos.sysmgmt/nim/methods/nim_master_recover -i en1 -S
error retrieving nim name, defaulting to host name.
Updating master definition
  Updated master attribute platform to chrp

```

```

Updated master attribute netboot_kernel to mp
Updated master attribute if1 to ent-Network1 lpar2 00096B4EAD9C
Updated master attribute cable_type1 to N/A
Updated ent-Network1 routing1 to default 10.1.2.1
Finished updating master definition
Resetting machines
  Reset master
Finished resetting machines
Resetting NIM resources
Finished resetting NIM resources
Checking NIM resources
Keeping LPP_PRUEBA
  Keeping SEC_ADAPTR
Finished checking NIM resources
nim_master_recover Complete

```

---

Example 5-120 shows that a new NIM network object named ent-Network1 is created.

*Example 5-120 NIM network objects*

---

```

(root@lpar2):/ # lsnim -c networks
ent-Network1   networks      ent
NET_EN1       networks      ent

```

---

Example 5-121 shows the attributes for the ent-Network1 NIM network object. Information for the new IP address is added to the ent-Network1 NIM network object.

*Example 5-121 NIM network object attributes*

---

```

(root@lpar2):/ # lsnim -l ent-Network1
ent-Network1:
  class      = networks
  type       = ent
  comments   = Generated during definition of lpar2
  Nstate     = ready for use
  prev_state = ready for use
  net_addr  = 10.1.2.0
  snm        = 255.255.255.0
  routing1 = default 10.1.2.1

```

---

**Note:** If the new IP address is configured on the same subnet than the old one there is no need to create a new NIM network object. The `nim_master_recover` method does not create a new NIM network object but updates the old NIM network object.

Example 5-122 shows the attributes for the NIM master. The `if1` attribute is changed to the NIM network object generated through the `nim_master_recover` method.

*Example 5-122 NIM master attributes*

---

```
(root@lpar2):/ # lsrim -l master
master:
  class           = machines
  type            = master
  max_nimesis_threads = 20
  if_defined      = chrp.mp.ent
  comments        = machine which controls the NIM environment
  platform        = chrp
  netboot_kernel  = mp
  if1             = ent-Network1 lpar2 00096B4EAD9C
  cable_type1     = N/A
  Cstate          = ready for a NIM operation
  prev_state      = ready for a NIM operation
  Mstate          = currently running
  serves          = LPP_PRUEBA
  serves          = SEC_ADAPTR
  serves          = boot
  serves          = nim_script
  master_port     = 1058
  registration_port = 1059
  reserved        = yes
  ssl_support     = yes
```

---

***Removing old NIM networks object and rename the new***

The `ent-Network1` NIM network objects has the information for the new IP address. The NIM network object keeping information about the old IP address must be removed and the `ent-Network1` NIM network objects must be renamed to the original NIM network object.

In our scenario we have two NIM network objects:

- ▶ `NET_EN1` containing information about the old IP address and network.

- ▶ ent-Network1 generated through **nim\_master\_recover** and containing information about the new IP address and network.

Example 5-123 shows the defined NIM network objects.

*Example 5-123 NIM network objects*

---

```
(root@lpar2):/ # lsrim -c networks
ent-Network1    networks    ent
NET_EN1        networks    ent
```

---

On Example 5-124 we show the removal of the NET\_EN1 NIM network object. The **nim** **command** is used to remove the NIM network object.

*Example 5-124 NET\_EN1 removal*

---

```
(root@lpar2):/ # nim -o remove NET_EN1
```

---

The only NIM network object defined in the NIM master is the one created through the **nim\_master\_recover** method as shown in Example 5-125.

*Example 5-125 NIM network objects*

---

```
(root@lpar2):/ # lsrim -c networks
ent-Network1    networks    ent
```

---

The next step is to rename, using the **nim** command, the ent-Network1 NIM network object to the original NIM network object named NET\_EN1. This step is shown in Example 5-126

*Example 5-126 Renaming the NIM network object*

---

```
(root@lpar2):/ # nim -o change -a new_name=NET_EN1 ent-Network1
```

---

Example 5-127 shows how the name change is reflected to the NIM master attributes. The if1 attribute is changed to reflect the new name for the NIM network object.

*Example 5-127 NIM master attributes*

---

```
(root@lpar2):/ # lsrim -l master
master:
  class           = machines
  type            = master
  max_nimesis_threads = 20
  if_defined      = chrp.mp.ent
  comments        = machine which controls the NIM environment
```

```

platform           = chrp
netboot_kernel     = mp
if1                = NET_EN1 1par2 00096B4EAD9C
cable_type1       = N/A
Cstate             = ready for a NIM operation
prev_state         = ready for a NIM operation
Mstate            = currently running
serves             = LPP_PRUEBA
serves             = SEC_ADAPTR
serves             = boot
serves             = nim_script
master_port        = 1058
registration_port  = 1059
reserved           = yes
ssl_support        = yes

```

---

### ***Checking for new NIM networks***

NIM master and NIM Client are now in different subnets. The NIM master needs information regarding to how to communicate to the NIM client. The need to define a new NIM network arises.

An additional NIM network can be defined by the NIM administrator or it can be added automatically when defining the client.

In our scenario we need an additional network to allow the client which is in the 10.1.1 subnet to communicate with the NIM server which is in the 10.1.2 subnet. We show here the two ways for performing the creation of the new NIM network object.

To create a NIM network object manually proceed as follows:

- ▶ Type the smitty nim\_net fastpath on the NIM server.
- ▶ Select **Define a Network** from the SMIT menu.
- ▶ Select the network type.
- ▶ Complete the values required to define the NIM network.

Example 5-128 shows the SMIT panel and values entered to define the proper network to allow communication between the NIM server and the NIM client.

*Example 5-128 NIM network creation.*

---

```

* Network Name                               [NET_EN2]
* Network Type                                 ent
* Ethernet Type                               Standard
+

```



	* <b>Network IP Address</b>	[10.1.1.0]	
	* <b>Subnetmask</b>	[255.255.255.0]	
	Default Gateway for this Network	[10.1.1.12]	
	Other Network Type		
	+		
	Comments	[ ]	

---

Another option is to create the client from the master and fill the required values to automatically create the NIM network needed to communicate the NIM master with the NIM client.

To create a NIM client object and automatically the NIM network object proceed as follows:

- ▶ Type the smitty nim\_mkmac fastpath on the NIM master
- ▶ Enter the hostname for the NIM client.
- ▶ Select the network type.
- ▶ Select the proper values for the network.

Example 5-129 shows the SMIT panel and values entered to define the NIM client along with the NIM network object. In this example the VLPAR3\_p5 NIM client object and the NET\_EN2 network object are created.

*Example 5-129 SMIT panel to define the NIM client and NIM network object*

	* NIM Machine Name	[VLPAR3_p5]	
	* Machine Type	[standalone]	+
	* Hardware Platform Type	[chrp]	+
	Kernel to use for Network Boot	[mp]	+
	Communication Protocol used by client	[ ]	+
	Primary Network Install Interface		
	* Cable Type	bnc	+
	Network Speed Setting	[ ]	+
	Network Duplex Setting	[ ]	+
	* <b>NIM Network</b>	[NET_EN2]	
	* Network Type	ent	
	* Ethernet Type	Standard	+
	* <b>Subnetmask</b>	[255.255.255.0]	
	* <b>Default Gateway Used by Machine</b>	[10.1.1.12]	
	* Default Gateway Used by Master	[10.1.2.1]	
	* Host Name	v1par3_p5	
	Network Adapter Hardware Address	[0]	
	Network Adapter Logical Device Name	[ ]	
	IPL ROM Emulation Device	[ ]	+/
	CPU Id	[ ]	

Machine Group  
Comments

□  
□

+

---

***Re-create the NIM clients***

The NIM clients must be re-created. For information on how to re-create the NIM clients refer to “Defining NIM clients” on page 77.

***Rebuild the /etc/niminfo file on the NIM clients***

The /etc/niminfo file on the NIM clients must be rebuilt.



# Network Installation Manager (NIM) and CSM

In this chapter, we give general information on what is Cluster System Management (CSM), then we describe how CSM uses NIM to perform its nodes installation.

For more details on CSM, refer to CSM manuals following the URL:

<http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.help.csm.doc/csm.html>

## 6.1 NIM and CSM

This chapter describes how NIM operates in a CSM environment. After some general information on CSM, we explain how CSM and NIM can coexist in a cluster environment.

### 6.1.1 General overview of CSM

As described in the introduction of this book, a cluster is a set of two or more networked connected computers. To manage this set of computers, a specific system management tool is required. Formerly, this was the role of Parallel Systems Support Programs (PSSP) which has been replaced by Cluster Systems Management (CSM). The basic challenges of CSM are the monitoring, the remote access, the diagnose of problems that can arise while installing, maintaining the Operating System on the different machine, and this is the main purpose of NIM.

Figure 6-1 shows the CSM cluster environment follow by a description of its components.

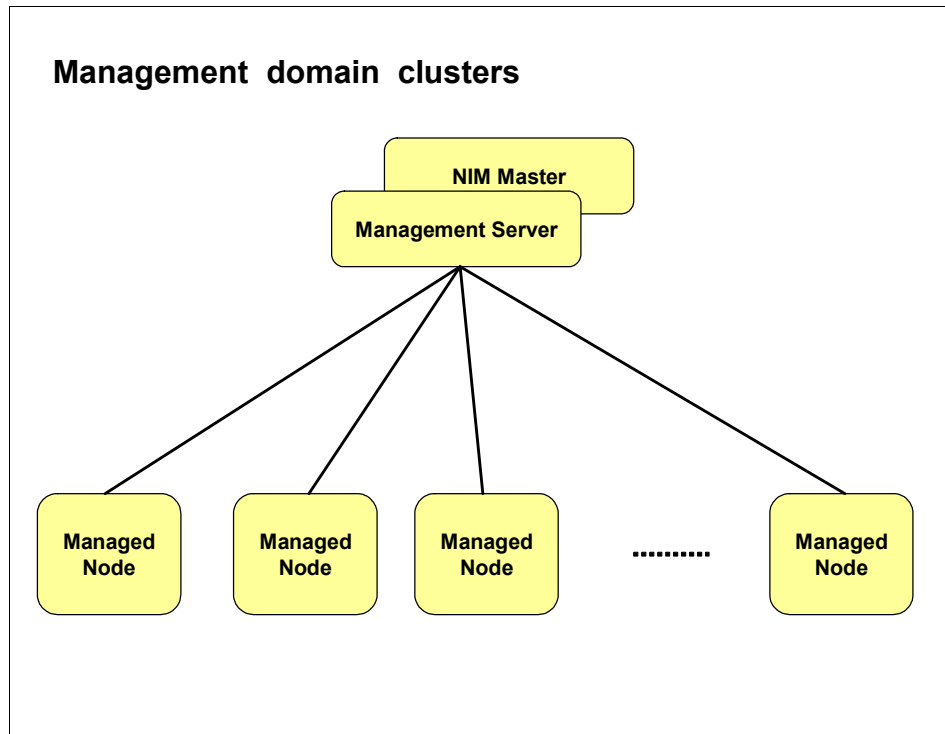


Figure 6-1 CSM cluster overview

## 6.1.2 The management server (MS)

This is a specific machine running AIX 5L and CSM server code. Although the objective of this book is not to give details on how to configure CSM, the basic steps on how an AIX 5L machine becomes a management servers are listed below:

### ***Step 1: Choosing the MS machine***

Among the AIX running machines, choosing the one that is going to become the MS is not difficult. You must know if this machine will also be the NIM master. In this case, refer to 2.3.1, “Choosing your NIM master” on page 45 for information related to the choice of a NIM master.

### ***Step 2: Run some checks on the AIX machine***

For instance, check the level of AIX, the level of the Reliable Scalable Clustering Technology (RSCT) product, the network configuration and so forth.

### ***Step 3: Install CSM package***

Basically, the CSM client is automatically installed with AIX, thus the AIX machine is cluster ready as a managed node. However, we need to install the CSM server code. To use this product, a licence is required. A 60 days trial licence is included in the AIX 5L package. After this period, a license fee is required.

### ***Step 4: Configure the CSM cluster***

Using several CSM tools, information related to the nodes are gathered to populate the CSM database.

### ***Step 5: Check the CSM cluster data***

By running some CSM commands, check if the content of the CSM database is what has been planned.

### ***Step 6: Work with CSM***

Using the CSM tools, you can install the nodes with NIM, and control and manage the AIX 5L cluster nodes with CSM.

## **The nodes**

They are the other managed machines in the cluster.

AIX may or may not have already been installed on these nodes. In case AIX has already been installed, a minimum CSM layer is necessary to join the cluster. In the other case (AIX not already installed), first AIX needs to be installed and then, the CSM layer has to be configured to have the node join the CSM cluster. The AIX installation (from a mksysb or from filesets) is not the responsibility of

CSM: it is the role of NIM. CSM is not designed to install or maintain a machine. CSM can help NIM to fill the client definitions of the machines objects once the NIM master configuration has been made and the resources created.

Another characteristic of a CSM cluster is that the management server is an AIX machine. Although, we can have a mix of AIX 5L nodes and Linux nodes. In this case, the MS can control and monitor the Linux nodes only if Linux is already installed. At the time this book was written, there are no defined procedures to use NIM to automatically install Linux on machines except for some manual operations, described in work on the NIM server as detailed in 4.2, “Using the OS\_install command” on page 138.

### 6.1.3 NIM and CSM

Unlike PSSP, CSM does not provide any tool to setup the NIM environment (from the NIM configuration to the different NIM operations such as installation or migration of a node). In CSM, there is no a script to perform these steps. PSSP has **setup\_server**.

Thus, the cluster administrator must built separately the CSM cluster and the NIM environment. For more detail on how to setup a CSM cluster, refer to the CSM documentation:

<http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.help.csm.doc/csm.html>

Basically, the different machines making up the cluster are seen as nodes for CSM and clients for NIM. These machines are the same, and the majority of their characteristics are similar in both databases: CSM and NIM.

To save time in building the NIM database and particular in defining the clients, there is one tool part of the CSM package which allows the system administrator to get nodes information from the CSM database and to fill the related client NIM information onto the NIM database. In fact, two commands are available: one for managing individual nodes **csm2nimnodes**, and the other, for machine groups **csm2nimgrps**.

Figure 6-2 on page 573 gives an overview of the relationships between the two databases: CSM registry and NIM ODM classes.

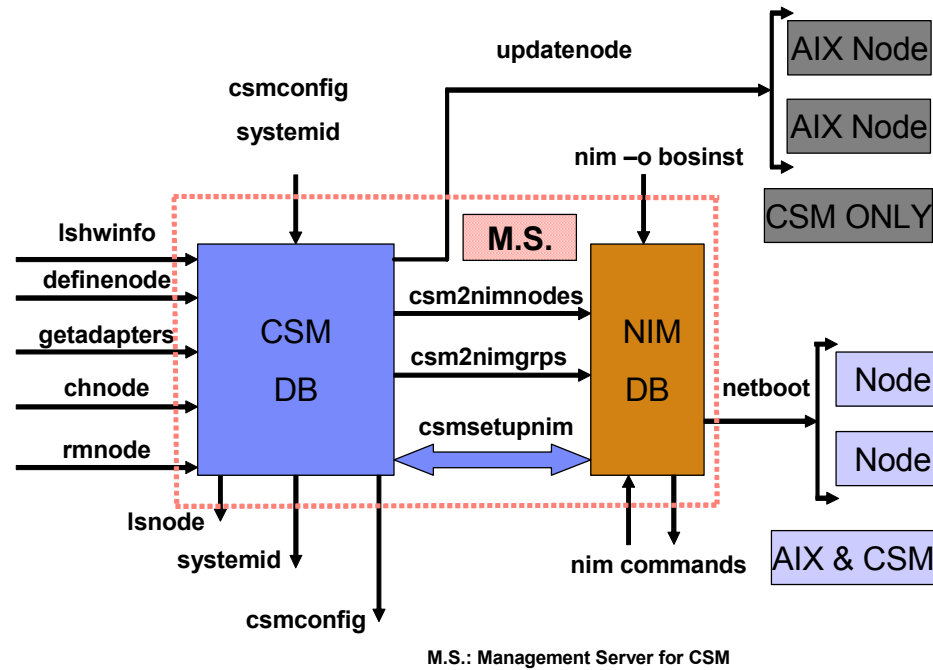


Figure 6-2 CSM and NIM coexistence

As shown in Figure 6-2, the `csm2nimnodes` and `csm2nimgrps` command link the CSM database to the NIM database while each program has its particular set of tools for node creation, modification or remove. The role of the `csmsetupnim` command is particular: it performs tasks onto the two databases to have automatically the node be part of the cluster for management purpose at the end of the installation.







# Basic NIM problem determination and tuning

This chapter describes basic NIM problem determination, solutions for network boot problems and procedures for producing debug output for NIM base operating system (BOS) installations.

In this chapter, we discuss the approach to identify an correct problems involved with NIM operations.

In this chapter, we describe the following topics:

- ▶ Troubleshooting methodologies
- ▶ Leds
- ▶ NFS
- ▶ Booting
- ▶ Log files
- ▶ Case study: NIM server performance
- ▶ Multi-threaded NIM nimesis daemon.

## 7.1 Troubleshooting methodologies

In this section we provide an approach to manage NIM troubleshooting. NIM bases its operations on two major components: the network and the operating system. Troubleshooting NIM consists in identifying the cause of the problem, and to discover if it is a NIM related problem, a network related problem or an AIX problem. After the component causing the problem has been identified it should be fixed through the proper action. Keep in mind that any troubleshooting methodology should start by examining the basics before delving into the most complex reasons behind an issue. A very simple example would be if you could not ping a NIM client you would check that the network cable was connected first rather than examining icmp packets with tcpdump. This “back to basics” approach can often speed up the problem resolution. The simplest solution to a problem is often the best approach.

There are several operations performed by NIM through all its functions. Here we introduce, as a sample, troubleshooting for a network boot problem. Network boot covers many of the operations performed by NIM. The troubleshooting procedure is guided through the flowcharts presented in this section.

### 7.1.1 Troubleshooting a network boot problem

In this scenario we work with two machines. One configured as a NIM master/Server and the other configured as a NIM client. The situation presented here is that the client has been booted to SMS menu to configure a network install boot operation in order to install the client. Remote IPL values have been entered in the client SMS configuration. On the Master side, lpp\_source and SPOT resources have been allocated to the client and a network install operation has been initiated. A problem has been detected while performing the network boot and the client is unable to boot from the server.

We follow the flowchart presented in Figure 7-1 on page 577 to solve the problem found in this scenario.

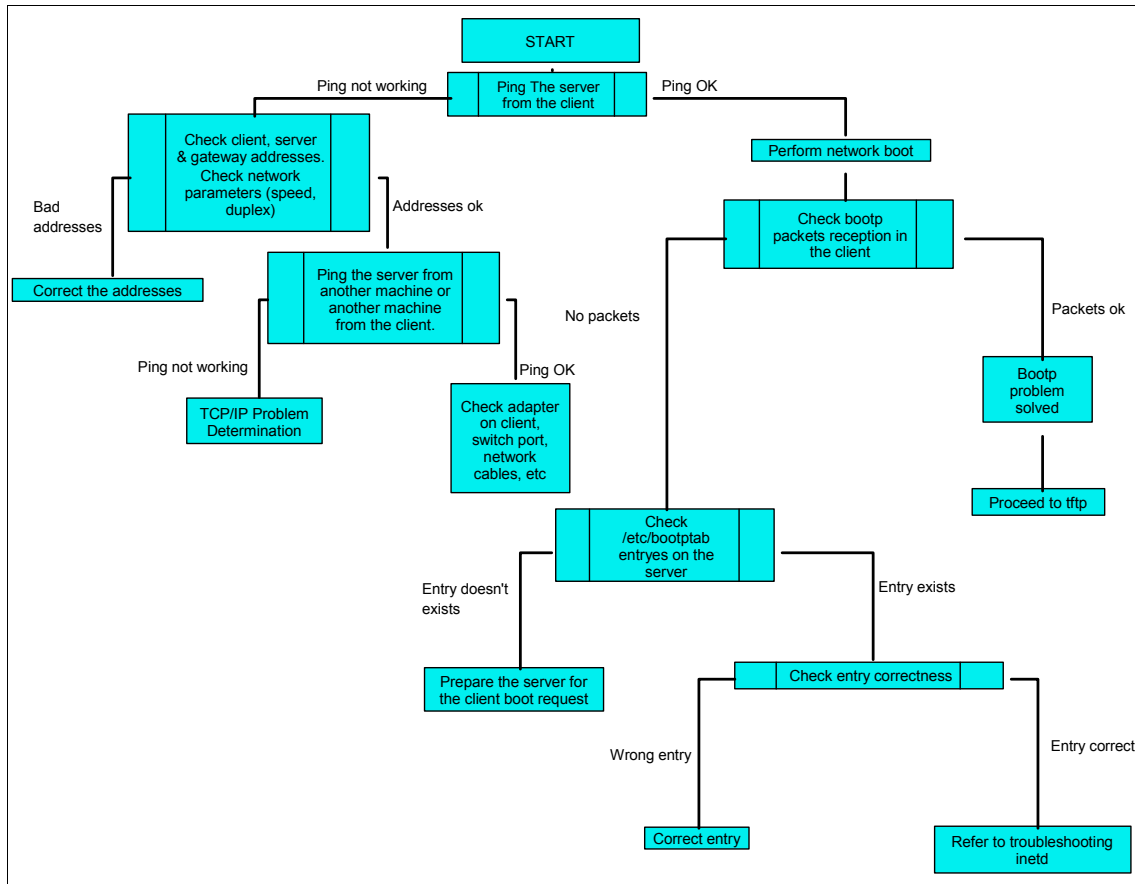


Figure 7-1 Troubleshooting a network boot problem flowchart

## Network communications

As a starting point, we try to find out if IP communication between machines can be established, (client and server). To do this, we perform a ping test from the client's SMS menu trying to ping our NIM master as shown in Example 7-1.

### Example 7-1 SMS ping menu

Version SF240\_202

SMS 1.6 (c) Copyright IBM Corp. 2000,2005 All rights reserved.

Ping Test

Interpartition Logical LAN: U9111.520.65C544E-V8-C3-T1

Speed, Duplex: auto,auto

Client IP Address: 10.1.1.70

Server IP Address: 10.1.1.1  
 Gateway IP Address: 000.000.000.000  
 Subnet Mask: 255.255.255.000  
 Spanning Tree Enabled: 1  
 Connector Type: none

### 1. Execute Ping Test

-----  
 Navigation keys:

M = return to Main Menu

ESC key = return to previous screen

X = eXit System Management Services

-----  
 Type menu item number and press Enter or select Navigation key:1

---

If the ping test fails, verify that the client, server, gateway addresses and network mask are specified correctly. Verify that the adapter configuration (network speed setting and duplex) are specified correctly.

If all settings are correct, try to ping the server from another machine. Try to ping another machine from the client to try to separate a network problem on the client from one on the server.

If the server can be pinged from another machine or other machines can not be pinged from the client, the network adapter on the boot client may be faulty or some problem with the switch port, the cables or the VLAN may exist.

### **Obtaining the boot image from the server**

At this point we have verified the communication between the NIM master/Server and the NIM client. The next stage on the network install operation is obtaining the boot image from the server. When a network boot is initiated on a client, a bootp request packet is sent from the client to the server. The server then replies with a packet to the client specifying the image to be loaded by the client. The client machine displays the number of packets sent and received for the bootp request as shown in Example 7-2.

*Example 7-2 BOOTP request success*

---

B00TP R = 1 B00TP S = 2

---

If a packet is sent from the client, but none is received, another packet will be sent.

Example 7-3 shows the output for a system that is not able to obtain a response packet from the server.

*Example 7-3 BOOTP request failure*

---

```
B00TP: B00TP request fail: 0
```

```
B00TP S = 4
```

---

If bootp packets continue to be sent but no response is received, the boot server may not be responding to the request.

Verify that the correct entry exists in the `/etc/bootptab` file on the server for the selected client.

The `/etc/bootptab` file contains entries, one for each client ready to perform a network boot operation, with the following information:

- ▶ `hostname_of_client`
- ▶ `bf=boot_file`
- ▶ `ip=client_ip_address`
- ▶ `ht=network_type`
- ▶ `sa=boot_server_address`
- ▶ `sm=client_subnet_mask`
- ▶ `ha=network_adapter_hardware_address` (required only if bootp requests are sent by broadcasting)

This entries are created at the moment that NIM configures the client machine for the operation.

Example 7-4 shows the content for the `/etc/bootptab` file in our environment.

*Example 7-4 /etc/bootptab content*

---

```
VLPAR3_p5:bf=/tftpboot/VLPAR3_p5:ip=10.1.1.70:ht=ethernet:sa=10.1.1.1:sm=255.255.255.0:
```

---

If an entry does not exist, either the NIM command used to set up the current operation failed, or the machine was reset before the boot operation could occur. Rerun the NIM `bos_inst`, `diag`, or `maint_boot` operation to prepare the server for the client boot request. If the entry exists in `/etc/bootptab`, verify that the specified data is correct. If a field contains incorrect data, the information that was used to define the machine or network in the NIM database was probably incorrect. Correct this problem by performing a NIM reset operation on the client machine,

correcting the invalid data in the client or network definition, retry the NIM operation, and reboot the client.

## Verifying inetd

If the `/etc/bootptab` file is correct, proceed to troubleshoot inetd following the flowchart in Figure 7-2.

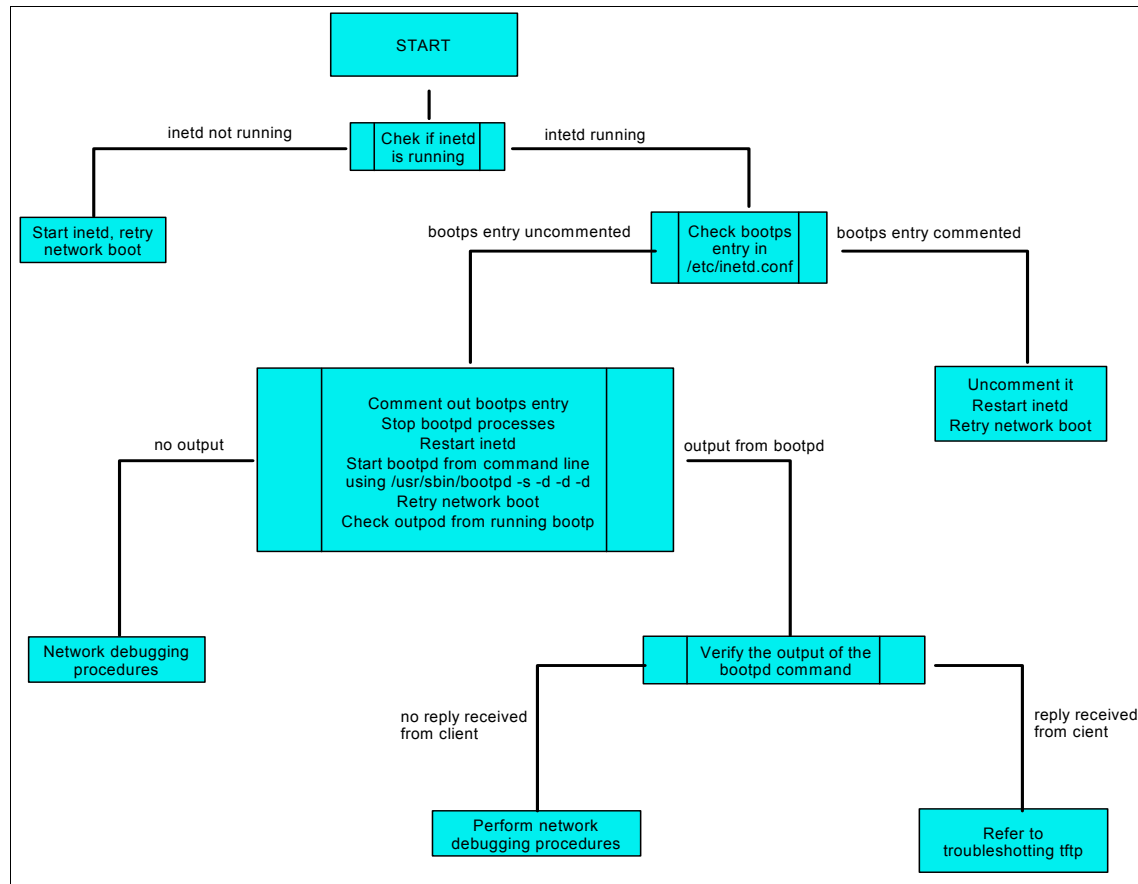


Figure 7-2 Troubleshooting inetd flowchart

Verify that the inetd daemon is running as shown in Example 7-5. If it is not running, start it and retry the network boot from the client. If the inetd daemon is running, it should automatically start the bootpd daemon when the bootp request is received at the server.

### Example 7-5 Inetd status verification

```
{nimmast}:/etc # lssrc -s inetd
```

Subsystem	Group	PID	Status
inetd	tcpip	102520	active

---

If the bootpd daemon is not started, verify that the bootps entry in the `/etc/inetd.conf` file is not commented out. If it is commented out, uncomment it and restart inetd with the **refresh -s inetd** command. Retry the network boot from the client.

The `/etc/inetd.conf` file should contain a line similar to the one shown in Example 7-6.

*Example 7-6 bootps entry*

---

```
# cat /etc/inetd.conf
.....
bootps  dgram  udp    wait   root   /usr/sbin/bootpd      bootpd /etc/bootptab
.....
```

---

If a bootp reply is still not received at the client, manually start the bootpd daemon in debug mode. To manually start the bootpd daemon in debug mode follow this steps:

- ▶ Comment out the bootps entry from the `/etc/inetd.conf` file on the server.
- ▶ Stop all running bootpd processes.
- ▶ Restart inetd using the **refresh -s inetd** command.
- ▶ Start bootpd from the command line, using the following command:

```
/usr/sbin/bootpd -s -d -d -d
```

Retry the network boot from the client. If no output is displayed from the running **bootpd** command, the client bootp request is not reaching the server.

Example 7-7 shows the output of the **bootpd** command running on the server in debug mode at the moment of receiving a bootpd request.

*Example 7-7 Bootpd output*

---

```
# /usr/sbin/bootpd -s -d -d -d
BOOTPD: bootptab mtime is Mon Jun 19 14:41:44 2006
BOOTPD: reading "/etc/bootptab"
BOOTPD: read 1 entries from "/etc/bootptab"
BOOTPD: dumped 1 entries to "/etc/bootpd.dump".
BOOTPD: bootptab mtime is Mon Jun 19 14:47:36 2006
BOOTPD: reading new "/etc/bootptab"
BOOTPD: read 1 entries from "/etc/bootptab"
BOOTPD: Received boot request.
```





.....

---

Example 7-9 shows the content of the .info file.

*Example 7-9 .info file content*

---

```
# cat VLPAR3_p5.info
#----- Network Install Manager -----
# warning - this file contains NIM configuration information
#       and should only be updated by NIM
export NIM_NAME=VLPAR3_p5
export NIM_HOSTNAME=VLPAR3_p5
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=nimmast
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export NIM_SHELL="nimsh"
export NIM_MASTERID=0051703A4C00
export NIM_LICENSE_ACCEPT=yes
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_ENV="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env"
export NIM_BOSINST_RECOVER="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env -a
hostname=VLPAR
3_p5"
export SPOT=nimmast:/AIX53ML4/SPOT_53_ML4/usr
export NIM_CUSTOM="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
location=nimmast:/export/nim
/scripts/VLPAR3_p5.script"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/installp/ppc/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 10.1.1.70:VLPAR3_p5 10.1.1.1:nimmast "
export NIM_MOUNTS=" nimmast:/AIX53ML4/LPP_53_ML4:/SPOT/usr/sys/inst.images:dir "
```

---

Figure 7-3 shows the flowchart used for troubleshooting tftp.

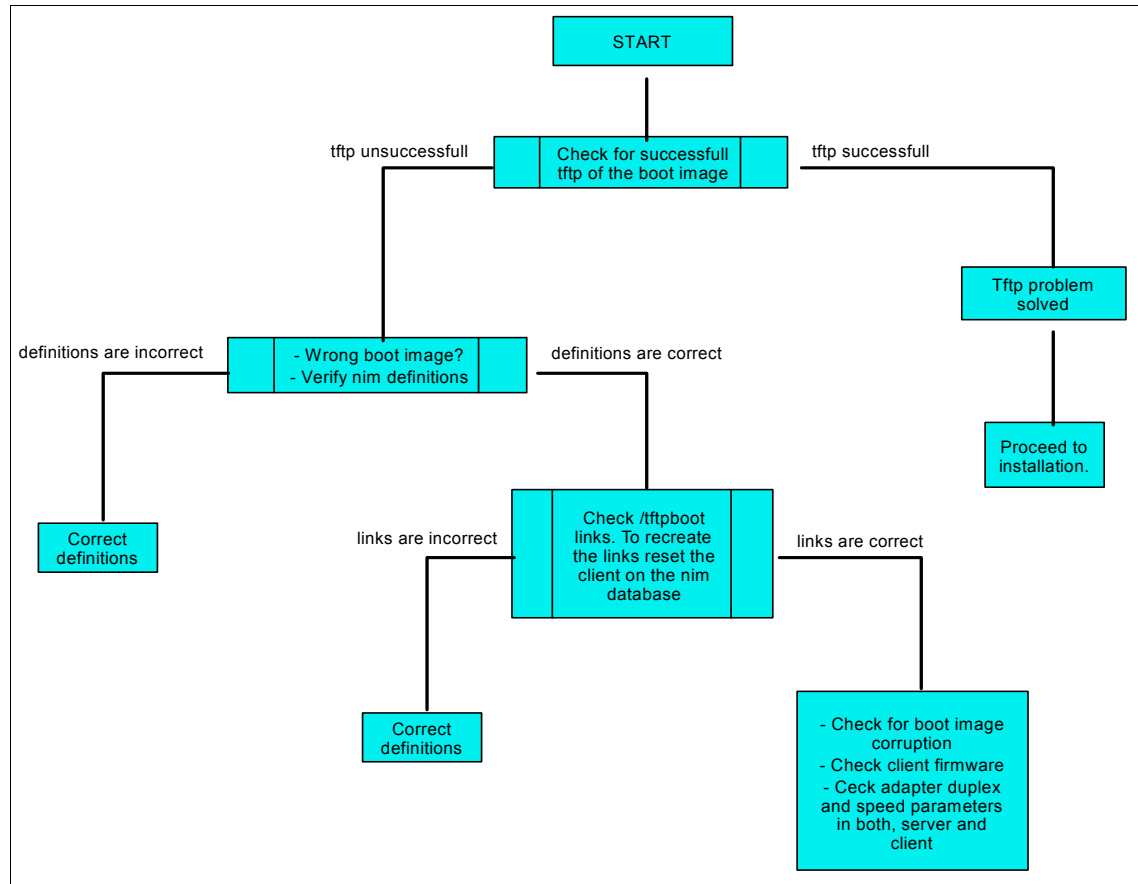


Figure 7-3 Troubleshooting tftp flowchart

The number of tftp packets transferred to the client will be displayed at the client machine as shown in Example 7-10.

*Example 7-10 tftp packet count*

---

```

FILE: /tftpboot/VLPAR3_p5
FINAL Packet Count = 23852
FINAL File Size = 12211712 bytes.
  
```

---

The boot image has been successfully retrieved at the client machine when the LED shows 299 or when the bottom third of the screen turns gray on other platform machines.

**Note:** The final file size value shown by the tftp transfer must be the same than the size of the boot image file as you can see in Example 7-8 on page 582 and Example 7-10 on page 584.

If the tftp of the boot image does not complete successfully, the client may be trying to get the wrong boot image. Verify that the client definition in the NIM database shows the correct platform and kernel type. If the data is incorrect, correct it, reset the client machine, rerun the NIM operation, and reboot the client over the network.

Verify that the /tftpboot directory on the boot server contains a link with the client name to the correct boot image. If the link does not exist, reset the client machine, rerun the NIM operation, and reboot the client over the network.

The response for the tftp server and the NIM server can also be tested from other machine through the **tftp** command. Using this command you can retrieve information about the content of the tftpboot directory on the server testing on this way the response of the tftp server.

Executing **tftp -g - <tftp\_server\_name> /tftpboot** is shown in Example 7-11.

*Example 7-11 tftp server test*

---

```
# tftp -g - nimmast /tftpboot
...SPOT_53_ML4.chspotaix5204.chspotaix5104.chSPOT_PRUEBA.chAIX53_ML04_d
ebSPOT-VI0.chrp.spotai5205.chReceived 144 Bytes in 0.0 Seconds
```

---

We get a list of the content of the directory as if we execute **echo \*** for the /tftpboot directory on the server as shown in Example 7-12.

*Example 7-12 directory listing on the server*

---

```
{nimmast}:/tftpboot # echo *
AIX53_ML04_debug.chrp.mp.ent SPOT-VI0.chrp.mp.ent
SPOT_53_ML4.chrp.mp.ent SPOT_PRUEBA.chrp.mp.ent spotaix5104.chrp.mp.ent
spotaix5204.chrp.mp.ent spotaix5205.chrp.mp.ent
```

---

If the link with the client name is pointing to the correct boot image and the tftp of the boot image does not complete successfully, the boot image may be corrupted. Re-create the boot image by performing a NIM check operation with the force flag on the SPOT. Also make sure the client has the latest version of the firmware installed. Also verify that the duplex settings for the network adapters in the server and the client, and the switch ports where they are connected are correct. If you have one of the adapters duplex incorrectly set, bootp may work but tftp may have some troubles when transferring the boot image. Tftp may also

fail due to MTU size mismatch between the NIM master network and the NIM client network.

## 7.1.2 Debugging NIM BOS installations

Due to problems in the network or in the NIM configuration, clients may fail to boot or install properly. When this happens, it may be necessary to obtain debug information in order to determine the cause of the problem.

For further information regarding debugging NIM BOS Installations please refer to the section Obtaining debug output for NIM BOS Installations on AIX 5L V5.3 Installation and Migration, SC23-4887-03.

## 7.1.3 Re-creating the SPOT from an existing directory

Rebuilding the NIM database during system recovery implies to define a SPOT. This may be a lengthy process if you are using the install media (CD). In the case that the NIM database was lost and the files for the previously defined SPOT are ok, you can use this operation to re-create the SPOT avoiding a fresh build.

After redefining the NIM master, one of the NIM methods is used to re-create the SPOT. The method is `/usr/lpp/bos.sysmgmt/nim/methods/m_mkspot`. The server name, the SPOT location and the SPOT name, must be given to the method.

Example 7-13 shows the re-creation of the SPOT which name is SPOT\_TEST, located in the /SPOT\_TEST directory in the server named master. Note the selected option `source=no` for the method. This option tells the method not to re-build the files again.

*Example 7-13 re-creation of a SPOT*

---

```
{nimmast}:/ # lsrim -l SPOT_TEST
0042-053 lsrim: there is no NIM object named "SPOT_TEST"
{nimmast}:/ # ls -l /SPOT_TEST/SPOT_TEST
total 72
-r-xr-xr-x  1 root    system      5670 Jun 14 2005 bosinst.data
-r-xr-xr-x  1 root    system      14262 Jun 08 2005 image.data
-r-xr-xr-x  1 root    system      5173 Aug 26 2005 lpp_name
drwxr-xr-x 24 bin     bin         4096 Jun 22 16:56 usr
{nimmast}:/ # /usr/lpp/bos.sysmgmt/nim/methods/m_mkspot -o -a
server=master -a location=/SPOT_TEST -a source=no SPOT_TEST
{nimmast}:/ # echo $?
0
{nimmast}:/ # lsrim -l SPOT_TEST
SPOT_TEST:
```

```

class      = resources
type       = spot
plat_defined = chrp
Rstate     = ready for use
prev_state = verification is being performed
location   = /SPOT_TEST/SPOT_TEST/usr
version    = 5
release    = 3
mod        = 0
oslevel_r  = 5300-04
alloc_count = 0
server     = master
if_supported = chrp.mp ent
Rstate_result = success

```

---

**Note:** If resources have been deleted using NIM commands, they cannot be recovered without re-creating them.

As you can see in Example 7-13 there was no SPOT in the NIM environment (shown on the first `lsnim` command execution) although the files that used to belong to an SPOT are still there (shown on the `ls` command execution). After the method execution to re-create the SPOT we check that the spot is created through the execution of the `lsnim` command.

## 7.2 Leds

If your system has an LED display, the three-digit LED shows values during NIM operations. These values are useful to understand which stage of the operation is being performed and, if it fails, in which stage fails. The list in “LED values during NIM operations” presents the values for the three-digit LED output.

### 7.2.1 LED values during NIM operations

The values on the first level of indentation are the normal values displayed during normal performance of the operations while the values on the second level are failure indicators. The values are presented in the order in which they are displayed in the three-digit LED:

- ▶ 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.
  - 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.
  - 623 The BOS installation program has encountered a fatal error.
- ▶ 624 Control passed to the BOS installation Program.
- ▶ c40 Extracting data files from media.
- ▶ c42 Extracting data files from diskette.
- ▶ c44 Initializing install data base with target disk information.
- ▶ c46 Normal install processing.
- ▶ c48 Prompting user for input.
- ▶ c50 Creating root volume group on target disks.
- ▶ c52 Changing from RAM environment to disk environment.
- ▶ c54 Installing either BOS or additional packages.
- ▶ c56 Running user-define customization.

- ▶ c58 Displaying message to turn key.
  - c41 Could not determine boot type or device.
  - c43 Could not access boot/install tape.
  - c45 Cannot configure console.
  - c47 Could not create pvid on disk.
  - c49 Could not create or form the jfs log.
  - c51 No paging devices found.
  - c53 Not enough space in /tmp to do preservation install.
  - c55 Could not remove the specified logical volume in a preservation install.
  - c57 Failure to restore BOS.
  - c59 Could not copy either device special files, device ODM, or vg information from RAM to disk.
  - c61 Failed to create boot image. Ensure enough space in /tmp.

Through the values displayed in the three-digit LED you can know the operation which failed and perform the proper actions to solve the problem.

Some of the most common errors encountered after the client machine has successfully received the boot image from the server are hangs with the LED showing 608, 611, or 613.

### **608 - tftp retrieve of client info file failure**

If a 608 hang is encountered, verify that the *ClientName.info* file exists in the /tftpboot directory. If it does not exist, retry the NIM operation to create it. If it does exist, verify that tftp access to the /tftpboot directory is not restricted in the /etc/tftpaccess.ctl file. It is also possible that the network adapter was not configured properly in the boot environment. Use debug-enabled network boot images to look for errors in the boot environment.

### **611 - Remote mount of NFS file system failure**

LED 611 hangs occur when the client machine is unable to mount a resource from the NIM master/resource server. Ensure that NFS is running on the master/resource server. Verify that the resources specified for the operation are exported properly by checking the /etc/exports and /etc/xtab files on the server. Also, confirm that the resources have permissions set correctly for reading. Debug-enabled network boot images can also be used to determine exactly which **mount** command is failing on the client. Also check the value of the nfso server portcheck option and usage of NFS reserved ports. For further information about usage of NFS reserved ports and the nfso server portcheck

option refer to [<place cross reference to chapter 5 section e Head 3 “NFS reserved ports”>](#)

### 613 - Failure setting up route tables

613 hangs usually occur because a route is incorrectly defined for a network in the NIM database. Verify that the correct gateways are specified between networks, and all gateways are functional. Use debug-enabled network boot images to determine which routes could not be defined.

## 7.2.2 Hang on LED 611 -- NFS mount problem

During bootup, the system hangs on LED 611. it may result from a shortened domain name. To show the exported lines for the lpp\_source and spot, execute:

```
# lsnim -Fl <client>
```

Make sure that every directory added to the exports list uses the fully qualified domain name (for example, client.xyz.com). A common problem is to have just "client" in the fields "HOSTS allowed root access" and "HOSTS & NETGROUPS allowed client access" instead of the fully qualified domain name.

To verify if a fully qualified domain name is needed, run:

```
# host <client>
```

The output should read '*<client>.xyz.com is <ip address>*'. Then run:

```
# host <ip address>
```

If it returns '*<client>.xyz.com is <ip address>*', then use the fully qualified domain name. The NIM master should be configured to use DNS in this case.

## 7.2.3 NIM installation hangs on LED 613

After initiating the rte installation, look at the client.info file in the /tftpboot directory to see if there is an incorrect default route set there. If there is, either remove the NIM default route or add the correct default route. (If both the client and the master are on the same network, you should not need any routes.)

To modify the clients network route in NIM:

```
# nim -Fo reset <client>
# nim -o deallocate -a subclass=all <client>
# lsnim -l <client> ==> if1 = <client_net> <client_hostname> <mac_addr>
```

Look for *<client\_net>* and make note of this name, for example:



```
if1          = NET2 LPAR7 22338000C003 ent
```

Then remove the route, run:

```
# smitty nim_rmroute
```

Select the `client_net` and select the route to remove.

To change the route, run:

```
# smitty nim_chroute
```

Select the `<client_net>` and change the route. Once you change the route, re-initiate the install.

## 7.3 NFS

When resources are allocated for use during NIM operations, they are NFS-exported to the client machines where the operations will be performed. If operations are performed simultaneously on many different clients, the `/etc/exports` and `/etc/xtab` files may become very large on the resource servers. This may cause size limits to be exceeded in the files, and it may also negatively affect NIM performance as the files are locked and modified for each resource allocation or deallocation.

The option to globally export the resources may set in environments where administrators are not concerned about who has access to the NIM resources. This prevents the repeated updates to the `/etc/exports` and `/etc/xtab` files. Setting global export of a NIM resource will make it readable by any machine in the network, not just those in the NIM environment. The resource will be globally exported as long as it is allocated to any client. When the resource is deallocated from all clients, it is unexported.

**Note:** Resources that are used exclusively by diskless and dataless clients may not be globally exported.

### 7.3.1 Exporting NIM resources globally

NIM resources can be exported globally using Web Based System Manager, SMIT or the command line.

#### Using Web Based System Manager

Follow these steps to enable global export of NIM resources using Web Based System Manager:

- ▶ From the NIM menu, select **Advanced Configuration. Export NIM Resources Globally.**
- ▶ Use the dialog to complete the task.

### Using SMIT

Follow these steps to enable global export of NIM resources using SMIT:

- ▶ Type the `smit nim_global_export` fastpath on the NIM Server.
- ▶ Select **enable** for the value of Enable/Disable Global Exporting of NIM Resources?

Example 7-14 shows the SMIT panel used when enabling global exports of NIM resources.

*Example 7-14 Enabling global exports of NIM resources using SMIT*

---

Export NIM Resources Globally

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

\* Enable/Disable Global Exporting of NIM Resources? [enable] [Entry Fields] +

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

---

To disable global exports of NIM resources using SMIT proceed as follows:

- ▶ Type the `smit nim_global_export` fastpath on the NIM Server.
- ▶ Select **disable** for the value of Enable/Disable Global Exporting of NIM Resources?

### Using the command line

To enable global exporting of NIM resources, set the attribute `global_export=yes` on the NIM master:

- ▶ `nim -o change -a global_export=yes master`

In Example 7-15 we can see the output generated by the `nim` command when enabling global exports of NIM resources.

**Example 7-15** Enabling global exports of NIM resources using the command line

---

```
{nimmast}:/ # nim -o change -a global_export=yes master
{nimmast}:/ # lsnim -l master
master:
  class                = machines
  type                 = master
  max_nimesis_threads = 20
  if_defined           = chrp.mp.ent
  comments              = machine which controls the NIM environment - code:524288
  global_export         = yes
  platform              = chrp
.....
```

---

To disable global exporting of NIM resources, remove the `global_export` attribute from the master by setting it to no:

► **`nim -o change -a global_export=no master`**

Example 7-16 shows the output from the `nim` command when disabling global exports of NIM resources.

**Example 7-16** Disabling global exports of NIM resources using the command line

---

```
{nimmast}:/ # nim -o change -a global_export=no master
{nimmast}:/ # lsnim -l master | grep glob
{nimmast}:/ #
```

---

Enabling or disabling global exports while there are resources allocated to clients could lead to situations where resources are exported with incorrect permissions. In order to change the `global_export` value, all NIM operations should be completed and resources deallocated. The `nim` command fails to change the `global_exports` value if resources are currently allocated to clients.

**Checking global export of NIM resources enablement**

The `lsnim` command can be used to check if global export of NIM resources is enabled. The `global_export` value is set to yes if global export of NIM resources is enabled or it is not set if it is disabled. Example 7-17 shows the output for a NIM master where global export of NIM resources is enabled. Refer to Example 7-16 to see how to check if global export of NIM resources is disabled.

**Example 7-17** `lsnim` output

---

```
{nimmast}:/ # lsnim -l master
master:
  class                = machines
```

```

type                = master
max_nimesis_threads = 20
if_defined          = chrp.mp.ent
comments           = machine which controls the NIM environment - code:524288
global_export      = yes
platform           = chrp
netboot_kernel     = mp

```

.....

---

Network performance is conditional to several factors which are outside of the scope of this publication. For further information regarding network tuning and NFS performance refer to AIX 5L Practical Performance Tools and Tuning Guide, SG24-6478 and AIX 5L Version 5.3 Performance Management Guide, SC23-4905-02.

## 7.4 Booting

This section explains how to troubleshoot a network boot problem.

### 7.4.1 Booting

In “Troubleshooting a network boot problem” on page 576 we go through all the stages of a network boot process. When troubleshooting a NIM client that is having problems in some of this stages it may be necessary to check the packets exchange between the machines, NIM master and NIM client. In this sections we present an overview of the network traffic between a NIM master and a NIM client.

In this scenario we have two machines, one configured as a NIM master/server and the other configured as a NIM client. Both machines are in the same subnet. The NIM server is named lpar2 while the NIM client is named v1par3\_p5. The results for the following examples are taken on the NIM master by running the command:

```
(root@lpar2):/ # tcpdump -a 'host lpar2 and v1par3_p5'
```

The `tcpdump` command with the arguments shown extracts the traffic between the hosts lpar2 and v1par3\_p5.

#### Pinging the NIM server

In this example, we ping the NIM server from the NIM client SMS menu. The ping is successful as shown in Example 7-18.

**Example 7-18** Successful ping to the NIM master

---

```
13:48:11.755361 arp who-has lpar2 tell v1par3_p5
13:48:11.755376 arp reply lpar2 is-at
13:48:11.757767 IP v1par3_p5 > lpar2: icmp 12: echo request seq 22136
13:48:11.757813 IP lpar2 > v1par3_p5: icmp 12: echo reply seq 22136
```

---

In this example the packets are reaching the NIM server and the reply is sent by the NIM server.

If the ping on the SMS menu of the NIM client reports the ping as unsuccessful We may have two situations:

- ▶ The packages are not reaching the NIM server in which case the tcpdump does not show any output.
- ▶ The replay is sent by the NIM master but do not get to the NIM client in which case we see the reply going out of the NIM server but the NIM client SMS menu shows an unsuccessful ping.

**Bootp**

The next example, Example 7-19 shows a successful exchange of packets between the NIM master and the NIM client during the bootp stage.

**Example 7-19** Successful bootp

---

```
14:35:52.162643 IP v1par3_p5.bootpc > lpar2.bootps: BOOTP/DHCP, Request from
22:33:80:00:80:03, length: 300
14:35:52.174248 IP lpar2.bootps > v1par3_p5.bootpc: BOOTP/DHCP, Reply, length: 300
```

---

Example 7-20 shows the network traffic during an unsuccessful bootp. The packets reach the NIM master but the reply is not sent by the NIM master to the NIM client. The problem to solve is the response from the bootp. Refer to “Obtaining the boot image from the server” on page 578.

**Example 7-20** Bootp failure

---

```
13:54:21.860380 arp who-has lpar2 tell v1par3_p5
13:54:21.860390 arp reply lpar2 is-at
13:54:21.870376 IP v1par3_p5.bootpc > lpar2.bootps: BOOTP/DHCP, Request from
22:33:80:00:80:03, length: 300
13:54:22.889784 IP v1par3_p5.bootpc > lpar2.bootps: BOOTP/DHCP, Request from
22:33:80:00:80:03, length: 300
13:54:24.909032 IP v1par3_p5.bootpc > lpar2.bootps: BOOTP/DHCP, Request from
22:33:80:00:80:03, length: 300
```

---

13:54:28.918683 IP v1par3\_p5.bootpc > lpar2.bootps: BOOTP/DHCP, Request from  
22:33:80:00:80:03, length: 300

---

## Tftp

In Example 7-21, we see the packet exchange between the NIM master and the NIM client. The boot image is transferred to the NIM client. There are omitted lines. This omitted lines are the packets transferred to the NIM client containing the boot image and their acknowledge.

### *Example 7-21 Successful tftp*

---

```
14:35:52.182347 IP v1par3_p5.4463 > lpar2.tftp: 28 RRQ "/tftpboot/v1par3_p5" octet
14:35:52.183812 IP lpar2.33444 > v1par3_p5.4463: udp 516
14:35:52.341956 IP v1par3_p5.4463 > lpar2.33444: udp 4
14:35:52.342017 IP lpar2.33444 > v1par3_p5.4463: udp 516
14:35:52.351681 IP v1par3_p5.4463 > lpar2.33444: udp 4
14:35:52.351731 IP lpar2.33444 > v1par3_p5.4463: udp 516
14:35:52.352386 IP v1par3_p5.4463 > lpar2.33444: udp 4
14:35:52.352423 IP lpar2.33444 > v1par3_p5.4463: udp 516
14:35:52.361754 IP v1par3_p5.4463 > lpar2.33444: udp 4
14:35:52.361800 IP lpar2.33444 > v1par3_p5.4463: udp 516
14:35:52.362371 IP v1par3_p5.4463 > lpar2.33444: udp 4
.....
.....
.....
14:38:04.485501 IP lpar2.33444 > v1par3_p5.4463: udp 516
14:38:04.494802 IP v1par3_p5.4463 > lpar2.33444: udp 4
14:38:04.494835 IP lpar2.33444 > v1par3_p5.4463: udp 4
14:38:04.495221 IP v1par3_p5.4463 > lpar2.33444: udp 4
14:38:04.504244 IP v1par3_p5.4463 > lpar2.33444: udp 4
14:38:04.504298 IP lpar2 > v1par3_p5: icmp 36: lpar2 udp port 33444 unreachable
14:38:26.504145 IP lpar2.1023 > v1par3_p5.shell: S 29956891:29956891(0) win 16384
<mss 1460>
14:38:27.503954 IP lpar2.1023 > v1par3_p5.shell: S 29956891:29956891(0) win 16384
<mss 1460>
14:38:30.504795 IP lpar2.1023 > v1par3_p5.shell: S 29956891:29956891(0) win 16384
<mss 1460>
14:38:35.368538 arp who-has lpar2 tell v1par3_p5
14:38:35.368551 arp reply lpar2 is-at
14:38:35.368820 IP v1par3_p5.filenet-tms > lpar2.tftp: 33 RRQ
"/tftpboot/v1par3_p5.info" octet
```

---

In Example 7-22, the tftp requests are reaching the NIM master but no reply is sent by the NIM master. In this case the tftp server is not responding to the request. Refer to “Verifying tftpd” on page 582.

*Example 7-22 Tftp failure*

---

```
15:02:12.180427 IP v1par3_p5.appserv-https > lpar2.tftp: 28 RRQ
"/tftpboot/v1par3_p5" octet
15:02:12.180486 IP lpar2 > v1par3_p5: icmp 36: lpar2 udp port tftp unreachable
15:02:12.190417 IP v1par3_p5.appserv-https > lpar2.tftp: 28 RRQ
"/tftpboot/v1par3_p5" octet
15:02:12.190458 IP lpar2 > v1par3_p5: icmp 36: lpar2 udp port tftp unreachable
15:02:12.200422 IP v1par3_p5.appserv-https > lpar2.tftp: 28 RRQ
"/tftpboot/v1par3_p5" octet
15:02:12.200460 IP lpar2 > v1par3_p5: icmp 36: lpar2 udp port tftp unreachable
15:02:12.210420 IP v1par3_p5.appserv-https > lpar2.tftp: 28 RRQ
"/tftpboot/v1par3_p5" octet
15:02:12.210457 IP lpar2 > v1par3_p5: icmp 36: lpar2 udp port tftp unreachable
```

---

## 7.5 Log files

This section provides information about files related to NIM operations. There are also other processes involved in NIM operations, so there are other log files that you should be checking. As mentioned in [<place cross reference to 5.e NIM communication within a firewall environment>](#) rsh, bootp, tftp and nfs are involved in NIM operations. Next you find a list of the files from where you can gather information about NIM and related processes which may help you at the moment of troubleshooting a NIM environment.

### 7.5.1 NIM related files

NIM creates and modifies a set of files and logs during any install or operations. You can use these files to find where is the problem for the installation that you are trying to perform. The format of the files is either text or alog. The text files content can be displayed with your favorite text editor or command. For the alog files you must use the **alog** command (as shown in Example 7-23) for the `/var/adm/ras/bosinstlog` file.

*Example 7-23 alog command execution*

---

```
# alog -o -f /var/adm/ras/bosinstlog | pg
Preparing target disks.
rootvg
Making boot logical volume.
```

```

hd5
Making paging logical volumes.
hd6
Making logical volumes.
.....

```

---

Table 7-1 shows the files and logs created and modified by NIM.

Table 7-1 NIM Log

File	Content	Text file	Alog file
/var/log/nimol.log	Installation output from the <b>nimol_install</b> command on the NIMOL server.	X	
/var/adm/ras/devinst.log	Log information from the BOS Installation.	X	
/var/adm/ras/emgr.log	Log information from the <b>emgr</b> command command execution.	X	
/var/adm/ras/bosinstlog	Log information from the BOS Installation.		X
/var/adm/ras/nim.alt_disk_install	Debug information from the <b>alt_disk_install</b> command execution on the client, if debug option has been selected.	X	
/var/adm/ras/alt_disk_inst.log	Output from the <b>alt_disk_install</b> command execution on the client, if verbose option has been selected.	X	
/var/adm/ras/nimsh.log	Output from the nimsh daemon,	X	
/var/adm/ras/nim.installp	Log information from the installp operation execution.	X	
/var/adm/ras/install_all_updates.log	Log information from the <b>install_all_updates</b> command execution.	X	
/var/adm/ras/nim.update	Log information from the <b>nim_update_all</b> command execution.	X	
/var/adm/ras/nim.setup	Log information from <b>nim_master_setup</b> command execution.	X	
/var/adm/ras/nim.script	Log information from a nim script execution.	X	



File	Content	Text file	Alog file
/var/adm/ras/eznim.cfg	Keeps a list of which resources are currently defined in the resource group used in the eznim environment.	X	
/var/adm/ras/nimlog	Log for console messages printed to stderr during a NIM operation.		X

## 7.5.2 RSH related files

Table 7-2 and Table 7-3 show the files related to rsh and rshd. The rshd daemon is the server which provides the function for remote command execution.

Table 7-2 rsh files

File	Purpose
\$HOME/.klogin	Specifies remote users that can use a local user account.
/usr/lpp/ssp/rcmd/bin/rsh	Link to AIX Secure /usr/bin/rsh that calls the SP Kerberos 4 rsh routine if applicable.
/usr/lpp/ssp/rcmd/bin/remsh	Link to AIX Secure /usr/bin/rsh that calls the SP Kerberos 4 rsh routine if applicable.

Table 7-3 rshd files

File	Purpose
\$HOME/.rhosts	Specifies remote users that can use a local user account on a network.
/etc/hosts.equiv	Specifies remote systems that can execute commands on the local system.
/etc/inetd.conf	Defines how the inetd daemon handles Internet service requests.
/etc/services	Defines the sockets and protocols used for Internet services.
/etc/inetd.conf	Contains the configuration information for the inetd daemon.

Information is logged also to the syslogd log file. Refer to “Enabling syslogd daemon logging” for information regarding to how to enable syslogd daemon logging.

## Enabling syslogd daemon logging

To enable the syslogd daemon logging, if it is not already enabled, you should follow these steps.

- ▶ Edit the `/etc/syslog.conf` file.
- ▶ Add a line with the message priority and destination file. Your line should look similar to the following:
 

```
*.debug                /tmp/syslog.out
```
- ▶ Create the empty log file using for example the **touch** command.
- ▶ Start the syslogd daemon, if not already running, or refresh the daemon if running using `src` commands as follow:
  - If syslogd daemon is already running:
 

```
refresh -s syslogd
```
  - If syslogd daemon is not running:
 

```
startsrc -s syslogd
```

Then you are able to monitor the output to find messages related to the failing component. Example 7-24 shows a line from the syslogd log file related to a rsh authentication failure.

*Example 7-24 syslogd log content*

---

```
.....
Jun 22 10:52:05 VLPAR3_p5 daemon:warn|warning rshd[311496]: Failed rsh
authentication from nimmast for local user root via remote user root
.....
```

---

### 7.5.3 BOOTP related files

Table 7-4 shows the files related to bootpd. This daemon Sets up the Internet Boot Protocol server. When a bootp request arrives, inetd starts the bootpd daemon. These are text files.

*Table 7-4 bootp files*

File	Purpose
<code>/etc/services</code>	Defines the sockets and protocols used for Internet services.
<code>/etc/bootptab</code>	Default configuration file.
<code>/etc/bootpd.dump</code>	Default dump file.

File	Purpose
/etc/inetd.conf	Contains the configuration information for the inetd daemon.

Bootpd also logs information through the syslogd daemon. See “RSH related files” on page 599 to find information on how to enable syslogd logging.

## 7.5.4 TFTP related files

Table 7-5 shows the files related to tftp and tftpd. The tftpd daemon is the server which provides the function for the Trivial File Transfer Protocol. These are text files.

Table 7-5 tftp and tftpd files

File	Purpose
/etc/tftpdaccess.ctl	Allows or denies access to files and directories.
/etc/inetd.conf	Contains the configuration information for the inetd daemon.
/etc/services	Defines the sockets and protocols used for Internet services.

The tftpd daemon also can be given the flag -v. With this option, the daemon, sends debugging information to the syslog daemon. See “RSH related files” on page 599 to find information on how to enable syslogd logging.

Example 7-25 shows the content of the syslogd output referring to a tftp transfer.

Example 7-25 tftpd output to syslogd

---

```

.....
Jun 22 14:05:18 nimmast daemon:info tftpd[463006]: [00000102] EZZ7044I
: 10.1.1.70 RRQ <file=Received , mode=/tftpboot, recognized
options: netascii>
Jun 22 14:05:18 nimmast daemon:info tftpd[463006]: [00000102] EZZ7029I
: Status Read request for 10.1.1.70: /tftpboot
Jun 22 14:05:18 nimmast daemon:info tftpd[463006]: [00000102] EZZ7046I
: Status Transaction completed successfully
.....

```

---

## 7.5.5 NFS related files

Table 7-6 shows the files related to nfs. These are text files.

Table 7-6 NFS files

/etc/rc.nfs	Contains the startup script for the NFS and NIS daemons.

For further information regarding nfs performance, refer to AIX 5L Practical Performance Tools and Tuning Guide, SG24-6478.

This chapter presents practical examples for a NIM environment and a POWER5 case study.

### ***NIM case study***

In the first section of this chapter we go through the performance tuning process for an AIX system that is using the Network Installation Management (NIM) software. Network Install Manager is a complex application which relies on several subsystems to provide software installation and maintenance in an AIX, or a mixed AIX/Linux environment (from AIX 5L V5.3).

NIM uses a client server model which provides clients with all the necessary resources for booting, installing, maintaining or diagnosing (AIX only) client machines. In a NIM environment, the following subsystems should be considered as candidates for performance tuning: network (TCP/IP), NFS, Virtual Memory Manager, Disk I/O and Logical Volume Manager.

### ***POWER5 case study***

This section considers specific POWER5 performance issues. We monitor the CPU performance of a POWER5-based system using a simple case scenario.

## 7.6 Case study: NIM server performance

In this case study we utilized an IBM @server pSeries model 690 server that was configured into four separate logical partitions (LPAR). Each partition included one 10/100 Ethernet adapter, one gigabit Ethernet adapter, and one internal SCSI hard disk. Each partition had two processors and 4GB of RAM. The partition configured as the NIM master (server) also has two fibre channel adapters and an additional local SCSI hard disk assigned to it.

For our first test, we had all partitions using the 10/100 Ethernet adapters, connected to a switch, with effective link parameters of 100Mbps, full duplex. The NIM server resources were allocated on the second internal SCSI hard disk. Later, we configured NIM to use the gigabit ethernet (connected to a GbitE switch), and also used the external fibre channel (FC) storage, connected via a Storage Area Network (a FC switch). A diagram of our test environment is presented in Figure 7-4.

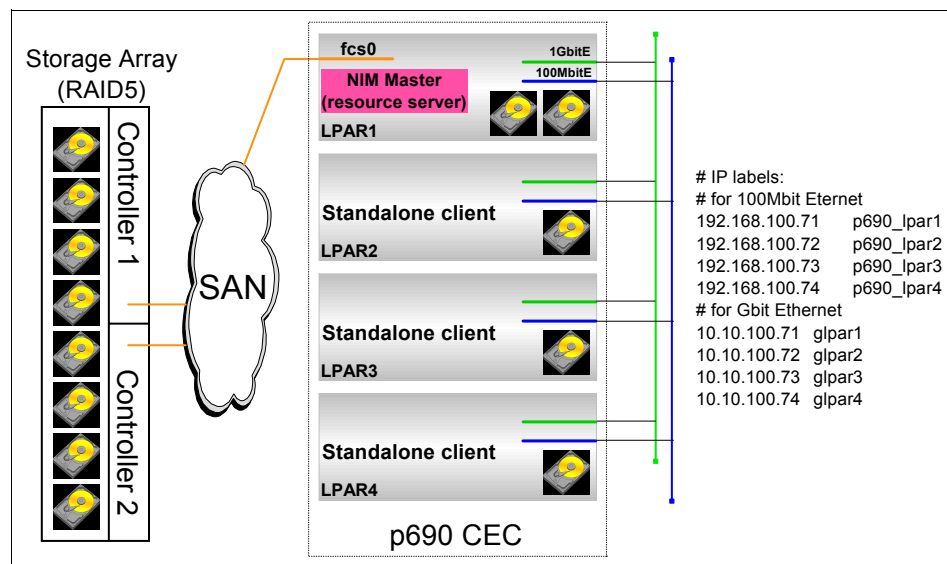


Figure 7-4 Test environment NIM diagram

### 7.6.1 Setting up the environment

We have installed the NIM master (in our case LPAR1) from AIX installation CD-ROMs with AIX 5L V5.3. For this purpose, we have assigned the CD-ROM drive available in the media drawer to LPAR1.

Once installed, we have configured the NIM master and start defining the resources to be used in our environment.

In our environment we want to install the three remaining LPARs (LPAR2, LPAR3, and LPAR4) from the NIM master. For this purpose, in the initial phase we need to define the following resources:

1. A NIM repository containing the software packages to be used for installing the clients (similar to the content of the AIX installation CD set). This type of resource is known as “lpp\_source”.
2. A repository containing the binaries (executables) and libraries to be used for executing various operations (programs) during clients’ installation, together with the kernel image used for booting the clients. This is known as “Shared Product Object Tree” (SPOT), and is in fact a directory similar to /usr file system.
3. The three clients (LPAR2, LPAR3, and LPAR4), which are defined in the NIM environment as “Standalone” machines (after installation they will boot from their own disk and will run an independent copy of the operating system).

These machines are defined using the MAC address of the network (Ethernet) adapter to be used for installation and an associated IP address.

While defining the resources mentioned before, we observed that defining the LPPSOURCE and SPOT type resources is very I/O disk resource demanding:

- In fact, creation of the LPPSOURCE consists of copying the necessary LPPs (Licensed Program Products) from the AIX installation CD-ROMs to a designated space on the disk.
- Also, creating the SPOT, is similar to an installation process, where the installation takes place on the defined disk (file system) space.

Once these resources are created, we proceed to installing the clients. Installing the clients in a NIM environment can be of three types: SPOT installation, bos rte, and mksysb. During initial client installation (of bos rte type), the following resources are used:

- The bootp server (to allow clients to boot over the network)
- The tftp server (to transfer the kernel to be loaded by the clients)
- The NFS subsystem (to run the install programs and to retrieve the necessary LPPs for installing the client)

Since the NIM software repository resides on a file system, and this file system is NFS exported to the NIM clients, the following subsystems are also involved during the installation process:

- The Virtual Memory Manager
- The TCP/IP subsystem
- The NFS subsystem

– The Logical Volume Manager

Thus, we found useful to tune these subsystems for obtaining the maximum performance for our NIM master. We started by monitoring an idle NIM master, and then gradually, tried to identify the bottlenecks during various NIM operations.

## 7.6.2 Monitoring NIM master using topas

During client installation process, to begin the performance tuning process for the system, we start by monitoring the system using **topas**.

*Example 7-26 topas output for NIM server*

---

```

Topas Monitor for host:  p690_lpar1          EVENTS/QUEUES  FILE/TTY
Fri Oct 22 17:23:42 2004 Interval: 2      Cswitch  1040  Readch    0
                                           Syscall  204  Writech   199
Kernel  6.2  |##                               Reads    0  Rawin    0
User    0.0  |                               Writes   0  Ttyout   199
Wait    0.0  |                               Forks    0  Igets    0
Idle    93.8 |#####                               Execs   0  Namei    2
                                           Runqueue 0.0  Dirblk   0
                                           Waitqueue 0.0
Network KBPS  I-Pack  O-Pack  KB-In  KB-Out
en1  11473.5  8387.0  705.0  458.8  22488.2
lo0   0.0     0.0    0.0    0.0    0.0
PAGING
Faults  0  Real,MB  4095
Disk  Busy%  KBPS    TPS  KB-Read  KB-Writ  Steals  0  % Comp  17.9
hdisk1 12.5  11136.0  90.0  22272.0  0.0    PgpsIn  0  % Noncomp 27.6
...(lines omitted)...

```

---

From the **topas** output, the only resource that is running at maximum speed is the ethernet adapter en1. A 10/100 adapter running at 100\_Full\_Duplex, has a transmit speed limit of approximately 10 Megabytes per second (10 MB/second = 10,000 KB/second). The **topas** output shows en1 at 11,437.5 KB/second. None of the other devices are overutilized at this point. The direct attached SCSI disk hdisk1, which contains the file systems being used for the NIM resources, is only 12.5 percent busy.

When a resource is running at its maximum speed, and is the limiting factor in the system, additional resources need to be added. In this case, tuning will not be able to compensate for the device data rate limit of the ethernet adapter.

To increase the network throughput bandwidth, we have chosen to allocate a Gigabit Ethernet adapter to the system. A Gigabit Ethernet adapter has a one direction maximum data rate of 100 Mbytes per second, 10 times the rate of a 100Mbit adapter.

## Creating a benchmark

One of the difficulties with just monitoring the NIM server as it handles NIM client requests, is that the workload varies. For tuning, it is desirable to have a representative workload that can be used as a benchmark. This benchmark workload can be run before and after tuning to see if any improvement occurs. This does not eliminate the need to monitor the system to determine actual workload benefits from tuning, but does provide a useful way to see if performance tuning is helping a related workload.

Since NIM uses NFS to transfer data between the server and clients, a simple workload is to mount an NFS directory on the clients and generate I/O with the `dd` command. The NIM server can `rsh` to the NIM clients which will be useful in automating the workload.

Example 7-27 shows how to export the directory that is being used for NIM resources. Then we use the remote shell (`rsh`) to mount the exported directory on each of the NIM clients.

---

### *Example 7-27 setting up for NIM benchmark run*

---

```
# exportfs -i -o root=glpar2:glpar3:glpar4 /dasbk
# for i in 2 3 4 ; do
> rsh glpar$i "mount 192.168.100.71:/dasbk /mnt"
> done
```

---

To facilitate the benchmarking effort we created two scripts, one for generating read I/O and one for generating write I/O as shown in Example 7-28 and Example 7-29.

---

### *Example 7-28 NIM write I/O benchmark script*

---

```
#!/usr/bin/ksh

for i in 2 3 4
do
    rsh glpar$i "dd if=/dev/zero of=/mnt/file$i bs=128K count=8000" &
done

#wait command waits for all background processes to finish before continuing
wait
```

---



---

### *Example 7-29 NIM read I/O benchmark script*

---

```
#!/usr/bin/ksh

for i in 2 3 4
do
    rsh glpar$i "dd if=/mnt/file$i of=/dev/null bs=128K count=8000" &
```



```
done
```

```
#wait command waits for all background processes to finish before continuing
wait
```

---

We first execute the **writenim.sh** with the **timex** command to get the total run time of the script.

*Example 7-30 Script for NIM write I/O benchmark*

---

```
# timex ./writenim.sh
8000+0 records in
8000+0 records out
8000+0 records in.
8000+0 records out.
8000+0 records in.
8000+0 records out.

real 386.79
user 0.02
sys 0.00
```

---

We observed that it took 386.79 seconds to write 3000 MB (1000 MB per NIM client). This gives us a throughput of 7.75 MB/second. We also collected a *topas* screen output (Example 7-31) which shows similar results to the actual workload from Example 7-26 on page 605.

*Example 7-31 topas output from write I/O benchmark script*

---

Topas Monitor for host: p690_lpar1						EVENTS/QUEUES		FILE/TTY			
Tue Oct 26 10:19:18 2004						Interval: 2		Cswitch	1750	Readch	0
						Syscall	207	Writech	95		
Kernel	13.0	####				Reads	0	Rawin	0		
User	0.0					Writes	0	Ttyout	95		
Wait	0.0					Forks	0	Igets	0		
Idle	87.0	#####				Execs	0	Namei	2		
						Runqueue	0.0	Dirblk	0		
Network	KBPS	I-Pack	O-Pack	KB-In	KB-Out	Waitqueue	0.0				
en1	12011.6	16283.0	1748.0	23809.5	213.7						
en0	0.1	3.0	0.0	0.1	0.0	PAGING		MEMORY			
lo0	0.0	0.0	0.0	0.0	0.0	Faults	1	Real,MB	4095		
						Steals	0	% Comp	12.8		
Disk	Busy%	KBPS	TPS	KB-Read	KB-Writ	PgspIn	0	% Noncomp	10.7		
hdisk1	84.5	11538.0	358.5	4.0	23072.0	PgspOut	0	% Client	10.9		

---

In order to get an accurate result from the read I/O script, we need to unmount all the related file systems. This is necessary to flush the caches, that is both the file

system cache of the NIM server as well as the NFS client caches of the NIM clients.

*Example 7-32 script for NIM read I/O benchmark*

---

```
# umount /dasbk
# mount /dasbk
# for i in 2 3 4 ; do
> rsh glpar$i umount /mnt
> rsh glpar$i mount 192.168.100.71:/dasbk /mnt
> done
# timex ./readnim.sh
8000+0 records in.
8000+0 records out.
8000+0 records in
8000+0 records out
8000+0 records in.
8000+0 records out.

real 268.74
user 0.02
sys 0.00
```

---

The read benchmark executed in 268.74 seconds, so the read throughput was 11.16 MB/second (3000 MB / 268.74 second). In Example 7-33 we collected a topas screen output which indicates that we have reached the limit of hdisk0 as well.

*Example 7-33 topas output from read I/O benchmark script*

---

```
Topas Monitor for host: p690_lpar1          EVENTS/QUEUES  FILE/TTY
Tue Oct 26 10:32:38 2004 Interval: 2      Cswitch 1232  Readch    0
                                           Syscall  204  Writech   170
Kernel  10.0 |###| Reads      0  Rawin     0
User     0.0 | | Writes     0  Ttyout   170
Wait    46.2 |#####| Forks     0  Igets    0
Idle    43.8 |#####| Execs    0  Namei    2
                                           Runqueue 0.0  Dirblk   0
Network KBPS  I-Pack  O-Pack  KB-In  KB-Out  Waitqueue 3.0
en1  11799.0  8635.0  801.0  488.0  23110.1
en0   0.5     5.0    0.0    1.0    0.0  PAGING
lo0   0.0     0.0    0.0    0.0    0.0  Faults  197  Real,MB  4095
                                           Steals   0  % Comp  12.8
Disk   Busy%   KBPS    TPS  KB-Read  KB-Writ  PgpsIn  0  % Noncomp  5.5
hdisk1 100.0  11494.0  302.5 22988.0  0.0  PgpsOut 0  % Client  5.7
```

---

### 7.6.3 Upgrading NIM environment to Gbit Ethernet

Now that we have a representative workload, we can add the gigabit ethernet adapter and rerun the benchmark workloads. This will give us an idea of what performance increase we may get in the actual workload.

Running the NIM script for write I/O resulted in a time of 260.40 seconds for a throughput of 11.5 MB/second. The execution is identical to Example 7-30 on page 607.

Output from the **topas** command was captured and is shown in Example 7-34.

*Example 7-34 The topas output from write I/O benchmark script*

---

```

Topas Monitor for host:  p690_lpar1          EVENTS/QUEUES  FILE/TTY
Tue Oct 26 11:17:07 2004  Interval: 2      Cswitch  14586  Readch    0
                               Syscall   64   Writech  148
Kernel  63.5  |#####|          | Reads    0  Rawin    0
User    0.0  |          |          | Writes   0  Ttyout   148
Wait    0.0  |          |          | Forks    0  Igets    0
Idle    36.5  |#####|          | Execs    0  Namei    0
                               Runqueue  2.5  Dirblk   0
Network KBPS  I-Pack  O-Pack  KB-In  KB-Out  Waitqueue  0.0
en0  72998.9  99075.0  9752.0  141.4K  1247.4
en1    0.6    8.0    1.0    0.8    0.3  PAGING          MEMORY
lo0    0.0    0.0    0.0    0.0    0.0  Faults    0  Real,MB  4095
                               Steals    0  % Comp   17.3
Disk  Busy%  KBPS    TPS  KB-Read  KB-Writ  PgpsIn  0  % Noncomp  50.5
hdisk1 100.0  42304.0  661.0  0.0  84608.0  PgpsOut  0  % Client  50.7
hdisk3  0.0    0.0    0.0    0.0    0.0  PageIn   0

```

---

The network throughput and CPU utilization has increased, but hdisk1 (the actual disk used for NIM repository) is now 100% busy, and is the current bottleneck. Although we increased the throughput of the networking component ten fold, our benchmark did not see the same amount of performance improvement.

This is typical of the performance tuning process. Increasing one resource often moves the bottleneck to a different component of the system.

Running the NIM script for read I/O resulted in a time of 175.87 seconds for a throughput of 17.1 MB/second. The execution is identical to Example 7-32 on page 608.

Output from the **topas** command was captured and is shown in Example 7-35.

*Example 7-35 topas output from read I/O benchmark script*

---

```

Topas Monitor for host:  p690_lpar1          EVENTS/QUEUES  FILE/TTY

```

```

Tue Oct 26 11:23:31 2004   Interval: 2           Cswitch    919  Readch      0
                               Syscall     59  Writech     162
Kernel   5.0  |##                               Reads       0  Rawin       0
User     0.0  |                               Writes       0  Ttyout     162
Wait     95.0 |#####                               Forks       0  Igets      0
Idle     0.0  |                               Execs       0  Namei      0
                               Runqueue   0.0  Dirblk     0
Network KBPS  I-Pack  O-Pack  KB-In  KB-Out  Waitqueue  8.0
en0  13896.1  9133.0  969.0  527.7  27264.4
en1   0.6     8.0    1.0    0.8    0.4  PAGING      MEMORY
lo0   0.0     0.0    0.0    0.0    0.0  Faults     526  Real,MB    4095
                               Steals      0  % Comp     17.2
Disk   Busy%  KBPS    TPS  KB-Read  KB-Writ  PgpsIn     0  % Noncomp  9.5
hdisk1 100.0  13556.0  372.5 27112.0  0.0  PgpsOut    0  % Client   9.7
hdisk3  0.0    0.0    0.0   0.0    0.0  PageIn    3388

```

## NIM workload results with gigabit ethernet

Utilizing the benchmark script, we did see a performance improvement in both read I/O and write I/O. Now we want to see what kind of improvement we get when the NIM server is handling NIM client requests. We collected topas output as well as iostat output while three NIM client installs were processing simultaneously as shown in Example 7-36 and Example 7-37.

*Example 7-36 NIM server topas output*

```

Topas Monitor for host:  p690_lpar1           EVENTS/QUEUES  FILE/TTY
Tue Oct 26 08:36:13 2004   Interval: 2           Cswitch    2143  Readch      0
                               Syscall    268  Writech     500
Kernel   15.0 |#####                               Reads       0  Rawin       0
User     0.0  |                               Writes       1  Ttyout     326
Wait     85.0 |#####                               Forks       0  Igets      0
Idle     0.0  |                               Execs       0  Namei      2
                               Runqueue   0.0  Dirblk     0
Network KBPS  I-Pack  O-Pack  KB-In  KB-Out  Waitqueue  8.0
en0  28098.3  18455.0  1901.0  1027.2  55169.4K
en1   0.5     4.0    2.0    0.2    0.7  PAGING      MEMORY
lo0   0.0     0.0    0.0    0.0    0.0  Faults     532  Real,MB    4095
                               Steals      0  % Comp     15.2
Disk   Busy%  KBPS    TPS  KB-Read  KB-Writ  PgpsIn     0  % Noncomp  38.8
hdisk1 100.0  27352.0  156.0 54704.0  0.0  PgpsOut    0  % Client   38.9

```

The output is similar to what was seen during the benchmark script runs. Network throughput has increased, but hdisk1 has become completely busy.

During the NIM client installation process, we collected **iostat** command output, using **iostat 5 >> iostat.out**. This started **iostat** collecting statistics every 5

seconds and saved the output to the file `iostat.out`. Once the client installs completed the command was stopped by pressing `Ctrl-C`.

*Example 7-37 NIM server iostat output*

---

```

tty:      tin          tout    avg-cpu:  % user   % sys    % idle   % iowait
          0.2          275.4          0.0     16.2     0.2     83.6

Disks:    % tm_act    Kbps    tps    Kb_read  Kb_wrtn
hdisk0    0.2         3.2     0.8     0         16
hdisk1    100.0      27371.2 167.8    136856    0

tty:      tin          tout    avg-cpu:  % user   % sys    % idle   % iowait
          0.0          146.2          0.3     14.6     0.0     85.1

Disks:    % tm_act    Kbps    tps    Kb_read  Kb_wrtn
hdisk0    0.2         4.0     1.2     0         20
hdisk1    100.0      26365.6 181.0    131828    0

tty:      tin          tout    avg-cpu:  % user   % sys    % idle   % iowait
          0.0          224.4          0.2     10.3     0.4     89.1

Disks:    % tm_act    Kbps    tps    Kb_read  Kb_wrtn
hdisk0    0.2         4.0     0.8     0         20
hdisk1    100.0      22391.2 186.4    111956    0

```

---

## 7.6.4 Upgrading the disk storage

Using a single locally attached SCSI drive poses many problems. One of the most important issues is that there is no data redundancy. Secondly, the performance is not sufficient to handle the client requests.

We have decided to use an IBM storage subsystem DS4500 (FASTT 900) to move the NIM server resources to. We have assigned two LUNs to the NIM server. The two LUNs reside on separate RAID groups. The RAID groups are comprised of 7 disks each and are configured in RAID5, with a stripe size of 64kB.

There are many ways of configuring these two DS4500 LUNs. Since the disk subsystem is handling the data redundancy, we do not need to use LVM mirroring. Our two main choices at this point are either a spread, or a striped logical volume:

- A spread logical volume also known as course striping, alternates data between the hdisks in a volume group on a physical partition level (PP).

- A striped logical volume alternates data between hdisks on a finer basis. With a striped logical volume, you can specify a stripe size from 4KB-128KB (must be a power of two).

We decided to use a JFS2 file system, so we also have additional choices on how to create the JFS2 log (in-line or on a separate logical volume).

### **Configuring the LVM**

AIX 5.3 has the device drivers and disk type pre-loaded for DS4500 disk devices. After configuring the DS4500 storage and assigning the LUNs to the server, the command **cfgmgr** detects the new storage and configures it to the system. After the disk is configured to the system, the disk need to assigned to a volume group. Once assigned to a volume group, the disk is automatically split into physical partitions. These physical partitions can then be made into logical volumes and the file system configured on the logical volumes.

To start the process we create a new volume group with the DS4500 disk devices. Example 7-38 we first list the disks available to the system with the **lsdev** command, then define the volume group with **mkvg**. Finally we check the characteristics of the volume group with **lsvg**.

#### *Example 7-38 Creating a new volume group*

---

```
# lsdev -Cc disk
hdisk0 Available 3s-08-00-8,0 16 Bit LVD SCSI Disk Drive
hdisk1 Available 5M-08-00-8,0 16 Bit LVD SCSI Disk Drive
hdisk2 Available 41-08-02      MPIIO Other FC SCSI Disk Drive
hdisk3 Available 41-08-02      MPIIO Other FC SCSI Disk Drive
hdisk4 Available 41-08-02      1742-900 (900) Disk Array Device
hdisk5 Available 4Q-08-02      1742-900 (900) Disk Array Device

# mkvg -y ds4500vg hdisk4 hdisk5
ds4500vg

# lsvg ds4500vg
VOLUME GROUP:      ds4500vg          VG IDENTIFIER:
0022be2a00004c00000000ff6b94f26
VG STATE:          active           PP SIZE:        32 megabyte(s)
VG PERMISSION:    read/write       TOTAL PPs:     1022 (32704
megabytes)
MAX LVs:          256              FREE PPs:      1022 (32704
megabytes)
LVs:              0                USED PPs:      0 (0 megabytes)
OPEN LVs:         0                QUORUM:        2
TOTAL PVs:        2                VG DESCRIPTORS: 3
STALE PVs:        0                STALE PPs:     0
ACTIVE PVs:       2                AUTO ON:       yes
MAX PPs per VG:  32512
```

MAX PPs per PV:	1016	MAX PVs:	32
LTG size (Dynamic):	1024 kilobyte(s)	AUTO SYNC:	no
HOT SPARE:	no	BB POLICY:	relocatable

---

The volume group contains two physical volumes, with a physical partition (PP) size of 32 megabytes. There are a total of 1022 PP, and the volume group has a logical track group (LTG) of 1024 kilobytes.

### Spread versus striped logical volume

We cannot be sure up front which type of logical volume will give us the best performance. To help determine which type to use, we will create both types of logical volumes and run the benchmark script.

#### *Example 7-39 Creating stripe and spread logical volumes*

---

```
# mklv -y'spreadlv' -t'jfs2' -e'x' ds4500vg 120
spreadlv
# mklv -y'stripe1v' -t'jfs2' '-S64K' ds4500vg 120 hdisk4 hdisk5
stripe1v
```

---

Now that the logical volumes are created, we can create the jfs2 file systems on these logical volumes.

#### *Example 7-40 Create and mount file systems*

---

```
# mklv -y'spreadlv' -t'jfs2' -e'x' ds4500vg 120
spreadlv
# mklv -y'stripe1v' -t'jfs2' '-S64K' ds4500vg 120 hdisk4 hdisk5
stripe1v
# crfs -v jfs2 -d'spreadlv' -m'/spreadfs' -A'No' -p'rw' -a agblksize='4096'
File system created successfully.
3931836 kilobytes total disk space.
New File System size is 7864320
# crfs -v jfs2 -d'stripe1v' -m'/stripefs' -A'No' -p'rw' -a agblksize='4096'
File system created successfully.
3931836 kilobytes total disk space.
New File System size is 7864320
# mount /spreadfs
# mount /stripefs
```

---

With the file systems mounted, we need to export the file systems and mount them on the NIM clients like we did in Example 7-27 on page 606.

#### *Example 7-41 Exporting file systems for benchmark testing*

---

```
# exportfs -i -o root=glpar2:glpar3:glpar4 /spreadfs
```

```
# exportfs -i -o root=glpar2:glpar3:glpar4 /stripefs
```

---

### **Benchmarking spread file system**

With the file systems exported, we can mount one of them and run the benchmark as shown in Example 7-42.

#### *Example 7-42 Mounting spreadfs on NIM clients*

---

```
# for i in 2 3 4; do
> rsh glpar$i "mount glpar1:/spreadfs /mnt"
> done
```

---

The hostname *glpar2* is the IP label (as associated in the */etc/hosts* file) for LPAR2 Gbit Ethernet interface, and so on (see the IP labels assignment in Figure 7-4 on page 603).

With the spreadfs file system NFS mounted on each of the NIM clients, we run the same script from Example 7-28 on page 606.

#### *Example 7-43 Script for NIM write I/O benchmark - gigabit and DS4500*

---

```
# timex ./writenim.sh
8000+0 records in.
8000+0 records out.
8000+0 records in
8000+0 records out
8000+0 records in.
8000+0 records out.

real 44.99
user 0.02
sys 0.01
```

---

With both gigabit ethernet and the DS4500 disk subsystem, the write throughput has increased dramatically to 66.7 MB/second (3000 MB / 44.99 second).

In Example 7-44 we show the full topas screen output as there is enough load for the other values to be of interest.

#### *Example 7-44 The topas output for NIM write benchmark - DS4500 spread and gigabit ethernet*

---

Topas Monitor for host:	p690_lpar1	EVENTS/QUEUES	FILE/TTY
Wed Oct 27 08:59:00 2004	Interval: 2	Cswitch 12640	Readch 0
		Syscall 61	Writech 163
<b>Kernel</b> 92.0	#####	Reads 0	Rawin 0
User 0.0		Writes 0	Ttyout 163
Wait 0.0		Forks 0	Igets 0
Idle 8.0	###	Execs 0	Namei 0



						Runqueue	0.0	Dirblk	0
Network	KBPS	I-Pack	O-Pack	KB-In	KB-Out	Waitqueue	0.0		
<b>en0</b>	<b>110.8K</b>	<b>151.0K</b>	<b>13837.0</b>	<b>219.8K</b>	<b>1861.7</b>				
en1	0.3	3.0	1.0	0.1	0.4	PAGING		MEMORY	
lo0	0.0	0.0	0.0	0.0	0.0	Faults	0	Real,MB	4095
						Steals	0	% Comp	17.2
Disk	Busy%	KBPS	TPS	KB-Read	KB-Writ	PgspIn	0	% Noncomp	50.4
<b>hdisk5</b>	<b>100.0</b>	<b>54720.0</b>	<b>53.5</b>	<b>0.0</b>	<b>106.9K</b>	PgspOut	0	% Client	50.4
<b>hdisk4</b>	<b>84.5</b>	<b>59408.0</b>	<b>140.0</b>	<b>0.0</b>	<b>116.0K</b>	PageIn	0		
hdisk2	0.0	0.0	0.0	0.0	0.0	PageOut	28542	PAGING SPACE	
						Sios	27255	Size,MB	512
Name	PID	CPU%	PgSp	Owner				% Used	1.0
<b>nfsd</b>	<b>409606</b>	<b>50.1</b>	<b>0.8</b>	<b>root</b>		NFS (calls/sec)		% Free	98.9
j2pg	147544	0.0	0.2	root		ServerV2	0		
topas	307366	0.0	1.2	root		ClientV2	0	Press:	
gil	73764	0.0	0.1	root		ServerV3	3412	"h" for help	
pilegc	61470	0.0	0.2	root		ClientV3	0	"q" to quit	

The topas output shows that some of the system resources are being fully utilized. This is a good thing. The CPU is only 8 percent idle, but is showing zero wait time. The new disks are fully utilized, but appear to be slightly imbalanced. And the Gbit Ethernet card throughput (one direction) is close to its maximum of 100 MB/second.

Now we run the read I/O benchmark making sure to flush the caches as shown in Example 7-45.

*Example 7-45 Script for NIM read I/O benchmark - DS4500 spread and gigabit*

```
# umount /spreadfs
# mount /spreadfs
# for i in 2 3 4 ; do
> rsh glpar$i umount /mnt
> rsh glpar$i mount glpar1:/spreadfs /mnt
> done
# timex ./readnim.sh
8000+0 records in.
8000+0 records out.
8000+0 records in.
8000+0 records out.
8000+0 records in
8000+0 records out

real 30.20
user 0.02
sys 0.00
```

With both gigabit ethernet and the DS4500 disk subsystem, the read throughput has increased dramatically to 99.3 MB/second (3000 MB / 30.20 second).

In Example 7-46 on page 616 we show the full topas screen output as there is enough load for the other values to be of interest.

*Example 7-46 The topas output for NIM read benchmark - DS4500 spread and gigabit ethernet*

---

```

Topas Monitor for host:  p690_lpar1          EVENTS/QUEUES  FILE/TTY
Wed Oct 27 09:04:54 2004  Interval: 2      Cswitch  6905  Readch    0
                                           Syscall  53   Writech   185
Kernel  52.2 |#####| Reads    0   Rawin     0
User      0.0 |      | Writes  0   Ttyout   185
Wait    24.0 |#####| Forks   0   Igets    0
Idle     23.8 |#####| Execs  0   Namei    0
                                           Runqueue 2.5 Dirblk    0

Network  KBPS  I-Pack  O-Pack  KB-In  KB-Out  Waitqueue 0.5
en0    102.5K 69925.0 7118.0 4028.4 201.1K
en1      0.3    4.0     1.0    0.2    0.4
lo0      0.0    0.0     0.0    0.0    0.0
PAGING
Faults   648  Real,MB  4095
Steals   0   % Comp   17.2
Disk     Busy%   KBPS     TPS  KB-Read  KB-Writ  PgpsIn   0   % Noncomp  38.9
hdisk4 56.5 53248.0 1524.0 104.0K 0.0  PgpsOut  0   % Client   39.0
hdisk5 43.0 49152.0 1360.0 98304.0 0.0  PageIn   25584
hdisk2   0.0    0.0     0.0    0.0    0.0  PageOut  0   PAGING SPACE
Sios     25602 Size,MB   512
                                           % Used    1.0
Name      PID  CPU%  PgSp  Owner  NFS (calls/sec) % Free    98.9
nfsd    409606 25.0 0.8 root  ServerV2      0
topas     254150 0.0   1.5  root  ClientV2      0
j2pg     147544 0.0   0.2  root  ServerV3     3198
gil       73764 0.0   0.1  root  ClientV3      0
aixmibd  442596 0.0   0.6  root
Press:
"h" for help
"q" to quit

```

---

Although the read throughput was higher, the resources do not report being as busy as with the write workload. The network adapter is still close to its limit at 100 MB/second.

Now that we have some numbers for a spread file system, we will run the same benchmark for the striped file system.

### ***Benchmarking stripe file system***

To prepare for running the same benchmark against the stripe file system, we need to unmount the spread file system from the NIM clients. Then the striped file system needs to be mounted, and the script run.

*Example 7-47 Script for NIM write I/O benchmark - gigabit and DS4500 striped*

```

# for i in 2 3 4 ; do
> rsh glpar$i "umount /mnt"
> rsh glpar$i "mount glpar1:/stripefs /mnt"
> done
# timex ./writenim.sh
8000+0 records in.
8000+0 records out.
8000+0 records in.
8000+0 records out.
8000+0 records in
8000+0 records out

real 98.00
user 0.02
sys 0.00

```

The striped file system finished in 98 seconds giving a throughput of 30.6 MB/s (3000 MB/98 seconds). This is much slower than the spread file system which finished in less than half the time at 45 seconds.

This may be a good indication that in our environment we are using large sequential reads and writes (typical NIM environment).

*Example 7-48 The topas output for NIM write benchmark - DS4500 stripe and Gigabit Ethernet*

```

Topas Monitor for host:  p690_lpar1          EVENTS/QUEUES  FILE/TTY
Wed Oct 27 09:25:51 2004  Interval: 2          Cswitch  7700  Readch    0
                          Syscall   54   Writech   142
Kernel  94.1 |#####| Reads     0   Rawin     0
User      0.0 |      | Writes    0   Ttyout   142
Wait     0.0 |      | Forks     0   Igets    0
Idle     5.9 |##  | Execs     0   Namei    0
                          Runqueue  17.9  Dirblk    0
                          Waitqueue 0.0

Network KBPS  I-Pack  O-Pack  KB-In  KB-Out  Waitqueue
en0  96998.7  127.4K 10872.0  183.2K 1512.8
en1     0.5    5.0    1.0    0.6    0.3
lo0     0.0    0.0    0.0    0.0    0.0

Disk  Busy%  KBPS  TPS  KB-Read  KB-Writ  PgpsIn  0  % Noncomp  77.1
hdisk5 100.0 49019.5 140.5  4.0 95584.0 PgpsOut 0  % Client  77.2
hdisk4 95.3 46992.8 782.1  20.0 91616.0 PageIn  0
hdisk2  0.0    0.0    0.0    0.0    0.0  PageOut 24054  PAGING SPACE
                          Sios     23294  Size,MB   512
                          % Used    1.0
Name      PID  CPU%  PgSp  Owner  NFS (calls/sec)  % Free  98.9
nfdsd   409606 52.2  0.8  root ServerV2  0
lrud     45078 26.1  0.1  root  ServerV2  0
j2pg    147544 0.0  0.2  root  ClientV2  0  Press:
topas   286806 0.0  1.5  root  ServerV3  2976  "h" for help

```

```
gil          73764  0.0  0.1 root          ClientV3      0  "q" to quit
```

---

Disk Busy% increased, but KBPS for the disks is lower. The extra overhead in splitting the I/Os into 64 KB strips between the DS4500 RAID LUNs resulted in decreased write performance. Course striping from implementing a spread file system appears to outperform fine striping as shown in Example 7-49.

After observing the write performance, it is time to finish the benchmark comparison by running the read I/O script.

*Example 7-49 Script for NIM read I/O benchmark - gigabit and DS4500 striped*

---

```
# umount /stripefs;mount /stripefs
# for i in 2 3 4 ; do
> rsh glpar$i "umount /mnt"
> rsh glpar$i "mount glpar1:/stripefs /mnt"
> done
# timex ./readnim.sh
8000+0 records in.
8000+0 records out.
8000+0 records in.
8000+0 records out.
8000+0 records in
8000+0 records out

real 29.46
user 0.02
sys 0.00
```

---

Read throughput for the striped file system is comparable and finished less than a second quicker than the spread file system. With both gigabit ethernet and the DS4500 disk subsystem, the read throughput for the striped file system has increased to 101.8 MB/second (3000 MB / 29.46 second).

In Example 7-50 we show the topas screen output for the NIM read benchmark to the striped file system.

*Example 7-50 The topas output for NIM read benchmark - DS4500 stripe and Gigabit Ethernet*

---

```
Topas Monitor for host:  p690_lpar1          EVENTS/QUEUES  FILE/TTY
Wed Oct 27 09:32:52 2004  Interval: 2          Cswitch  7185  Readch      0
                          Syscall    59    Writech    228
Kernel  53.5  |#####|          Reads     0    Rawin      0
User      0.0  |          |          Writes    0    Ttyout    228
Wait    25.0  |#####|          Forks     0    Igets     0
Idle     21.5  |#####|          Execs     0    Namei     0
                          Runqueue  0.5  Dirblk     0
Network  KBPS  I-Pack  O-Pack  KB-In  KB-Out  Waitqueue  5.0
```

```

en0      104.4K 71323.0 7186.0 4107.7 204.8K
en1       0.5   3.0   1.0   0.6   0.5  PAGING      MEMORY
lo0       0.0   0.0   0.0   0.0   0.0  Faults      639  Real,MB    4095
          Steals      0  % Comp    17.2
Disk Busy%   KBPS     TPS KB-Read KB-Writ PgpsIn  0  % Noncomp 46.4
hdisk4 89.0 52166.0 1377.0 101.9K 0.0 PgpsOut  0  % Client  46.4
hdisk5 70.0 52140.0 1361.0 101.8K 0.0 PageIn   26065
hdisk2    0.0   0.0   0.0   0.0   0.0  PageOut   0  PAGING SPACE
          Sios      26060  Size,MB   512
          % Used    1.0
Name      PID   CPU%  PgSp  Owner
 nfsd   409606 25.0 0.8 root  NFS (calls/sec) % Free    98.9
topas     266346  0.0  1.5  root  ServerV2      0
j2pg     147544  0.0  0.2  root  ClientV2      0  Press:
aixmibd  442596  0.0  0.6  root  ServerV3     3257  "h" for help
rpc.lockd 385272  0.0  0.2  root  ClientV3      0  "q" to quit

```

The disk Busy% is much higher for the striped file system, but the load is better balanced. The rest of the resource utilization is similar to the spread file system.

## 7.6.5 Real workload with spread file system

Although the striped file system outperformed the spread file system for read operations, the difference was small. The difference between write throughput was significant, the spread file system had more than double the write throughput of the striped file system. Because of these results, we select the spread file system.

**Important:** Little benefit is gained from doing a similar feature twice. For example it is common knowledge that with database applications, you want to avoid double buffering. Double buffering is where both the application and the operating system use memory as cache. This consumes memory that could be used for other operations, and wastes CPU cycles doing redundant caching. Likewise with striping. The DS4500 LUNs are already striped across disks. Using LVM striping across DS4500 LUNs results in, for lack of a better term, double striping. The DS4500 controller is going to receive small stripes alternating between different disk groups. This extra striping is not benefiting the disk subsystem, and could even be causing performance degradation.

In summary, it is important to understand the workload the application is generating so as to make the most efficient use of system resources.

With the NIM resources moved to the spread file system, we can now monitor a real workload. For this we will simultaneously restore three NIM clients and monitor the output with **topas**, **iostat**, and **sar** as shown in Example 7-51.

*Example 7-51 The topas output after Gigabit Ethernet and DS4500 storage*

```

Topas Monitor for host:  p690_lpar1          EVENTS/QUEUES  FILE/TTY
Wed Oct 27 17:18:50 2004  Interval: 2      Cswitch  5126  Readch    0
                               Syscall   218   Writech   645
Kernel  34.5  |#####| Reads     0   Rawin     0
User     0.2  |#| Writes    1   Ttyout   189
Wait     4.5  |##| Forks     0   Igets    0
Idle    61.2  |#####| Execs     0   Namei    2
                               Runqueue  0.0   Dirblk   0

Network KBPS  I-Pack  O-Pack  KB-In  KB-Out  Waitqueue  0.0
en0  73874.8  47437.0  4809.0  2651.3  141.7K
en1   0.3     5.0    1.0    0.2    0.4  PAGING
lo0   0.0     0.0    0.0    0.0    0.0  Faults   175  Real,MB  4095
                               Steals    0   % Comp   18.1
Disk  Busy%  KBPS    TPS  KB-Read  KB-Writ  PgpsIn    0  % Noncomp  37.2
hdisk5  31.0  36776.0  261.0  73552.0  0.0  PgpsOut    0  % Client   37.3
hdisk4  28.0  35480.0  199.0  70960.0  0.0  PageIn   17942
hdisk2  0.0   0.0    0.0    0.0    0.0  PageOut    0  PAGING SPACE
                               Sios     18022  Size,MB  512
Name      PID  CPU%  PgSp  Owner  NFS (calls/sec)  % Used  1.0
nfsd     409606  0.0  0.8  root  ServerV2    0  % Free  98.9
topas    348170  0.0  1.2  root  ClientV2    0
sadc     393360  0.0  0.2  root  ServerV3   2232  Press:
nimesis  262222  0.0  1.4  root  ClientV3    0  "h" for help
gil      73764  0.0  0.1  root  "q" to quit

```

The three NIM clients definitely made use of the additional network and storage resources. They have not “maxed out” the available resources based on the benchmark testing.

To collect iostat information we executed the command **iostat 5 60 >> iostat.out** as shown in Example 7-52. This collected statistics every 5 seconds for 60 intervals, for a total of five minutes. We scanned through the output and Example 7-52 contains the interval where activity was the highest.

*Example 7-52 The iostat output after gigabit ethernet and DS4500 storage*

```

tty:      tin      tout  avg-cpu:  % user  % sys  % idle  % iowait
          0.4      80.8          0.2   25.7   73.1    1.0

Disks:    % tm_act  Kbps   tps   Kb_read  Kb_wrtn
hdisk0    0.0      0.0    0.0    0        0
hdisk1    0.0      0.0    0.0    0        0
hdisk3    0.0      0.0    0.0    0        0
hdisk2    0.0      0.0    0.0    0        0
dac0      0.0     27161.6  97.4  135808    0
dac0-utm  0.0      0.0    0.0    0        0
dac1      0.0     25395.2  97.0  126976    0

```

dac1-utm	0.0	0.0	0.0	0	0
hdisk4	18.8	27161.6	97.4	135808	0
hdisk5	20.0	25395.2	97.0	126976	0
cd0	0.0	0.0	0.0	0	0

---

The dac0 and dac1 items in the Disks: column are the DS4500 controllers. If there were more disks per controller and activity on the disks, the dac0 and dac1 would show a cumulative value. As there is only one disk per controller in our configuration, the values are the same for the disk and its associated controller.

The DS4500 system is handling the I/O requests and has some throughput left over. The load is spread fairly evenly over the two DS4500 hdisks.

The **sar** command is another useful way of collecting disk statistics. To collect the disks statistics in Example 7-53 we executed the command **sar -d 5 60 >> sar.out**. We then scanned the output and selected the interval with the highest utilization.

*Example 7-53 The sar output after gigabit ethernet and DS4500 storage*

---

```
117:15:28  device  %busy  avque  r+w/s  Kbs/s  await  avserv
...(lines omitted)....
          dac0    0    0.0    145    33231    0.0    0.0
          dac0-utm 0    0.0     0     0     0.0    0.0
          dac1    0    0.0    150    32261    0.0    0.0
          dac1-utm 0    0.0     0     0     0.0    0.0
          hdisk4  24    0.0    145    33231    0.0    0.0
          hdisk5  24    0.0    150    32261    0.0    0.0
...(lines omitted)....
```

---

The output from **sar -d** is similar to **iostat**, but with **sar** we get some additional values that can be useful.

Zero values for *avque*, *await*, *avserv* are desirable. Nonzero values should be subject for further investigation and tuning.

## 7.6.6 Summary

Performance tuning is an iterative process. This case study went through a couple of basic iterations of the process. It is important to accurately identify system bottlenecks and then make the correct choice as to add resources, tune resources, or leave it alone. Performance tuning on a production system is risky. Having system backups and system documentation can go a long way in recovering from bad tuning choices. An understanding of the actual system workload and the effects of tuning commands is the important first step in the performance tuning process.

## 7.6.7 Multi-threaded option for the NIM nimesis daemon.

If you are installing a large number of nodes, enable the multi-threaded option on the NIM nimesis daemon by setting the **max\_nimesis\_threads** value. Setting this value improves the performance of NIM when you are working with large numbers of nodes. The multi-threaded option provides better handling of the volume of client information change requests and client state changes. Without the use of the multi-threaded option, the NIM master can become overloaded by activity on the NIM database and the number of active processes, resulting in simultaneous failures during the installation of a large number of client machines. The multi-threaded nimesis daemon will serialize and buffer NIM client requests to protect the NIM master from process overload, without causing significant performance degradation.

The number of threads assigned to this daemon determines how many simultaneous NIM client requests can be handled in the NIM environment. Because most of the NIM client requests are processed rapidly, it is not necessary to have one thread for every client installing. The number of threads needed to support the activities in a NIM environment is dependent upon several items. The following should be considered when determining the number of threads:

- ▶ Number of clients that will be operated on at the same time
- ▶ Processing capacity of the NIM master machine
- ▶ What type of operations are planned

In general, one thread can support two to four clients that are installing BOS at the same time. For example, when installing 150 machines, 50 to 75 threads is sufficient. The number of threads is highly dependent on the processing power of the NIM master machine, and slower master machines may require more threads.

For smaller NIM environments, enabling the multi-threaded daemon can monopolize system resources on the master that will not be used. For example, when installing 50 machines simultaneously, 20 to 25 threads or even the single-threaded daemon would suffice.

**Note:** The multi-threaded option alone will not allow more machines to be installed simultaneously. The multi-threaded option should be used in conjunction with global export of NIM resources, distribution of NIM resources throughout the NIM environment (i.e. resource servers), and a network environment capable of handling a large volume of throughput.



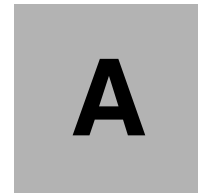
You can specify a value from 20 to 150 (the default is 20). The general rule is to set it for approximately half the number of nodes on which you are operating simultaneously. For example, if you want to install 100 nodes, set the value to 50. Issue the following command to set the `max_nimesis_threads` value:

```
# nim -o change -a max_nimesis_threads=50 master
```

When resources are allocated for use during NIM operations, they are NFS-exported to the client machines where the operations will be performed. If operations are performed simultaneously on many different clients, the `/etc/exports` and `/etc/xtab` files may become very large on the NIM master. This may cause size limits to be exceeded in the files, and it may also negatively affect NIM performance as the files are locked and modified for each resource allocation or deallocation. If you are installing a large number of nodes, set the **`global_exports`** attribute to "yes" on the NIM master. It is a good idea to set this attribute when you are simultaneously running NIM operations to many nodes.

```
# nim -o change -a global_export=yes master
```





## General LPAR sizing and creation with mig2p5 tools

The *mig2p5* tools were developed to provide a set of user-friendly tools that enable you to facilitate the migration of workloads from older AIX machines to POWER5 LPARs (with either dedicated or shared processors). The *mig2p5* files were written by the IBM Yorktown research team and were incorporated into the **nim\_move\_up** tool written by the STG team in Austin.

The *mig2p5* tools have been designed to work with the **nim\_move\_up** tool but can also be used as standalone scripts. We provide a brief introduction on how to use these scripts outside of the **nim\_move\_up** environment.

## The mig2p5 tools

The *mig2p5* tools are set of perl scripts. These scripts define a process for automating the sizing and creation of POWER5 LPARS. The basic flow of this process is described as follows and illustrated in Figure A-1.

1. Collect the source LPARs CPU and memory utilization. This is performed by **getSrcUtil**.
2. Collect the source LPARs hardware configuration. This is performed by **getSrcCfg**.
3. Capture the target POWER5 machine's hardware configuration. **getTgtRsrc** performs this task.
4. Generate the target LPAR's required configuration, based on the hardware requirements and CPU/memory utilization collected in step one. **genTgtCfg** executes this task.
5. Create the target LPAR on the POWER5 machine via **createTgtLPAR**.

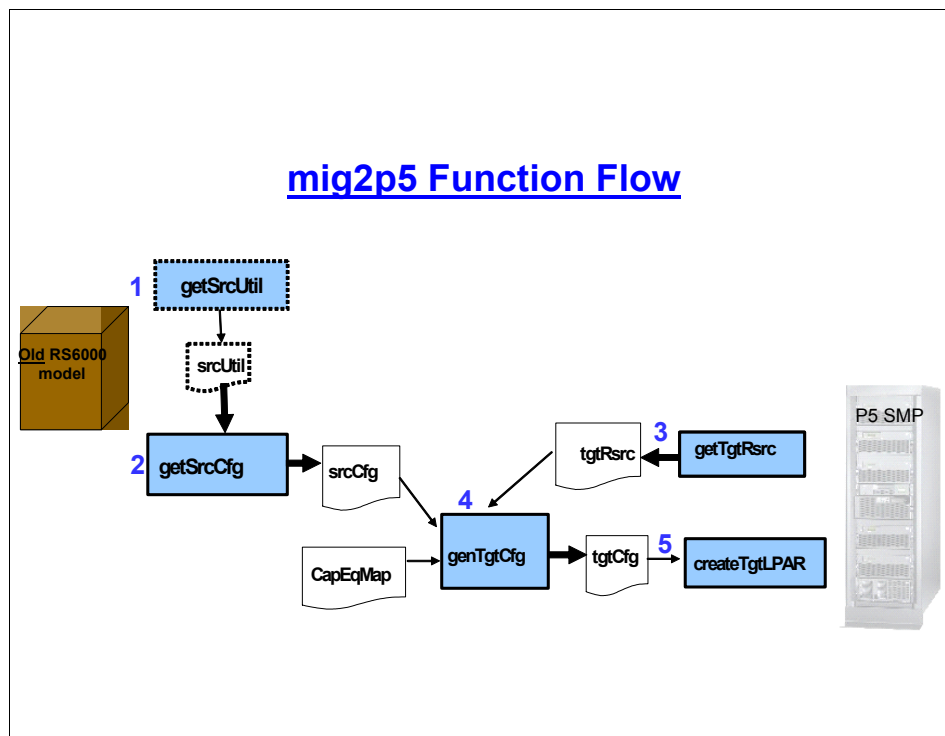


Figure A-1 *mig2p5* function flow

The *mig2p5* scripts are listed as follows:

```
/usr/lpp/bos.sysmgt/nim/methods/getSrcUtil  
/usr/lpp/bos.sysmgt/nim/methods/getSrcCfg  
/usr/lpp/bos.sysmgt/nim/methods/getTgtRsrc  
/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg  
/usr/lpp/bos.sysmgt/nim/methods/createTgtLPAR
```

They are incorporated with the **nim\_move\_up** utility but can be used outside of this to size and create new POWER5 LPARs.

### **/usr/lpp/bos.sysmgt/nim/methods/getSrcUtil**

This script generates a file (*srcUtil*) containing the systems CPU and memory utilization over a period of time (which you can specify). It collects **vmstat** information which is used by the **genTgtCfg** script to size the new POWER5 LPAR. To collect data over a longer period of time, you can adjust the interval and loop count. The following will collect performance data on LPAR4 for a 24 hour period.

```
{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./getSrcUtil -s LPAR4 -o  
/tmp/mig2p5/lpar4util.out -i 10 -l 9000
```

### **/usr/lpp/bos.sysmgt/nim/methods/getSrcCfg**

This script generates a file (*srcCfg*) with the source systems hardware configuration.

### **/usr/lpp/bos.sysmgt/nim/methods/getTgtRsrc**

This script generates a file (*tgtRsrc*) containing information about hardware resources available on the target POWER5 machine.

### **/usr/lpp/bos.sysmgt/nim/methods/genTgtCfg**

This generates the new LPAR definition which will be used to create the new LPAR on the POWER5 system. It uses the *capEqMap* file (*/usr/lpp/bos.sysmgt/nim/methods/capEqMaps*), and the *srcCfg* and *tgtRsrc* files during this process. It creates a file (*tgtCfg*) that contains the target LPARs configuration.

### **/usr/lpp/bos.sysmgt/nim/methods/createTgtLPAR**

This script creates the new LPAR definition and profile on the POWER5 system, based on the *tgtCfg* file.

## Standalone mig2p5 tool

As shown in Figure A-2 on page 629, the mig2p5 tools can be issued from any standalone AIX 5L V5.3 machine. When issuing the tools from a standalone AIX instance, the following occurs:

1. From the standalone AIX system, **getSrcUtil** and **getSrcCfg** connect to the source system (via rsh or ssh) and collect CPU/memory utilization and hardware configuration information. Refer to 4.5, “NIM mksysb migration and nim\_move\_up POWER5 tools” on page 206 and/or *Setting up secure script execution between SSH clients and the HMC at* (at the Web site below) for more information on configuring SSH keys for communication between an AIX host and a HMC.

<http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp>

2. **getTgtRsrc** connects the POWER5 HMC via ssh and gathers hardware resource information for the POWER5 machine.
3. **genTgtCfg** generates the target LPAR's required configuration.
4. The **createTgtLPAR** scripts creates the new LPAR definition on the POWER5 machine.

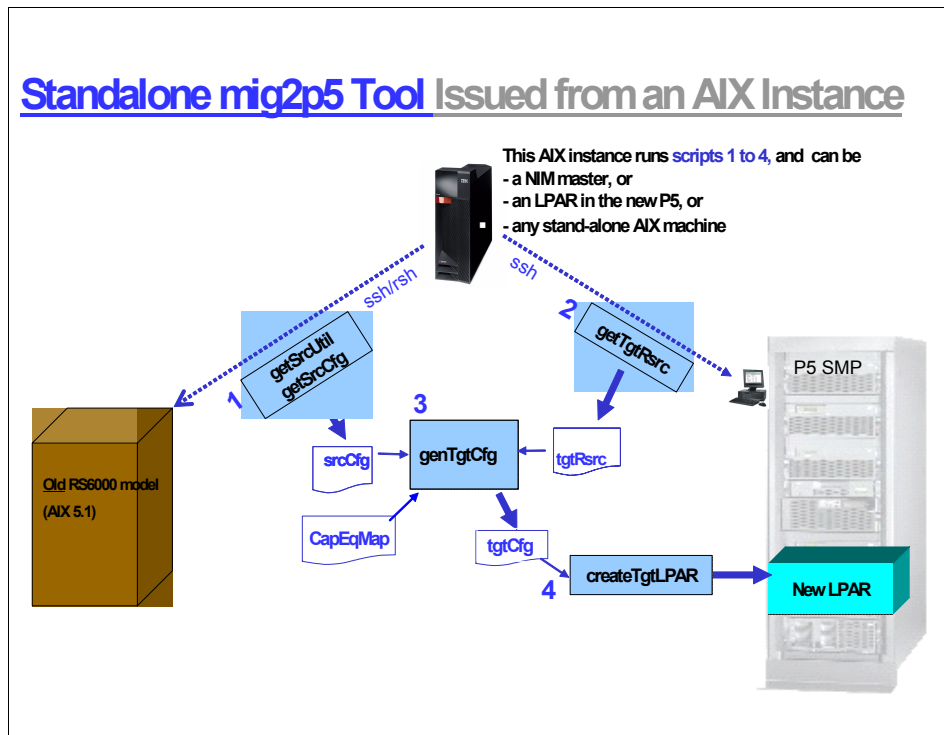


Figure A-2 Using mig2p5 to size and create a POWER5 LPAR

The following are examples of using the mig2p5 tools from a NIM master. These scripts are currently delivered via the **bos.sysmgt.nim.master** fileset. You could copy these scripts to another AIX system if you wanted to run them somewhere other than the NIM master.

## Get the source LPARs hardware configuration

Example A-1 shows how to get the source LPARs' hardware configuration.

*Example: A-1 Acquiring the source LPARs hardware configuration*

```
{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./getSrcCfg -s lpar4 -o
/tmp/mig2p5/lpar4cfg.out -util /tmp/mig2p5/lpar4util.out
```

INPUT:

```
Source Hostname           = lpar4
File name for source configuration= /tmp/mig2p5/lpar4cfg.out
File containing utilization data = /tmp/mig2p5/lpar4util.out
-----
```

Getting the type & model of the given source machine.....

```
-----
Getting the info about CPUs and Memory in the source machine.....
-----
```

```
Adding utilization info from the /tmp/mig2p5/lpar4util.out
File.....
```

```
-----
Getting the info about volume groups in the source machine.....
-----
```

```
-----
Getting the info about network adapters in the source machine....
-----
```

## Get the source LPARs CPU and memory utilization

Example A-2 shows how to get the source LPARs' hardware configuration.

*Example: A-2 Getting the source LPARs CPU and memory utilization*

```
{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./getSrcUtil -s LPAR4 -o
/tmp/mig2p5/lpar4util.out -i 10 -l 9000
```

```
The number of vmstat loops :           9000
The Average of Active virtual pages:    229 MB
The Average of Size of the free list:    6 MB
The sum of the CPU %idle:                291
The Average of CPU idle:                 97.00
The Average of CPU utilization:          3.00
```

## Flags for getSrcCfg command

Example A-3 shows how to get the source LPARs' hardware configuration.

*Example: A-3 Shows flags for getSrcCfg command*

```
{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./getSrcCfg
```

SYNTAX:

```
getSrcCfg -s <srcHostname> [-o <srcCfgFile>] [-util <utilFile>]
```

where

<srcHost> : Hostname of the source machine

<utilFile> : File Containing the CPU and Memory utilization

values for the source

<srcCfgFile> : Output file into which the source configuration



will be written

-h : prints this syntax help. This flag cannot be used with other flags.

NOTE: This version only works for <srcHostname> running AIX 5.0 and above (as it uses the lsconf command).

---

## Flags for the getSrcUtil command

Example A-4 shows how to get the source LPARs' hardware configuration.

*Example: A-4 Shows the flags for the getSrcUtil command*

---

```
{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./getSrcUtil
```

SYNTAX:

```
getSrcUtil -s <srcHost> [-o <srcUtilFile> ] [-i <interval>] [-l <loops>]
] [-h] [-v]
```

where flag

-o writes the output to a file. Default file:  
/tmp/srcUtil

-i denotes the <interval> in # seconds. Default: 1

-l denotes the number of loops; Default: 10

-v Version number

---

## Get the target POWER5 machine resources via its HMC

Example A-5 shows how to get the source LPARs' hardware configuration.

*Example: A-5 Gets the target POWER5 machine resources*

---

```
{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./getTgtRsrc -hmc hmcsark
-u hscroot -m SARK-520-65C544E -viosEth VI02 -viosDisk VI02 -o
/tmp/mig2p5/tgtrsc.out
```

INPUT:

HMC Host	= hmcsark
HMC Username	= hscroot
Managed System name	= SARK-520-65C544E
Ethernet VIO Server(s)	= VI02
Disk VIO Server(s)	= VI02

```

Output file                               = /tmp/mig2p5/tgtrsc.out
-----
Getting the type & model of the given P5 machine.....
cmd = ssh hscroot@hmcsark lssyscfg -m SARK-520-65C544E -r sys
-Ftype_model
Type-Model = 9111-520
-----
Getting the number of processing units in the given P5 machine.....
cmd = ssh hscroot@hmcsark lshwres -m SARK-520-65C544E -r proc --level
sys -Fconfigurable_sys_proc_units
available_proc_units = 2.0
-----
Getting the amount of memory in MBs in the given P5 machine.....
Configurable Memory = 16384
Available memory = 11648
LMB Size = 64
-----
Getting the available physical I/O slots.....
-----
Getting the volume groups info in the VIO Disk server(s).....
Command:
  ssh hscroot@hmcsark viosvr cmd -m SARK-520-65C544E -p VI02 -c
  "\"lsvg\"|"
Command:
  ssh hscroot@hmcsark viosvr cmd -m SARK-520-65C544E -p VI02 -c "\"lsvg
  rootvg\"|"
VIO Server = VI02; VG = rootvg; Size = 41088MB
-----
Getting the available VLANs info in the VIO server(s).....
Command:
  ssh hscroot@hmcsark lshwres -m SARK-520-65C544E -r virtualio
  --subtype eth --level lpar --filter "lpar_names=VI02"

-Flpar_name:slot_num:port_vlan_id:is_trunk
VIO Server = VI02; VLAN Id = 1; Slot number = 30

```

---

## Flags for the getTgtRsrc command

Example A-6 shows how to get the source LPARs' hardware configuration.

*Example: A-6 Shows the flags for the getTgtRsrc command*

---

```
{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./getTgtRsrc
```

**SYNTAX:**

```
getTgtRsrc -hmc <hmcHost> -u <hmcUser> -m <managedSys> [-viosEth
<vioEthServers>] [-viosDisk <vioDiskServers>] -o <outFile>
```

where

<hmcHost> : Hostname of the target HMC

<hmcUser> : Name of the user on the HMC who can execute

Administrative commands

<managedSys> : Name of the target managed system

<vioEthServers> : LPAR names of the VIO servers for Ethernet.

If you have multiple VIO server LPARs, separate the names with comma.

<vioDiskServers> : LPAR names of the VIO servers for Hard Disks.

If you have multiple VIO server LPARs, separate the names with comma.

<outFile> : Name of the file into which the output will be written

-h : Prints this syntax help. Cannot be used with other flags.

E.g.: ./getTgtRsrc -hmc myhmc -u ssh\_user1 -m myP5 -vio vioLP1,vioLP2  
-o tgtRsrc

## Flags for the genTgtCfg command

Example A-7 shows how to get the source LPARs' hardware configuration.

*Example: A-7 Shows the flags for the genTgtCfg command*

```
{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./genTgtCfg
```

**SYNTAX:**

```
genTgtCfg.pl -s <srcCfgFile> -m <tgtRsrcFile> -e <eqMapsFile>
-lpar <lparName> -prof <profileName> -o <tgtLPARFile>
[-dedCpu] [-dedEth] [-dedDisk]
```

where

<srcCfgFile> : Name of the file containing the source system info

<tgtRsrcFile> : Name of the file containing the target machine info

<eqMapsFile> : Name of the file containing the capacity equivalence info

<lparName> : Name of the LPAR to be created in the target machine

<profileName> : Profile for the above LPAR

<tgtLPARFile> : Name of the file into which the target LPAR configuration info will be written.

-dedCpu : Use dedicated CPUs for the target LPAR

-dedEth : Use dedicated Ethernet adapters for the target LPAR

```

-dedDisk      : Use dedicated hard disks for the target LPAR
-T : assess user_defined volume groups other than rootvg from the original
client
-h : Prints this syntax help. This flag cannot be used with other flags.
E.g.: ./genTgtCfg -s srcCfg -m tgtSmpRsrc -e eqMaps -lpar LP1 -prof Profile1 -o
tgtLPARcfg

```

---

## Generate the target LPARs profile information

Example A-8 shows how to get the source LPARs' hardware configuration.

*Example: A-8 Shows the generation of the target LPARs profile information*

---

```

{nimmast}:/usr/lpp/bos.sysmgt/nim/methods # ./genTgtCfg -s
/tmp/mig2p5/lpar4cfg.out -m /tmp/mig2p5/tgtrsc.out -e
/usr/lpp/bos.sysmgt/nim/methods/capEqMaps -lpar newlpar -prof normal -o
/tmp/mig2p5/newlparcfg.out

```

### INPUT:

```

Source Configuration file           : /tmp/mig2p5/lpar4cfg.out
Target Machine information file      : /tmp/mig2p5/tgtrsc.out
Capacity equivalence maps file      :
/usr/lpp/bos.sysmgt/nim/methods/capEqMaps
Target LPAR name                     : newlpar
Name of the profile to be created in LPAR: normal
Target LPAR file                     : /tmp/mig2p5/newlparcfg.out
Use dedicated CPUs?                  : No
Use dedicated Ethernet adapters?     : No
Use dedicated Hard disks?            : No

```

-----  
Reading the Source Configure from the given file.....

```

10002
src_mem_util = 99

```

```

Machine Type-model of the source machine = 7040-671
Number of CPUs in the source machine    = 2
Amount of memory in the source machine  = 1024

```

```

Network slots in source machine         = 10/100 Mbps Ethernet PCI
Adapter II (1410ff01)

```

---

```

Reading the target P5 managed system configuration.....

Available processing units = 2.0
Available memory          = 11648

IO slots = 21010002/none/Universal Serial Bus UHC
Spec:21030002/none/Fibre Channel Serial Bus:21040002/none/Empty
slot:21050002/none/Fibre

Channel Serial Bus:21030003/none/Empty slot:21040003/none/PCI 1Gbps
Ethernet Fiber 2-port

Storage IO slots =

Network IO slots = 21040003/none/PCI 1Gbps Ethernet Fiber 2-port

VIO Volume groups = rootvg,VI02,41088

VIO Hard disks =

VIO VLANs      = 1,VI02,30
src_freq = 1000, tgt_freq = 1500
-----
Calculating the resource requirements of target LPAR.....

des_procs = 0.1
des_memory = 1024
New Physical io slots =
-----
Creating the target LPAR configuration file.....

```

## Generated Target LPAR configuration information

Example A-9 shows how to get the source LPARs' hardware configuration.

*Example: A-9 Shows the generated target LPAR configuration information*

```

{nimmast}:/tmp/mig2p5 # cat newlparcfg.out
LPAR_NAME = newlpar
PROFILE_NAME = normal
# PROC_UNITS = <min> <desired> <max>
PROC_UNITS = 0.1 0.1 0.2

```

```
# MEMORY = <min MB> <desired MB> <max MB>
MEMORY = 512 1024 2048
# VIO_VG_INFO = <vgname_src>,<size in MB>,<vgname_vio>,<lpar_name>
#   Where <vgname_src> is the VG name in source machine, and
#       <vgname_vio> is the VG name in VIO server LPAR
VIO_VG_INFO = rootvg,17344,rootvg,VI02
# VIO_VLAN_INFO = <vlan id>,<lpar name>,<slot number>
VIO_VLAN_INFO = 1,VI02,30
```

---

At this point you can either chose to create the LPAR with the **createTgtLPAR** script or if all you required was the sizing information (shown in Example A-9 on the *newlparcfg.out* file) then you can stop now. If you want to automate the whole process from sizing to LPAR creation to NIM installation then you should use the **nim\_move\_up** tool alone.

## Create the target LPAR definition on the POWER5 machine

Example A-10 shows how to get the source LPARs' hardware configuration.

*Example: A-10 Creates the target LPAR definition*

---

```
{nimmast}:/usr/lpp/bos.sysmgmt/nim/methods # createTgtLPAR -m
/tmp/mig2p5/tgtrsc.out -t /tmp/mig2p5/newlparcfg.out
INPUT:
Target Machine configuration file = /tmp/mig2p5/tgtrsc.out
Target LPAR configuration file   = /tmp/mig2p5/newlparcfg.out
Managed system = SARK-520-65C544E
HMC Hostname = hmcsark
User configured for remote ssh to HMC = hscroot
LPAR name: newlpar
Profile name: normal
Proc units: Min=0.6, Desired=1.3, Max=2.0
CPUs: Min=1, Desired=2, Max=2
Memory: Min=3520, Desired=7040, Max=14080
Virtual IO VGs:
rootvg,17344,rootvg,VI02

Virtual IO VLANs:
1,VI02,30

-----
Creating the target LPAR profile with proc units and memory.....
Command:
ssh hscroot@hmcsark mkssyscfg -r lpar -m SARK-520-65C544E -i
```

```
name=newlpar,profile_name=normal,lpar_env=aixlinux,proc_mode=shared,sharing_mod
e=cap,min_proc_units=0.6,desired_proc_units=1.3,max_proc_units=2
```

```
.0,min_procs=1,desired_procs=2,max_procs=2,min_mem=3520,desired_mem=7040,max_me
m=14080,boot_mode=norm
```

```
-----
Discovering the LPAR id assigned to the above LPAR.....
```

```
Command:
```

```
ssh hscroot@hmcsark lssyscfg -r lpar -m SARK-520-65C544E --filter
lpar_names=newlpar -Flpar_id
LPAR ID = 12
```

```
-----
Adding Physical I/O slots to the LPAR profile.....
```

```
.....No physical I/O slots to add to the LPAR profile.
```

```
-----
Adding Virtual I/O to the LPAR profile.....
```

```
-----
Getting the list of existing vscsi devices in the VIO server....
```

```
Command:
```

```
ssh hscroot@hmcsark viosvrcmd -m SARK-520-65C544E -p VIO2 -c "\lsdev
-virtual\""
```

```
-----
Determining the free virtual slot number in the VIO server.....
```

```
Command:
```

```
ssh hscroot@hmcsark lshwres -r virtualio --rsubtype slot -m SARK-520-65C544E
--level slot --filter "lpar_names=VIO2" -Fslot_num
```

```
-----
Creating virtual scsi adapter in the VIO server.....
```

```
Adding the vscsi adapter to the VIO server using DR operation.
```

```
Command = ssh hscroot@hmcsark chhwres -r virtualio --rsubtype scsi -m
SARK-520-65C544E -o a -p VIO2 -s 42 -a
```

```
"adapter_type=server,remote_slot_num=2,remote_lpar_name=newlpar"
```

```
-----
Getting the default and current profile names of VIO server...
```

```
Command:
```

```
ssh hscroot@hmcsark lssyscfg -r lpar -m SARK-520-65C544E --filter
"lpar_names=VIO2" -Fdefault_profile:curr_profile
```

```
-----
Adding the vscsi adapter to the default profile of VIO server....
```

```
Command:
```

```
ssh hscroot@hmcsark chsyscfg -r prof -m SARK-520-65C544E -i
"name=normal,lpar_name=VIO2,virtual_scsi_adapters+=42/server/12/newlpar/2/1"
```

```
-----
Configuring the virtual scsi adapter in the VIO server.....
```

```
Command:
```

```

ssh hscroot@hmcsark viosvr cmd -m SARK-520-65C544E -p VI02 -c "\"cfgdev\""
:-----
Determining the OS-generated vscsi device name for the new adapter....
Command:
ssh hscroot@hmcsark viosvr cmd -m SARK-520-65C544E -p VI02 -c "\"l1sdev
-virtual\""
-----
Creating a logical volume in the VIO server.....
Command:
ssh hscroot@hmcsark viosvr cmd -m SARK-520-65C544E -p VI02 -c "\"mklv -lv
rootvg_12 rootvg 17G\""
rootvg_12
-----
Attaching the above LV to the vscsi device created earlier.....
Command:
ssh hscroot@hmcsark viosvr cmd -m SARK-520-65C544E -p VI02 -c "\"mkvdev -vdev
rootvg_12 -vadapter vhost12 -dev vtscsi12\""
vtscsi12 Available
-----
Creating the virtual scsi adapter for client LPAR.....
Command:
ssh hscroot@hmcsark chsyscfg -r prof -m SARK-520-65C544E -i
name=normal,lpar_name=newlpar,virtual_scsi_adapters+=2/client//VI02/42/1
-----
Creating the virtual ethernet adapters for client LPAR.....
Command:
ssh hscroot@hmcsark chsyscfg -r prof -m SARK-520-65C544E -i
name=normal,lpar_name=newlpar,virtual_eth_adapters+=3/0/1//0/1

```

---

## Flags for the createTgtLPAR command

Example A-11 shows how to get the source LPARs' hardware configuration.

*Example: A-11 Shows the flags for the create TgtLPAR command*

---

```
{nimmast}:/usr/lpp/bos.sysmgmt/nim/methods # ./createTgtLPAR
```

### SYNTAX:

```
createTgtLPAR.pl -m <tgtMachFile> -t <tgtCfgFile> -v <vginfo_file> -T
```

where

<tgtMachFile> : File containing the information about the target machine

<tgtCfgFile> : File containing the description of the target LPAR

<vginfo\_file> : File to which the vhost and slot number of each VG is

written

-T : Create user VGs as well.

-h : Prints this syntax help. Cannot be used with other flags.



E.g.: `./createTgtLPAR -m myP5 -t myLPAR`

---





**B**

# **Automatic provisioning NIM and Tivoli Provisioning Manager (TPM)**

This appendix describes the implementation of NIM as an automated provisioning tool with Tivoli Provisioning Manager.

## TPM Overview

IBM Tivoli Provisioning Manager (TPM) provides the capability to automate the provisioning of servers (real and/or virtual), software, network connections, and storage.

Tivoli Provisioning Manager permits the rapid deployment of resources within your infrastructure. It is an end-to-end automation tool. It captures a customer's existing procedures by linking together their systems management tools and executing them and new processes in a repetitive error-free manner either within or across organizational boundaries.

## Why to use TPM?

TPM helps boost server-to-administrator ratios and reduce human error by automating the manual and repetitive steps required to provision a production server or to update it with the latest fixpack or security patch. By utilizing existing hardware, software, and network devices without rewiring, you can minimize implementation times and achieve a fast return on investment. In addition, IBM Tivoli Provisioning Manager allows you to create, customize, and use automation workflows. Prebuilt workflows provide control and configuration of major vendors' products, while customized workflows can implement your company's data center best practices and procedures.

There is a central Web site called IBM Tivoli Open Process Automation Library where you can check if TPM is providing the workflows to automate your configuration or installation:

<http://catalog.lotus.com/wps/portal/tpm/>

Search for "nim" for example.

## Architecture picture

This is a global architecture overview including the Tivoli Intelligent Orchestrator part. TPM represented by the two following components: the Deployment Engine and the Data Center Model as shown in Figure B-1 on page 643.

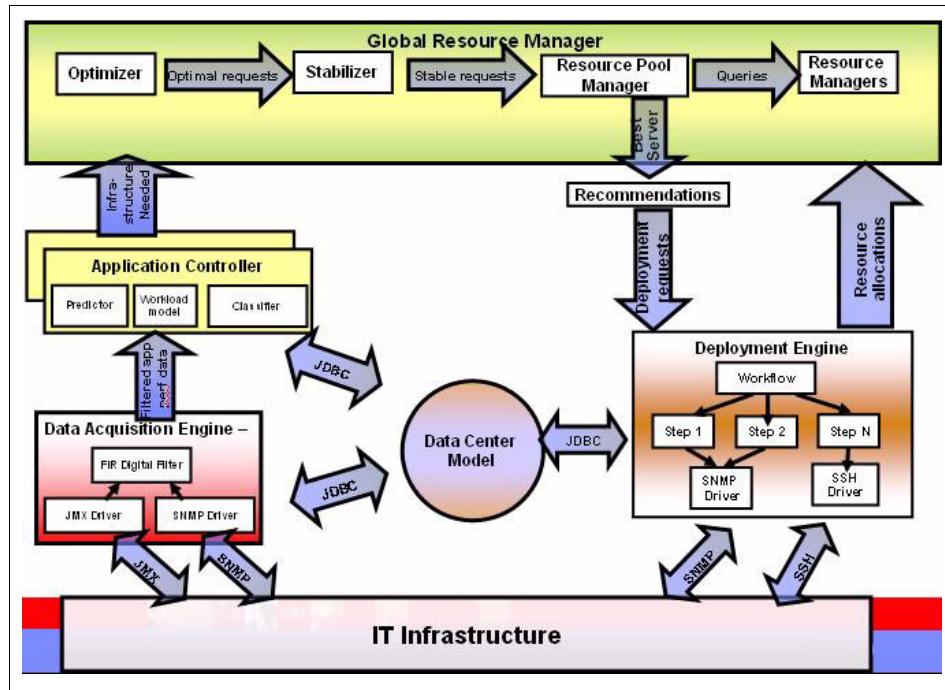


Figure B-1 TPM architecture

## TPM terminology

This section contains TPM terminology used in this appendix.

### Data Center Model (DCM)

It represents physical and logical devices within the infrastructure that can be managed by TPM. Each object has a unique identifier and this ID is used by the workflows during the provisioning mechanism.

### Workflow

It is a usually a best practice to achieve an activity, for instance, installing an operating system. A workflow is composed of transitions which are the steps to execute to complete the activity. A step could be a command, a piece of code or another workflow.

## Drivers

A driver is a package containing several workflows, documentation and scripts. TPM is providing some drivers for IBM @server pSeries platform and AIX 5L which contain all the workflows to automate tasks. For example:

- ▶ pSeries-Server.tcdriver
- ▶ LPAR-cpu.tcdriver
- ▶ Aix-Operating-System.tcdriver
- ▶ NIM.tcdriver
- ▶ AIX-LVM.tcdriver

## Boot server

A server with capabilities to deploy images, install operating system, backup images. The Network Installation Manager (NIM) Master is considered as a boot server.

## Software definition

Represents a piece of software and the information required to install and configure it. It acts as a container that brings together all the information about a piece of software.

## Software installable

An installable file is the actual software package or image file that is distributed and installed on a target system.

## Host platform

A hosting server like a pSeries machine, on which Logical Partition can be created. An ESX Server is also a HostPlatform as it can host Virtual Machines.

## Server template

It contains information on the required characteristics to create a server: CPU, memory, disk, etc. Each value affected to a machine is deducted from the HostPlatform.

## Discovery technology

Some code is provided within TPM to discover automatically information on many types of hardware, for example, the Hardware Management Console (HMC) for the IBM @server pSeries, IBM Director for xSeries®, and NIM objects.

## TPM drivers for NIM

Table B-1 shows all the workflows available for the NIM driver.

Table B-1 TPM drivers available for NIM

File	Description
/repository/nim_ckstatus.sh	Script to check NIM installation status using NIM transition.
/repository/nim_cust.sh	Script to launch a NIM customization operation.
/repository/nim_define_client.sh	Script to define a server as a client to the NIM master.
/repository/nim_install.sh	Script to launch a NIM BOS install.
/repository/nim_mk_clone.sh	Script to create a mksysb.
/repository/inventory_export.pl	Standalone script used to export NIM inventory for population of DCM.
/doc/NIM.html	README.doc
/workflow/Get_NIM_TimeOut.wkf	Workflow to get the NIM timeout from a device or use default.
/workflow/Check_NIM_Install_Status.wkf	Workflow to drive nim_ckstatus.sh script.
/workflow/Install_NIM_Client.wkf	Workflow to drive nim_install.sh script.
/workflow/Customize_a_NIM_Client.wkf	Workflow to drive nim_cust.sh script.
/workflow/NIM_Copy_File_from_Local.wkf	Workflow to copy a local file to a Server.
/workflow/NIM_Remote_Remove_File.wkf	Workflow to remove a file from a Server.
/workflow/Get_NIM_Client_Attributes.wkf	Workflow to get NIM client variable information from a Server.
/workflow/Get_NIM_Software_Attributes.wkf	Workflow to get NIM software variable information from a Software Product or Software Stack.
/workflow/NIM_Create_Clone_Image.wkf	Workflow to call nim_mk_clone.sh
/workflow/Create_Clone_AIX_Image.wkf	Workflow to create a mksysb of a client.
/workflow/Define_NIM_Client.wkf	Workflow to drive nim_define_client.sh script.

File	Description
/workflow/Define_NIM_Client_from_DCM.wkf	Workflow to define a server as a NIM client to a NIM master.
/workflow/AIX_Image_Install.wkf	Workflow to BOS install an AIX server.
/workflow/AIX_Reset_Install.wkf	Workflow to Reset a NIM install.
/workflow/AIX_Software_Install.wkf	Workflow to Install software using the NIM management server onto a server.
/workflow/AIX_Software_Uninstall.wkf	Workflow to uninstall software using the NIM management server from a server.

## Workflow to define a NIM client

This is the workflow to define a NIM client. Look at all the ids passed as parameters. TPM is an object oriented application and provides a flexible way to write automation code for an activity.

The workflow will retrieve the NIM master name & IP address from the Data Center Model (in which you have to define the "NIM master" boot server. Then the workflow will transfer and execute a script "define\_nim\_client.sh" on the NIM master. See Example B-1.

*Example: B-1* Workflow to define a NIM client

---

```
# -----
# Licensed Materials - Property of IBM
# 5724-F75
# (C) Copyright IBM Corp. 2003, 2004
# All Rights Reserved
# US Government Users Restricted Rights -Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#

#Defines a server in the DCM as a NIM client to a specified NIM master
workflow Define_NIM_Client_from_DCM(in NIM_Master_ID, out NIM_name, out
Return_code, in Server_ID, out Stderr) LocaleInsensitive
  var Unmanaged_Network_Interface_ID
  var IP_address
  var Unmanaged_Network_NIC_ID
  var NIM_name_key
  var IF_Name
  var Copied_script_location
```



```

var mac_address
var NIM_masterID_key
var Server_name
var Copied_script_name
NIM_name_key = "NIM.name"
NIM_masterID_key = "NIM.masterID"

NIM_name =
DCMQuery(/Server[@id=$Server_ID]/property[@componentId="5"]/@NIM.name)

java:com.thinkdynamics.kanaha.de.javaplugin.datacentermodel.FindServer(
<null>, <null>, <null>, <null>, <null>, Server_name, <null>, <null>,
Server_ID, Unmanaged_Network_Interface_ID)

java:com.thinkdynamics.kanaha.de.javaplugin.datacentermodel.FindNetwork
Interface(IP_address, <null>, <null>, IF_Name,
Unmanaged_Network_Interface_ID, Unmanaged_Network_NIC_ID, <null>,
<null>)

try

java:com.thinkdynamics.kanaha.de.javaplugin.datacentermodel.FindNic(<nu
ll>, <null>, mac_address, <null>, Unmanaged_Network_NIC_ID, <null>,
<null>, <null>, <null>, <null>)
catchall
endtry

# Replace spaces with '_' in Server_name
var Copy_server_name=Jython[""]

java:com.thinkdynamics.kanaha.de.javaplugin.stringoperations.ReplaceAll
(Server_name, " ", "_", Copy_server_name)

Copied_script_name = Jython[ "nim_define_client_" + Copy_server_name
+ ".sh" ]

NIM_Copy_File_from_Local(NIM_Master_ID, Copied_script_name, "/tmp",
Copied_script_location, "nim_define_client.sh")

Define_NIM_Client(IF_Name, IP_address, mac_address, NIM_Master_ID,
NIM_name, Return_code, Stderr, NIM_name, Copied_script_location,
Server_name)

NIM_Remote_Remove_File(NIM_Master_ID, Copied_script_name, "/tmp")

```

```
#Sets the NIM.masterID variable on the server
DCMInsert parent=DCMQuery(/Server[@id=$Server_ID]/@id) <<EOINSERT
  <property component="DEPLOYMENT_ENGINE" name="NIM.masterID"
value="$NIM_Master_ID" />
EOINSERT

#Sets the NIM_name variable on the server
DCMInsert parent=DCMQuery(/Server[@id=$Server_ID]/@id) <<EOINSERT
  <property component="DEPLOYMENT_ENGINE" name="NIM.name"
value="$NIM_name" />
EOINSERT
```

---

Once you have defined the NIM Client. You can use the AIX\_Image\_Install workflow to install the operating system on the client partition.

## Automated provisioning scenario

In this scenario, we describe the process to create a partition with Tivoli Provisioning Manager, and how the NIM master is used to install the Operating System. It discusses some processes to automate such a procedure.

TPM 3.1 is providing a driver called pSeries-Server.tcdriver. It contains a workflow to create a Logical Partition on POWER4 or on POWER5 technologies.

Before using a workflow within TPM, we have to define the virtual objects that represent the real infrastructure. We create the following objects:

- ▶ A HostPlatform that represents the IBM System p p570. This object is associated with the pSeries-Server.tcdriver.
- ▶ A boot server that represents the NIM master.
- ▶ A software definition for the AIX 5L operating system.
- ▶ A software installable for the AIX 5L V5.3 and information for install as shown in Figure B-2 on page 649.

Key	Component	Value
NIM.bid	Deployment engine	BID_NP_HD0
NIM.fb_script	Deployment engine	ADD_ROUTE
NIM.mksysb	Deployment engine	MK_PBENCH_53ML2_291105
NIM.spot	Deployment engine	SPOT_530_ML02
NIM.Timeout	Deployment engine	12000
OS_Type	Entire system	AIX

Total: 6, displayed: 6.

Figure B-2 Variable in AIX 5L V5.3

## Partition creation

Any object in TPM has a unique identifier called a DCM ID. So our four required objects have each their own ids and they will be provided as inputs to the workflow.

A key notion within TPM is the logical operation as you can see in Figure B-3 on page 649. The workflow is implementing a logical operation called HostPlatform.CreateVirtualServer.

Workflow Name	Logical Device Operation
Odina.pBenchLite_pSeries_Create_Virtual_Server	HostPlatform.CreateVirtualServer
Odina.pBenchLite_pSeries_Destroy_Virtual_Server	HostPlatform.DestroyVirtualServer
pSeries_Expand_Resource_Allocation	HostPlatform.ExpandResourceAllocation
pSeries_Reduce_Resource_Allocation	HostPlatform.ReduceResourceAllocation

Figure B-3 Workflows

We have associated the HostPlatform with our custom workflow so when the logical operation HostPlatform.CreateVirtualServer is called with the HostPlatformID as an input parameter, TPM automatically calls the following implementation:

```
Odina.pBenchLite_pSeries_Create_Virtual_Server(HostPlatformID,
ServerTemplateID, Name, out ServerID) implements
HostPlatform.CreateVirtualServer
```

After the execution of this workflow, we will have a partition created on the machine and a virtual object in TPM (with an id called ServerID) to execute the others workflows.

### Partition installation

Once the partition has been created, we will use the NIM driver to define the client and install the AIX 5L image. The same principle is applied here; the BootServer representing the NIM master is associated with a driver which contains implemented workflows. We have the inputs DCM ID of the NIM master and the ServerID which is the output of the previous creation step. For example:

```
Odina.pBenchLite_Define_NIM_Client_from_DCM(NIM_Master_ID, out
NIM_name, out Return_code, Server_ID, out Stderr)
```

We have the inputs DCM ID of the NIM master which is the BootServerID and the one for the SoftwareInstallable which is the SoftwareStackImageID. The ServerID is the DestinationDeviceID. For example:

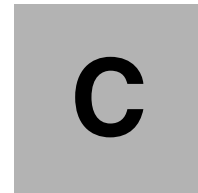
```
Odina.pBenchLite_AIX_Image_Install(BootServerID,
SoftwareStackImageID, DestinationDeviceID) implements
BootServer.InstallImage
```

## Ways to automate (interfaces, cmd line, web services)

TPM has the technical workflow to execute commands within the real infrastructure but you still need to launch the workflows from the TPM Web Interface and fill-in the rights parameters. If you want to automate the execution of workflows, you have several options:

- ▶ Using the TPM provided scheduler.
- ▶ Using Tivoli Intelligent Orchestrator to sense the environment (it gets performance information from the servers running the business applications) and give orders to TPM to execute deployments.
- ▶ Using scripts associated to a custom scheduler that invoke the TPM command line interface.
- ▶ Using TPM exposed Web services to launch workflows from any other application or process.

In the On Demand IN Action Project, we used a web service approach with a reservation portal as a front end.



# **NIM and System i OS installation for an AIX 5L LPAR**

This appendix provides information on how NIM runs on a System i machine.

## IBM System i operating system installation for an AIX 5L LPAR

IBM System i and IBM System p rely on the same hardware platform and processor (POWER 5). The only differences come from the microcode and the HMC.

In fact, once the System i machine has been partitioned, Network Installation Manager (NIM) does not make any difference between an AIX 5L System p partition and an AIX 5L System i partition.

This AIX 5L System i partition can be a NIM master or a NIM client. This partition can also be used as a resource server. Basically there is no difference from NIM perspective between both systems. Figure 7-5 shows an example of a mixed NIM environment with System p machines and System i machines.

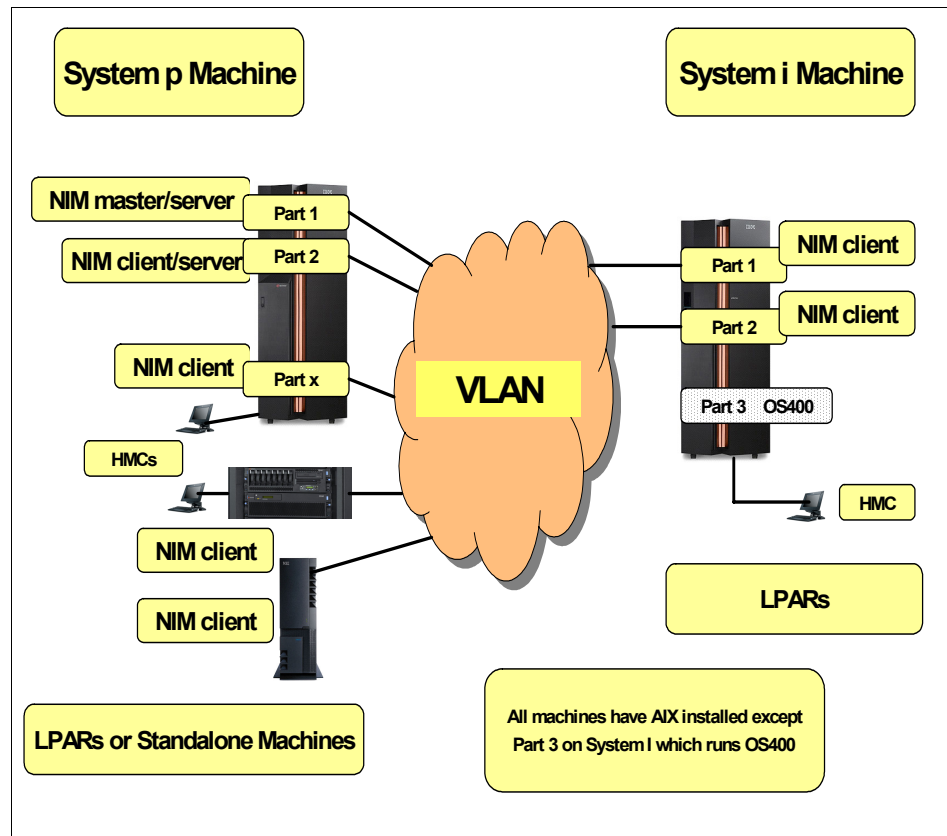


Figure 7-5 NIM environment with System i partitions

Refer to Chapter 3, “Basic configuration of a Network Installation Manager (NIM) environment” on page 49 for details on how to configure and use a NIM environment.







## Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

### Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG247296>

Alternatively, you can go to the IBM Redbooks Web site at:

[ibm.com/redbooks](http://ibm.com/redbooks)

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG247296.

### Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
<b>SG247296.zip</b>	Scripts used in this book

## System requirements for downloading the Web material

The following system configuration is recommended:

**Operating System:** AIX

## How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

# Abbreviations and acronyms

<b>AIX</b>	Advanced Interactive Executive	<b>GUI</b>	Graphical User Interface
<b>ACL</b>	Access Control List	<b>HA</b>	High Availability
<b>APAR</b>	Authorized Program Analysis Report	<b>HACMP</b>	High Availability Cluster Multi-Processing
<b>API</b>	Application Programming Interface	<b>HANIM</b>	High Availability NIM
<b>ARP</b>	Address Resolution Protocol	<b>HBA</b>	Host Bus Adapter
<b>BFF</b>	Backup File Format	<b>HMC</b>	Hardware management Console
<b>BOS</b>	Base Operating System	<b>HTML</b>	HyperText Markup Language
<b>CDE</b>	Common Desktop Environment	<b>HTTP</b>	HyperText Transfer Protocol
<b>CEC</b>	Central Electronic Complex	<b>HTTPS</b>	HTTP-Secure
<b>CHRP</b>	Common Hardware Reference Platform	<b>I/O</b>	Input / Output
<b>CLI</b>	Command Line Interface	<b>IBM</b>	International Business Machines Corporation
<b>CNAME</b>	Cannonical Name	<b>ICMP</b>	Internet Control Messaging Protocol
<b>COSI</b>	Common OS Image	<b>ID</b>	Identification
<b>CPU</b>	Central Processing Units	<b>IDE</b>	Integrated Development Environment
<b>CPUID</b>	CPU Identification	<b>IP</b>	Internet Protocol
<b>CSM</b>	Cluster Systems Management	<b>IPL</b>	Initial Program Loading
<b>CSP</b>	Concluding Service Pack	<b>ITSO</b>	International Technical Support Organization
<b>CWD</b>	Current Working Directory	<b>JFS</b>	Journalled File System
<b>DHCP</b>	Dynamic Host Configuration Program	<b>LAN</b>	Local Area Network
<b>DLPAR</b>	Dynamic LPAR	<b>LDAP</b>	Lightweight Directory Access Protocol
<b>DMZ</b>	Demilitarized Zone	<b>LED</b>	Light Emmiting Diode
<b>DNP</b>	Dynamic Node Priority	<b>LIC</b>	Licensed Internal Code
<b>DNS</b>	Dynamic Name Service	<b>LPAR</b>	Logycal Partition
<b>FC</b>	Fibre Channel	<b>LPP</b>	Licensed Program Product
<b>FQDN</b>	Fully Qualified Domain Name	<b>LTG</b>	Logical Track Group
<b>FTP</b>	File Transfer Protocol	<b>LUN</b>	Logical Unit Number
<b>GSA</b>	Global Storage Architecture	<b>LV</b>	Logical Volume

<b>LVD</b>	Low-Voltage Differential (SCSI)	<b>ROM</b>	Read-Only Memory
<b>LVM</b>	Logical Volume Manager	<b>RPC</b>	Remote Procedure Call
<b>MAC</b>	Media Access Card	<b>RPD</b>	RSCT Peer Domain
<b>ML</b>	Maintenance Level	<b>RPM</b>	RPM Package Manager
<b>MPIO</b>	Multi-Path I/O	<b>RSA</b>	Remote Supervisor Adapter
<b>MTU</b>	Maximum Transfer Unit	<b>RSCT</b>	Reliable Scalabel Clustering Technology
<b>NFS</b>	Network File System	<b>RSH</b>	Remote Shell
<b>NIC</b>	Network Interface Card	<b>SAN</b>	Storage Area Network
<b>NIM</b>	Network Install Manager	<b>SCSI</b>	Small Computer Systems Interface
<b>NIMSH</b>	NIM Service Handler	<b>SDD</b>	Sub-system Device Driver
<b>NIS</b>	Network Information Service	<b>SDDPCM</b>	SDD Path Control Module
<b>NTP</b>	Network Time Protocol	<b>SDK</b>	Systems Development Kit
<b>ODM</b>	Object Data Manager	<b>SLES</b>	SUSE Linux Enterprise Server
<b>PC</b>	Personal Computer	<b>SME</b>	Systems management Environment
<b>PCI</b>	Peripheral Component Interconnect	<b>SMIT</b>	System Management Interface Tool
<b>PCI-X</b>	PCI-Extended	<b>SMS</b>	System management Services
<b>PCM</b>	Path Control Module	<b>SOE</b>	Standard Operating Environment
<b>PID</b>	Process ID	<b>SPOC</b>	Single Point Of Control
<b>PP</b>	Physical Partition	<b>SPOT</b>	Shared Product Object Tree
<b>PSSP</b>	Parallel Systems Support Program	<b>SRC</b>	System resource Controller
<b>PTF</b>	Program Temporary Fix	<b>SRPM</b>	Source RPM
<b>PVID</b>	Physical Volume ID	<b>SSH</b>	Secure Shell
<b>PXE</b>	Pre-boot eXecution Environmnet	<b>SSL</b>	Secure Socket Layer
<b>RAID</b>	Redundant Array of Independent Disks	<b>SUMA</b>	Service Update Management Assistant
<b>RAM</b>	Random Access Memory	<b>TCB</b>	Trusted Computing Base
<b>RAS</b>	Reliabilty Availability Serviceability	<b>TCP</b>	Transmission Control Protocol
<b>RFC</b>	Request For Comment	<b>TCP/IP</b>	Transmission Control Protocol / Internet Protocol
<b>RHEL</b>	RedHat Enterprise Linux	<b>TFTP</b>	Trivial File Transfer Protocol
<b>RM</b>	Resource Manager	<b>TID</b>	Thread ID
<b>RMC</b>	Resource Monitoring and Control		

	<b>TL</b>	Technology Level
	<b>TOC</b>	Table Of Contents
	<b>TPM</b>	Tivoli Provisioning Manager
	<b>TPS</b>	Transactions Processing System
	<b>TSM</b>	Tivoli Storage Manager
	<b>UDP</b>	Universal Datagram Protocol
	<b>URL</b>	Universal Resource Locator
	<b>VG</b>	Volume Group
	<b>VIO</b>	Virtual I/O
	<b>VIOS</b>	VIO Server
	<b>VIPA</b>	Virtual IP Address
	<b>VLAN</b>	Virtual LAN
	<b>VLSI</b>	Very Large Scale Integration
	<b>VSCSI</b>	Virtual SCSI
	<b>XML</b>	Extended Markup Language



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 662. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Linux Applications on pSeries redbook*, SG24-6033-01
- ▶ AIX 5L Practical Performance Tools and Tuning Guide, SG24-6478-00
- ▶ Advanced POWER Virtualization on IBM System p5, SG24-7940-01

## Other publications

These publications are also relevant as further information sources:

- ▶ "AIX 5L V5.3 Installation and Migration, SC23-4887-03"

## Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ SUMA is a complement to the UNIX servers product family portion of the IBM Support Fix Central Web site located at:  
<http://www.ibm.com/eserver/support/fixes/>
- ▶ RPM Package manager Web site:  
<http://www.rpm.org/>
- ▶ AIX 5.3 TL4 Release Notes  
[http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304\\_base\\_relnotes.pdf](http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.resources/RELNOTES/5304_base_relnotes.pdf)
- ▶ VIO Server documentation Web site  
<http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/topic/iphb1/iphb1kickoff.htm>

- ▶ SUMA Whitepaper  
<http://www.ibm.com/servers/aix/whitepapers/suma.html>
- ▶ IBM AIX 5L Service Strategy Web site:  
[http://www.ibm.com/servers/eserver/support/unixservers/aix\\_service\\_strategy.html](http://www.ibm.com/servers/eserver/support/unixservers/aix_service_strategy.html)
- ▶ CSM documentation Web site  
<http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.help.csm.doc/csm.html>

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)



# Index

## A

adapter definition file 431  
 AIX migration 180  
 alog 68  
 alt\_disk\_install 169, 261  
 alternate boot image 280  
 alternate disk migration 123, 165, 169, 261, 266, 276  
 Alternate NIM master 42  
 alternate NIM master 125, 128, 130, 133–134, 308, 312, 318, 397, 414  
 alternate\_disk\_install 131  
 alternate\_master 11, 13  
 APAR 321, 350  
 attribute 424  
 Authorized Program Analysis Report 350  
 availability 44, 48

## B

backup 2–4, 460  
 backup file 52  
 Backup File Format (BFF) 360  
 basic resources 17, 30–32  
 BFF 20–21, 123, 359  
 BFF package 364  
 bffcreate 69, 71, 115  
 boot file 65–66, 68  
 boot image 34  
 BOOTP 65, 82, 85, 92, 284–286, 448  
 bootp 49, 66, 446  
 BOOTP client 448  
 BOOTP server 448  
 bootpd 64–66  
 BOS install 17, 33, 42  
 BOS installation 199, 423, 476, 482, 484  
 bos rte install 50  
 BOS run-time environment 278  
 bosinst.data 128, 199, 204, 207, 211–213, 217, 229, 395, 474  
 bosinst\_data 11–12, 15–17, 23, 46–47, 112, 116, 118, 120–121  
 BOSINST\_DEBUG 199  
 Bundles 375

bundles 359

## C

cdmount 70  
 check 422  
 class name 13  
 classes 11–12, 18  
 Client 10, 30  
 client CPUID 434  
 clients 8, 10, 26, 28–29, 36, 40, 42, 44–45, 47  
 cloning 263, 485  
 Cluster Systems Management (CSM) 2, 4  
 command  
   /usr/sbin/rsct/install/bin/cfgct 471  
   /usr/sbin/rsct/install/bin/uncfgct -n 471  
 alt\_disk\_install 159, 281  
 alt\_disk\_install -x altinst\_rootvg 284  
 backupios 392  
 bicheck 212  
 blvset 170  
 bootlist 167, 188, 291  
 bootptodhcp 288  
 bosboot 280  
 cfgdev 386  
 cfgmgr 388  
 chmod 126  
 chresponse 410  
 claddserv 312  
 clfindres 317  
 clRGmove 316  
 compare\_report 325, 342, 344, 347  
 cp 371  
 cthactrl 406–407  
 cthatsctrl 406  
 cthatstune 406  
 ctskeygen 488  
 ctsthl 408, 486  
 dig 323  
 exportfs 305  
 gencopy 39  
 geninstall 39, 359  
 geninv 344  
 grep 357

host 323  
 installios 391  
 installp 21, 126, 360, 365  
 instfix 342  
 inutoc 126  
 lppchk 258, 474  
 lppmgr 334  
 lscfg 287  
 lscmg 406  
 lscosi 303–304  
 lspp 342  
 lsnim 155, 161, 453  
 lsnim -a info 192  
 lsnim -c groups 16  
 lsnim -c machines 12  
 lsnim -c networks 14  
 lsnim -l 13  
 lsnim -t spot 19  
 lsrpdomain 400  
 lsrpnode 401  
 lsrsrc 402, 486  
 lsrsrs 317  
 lssrc 317  
 lsts 305  
 lsvg 172  
 mkcondition 410  
 mkcondresp 412  
 mkcosi 302–303  
 mkininstallp 360  
 mkszfile 479  
 mkts 304  
 netstat 450  
 nfs0 457  
 nim 39, 224  
 nim\_clients\_setup 42, 119  
 nim\_master\_setup 41, 50, 114  
 nim\_move\_up 43, 206, 218, 223  
 nim\_move\_up -r 245  
 nimadapters 423, 429, 431  
 nimadm 219  
 nimclient 433, 443  
 nimclient -C 436  
 nimclient -P 444  
 nimclient -p 444  
 nimconfig -c 440  
 nimdef 119  
 niminit 268  
 niminv 325  
 nimquery 434  
 nslookup 323  
 oslevel 342, 474, 490  
 oslevel -r 20  
 preprnode 400  
 restore 365  
 restvg 474  
 rmcosi 304  
 rmts 307  
 rpm 349, 369  
 rpmbuild 369  
 savevg 473  
 script 170, 203  
 showled 199  
 snap 155, 170, 172  
 startcondresp 412  
 startprdomain 400  
 stopprnode 413  
 suma 321, 327, 335  
 vmstat 233  
 wsm 96  
 commands  
   lsnim 464  
   mount 427  
   nimadapters 423–424  
   nimclient 433  
 common OS image 123, 300, 302  
 Concluding Service Pack 488  
 Concluding Service Pack (CSP) 351  
 Concluding Service Packs 51  
 console 10, 16, 23, 36, 468  
 control\_flow stanza 199  
 COSI 123, 299  
   operations 302  
 CPU utilization statistics 235  
 cron 132, 157, 175, 196, 321, 329  
 crontab 132, 330–331, 333  
 cryptographic authentication 441–442  
 CSP 51–54, 351  
 Cstate 13, 35–36, 193, 214, 256–257  
 ctrl\_hosts 139  
 customization 259  
  
**D**  
 daemon  
   dhcpsd 291  
   inetd 286  
   nim 449  
   nimesis 445

- nimreg 449
- NIMSH 433
- nimsh 434
- rshd 263, 434, 460
- dataless 8, 11, 29, 77, 96, 125, 299–302, 304–305, 395
  - operations 302
- dataless machine 300
- debug output 199
- Device drivers 17
- DHCP 285–286
- DHCP server 285, 290
- diagnostic 488
- directory
  - /ftpboot 179, 469
  - /usr/es/sbin/cluster/utilities 312
  - /usr/lpp/bos.sysmgmt/nim/methods 433
  - RPMS/ppc 138
- Disaster recovery 3
- diskless 8, 11, 29, 77, 96, 125, 299–302, 304, 306, 395
  - operations 302
- diskless machine 300
- DNS 33, 485
- DVD 52, 68–70, 72, 82, 90, 98

**E**

- Etherchannel 48
- Ethernet 2, 471
- EXISTING\_SYSTEM\_OVERWRITE 477
- EZ NIM 30, 114

**F**

- Fibre Channel 245, 471
- fibre channel 48
- file
  - .toc 422
  - /etc/bootptab 288, 445, 469
  - /etc/ct\_node\_id 471, 487
  - /etc/dhcpsd.cnf 285, 288
  - /etc/environment 443
  - /etc/exports 446
  - /etc/filesystems 284
  - /etc/group 263
  - /etc/hosts 223
  - /etc/inetd.conf 286
  - /etc/irs.conf 311
  - /etc/motd 175

- /etc/netsvc.conf 311
- /etc/niminfo 434, 445, 454, 456
- /etc/passwd 263
- /etc/security/passwd 263
- /etc/sendmail.cf 175
- /etc/tunables/nextboot 172, 263
- /usr/lpp/bosinst/bosinst.template.README 476
- /usr/sbin/cluster/netmon.cf 311
- /var/adm/ras/nimsh.log 434
- /var/ct/cfg/ct\_has.thl 486
- /var/ct/cfg/ctrmc.acls 471, 486–487
- ~/.rhosts 268, 460, 483
- bosinst.data 128
- bosint.data 474
- ct\_has.pkf 488
- ct\_has.qkf 488
- image.data 474
- secondary adapter rules 424
- fileset
  - bos.alt\_disk\_install.rte 250
- filesets 8, 17, 20–22, 27, 29–30, 36, 39
- Firewall 459
- firewall 433, 443, 445
- firstboot script 128
- fixes 8, 39
- FTP 321

**G**

- Groups class 12, 16

**H**

- HA NIM 123–124
- HACMP 123, 165, 169–170, 308–313, 315, 317–319
  - application server 312
  - DNP 318
  - Dynamic Node Priority 318
  - resource group 312
- HACMP configuration 309, 312
- HANIM 123, 155, 308, 314
- hardlink 75
- Hardware Management Console (HMC) 155
- High Availability NIM 155
- High Availability Cluster Multi-Processing 308
- HMC 5, 8, 10, 180, 192, 203, 223, 228, 486
- HMC command
  - vtmenu 200
- hostname 486

HTTP 321  
 HTTPS 321  
 hypervisor 51

**I**

IBM System p 3–5  
 image.data 128, 160, 176, 204, 249, 251, 275, 353,  
 359, 392, 474  
 image\_data 11, 17, 46–47  
 inetd 64–68  
 install 2–4  
 installation method 23, 33  
 installing 1  
 installp 61, 69–70, 72, 74  
 installp bundle 461  
 installp\_bundle 131, 375–376  
 interim fix 351  
 IP label 32  
 irs.conf 63–64

**K**

kernel 34, 460

**L**

LDAP 485  
 LED code 472  
 LED code 605 472  
 Linux 50  
 Linux clients 138  
 Linux distribution 294  
 linux\_source 138–140  
 local disk caching 272  
 LPAR 2  
 lpp\_source 11–12, 15–16, 18, 20–22, 25–27,  
 30–31, 35–37, 39–41, 43–44, 46, 52–54, 56–57, 61,  
 68–72, 81, 102–103, 111–112, 115–116, 118,  
 120–121, 126–127, 130, 138, 140, 145, 147–148,  
 150, 156, 162, 178, 180, 214, 218–220, 224, 226,  
 228, 231–233, 236, 248–249, 254, 256, 258–259,  
 262, 266, 273, 276, 303, 310–311, 324–325, 334,  
 336–342, 346–347, 359, 367–368, 373–374, 376,  
 399, 418, 422  
 lppmgr 37, 39–40, 420  
 lsnim 56, 59–60, 70–71, 76, 78–79, 120–121  
 lssrc 64  
 LUN 189

**M**

MAC address 287  
 mac\_group 11–12, 16  
 machines 8–14, 16–17, 23, 26–28, 35, 50, 54–55,  
 57, 59–60, 62, 77–79, 96, 114, 117, 119–121  
 Machines class 12  
 maintain 2–4  
 maintenance 488  
 Maintenance Level 321  
 maintenance level 46  
 manual network boot 215  
 Master 10, 15, 26–28, 30, 48  
 master 8–12, 16, 19, 21–23, 26–32, 35–37, 40–42,  
 44–45, 47–48, 418  
 memory utilization statistics 235  
 microcode 154  
 micro-partition 210  
 Micro-partitioning 5  
 Migration 192  
 minimum AIX image 10  
 mksysb 3, 11, 15, 20, 22–23, 25, 27–28, 33, 39–43,  
 46–47, 50–52, 54–55, 68, 74, 81, 93, 114–118,  
 120–121, 123, 128, 146, 148–153, 155, 159,  
 164–166, 169, 174, 197–198, 206–207, 209,  
 211–219, 221–222, 224–225, 247–248, 250–251,  
 253, 255–256, 261, 264, 271, 353–358, 391–395,  
 472  
 mksysb image 460–462, 473  
 mksysb installation 222  
 mksysb migration 43, 206, 213, 215, 470  
 ML 2  
 Mstate 13, 35, 193

**N**

Name resolution 26  
 naming convention 9, 46–47, 51–55  
 net\_addr 56, 117  
 netboot 291  
 Netmask 55  
 netname 57–58  
 netsvc.conf 63  
 network boot 3, 467  
 network install 446  
 Networks class 12, 14  
 NFS 17–18, 27, 448  
 NFS reserved ports 450–451  
   disabling 456  
   enabling 453

- NFS server 457
  - NFS server portcheck 458
  - nfs\_reserved\_port 453, 456
  - NIM 1–5
  - NIM client 10, 32–33, 45
    - maintenance 488
  - NIM client installation 83
  - NIM client migration 170
  - NIM database 11–12, 14, 18, 49, 62, 75, 77, 178, 449
  - nim group 2
  - NIM install 446
  - NIM master takeover 124
  - NIM methods
    - /usr/lpp/bos.sysmgmt/nim/methods/c\_cfgadptrs 423
  - NIM migration 470
  - NIM name 15
  - NIM network 55–56, 77–78, 100
  - NIM operation 434, 459
    - bos\_inst 220
    - change 456
    - check 422
    - COSI operations 302
    - customization 432
    - dataless client 302
    - installing RPM packages 373
    - nim\_move\_up 43
    - nimadm 283
  - Nim operation
    - diskless client 302
  - NIM operations
    - nim -o bos\_inst 423, 425
    - nim -o cust 423, 425
  - NIM resource 9, 15
    - BOSINST\_DATA 224
    - bosinst\_data 199, 214, 460, 476
    - deallocating 289
    - exclude\_files 224
    - image\_data 482
    - installp\_bundle 376
    - mksysb 213, 461, 463, 466
    - SPOT 214, 461
  - NIM server 2
  - NIM synchronization 124, 126–128, 130–134
  - NIM takeover 135, 410
  - NIM terminology 7
  - NIM update 419
  - nim\_bosinst 82–83
  - NIM\_MKSYSB\_SUBDIRS 52, 54–55
  - nim\_move\_up 123, 153, 206, 217–227, 229, 231, 233, 235–236, 245–249, 254, 256–261
  - nim\_script 52, 57, 59–60, 116–117, 121
  - nimadm 261–264, 274
    - Local Disk Caching 263
  - nimadm migration 266
  - nimclient 60
  - nimconfig 57–59, 61, 70
  - nimesis 57–58, 60, 77, 115
  - niminit 58, 61–62
  - nimlog 68
  - NIMSH 433, 435–436, 441–442
    - cryptographic authentication 435
    - NIM Service Handler 433
  - nimsh 42, 44
  - NIMSH secondary port 443
  - Nstate 14
  - NTP 485
- ## O
- Object Data Management 11
  - Object Data Manager 60
  - ODM 11, 60, 278
  - OpenSSH 139, 218, 222
  - OpenSSL 139, 433, 436–438
  - OS\_install 49–50, 123, 138–144
  - os\_resources 139
  - osinstall\_config 139, 144
  - oslevel 23, 28, 36, 53
- ## P
- pax 70–71, 75
  - port mapper 448
  - portmapper 459
  - ports 446
  - Pre-boot eXecution Environment 285
  - prev\_state 56, 70, 78–79, 117–118, 121
  - Protocols 447
  - protocols 446
  - PTF 36, 321, 350
  - public key 439
  - pull 10–11, 23, 34
  - pull mode 10, 35
  - pull operation 442
    - disabling 445
  - push installation 482–483
  - push mode 10

push operation 442  
   disabling 444  
 PXE 284, 294

## R

ramdisk 17, 285  
 RECOVER\_DEVICES 477  
 recovering devices 472  
 registration ports 442  
 remote command execution 433  
 Remote IPL 181  
 Remote Procedure Call 448  
 Remote shell 447  
 res\_group 11, 16, 119–120  
 resource  
   mksysb 465  
   SPOT 465  
 Resource server 10  
 resource server 8–10, 15, 27, 145–151, 300, 306,  
 462  
 resource\_servers 139  
 resources  
   adapter\_def 423, 430  
 resources class 18  
 restore process 468  
 RMC 471, 487  
   ACL 487  
   security files 488  
   trusted host list file 488  
 rootvg 23, 28, 41, 44, 171  
 rootvg cloning 176  
 RPM 20–21, 39, 123, 359, 369  
 RSCT 310, 471  
 RSCT Peer Domain 123  
 rshd 433  
 Rstate 16, 19, 21–23, 59–60, 70, 75, 117–118  
 rte 22, 27, 33, 44, 50, 52, 61, 68, 73–74, 81, 83  
 rte installation 486  
 Run Time Environment 33

## S

SAN 213, 485  
 SAN disks (LUNs) 486  
 script 11, 17, 22, 25, 46–47  
   nim\_clients\_setup 115  
   pre\_migration 156  
 SCSI 210, 245, 471–472  
 SDD 154, 170, 195

SDDPCM 154, 170  
 secondary adapter 424–425, 427  
 secondary adapter attribute 426  
 secondary adapter definitions 429  
 secondary adapter stanza 431  
 Secondary adapter support 423  
 server farm 1  
 service console 220  
 Service Pack (SP) 351  
 Service Update Management Assistant 123, 321  
 Shared Ethernet Adapter (SEA) 378  
 Shared Ethernet adapter (SEA) 388  
 Shared Virtual Ethernet 388  
 show\_progress 43  
 simage 53  
 simages 21–22, 31, 39, 70–71, 118  
 SMIT 9, 22, 29, 32, 37–41  
 SMIT fastpath  
   smit nim\_perms 444  
   smitty clstart 315  
   smitty nim\_adapt\_def 430  
   smitty nim\_bosinst 484  
   smitty nim\_chmac 452  
   smitty nim\_client\_inst 438  
   smitty nim\_config\_services 435  
   smitty nim\_global\_nfs 451, 457  
   smitty nim\_mac 483  
   smitty nim\_mkres 465, 475  
   smitty nim\_ssl 439  
   smitty nim\_switch\_master 454, 483  
   smitty suma\_config\_base 326  
   smitty suma\_task\_defaults 327  
   smitty tscosi\_client 301  
 SMS 10, 36, 181, 188  
 SMS menu 82–83, 91  
 snm 56, 117  
 software maintenance 2, 422  
 source 16, 27, 29, 36, 40  
 SPOT 16–20, 22–23, 29–31, 33–35, 40–41, 43, 46,  
 52, 54, 61, 72, 74–76, 82–83, 103, 105, 110–111,  
 115, 305, 418, 464  
 spot 11–12, 15, 18–19, 27, 29, 31–32, 35  
 SPOT copy 44  
 SSH keys 228  
 SSL authentication 439  
 SSL certificates 436  
 SSL directory structure 437  
 standalone 8, 10–13, 32, 35, 60–61, 77–80, 82,  
 119–121

SUMA 123, 310, 324  
 SUMA tasks 328  
 SYSLOG 66, 68  
 system backup 3  
 System Resource Controller 433  
 systems administrators 1

## T

takover 314  
 tape 52  
 TCB 268, 270  
 TCP socket 447  
 TCP/IP 460  
 Technology Level 36, 321, 488  
 technology levels 2  
 TFTP 65, 67, 82, 92, 285, 448, 459  
 tftp 292, 446  
 tftpboot 51, 57, 59, 65–68, 92–93, 115–117, 122  
 tftpd 64, 66–68  
 thin server 300, 304  
 thin server command  
   chcosi 301  
   cpcosi 301  
   dbts 301  
   lscosi 301  
   lsts 301  
   mkcosi 301  
   mkts 301  
   rmcosi 301  
   rmts 301  
   swts 301  
 thin servers 29  
 TIME\_WAIT 443  
 Tivoli Storage Manager 155, 174  
 TL 2  
 toc 71  
 TSM 174

## U

UDP 448  
 upgrade 2–3  
 upgrading 1  
 user volume groups 28

## V

VIO 210, 377  
 VIO client 379, 390

VIO server 5, 221–223, 245, 378–379  
 Virtual Ethernet 385  
 virtual host device 386  
 Virtual I/O 221  
 virtual I/O 2  
 virtual I/O resources 245  
 Virtual optical device 387  
 Virtual SCSI 378–379, 385  
 virtual SCSI adapter 245  
 Virtual SCSI device 386  
 Virtual shared ethernet 378  
 Virtual Target Device mapping (VTD) 386  
 VLAN 51  
 volume group structure 174  
 Volume groups backups 28  
 vterm 192

## W

Web-Based System Management 444  
 WebSM 96, 109  
 Websphere 51  
 wget 349

## Z

zoning 471, 485









Draft Document for Review March 13, 2007 6:19 pm

# NIM from A to Z in AIX 5L



**Redbooks**

**Configuring NIM  
illustrated**

**Sample best  
practices**

**NIM case scenarios  
included**

With LPARs and micro-partitioning, NIM is more important than ever. I believe that most of our bigger and more significant customers are using it or will soon be doing so, and the existing Redbook is woefully out of date. ( NIM: From A to Z in AIX 4.3 SG24-5524-00 published 14 February 2000 )

We think the NIM A-Z needs to retire and be replaced by something that really reflects the current systems and installation challenges. Not an update but a rewrite of the redbook that would provide clients with task based information on NIM setup and administration for a variety of scenarios. One obvious inclusion would be NIM in an LPAR environment. We are sure the NIM setup is a major pain point for a lot of clients, providing them with helpful technical documentation increases satisfaction and reduces technical support call rates.

The field is proposing a complete re-write to NIM on AIX 5L version 5.3.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:  
[ibm.com/redbooks](http://ibm.com/redbooks)**

SG24-7296-00

ISBN

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine:fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.  
Draft Document for Review March 13, 2007 6:19 pm

**7296spine.fm 673**



**Redbooks**

# NIM from A to Z in AIX 5L

(1.5" spine)  
1.5"<->1.998"  
789 <->1051 pages



**Redbooks**

# NIM from A to Z in AIX 5L

(1.0" spine)  
0.875"<->1.498"  
460 <-> 788 pages



**Redbooks**

# NIM from A to Z in AIX 5L

(0.5" spine)  
0.475"<->0.875"  
250 <-> 459 pages



**Redbooks**

# NIM from A to Z in AIX 5L

(0.2" spine)  
0.17"<->0.473"  
90 <-> 249 pages

(0.1" spine)

0.1"<->0.169"

53 <-> 89 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine:fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.  
Draft Document for Review March 13, 2007 6:19 pm

**7296spine.fm 674**



**Redbooks**

# NIM from A to Z in AIX 5L

(2.5" spine)  
2.5"<->mm.n" **1315<-> mmn pages**



**Redbooks**

# NIM from A to Z in AIX 5L

(2.0" spine)  
2.0"<->2.498" **1052 <-> 1314 pages**