IBM AIX Enhancements and Modernization

Navdeep Dhaliwal
Ahmed Mashhour
Armin Röll
Liviu Rosca
Note: Before using this information and the product it supports, read the information in “Notices” on page xv.
## Contents

**Figures** ................................................................. vii

**Tables** ................................................................. ix

**Examples** .............................................................. xi

**Notices** ................................................................. xv
**Trademarks** ............................................................ xvi

**Preface** ................................................................. xvii

**Authors** ................................................................. xvii

**Now you can become a published author, too!** ......................... xviii

**Comments welcome** .................................................. xvi

**Stay connected to IBM Redbooks** .................................... xviii

**Chapter 1. General enhancements** .................................. 1

1.1 Live Kernel Update .................................................. 2
1.1.1 Live Kernel Update concepts .................................... 2
1.1.2 Live Kernel Update modes ......................................... 3
1.1.3 Live Kernel Update procedure .................................... 3
1.1.4 Live Kernel Update management types ......................... 4
1.1.5 Live Kernel Update methods .................................... 4

1.2 Server Flash Caching ................................................ 7
1.2.1 Flash Caching Concepts ........................................... 7
1.2.2 Implementation modes ............................................ 8

1.3 Multipath I/O .......................................................... 10
1.3.1 AIX Path Control Module (AIXPCM) ............................ 10
1.3.2 Subsystem Device Path Control Module (SDDPCM) ............ 16

1.4 iSCSI software initiator ............................................. 16
1.4.1 iSCSI overview ..................................................... 16
1.4.2 Configuring the initiator ......................................... 17

1.5 Network Installation Manager ....................................... 19
1.5.1 Object Classes ..................................................... 19
1.5.2 HTTP Service ....................................................... 21
1.5.3 Secure Socket Layer (SSL) ....................................... 22
1.5.4 Live Kernel Update ............................................... 22
1.5.5 nimadm support for MultiBOS environments ................... 28

1.6 Logical Volume Manager ............................................ 28
1.6.1 LVM reclamation support ........................................ 28

1.7 JFS2 ................................................................. 29
1.7.1 JFS2 defragger ..................................................... 29
1.7.2 JFS2 space reclaim ............................................... 30

1.8 Multiple alternative disk clones .................................. 31
1.8.1 Cloning concepts ................................................ 31

1.9 Active Memory Expansion ........................................... 36

1.10 Unicode support .................................................... 37
1.11 AIX Toolbox for Linux Applications ............................... 38

**Chapter 2. Security enhancements** ................................ 39
Chapter 6. Disaster Recovery and High Availability

6.1 IBM VM Recovery Manager for Power ............................................ 102
   6.1.1 IBM VM Recovery Manager versions and life cycle ..................... 102
   6.1.2 IBM VM Recovery Manager HA components .................................. 103
   6.1.3 IBM VM Recovery Manager DR components .................................. 104

6.2 PowerHA .......................................................... 107
   6.2.1 PowerHA editions ............................................................. 107
   6.2.2 PowerHA versions and life cycle ............................................ 107
   6.2.3 AIX requirements for various PowerHA levels ............................. 108
   6.2.4 PowerHA licensing ........................................................... 108

Chapter 7. AIX fundamentals .................................................. 111

7.1 Logical Volume Manager .................................................... 112
   7.1.1 Introduction to the LVM ....................................................... 112
   7.1.2 LVM components .............................................................. 113
   7.1.3 Principles to optimize LVM disks .......................................... 115
   7.1.4 LVM strategies of various storage types ................................. 116
   7.1.5 LVM configuration ............................................................. 117

7.2 AIX JFS2 .............................................................. 124
   7.2.1 JFS2 functions ................................................................. 124
   7.2.2 JFS2 features ................................................................. 125

7.3 Role Based Access Control .................................................. 125
   7.3.1 RBAC elements ............................................................... 126
   7.3.2 RBAC Kernel Security Table and exact files .............................. 128
   7.3.3 Customizing RBAC role for certain tasks .................................. 128
   7.3.4 Domain RBAC ................................................................. 132
   7.3.5 Domain RBAC implementation scenario .................................... 132

7.4 Encrypted file systems ....................................................... 135
   7.4.1 EFS commands used .......................................................... 136
   7.4.2 Sample scenario of EFS ...................................................... 136
   7.4.3 Integrate EFS keystore with OpenSSH key authentication .............. 139

7.5 AIXpert and PowerSC integration .......................................... 143
   7.5.1 Using AIXpert ................................................................. 143
   7.5.2 Using AIXpert to generate a compliance report by pscxpert PowerSC .. 145
<table>
<thead>
<tr>
<th>Section Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6 AIX Auditing subsystem and AHAFS</td>
<td>146</td>
</tr>
<tr>
<td>7.6.1 AIX Auditing subsystem</td>
<td>146</td>
</tr>
<tr>
<td>7.6.2 Implementing AIX audit subsystem for exact event</td>
<td>147</td>
</tr>
<tr>
<td>7.6.3 AHAFS</td>
<td>149</td>
</tr>
<tr>
<td>7.7 MultiBOS</td>
<td>152</td>
</tr>
<tr>
<td>7.7.1 Standby BOS setup</td>
<td>152</td>
</tr>
</tbody>
</table>

**Related publications**

- IBM Redbooks                                           | 161  |
- Online resources                                        | 161  |
- Help from IBM                                           | 164  |
Figures

1-1  Mind blown emoji .......................................................... 37
4-1  PowerVC integration into of HMC-managed Power system environments .... 54
4-2  PowerVC integration into NovaLink-managed Power system environments 55
4-3  Creating a cloud connection ............................................ 58
4-4  Selecting your cloud provider for cloud connection .................... 59
4-5  Entering configuration details for OpenStack connection ................ 59
4-6  Saving your cloud connection .......................................... 60
4-7  Creating a template from scratch in Cloud Automation Manager ........ 61
4-8  Entering new template metadata ...................................... 61
4-9  Opening the in-line editor page ....................................... 62
4-10 Adding your Terraform template ...................................... 63
4-11 Navigating to parameter page ........................................ 65
4-12 Generating parameters for your template ................................ 66
4-13 Selecting template to deploy ......................................... 67
4-14 Entering basic details for instance deployment ........................ 67
4-15 Entering additional details for instance deployment .................. 69
4-16 Successful instance deployment ...................................... 70
5-1  iSCSI Data Flow ............................................................ 97
Tables

1-1  MPIO path_status definitions ......................................................... 12
1-2  List of object classes in Network Installation Manager ......................... 19
1-3  Management object types and description ........................................ 20
2-1  Trusted Execution different policies .................................................. 41
4-1  Sample IBM PowerVC details .......................................................... 57
4-2  Cloud Automation Manager cloud connection for IBM PowerVC connectivity .... 58
4-3  AIX specific Ansible modules available in v2.8 of Ansible ...................... 72
4-4  Package list required for Kubernetes ............................................... 76
5-1  PowerVM NovaLink and Hardware Management Console capabilities ........... 89
5-2  Comparison between PEP editions .................................................... 94
6-1  Requirements for VMR HA components .............................................. 104
6-2  Requirements for VMR DR components ............................................. 106
7-1  RBAC Kernel Security Tables .......................................................... 128
7-2  RBAC definition file .......................................................................... 128
7-3  Listing AIXpert security levels ......................................................... 144
7-4  Available predefined event producers ............................................... 150
Examples

1-1 Checking current AIX environment .................................................. 5
1-2 Authentication between the AIX server and HMC ............................... 6
1-3 Authentication between the AIX server and PowerVC ......................... 6
1-4 Editing /var/adm/ras/liveupdate/lvupdate.data .............................. 6
1-5 Using geninstall command with -k option to perform LKU .................. 7
1-6 Creating cache pool and cache partition in a dedicated flash LUN .......... 8
1-7 Assigning cache partition to target device that needs to be cached ....... 9
1-8 Starting caching on hdisk2 target and monitor the status ................... 9
1-9 Creating cache pool and cache partition in VIOS server ...................... 9
1-10 Assigning the cache partition to a vhost in VIOS server ................. 9
1-11 Assigning cache partition to a target hdisk2 device in AIX LPAR ....... 9
1-12 Starting caching on hdisk2 target and monitor the status ................ 9
1-13 Using Ismpio to check path status ............................................... 12
1-14 Displaying parent adapters and remote ports ............................... 13
1-15 Sample output of MPIO detailed statistics .................................. 14
1-16 Dynamic queue_depth update for device ...................................... 15
1-17 Changing default reserve_policy to no_reserve for AIXPCM FC disk ... 15
1-18 Changing default algorithm to shortest_queue for AIXPCM iSCSI disk ... 15
1-19 Disabling paths using cpath ...................................................... 15
1-20 File based iSCSI discovery ....................................................... 18
1-21 ODM based iSCSI discovery ..................................................... 18
1-22 Enabling nimhttp subsystem ..................................................... 21
1-23 Using nimhttp without the use of cryptographic authentication ......... 21
1-24 Creating a Hardware Management Console resource ....................... 23
1-25 Using nimquery to define Central Electronic Complex object .......... 24
1-26 Manually defining a Central Electronic Complex ............................ 25
1-27 Registering Network Installation Manager client using niminit on client partition .......................... 25
1-28 Adding management source and LPAR id to an AIX machine ............ 25
1-29 Creation of Live Kernel Update data resource ............................... 26
1-30 Live Kernel Update initiated from Network Installation Manager master 27
1-31 defragfs command ............................................................ 29
1-32 Checking file system space reclaim support .................................. 30
1-33 Checking the system physical volume ........................................... 32
1-34 Initiating the cloning process to hdisk1 ....................................... 32
1-35 Check the new volume group created after the clone ....................... 32
1-36 Check how to wake up the clone and put it in a sleep ..................... 33
1-37 Change root directory to access altinst_rootvg clone ...................... 33
1-38 Put the altinst_rootvg into sleep state ....................................... 34
1-39 Attempting to create another clone with error from mksysb image ...... 35
1-40 Renaming the first altinst_rootvg clone ..................................... 35
1-41 Create a second clone and rename it .......................................... 36
1-42 Using yum.sh to install and configure AIX Toolbox repositories ......... 38
2-1 Checking CLIC library and kernel extention availability in the system ... 41
2-2 Checking TE status and enable it .............................................. 42
2-3 Permitting ls command as per TSD database ................................ 42
2-4 New ls untrusted command not running ...................................... 42
2-5 Disabling the TE integrity check ................................................ 43
4-1 Sample provider.tf for Terraform deployment to PowerVC .................. 56
4-2 Terraform template for creation of a IBM PowerVC partition ........................................ 63
4-3 Installing Ansible using YUM on AIX ................................................................. 71
4-4 Running yum to install Kubernetes dependencies on AIX ........................................... 76
4-5 Checking packages installed .............................................................................. 77
5-1 Creating SSP cluster ......................................................................................... 86
5-2 Creating a logical unit to be used or a client logical partition .............................. 86
5-3 Mapping the LU to virtual host with a VTD name ............................................... 86
5-4 Adding a second VIOS node ............................................................................... 86
5-5 Increasing dynamically the LU size ...................................................................... 87
5-6 Extracting mksysb from VIOS 3.1 Flash Media .................................................... 98
5-7 Extracting mksysb from VIOS 3.1 DVD Installation Media ..................................... 98
7-1 Extending the volume group to have a new disk for mirror ................................ 117
7-2 getting into mirrorvg SMIT menu ....................................................................... 117
7-3 Setting the preferred read copy by mklv or change it by chlv .............................. 119
7-4 Checking free pps .............................................................................................. 119
7-5 Using mklv command to create raw LV ............................................................... 120
7-6 Unmounting a file system from command line ...................................................... 120
7-7 Unmounting a file system using SMIT ................................................................. 120
7-8 Deactivating the volume group using varyoffvg command ................................. 121
7-9 Performing exportvg command ......................................................................... 121
7-10 Deleting the disk by rmdev command ............................................................... 121
7-11 Listing hdisk2 information using lsvg hdisk2 .................................................. 121
7-12 Listing volume group information ..................................................................... 122
7-13 Exporting the volume group .............................................................................. 122
7-14 Removing the disk using rmdev command ......................................................... 122
7-15 Checking disk growth ........................................................................................ 123
7-16 Turning off bad block relocation policy ............................................................ 123
7-17 Checking mklv command full path and check if it has security attributes ........ 129
7-18 Checking if any role associated to the access authorization .............................. 129
7-19 Creating a role using mkrole command ............................................................. 129
7-20 Using setkst command to update kernel security tables .................................... 129
7-21 Using chuser command to add the new role ...................................................... 129
7-22 Listing roles assigned to the current logged in user ........................................... 130
7-23 Checking if enhanced_RBAC is enabled ........................................................... 130
7-24 Enabling enhanced_RBAC and reboot ............................................................. 130
7-25 Using mkauth command to create authorization .............................................. 130
7-26 Listing the authorization created ...................................................................... 130
7-27 Using tracepriv for a certain command ............................................................. 131
7-28 Using setsecattr command to bind the security attributes to the authorization .... 131
7-29 Listing the complete authorization associated with the command .................. 131
7-30 Using mkrole command .................................................................................... 131
7-31 Adding the new role to the user ........................................................................ 131
7-32 Updating the kernel security table using setkst command ................................. 131
7-33 Checking enhanced_RBAC status ...................................................................... 132
7-34 Creating a user for the job of mounting the file systems ..................................... 133
7-35 Checking authorization associated with mount command ................................. 133
7-36 Checking authorization associated with umount command ............................... 133
7-37 Creating a role with both authorizations and list it ............................................ 133
7-38 Assigning the new role created to the designated user ....................................... 133
7-39 Updating kernel security table .......................................................................... 133
7-40 Testing the role by the new user ....................................................................... 134
7-41 Creating RBAC domains ................................................................................... 134
7-42 Checking the file system .................................................................................... 134
7-43  Assigning a domain to exact file system ......................................... 134
7-44  Assigning a domain to a user .......................................................... 134
7-45  Updating the kernel security table .................................................... 135
7-46  Attempting to mount the file systems ................................................. 135
7-47  Enabling EFS using efsenable .......................................................... 137
7-48  Checking EFS directories ............................................................... 137
7-49  Creating encrypted file systems ....................................................... 137
7-50  Creating the users example ............................................................. 137
7-51  Creating the EFS directory and set the inheritance .............................. 138
7-52  Checking the EFS enablement .......................................................... 138
7-53  Loading the EFS keystore into the shell .......................................... 138
7-54  Listing encrypted file ................................................................. 138
7-55  Listing encrypted file attributes .................................................... 139
7-56  Encrypting individual file ............................................................... 139
7-57  Enabling public key authentication .................................................. 140
7-58  Enabling EFS login for client and server ......................................... 140
7-59  Restarting sshd on both client and server machines ............................ 140
7-60  Creating a client user and generate a public key ................................ 140
7-61  Copying public keys on to the server in authorized_keys ...................... 141
7-62  Creating keystore and insert the public key cookies ............................ 142
7-63  Verifying login authentication ........................................................ 142
7-64  Setting specific AIXpert security level .......................................... 144
7-65  Implementing AIXpert low level security in the AIX ............................ 145
7-66  Applying low level security with verbose mode .................................. 145
7-67  Checking the rule in XML file and in /etc/security/user ....................... 145
7-68  Checking security settings against the previously applied set of rules .... 146
7-69  Generating a compliance report by pcsxpert ................................... 146
7-70  Checking the .txt version compliance report ...................................... 146
7-71  Changing the audit mode in /etc/security/audit/config ......................... 148
7-72  Define a class that has the desired event name in audit config file ......... 148
7-73  Adding the class to a specific user ................................................ 148
7-74  Adding auditstream command redirected to an output file .................... 148
7-75  Defining an audit object to monitor its execution ............................ 149
7-76  Defining an event format ............................................................. 149
7-77  Recycling audit subsystem ........................................................... 149
7-78  Stopping the audit and monitor the events ....................................... 149
7-79  Mounting /aha file system .......................................................... 151
7-80  Editing aha-pl.inp file to monitor /tmp file system ............................ 151
7-81  Mounting aha file system and start monitoring .................................. 151
7-82  Previewing the standby BOS instance creation .................................. 153
7-83  Creating multibos instance ............................................................ 154
7-84  Checking the standby logical volumes .............................................. 155
7-85  Setting the boot list to point to standby instance first ....................... 155
7-86  Rebooting the system and check which instance we booted off ............... 156
7-87  Updating the current active and standby boot image .......................... 156
7-88  Checking multibos unique tagging ................................................ 157
7-89  Removing the multibos instance .................................................. 157
Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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 actual people or business enterprises 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. The sample programs are provided “AS IS”, without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.
Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

<table>
<thead>
<tr>
<th>AIX®</th>
<th>POWER®</th>
<th>Redbooks (logo) ®</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2®</td>
<td>POWER8®</td>
<td></td>
</tr>
<tr>
<td>DS8000®</td>
<td>POWER9™</td>
<td></td>
</tr>
<tr>
<td>Global Technology Services®</td>
<td>PowerHA®</td>
<td>System Storage™</td>
</tr>
<tr>
<td>IBM®</td>
<td>PowerPC®</td>
<td>SystemMirror®</td>
</tr>
<tr>
<td>IBM Cloud™</td>
<td>PowerVM®</td>
<td>Terraform®</td>
</tr>
<tr>
<td>IBM Spectrum®</td>
<td>Redbooks®</td>
<td>Tivoli®</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XIV®</td>
</tr>
</tbody>
</table>

The following terms are trademarks of other companies:

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Ansible, OpenShift, Red Hat, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

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

VMware, and the VMware logo are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

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

This IBM® Redbook publication is a comprehensive guide that covers IBM AIX® operating system layout capabilities, distinct features, system installation and maintenance including AIX security and trusted environment and compliance integration, with the benefits of IBM Power Virtualization Management (PowerVM®), IBM Power Virtualization Center (PowerVC) including the cloud capabilities and automation flavors. The objective of this book is to introduce IBM AIX modernization features and integration with different environments:

- General AIX enhancements
- AIX Live Kernel Update individually or using Network Installation Manager.
- AIX security features and integration.
- AIX networking enhancements.
- PowerVC integration and features for cloud environments.
- AIX deployment using Terraform® and Cloud Automation Manager.
- AIX automation using configuration management tools.
- AIX integration with Kubernetes.
- PowerVM enhancements and features.
- Latest disaster recovery solutions.
- AIX Logical Volume Manager and JFS2.
- AIX installation and maintenance techniques.

Authors

This book was produced by a team of specialists from around the world working at IBM Redbooks, Austin Center.

Navdeep Dhaliwal is a Hybrid Cloud DevOps Engineer with IBM Global Technology Services® in Australia working primarily on AIX automation. He has over 15 years of experience in AIX professionally, with 13 of those years working for IBM Australia. He graduated with distinction from the University of Manitoba and holds a Bachelors of Computer Science. His areas of expertise include IBM PowerVM, Power Systems, AIX, Linux, Chef, Ansible, and Cloud technologies.

Ahmed Mashhour is a Power Systems Global subject matter expert (SME). He is IBM AIX, Linux, and IBM Tivoli® certified with 14 years of professional experience in IBM AIX and Linux systems. He is an IBM AIX back-end SME who supports several customers in the US, Europe, and the Middle East. His core experience is in IBM AIX, Linux systems, clustering management, virtualization tools, and various Tivoli and database products. Mash has authored several publications inside and outside IBM, including co-authoring other IBM Redbooks® publications. He has hosted IBM AIX, PowerVM, PowerHA® and Spectrum Scale classes more than 70 times around the world.

Armin Röll works as a Power Systems IT specialist in Germany. He has 24 years of experience in Power Systems and AIX pre-sales technical support and, as a team leader, he fosters the AIX skills community. He holds a degree in experimental physics from the University of Hamburg, Germany. He co-authored the AIX Version 4.3.3, the AIX 5L Version 5.0, the AIX 5L Version 5.3, the AIX 6.1 Differences Guide and the AIX 7.1 Differences Guide IBM Redbooks.
Liviu Rosca is a Senior Product Support and IT Specialist with IBM Global Technology Service in Romania. He has 17 years of experience in Power Systems, IBM AIX and IBM PowerHA. His areas of expertise include Power Systems, AIX, PowerHA, networking, security and telecommunications. He teaches AIX and PowerHA classes. He co-authored other IBM Redbooks publications.

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

Amit Agarwal, Javier Bazán, Petra Buehrer, Ben Castillo, Rahul Chandrakar, Bert Dufranse, Axel Ismirlian, Joefon Jann, Ann Lund, Amartey Pearson, Ian Robinson, Mohamed Soliman, Robert Thompson

IBM

Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

► Use the online Contact us review Redbooks form found at:
  ibm.com/redbooks

► Send your comments in an email to:
  redbooks@us.ibm.com

► Mail your comments to:
  IBM Corporation, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

► Find us on Facebook:
  http://www.facebook.com/IBMRedbooks
Follow us on Twitter:
http://twitter.com/ibmredbooks

Look for us on LinkedIn:
http://www.linkedin.com/groups?home=&gid=2130806

Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

Stay current on recent Redbooks publications with RSS Feeds:
http://www.redbooks.ibm.com/rss.html
General enhancements

This chapter explains IBM AIX general enhancements which include the following topics:

- “Live Kernel Update” on page 2
- “Server Flash Caching” on page 7
- “Multipath I/O” on page 10
- “iSCSI software initiator” on page 16
- “Network Installation Manager” on page 19
- “Logical Volume Manager” on page 28
- “JFS2” on page 29
- “Multiple alternative disk clones” on page 31
- “Active Memory Expansion” on page 36
- “Unicode support” on page 37
- “AIX Toolbox for Linux Applications” on page 38
1.1 Live Kernel Update

The AIX operating system provides the Live Kernel Update (LKU) function that eliminates the workload downtime that is associated with AIX system restart that is required by previous AIX releases when fixes to the AIX kernel are deployed. The workloads on the system are not stopped in a Live Kernel Update operation, yet the workloads can use the interim fixes after the Live Kernel Update operation.

AIX 7.2 Live Kernel Update aims to achieve zero downtime while updating OS patches without the need to disrupt business critical workloads. This can save organizations substantial costs and help them avoid a data breach in the event of critical security patches, which often happens due to not completing a maintenance window on time.

1.1.1 Live Kernel Update concepts

In order to preform the functionality of Live Kernel Update, we have to understand the Live Kernel Update concepts and modes.

In the Live Kernel Update function, the logical partition (LPAR) where the operation is started is named the original partition. The operation involves another LPAR that is named the surrogate partition. Check pointing a workload means freezing a running process and saving its current state. Check pointing processes on an LPAR and restarting them later on another LPAR is named mobility.

If you plan to install updates by using the Live Kernel Update function, before you begin the installation, it is recommended to back up your system so that you can return to the previous operating level, if necessary, by restoring the system from the backup or by restarting your system from an alternate disk copy. The updates that are installed by using the Live Kernel Update function are always committed. therefore, you cannot reject the updates later.

The updates for a service pack, technology level, and interim fixes are applied before starting the surrogate partition, and the running workloads are transferred from the original partition to the surrogate partition. The Live Kernel Update process involves the following steps:

- If updates to a service pack or technology level are specified to be installed by using the Live Kernel Update function, the updates are applied and committed first on the original partition.
- If any interim fixes are specified along with the service pack and technology level updates, the interim fixes are installed on the original partition.
- The root volume group of the original partition (orig-rootvg) is cloned.
- If only interim fixes are specified for the Live Kernel Update operation, the interim fixes are applied on the cloned volume group that serves as the boot volume group for the surrogate partition (surr-boot-rootvg).
- After the surrogate partition is started and while the workloads are still running on the original partition, the root volume group of the surrogate partition is mirrored (surr-mir-rootvg).
- The workload processes are check pointed and moved to the surrogate partition.
- Workloads resume on the surrogate partition in a chrooted environment (changed root directory) on the original root volume group (orig-rootvg). During this process, the workloads continue to run without being stopped, although a short blackout time occurs when these workloads are suspended.
- If the Live Kernel Update operation fails after step 1 and step 2, the updates and interim fixes installed on the system in these steps are not uninstalled. If the cause of the Live Kernel Update failure is corrected, you can attempt the Live Kernel Update operation again instead of restarting the original LPAR. In this scenario, updates or interim fixes are
not specified for the Live Kernel Update operation because the updates are already installed.

The Live Kernel Update feature is intended for applying interim fixes that contain kernel changes or kernel extension changes that require a reboot. The interim fix might contain other files (for example, commands and libraries), and the Live Kernel Update feature does not change anything about the way these files are applied. For example, a shared library will be modified on the file system, but any running processes continues to use the old version of the library. Therefore, applications that require a library fix must be stopped and restarted to load the new version of the library after the fix is applied. In AIX version 7.2 with the 7200-01 Technology Level, or later, you can use the `genld -u` command to list the processes that are using the old version of any shared libraries or other objects that are updated. You can use the list that is displayed from the `genld -u` command to identify the processes that must be stopped and restarted to load the updated objects.

The mirror copy of the original root volume group (rootvg) is retained after the Live Kernel Update operation is complete. Thus, if you have installed only interim fixes with the Live Kernel Update function and if you want to return to the state of the system before you applied the interim fixes, the LPAR can be restarted from the disk that was specified as the mirror volume group (mirrorvg).

Alternatively, you can choose to install any updates or interim fixes on the original LPAR by using any installation method that is supported by the AIX operating system. After these updates or fixes are installed, you can use the Live Kernel Update function to load the updated kernel software without restarting the system.

### 1.1.2 Live Kernel Update modes

Live Kernel Updates has two modes, either a preview mode or a an automated mode.

**Preview mode**

In preview mode, estimation of the total operation time, estimation of application blackout time, and estimation of resources such as storage and memory are provided to the user. These estimations are based on the assumption that the surrogate partition has the same resources in terms of CPU, memory, and storage as the original partition. All the provided inputs are validated and the Live Kernel Update limitations are checked.

**Automated mode**

In automated mode, a surrogate partition with the same capacity as the original partition is created, and the original partition is turned off and discarded after the Live Kernel Update operation completes.

### 1.1.3 Live Kernel Update procedure

The Live Kernel Update process involves the following steps:

1. Back up the system by using your preferred backup method. A backup is required if you want to restore the system to its previous state before the updates or interim fixes were installed.
2. Install the updates and interim fixes by using any supported installation method (Network Installation Manager or installp).
3. If you must restart the system to apply the updates or interim fixes, you can use the Live Kernel Update function instead of restarting the system. The Live Kernel Update operation starts either through the `geninstall` command or Network Installation Manager (NIM).
4. The root volume group of the original partition (orig-rootvg) is cloned.
5. After the surrogate partition is started and while the workloads are still running on the original partition, the root volume group of the surrogate partition is mirrored (surr-mir-rootvg).
6. The workload processes are check pointed and moved to the surrogate partition.
7. Workloads resume on the surrogate partition in a chrooted environment (changed root directory) on the original root volume group (orig-rootvg). During this process, the workloads continue to run without being stopped, although a short blackout time occurs when the workloads are suspended.
8. If the Live Kernel Update operation fails, correct the cause of the failure, and then retry the process starting at step 3.

1.1.4 Live Kernel Update management types

The LPAR can be managed by either Hardware Management Console (HMC) or IBM Power Virtualization Center (PowerVC).

**HMC based Live Kernel Update operation**

If the LPAR is managed by an HMC, you must authenticate to the HMC. You can authenticate to the HMC by using the `hmcauth` command or by defining an HMC object through Network Installation Manager (NIM).

The `hmclientliveupdate` HMC role has all the privileges required for the Live Kernel Update operation. If a user is defined on the HMC with this role, the authentication can be done with this user rather than the hscroot user.

When you run the Live Kernel Update operation, the value of the lpar_id attribute changes. You can request a specific value for the lpar_id attribute in the lvupdate.datafile, but it cannot be the same as the original value.

**PowerVC based Live Kernel Update operation**

If the LPAR is managed by PowerVC, you can authenticate with the PowerVC by using the `pvcauth` command or by defining a PowerVC object through NIM.

When you run the Live Kernel Update operation, the value of the lpar_id attribute changes. But, you cannot request a specific value for the lpar_id attribute in the lvupdate.datafile. If multiple profiles are associated with the LPAR, only the active profile is maintained by the Live Kernel Update operation. The other profiles are not preserved after the Live Kernel Update operation is complete. The virtual adapter ID values, also known as slot numbers, might change during the Live Kernel Update operation.

1.1.5 Live Kernel Update methods

Live Kernel Update can either be done by a normal `geninstall` command or through a NIM environment.

**Note:** The Live Kernel Update operation does not require you to specify any updates or interim fixes because the updates are installed on the system.
Live Kernel Update by NIM
Performing the Live Kernel Update operation by using NIM requires a preparation, since starting AIX Live Kernel Update operation on target machine can be initiated from NIM master or from NIM client.

See more details in 1.5, “Network Installation Manager” on page 19

Live Kernel Update using geninstall command
You can always run the Live Kernel Update using `geninstall` command either in one step pointing to the Technology Level, Service Pack or the interim fix, or by updating the AIX, then verifying the Live Kernel Update that would not make you reboot the AIX server.

The first part is to check your current AIX server and to have at least two free disks with proper multi pathing setup as shown in Example 1-1. We recommend having three disks to create alt_disk_copy beforehand.

Example 1-1 Checking current AIX environment

```bash
# ifconfig en0
en0: flags=1e084863,814c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD(ACTIVE),LARGESEND,CHAIN> inet 9.47.64.99 netmask 0xfffff000 broadcast 9.47.79.255
tcp_sendspace 262144 tcp_recvspace 262144 rfc1323 1

# lspv
hdisk0 00f6db0a6c7aece5 rootvg active
hdisk1 none None
hdisk2 none None
hdisk3 none None

# lsdev -Ccdisk
hdisk0 Available C3-T1-01 MPIO IBM 2076 FC Disk
hdisk1 Available C3-T1-01 MPIO IBM 2076 FC Disk
hdisk2 Available C3-T1-01 MPIO IBM 2076 FC Disk
hdisk3 Available C3-T1-01 MPIO IBM 2076 FC Disk

# lspath
Enabled hdisk0 fSCSI0
Enabled hdisk1 fSCSI0
Enabled hdisk2 fSCSI0
Enabled hdisk0 fSCSI1
Enabled hdisk1 fSCSI1
Enabled hdisk2 fSCSI1
Enabled hdisk0 fSCSI1
Enabled hdisk1 fSCSI1
```
We recommend creating alt_disk_copy beforehand. See 1.8, “Multiple alternative disk clones” on page 31 for more information on creating the rootvg clone.

The second step is to authenticate between AIX server and the controlling environment, either HMC or PowerVC as shown in Example 1-2 on page 6 and Example 1-3 on page 6.

**Example 1-2  Authentication between the AIX server and HMC**

```
# hmcauth -u hscroot -a 9.47.66.228
Enter HMC password:

# hmcauth -l
Address : 9.47.66.228
User name: hscroot
Port     : 12443
```

**Example 1-3  Authentication between the AIX server and PowerVC**

```
# pvcauth -a PVC_HOSTNAME_OR_IP -u powervc_username -P powervc_password

# pvcauth -a 5.5.55.121 -u root
    Enter password for root:

# pvcauth -u pvadmin -p abc@123 -a pvc_server
```

After confirming the authentication, make sure you create the Live Kernel Update data file from the lvupdate template file which is located at the path /var/adm/ras/liveupdate/lvupdate.template.

Copy it to same location with the file name /var/adm/ras/liveupdate/lvupdate.data.

As shown in Example 1-4 we set nhdisk to hdisk2 which will be the disk used to create a copy of the original rootvg and boot the surrogate partition, and we set mhdisk to hdisk3 which will be the disk used for the mirrored rootvg on the surrogate partition. The HMC lpar_id is the new surrogate LPAR id, however it can be left blank which will let the system to select the next available id for the new surrogate partition, and the HMC or PowerVC authentication that has validated earlier in the Example 1-3.

See Example 1-4 that describes setting lvupdate.data content.

**Example 1-4  Editing /var/adm/ras/liveupdate/lvupdate.data**

```
general:
    kext_check =

disks:
    nhdisk = hdisk2
    mhdisk = hdisk3

hmc:
    lpar_id = 88
    management_console = 9.47.66.228
    user = hscroot
```
Last step is to run the Live Kernel Update, either for a Technology Level, Service Pack, or an interim fix. `geninstall` with `-k` option is used to perform the operation as described in Example 1-5.

**Example 1-5   Using geninstall command with -k option to perform LKU**

```bash
# oslevel -s
7200-03-01-1838
# geninstall -Yk -d /patches/aix/aix72tl3sp3 update_all
```

Example 1-5 will update the AIX server from 7200-02-01 to 7200-03-03 without rebooting the AIX server.

If you have already updated the AIX server either a Technology Level, Service Pack or interim fix by `smitty update_all` or `emgr` commands and you have not rebooted yet, there is still a chance to activate the updates without a reboot, using same `geninstall` command with only `-k` option.

**Note:** The `geninstall` `-k` operation does not update RPM packages. RPM packages should be updated before performing an LKU operation.

### 1.2 Server Flash Caching

Flash Cache is referred to as server side caching of data. It allows an LPAR to use SSDs or flash storage as read-only cache to improve read performance for spinning disks. The cache can be significantly smaller than the data it is caching and can be direct attached or SAN based.

It is an emerging technology in recent years because of its own advantages and one of them is faster access of to data. This has been introduced so enterprise-class customers can have faster access to data in their production environment which was much essential. In order to make use of flash cache, you need to have a set of flash drives and solid-state drive (SSD) capable adapters connected to the flash drives.

#### 1.2.1 Flash Caching Concepts

The caching functionality can be enabled dynamically while the workload is running, the act of starting to cache does not require the workload to be brought down to a quiescent state. The caching is also completely transparent to the workload. Please consider the following items:

- The cache devices can be server-attached flash (built-in SSD drives in the server), flash devices directly attached using SAS controllers, or flash resources in SAN.
- If a target device is cached all read requests are routed to the caching software.
- If a particular block is found to be in the flash cache, then the I/O request is served from the cache device.
- If a block requested is not found in the cache, or if it is a write request, it falls through to the original storage.

**The following terms should be considered.**

Flash Cache has several terms and concepts that form the structure.
**Cache Device**
Any SSD or flash storage used for caching.

**Cache Pool**
A group of cache devices that is only utilized for storage caching. It provides simplified management of multiple flash disk devices. Initially only a single cache pool is supported, however additional devices can be added to expand a cache pool as needed.

**Cache Partition**
A logical cache device that is carved out from the Cache Pool. It provides flexibility and better utilization of flash storage for caching. Multiple partitions can be utilized or expanded as needed for a larger working set.

**Target Device**
A storage device that is being cached. A single cache partition can be used to cache one or more target devices. You can always enable or disable caching on one or more target devices.

**Cache Management**
The command set that is available on AIX and on the VIOS to create a Cache Pool, carve it up into Cache Partitions and assign them to workloads or AIX partitions (LPARs).

**Cache Engine**
This is the core of the caching software that makes decisions on what blocks in the storage need to be cached, and retrieve data from the cache as opposed to the primary storage.

### 1.2.2 Implementation modes

AIX server-side flash caching is supported in several configurations. These configurations differ on how the cache device is provisioned to the AIX LPAR. These modes are discussed in the following sections.

**Dedicated**
In dedicated mode, the cache device is directly provisioned to the AIX LPAR. A cache pool needs to be created on this device, on which only one cache partition may be created. The cache partition can then be used as a cache for any number of target devices on this LPAR.

Since the cache device is dedicated to this LPAR, the LPAR will not be able to use Live Partition Mobility. If the LPAR needs to be migrated to another service, the caching will need to be manually stopped, and the cache device un-configured in preparation for the migration.

In the following scenario, we have hdisk1 as a cache version, and hdisk2 as a target device.

The first step is to create a cache pool and a cache partition as shown in Example 1-6

**Example 1-6 Creating cache pool and cache partition in a dedicated flash LUN**

```
# cache_mgt pool create -d hdisk1 -p pool1
# cache_mgt partition create -p pool1 -s 1000M -P part1
```

Next step is to assign the cache partition to target devices hdisk2 in which we would like to cache as described in Example 1-7.
Example 1-7  Assigning cache partition to target device that needs to be cached

```bash
# cache_mgt partition assign -t hdisk2 -P part1
```

Next we have to start caching on the target device and monitor the status as shown in Example 1-8 on page 9.

Example 1-8  Starting caching on hdisk2 target and monitor the status

```bash
# cache_mgt cache start -t hdisk2
# cache_mgt monitor get –h -s
```

**Virtual SCSI mode**

In the virtual SCSI mode, the cache device is assigned to the VIOS. In this case, the cache pool is created on the VIOS.

The cache pool is then carved up into several partitions on the VIOS. Each cache partition can then be assigned to a virtual host (vhost) adapter. After this, once the device is discovered on the AIX LPAR, the partition can be used to cache a target device. Since this cache device is virtual, the partition can be migrated to another server.

Prior to the start of the migration, the cache is automatically stopped on the source. As part of the migration a cache partition of the same size is created dynamically on the target VIOS (if the target VIOS also has the caching software installed and has a cache pool available).

During the migration the cache partition is made available to the LPAR. Once the migration is completed, caching is automatically started on the destination. In this case, the cache will start in an empty unpopulated state.

In the following scenario, the cached device is hdisk1 on the VIOS and the target device is hdisk2 on the AIX LPAR.

In VIOS, we will do the following:

1. Create a cache pool and a cache partition on VIOS server side as shown in Example 1-9

   ```bash
   Example 1-9  Creating cache pool and cache partition in VIOS server
   # cache_mgt pool create -d hdisk1 -p pool1
   # cache_mgt partition create -p pool1 -s 100M -P part1
   ```

2. Assign the cache partition to a virtual host (vhost) in VIOS server as described in Example 1-10

   ```bash
   Example 1-10  Assigning the cache partition to a vhost in VIOS server
   # cache_mgt partition assign -P part1 –v vhost0
   ```

In AIX LPAR, we will do the following:

1. Assign the cache partition to a target device in AIX LPAR as described in Example 1-11

   ```bash
   Example 1-11  Assigning cache partition to a target hdisk2 device in AIX LPAR
   # cache_mgt partition assign –t hdisk2 –P cachedisk0
   ```

2. Start caching on the target device and monitor the status as shown in Example 1-12

   ```bash
   Example 1-12  Starting caching on hdisk2 target and monitor the status
   # cache_mgt cache start –t hdisk2
   ```
1.3 Multipath I/O

Multipath I/O (MPIO) provides the ability for a device to be accessed through one or more storage paths. To use MPIO, you need two main components:

- An MPIO capable device driver that controls the target device.
- A path control module (PCM) that provides path management functions.

As of AIX 7.2 TL 3, the parallel SCSI, Internet Small Computer Systems Interface (iSCSI) and Fibre Channel (FC) disk device drivers and their device methods support MPIO disk devices. In this section we will discuss the AIX path control module.

1.3.1 AIX Path Control Module (AIXPCM)

The native AIX path control module (AIXPCM) performs path management functions and health checks. The health check capabilities of AIXPCM allow it to:

- Check the paths and determine which paths are currently usable for sending I/O.
- Enable a path that was previously marked failed because of a temporary path fault.
- Check currently unused paths that would be used if a failover occurred.

AIXPCM benefits from being integrated into the operating system which removes the need for separate maintenance activities related to updating the PCM. By being integrated, your regular operating system maintenance will result in the PCM being updated as well simplifying management of your AIX environments.

When using the default AIXPCM you will be able to set two types of attributes, device attributes and path attributes. These attributes can be modified to suit your system configuration based on your individual requirements and be defined on a per device basis if required.

I/O Algorithms

One attribute that can be changed is the algorithm attribute. Algorithm determines the way I/O is distributed across the paths available for a device. There are three different algorithms for I/O distribution:

- **fail_over**: In failover mode I/O is sent over one enabled path, another path is only used if this path fails. The initial path that is selected is the one with the highest priority (the lowest path priority value). If this path fails, the next highest priority path is selected for I/O operations. This is the default algorithm for iSCSI and Fibre Channel devices.

- **round_robin**: In this mode I/O is distributed across multiple enabled paths. For any device that has preferred and non-preferred paths or active and passive paths, only a subset of paths are used. Paths that have a higher path priority receive a larger share of I/O operations in this mode.

- **shortest_queue**: In this mode, an I/O path is distributed across multiple enabled paths. For any device that has preferred and non-preferred paths or active and passive paths, only a subset of paths are used. In this mode path priority is ignored and paths are selected based on the amount of
pending I/O operations only. The path with the lowest number of pending I/O operations is selected for I/O.

Health check mode and interval
Another key device attribute is the hcheck_mode attribute. This attribute determines which paths are probed when the health check capability is used. Health checking is only performed on devices which have a state of open, a device that is not in use will not have its paths health checked. Health checking is also not performed on any disabled or missing paths. There are three health check modes:

- **enabled** In this mode the healthcheck command is sent to all paths that have are enabled for the device. This includes paths that have failed.
- **failed** In this mode the healthcheck command is sent to all paths that are in failed state for the device.
- **nonactive** In this mode the healthcheck command is sent to all paths that do not have any active I/O, this includes paths that are in enabled and failed state. This is the default health check mode that is configured on AIX.

Along with hcheck_mode you can also configure how often the health check is performed by configuring the hcheck_interval value. This attribute can be set to any value between 0 and 3600 and represents the time in seconds between polling. If a value of 0 is specified this indicates that health checking should be disabled on the device. The default value for hcheck_interval is set to perform health checking every 60 seconds.

Reservation policies
If your disks require concurrent access from multiple initiators, another attribute you may need to modify is the device attribute reserve_policy. This device attribute is required for all MPIO devices regardless of the PCM in use. This value describes the type of reservation policy that is set on a device. For MPIO devices, the following reservation policies exist:

- **no_reserve** This policy does not apply any reservation on the target device allowing initiators (paths) on the same system, as well as on other systems, access to the target device. This is the recommended policy for devices where disks are shared between hosts as well as devices that have the shortest_queue or round_robin algorithms configured.
- **single_path** This is the default policy when using AIXPCM. This policy places a SCSI2 reserve on a target device so that the device can only be accessed on the path it was reserved on. This policy prevents other paths on the same system from accessing the storage without first sending a bus device reset to release the reserve on the device.
- **PR_exclusive** This policy applies a SCSI3 persistent-reserve with exclusive-host methodology on the device when the device is opened to exclusively lock it to a single host. A PR_key_value attribute must also be set on the device when using this mode to uniquely identify the host.
- **PR_shared** This policy applies a SCSI3 persistent-reserve with shared-host methodology when the device is opened. Initiators from other host systems must register before they can access the device. A PR_key_value attribute must also be set on the device when using this mode to uniquely identify the host.

Queue depths
The queue_depth attribute for MPIO devices specifies the number of commands that AIX can concurrently send to the device. The desired value for this will be dependent on multiple
factors such as your storage systems throughput and latency as well as application requirements.

Checking paths using AIXPCM
The main command used to display information about your MPIO configuration is the `lsmpio` command as shown in Example 1-13. In this example we used the `-l` flag to specify the disk we wanted to see the status of, by omitting this flag you will see the status of all MPIO disks. The path_status column provides information on preferred paths as well as path failures. The possible values for path_status are described in Table 1-1.

**Example 1-13 Using lsmpio to check path status**

```
# lsmpio -l hdisk0
name    path_id  status   path_status  parent  connection
hdisk0  0        Enabled  Non          fSCSI0  5005076802360f79,0
hdisk0  1        Enabled  Sel,Opt      fSCSI0  5005076802360f7a,0
hdisk0  2        Enabled  Non          fSCSI1  5005076802260f79,0
hdisk0  3        Enabled  Opt          fSCSI1  5005076802260f7a,0
```

**Table 1-1 MPIO path_status definitions**

<table>
<thead>
<tr>
<th>path_status value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opt</td>
<td>Indicates that the path is an optimized path. This value indicates a path that attaches to a preferred controller in a device that has multiple controllers. The PCM selects one of the preferred paths for I/O operations whenever possible.</td>
</tr>
<tr>
<td>Non</td>
<td>Indicates that the path is a non-optimized path. On a device with preferred paths, this path is not considered as preferred path. The PCM avoids the selection of this path for I/O operations, unless all preferred paths fail.</td>
</tr>
<tr>
<td>Act</td>
<td>Indicates that the path is an active path on a device that has active and passive controllers. The PCM selects active paths for I/O operations on such a device.</td>
</tr>
<tr>
<td>Pas</td>
<td>Indicates that the path is a passive path on a device that has active and passive controllers. The PCM avoids the selection of passive paths.</td>
</tr>
<tr>
<td>Sel</td>
<td>Indicates that the path is being selected for I/O operations, at the time when the lsmpio command is to be run.</td>
</tr>
<tr>
<td>Rsv</td>
<td>Indicates that the path has experienced an unexpected reservation conflict. This value might indicate a usage or configuration error, with multiple hosts accessing the same disk.</td>
</tr>
<tr>
<td>Fai</td>
<td>Indicates that the path experienced a failure. It is possible for a path to have a status value of Enabled and still have a path_status value of Fai. This scenario indicates that operations sent on this path are failing, but AIX MPIO has not marked the path as Failed. In some cases, AIX MPIO leaves one path to the device in Enabled state, even when all paths are experiencing errors.</td>
</tr>
<tr>
<td>Deg</td>
<td>Indicates that the path is in a degraded state. This scenario indicates that the path was being used for I/O operations. Those operations experienced errors, thus causing the PCM to temporarily avoid the use of the path. Any additional errors might cause the path to fail.</td>
</tr>
</tbody>
</table>
To see connectivity information for a Fibre Channel adapter you can use the `lsmpio -are` command. This command will show you connectivity status for each of your Fibre Channel adapters along with the remote ports they are connected to on your storage. This will also provide you with error statistics for the connection to each remote port as shown in Example 1-14.

**Example 1-14  Displaying parent adapters and remote ports**

```
# lsmpio -are
Adapter Driver: fSCSI1 -> AIX PCM
  Adapter WWPN: c0507608eb2b005a
  Link State: Up
  Connection Errors
  Last 10 Minutes: 0
  Last 60 Minutes: 0
  Last 24 Hours: 0
  Total Errors: 0

  Connection Errors
    Last 10  Last 60  Last 24
    Minutes  Minutes Hours
  5005076802260f79  0      0      0
  5005076802260f7a  0      0      0

Adapter Driver: fSCSI0 -> AIX PCM
  Adapter WWPN: c0507608eb2b0058
  Link State: Up
  Connection Errors
  Last 10 Minutes: 0
  Last 60 Minutes: 0
  Last 24 Hours: 0
  Total Errors: 0

  Connection Errors
    Last 10  Last 60  Last 24
    Minutes  Minutes Hours
  5005076802360f79  0      0      0
  5005076802360f7a  0      0      0
```

You can also view statistics related to each device by using the `lsmpio -Sd` command. This command provides detailed statistics for your MPIO devices. By using this command you can see the number of errors on a path as well as what type of error was detected by the AIXPCM.

In Example 1-15 we show sample output of the detailed statistics provided when using AIXPCM.
### Example 1-15  Sample output of MPIO detailed statistics

```
# lsmpio -Sd
Disk: hdisk0
Path statistics since Sun Oct 27 14:13:18 CDT 2019
Path 0: (fSCSI0:5005076802360f79,0)
  Path Selections: 0
  Adapter Errors: 0
    Software: 0
    Hardware: 0
    Transport Dead: 0
    Transport Busy: 0
    Transport Fault: 0
    No Device Response: 0
    Target Port ID Changed: 0
  Command Timeouts: 0
  Reservation Conflicts: 0
  SCSI Queue Full: 0
  SCSI Busy: 0
  SCSI ACA Active: 0
  SCSI Task Aborted: 0
  SCSI Aborted Command: 0
  SCSI Check Condition: 3
    Medium Error: 0
    Hardware Error: 0
    Not Ready: 0
    Other: 3
  Last Error: SCSI Check Condition
  Last Error Time: Sun Oct 27 14:13:22 CDT 2019
  Path Failure Count: 0
    Due to Adapter Error: 0
    Due to I/O Error: 0
    Due to Health Check: 0
    Due to SCSI Sense: 0
    Due to Qualifier Bit: 0
    Due to Opening Error: 0
    Due to PG SN Mismatch: 0
  Last Path Failure: N/A
  Last Path Failure Time: N/A
Disk: hdisk0
Path statistics since Sun Oct 27 14:13:18 CDT 2019
Path 1: (fSCSI0:5005076802360f7a,0)
  Path Selections: 173748
  Adapter Errors: 0
    Software: 0
    Hardware: 0
    Transport Dead: 0
... Output truncated ...
```

### Updating attributes and modifying defaults

These attributes can be updated to your desired values by using the `chdev` command. If the device is in use, you will need to use the `-U` flag in conjunction with the `chdev` command to update the device while it is in use. As of AIX 7.2 TL 3 queue_depth can now also be updated dynamically on an in use device.
You can update a value while a device is in use by using the command `chdev -l <device> -a <ATTRIBUTE>=<VALUE> -U` as shown in Example 1-16 where we update the queue_depth to 32. The same method can be used to update the algorithm attribute.

Example 1-16  Dynamic queue_depth update for device

```bash
# lsattr -El hdisk0 -a queue_depth
queue_depth 20 Queue DEPTH True+
# chdev -l hdisk0 -a queue_depth=32 -U
hdisk0 changed
# lsattr -El hdisk0 -a queue_depth
queue_depth 32 Queue DEPTH True+
```

You can also update the default values for any of these attributes so that future disks that are discovered have their attributes using the `chdef` command so that future disks that are discovered have your desired value set for the attribute.

To change the default value for a Fibre Channel (FC) device using AIX PCM you can run the command `chdef -t mpioosdisk -c disk -s fcp -a <attribute>=<value>` as shown in Example 1-17 on page 15.

Example 1-17  Changing default reserve_policy to no_reserve for AIXPCM FC disk

```bash
# chdef -t mpioosdisk -c disk -s fcp -a reserve_policy=no_reserve
reserve_policy changed
```

Similarly, you can change the default attribute for iSCSI devices by running `chdef -t mpioosdisk -c disk -s iscsi -a <attribute>=<value>` as shown in Example 1-18.

Example 1-18  Changing default algorithm to shortest_queue for AIXPCM iSCSI disk

```bash
# chdef -t mpioosdisk -c disk -s iscsi -a reserve_policy=no_reserve
reserve_policy changed
```

To adjust the priority on a path, or disable and enable paths, you can use the `chpath` command. To disable or enable a path you can run the command `chpath -l <device> -p <parent adapter> -s <disable|enable> -i <path id>`. You can also omit specifying the path to disable, or enable all paths on a specific adapter and run the command `chpath -l <device> -p <parent adapter> -s <disable|enable>`.

In Example 1-19 we demonstrate disabling all paths on `fscsi0` for `hdisk0`. It is important to note, using `chpath` to disable a path disables future I/O over that path only, not I/O that is currently in progress.

Example 1-19  Disabling paths using chpath

```bash
# lsmpio -l hdisk0
name path_id status path_status parent connection
hdisk0 0 Enabled Non fSCSI0 5005076802360f79,0
hdisk0 1 Enabled Sel,Opt fSCSI0 5005076802360f7a,0
hdisk0 2 Enabled Non fSCSI1 5005076802260f79,0
hdisk0 3 Enabled Sel,Opt fSCSI1 5005076802260f7a,0
# chpath -l hdisk0 -p fSCSI0 -s disable
paths Changed
# lsmpio -l hdisk0
name path_id status path_status parent connection
```
1.3.2 Subsystem Device Path Control Module (SDDPCM)

Previously, when connecting to IBM SVC and Storewize storage devices the recommended storage driver was SDDPCM. On POWER9 systems SDDPCM is no longer supported and IBM recommend the use of AIXPCM, in addition, the IBM recommended PCM to utilize on SVC and Storewize devices running microcode levels V7.6.1 and above is now AIXPCM.

AIXPCM has evolved and now includes all significant functionality available within SDDPCM.

Instructions on how to migrate from SDDPCM to AIXPCM are documented on the IBM Support page at:


**Note:** When migrating from SDDPCM to AIXPCM your device attributes will reset to system defaults including values for queue_depth, algorithm and reserve_policy. If you were utilizing the load_balance algorithm prior to migration, the shortest_queue algorithm from AIXPCM provides the same functionality. AIXPCM also defaults to using the reserve_policy of single_path. When migrating to AIXPCM you can change the reserve_policy on your devices to no_reserve to mimic the behavior of SDDPCM.

1.4 iSCSI software initiator

As of AIX 7.2 TL 3 and above, AIX now has native support for using AIX as an iSCSI initiator using any network adapter. In this section we will discuss what iSCSI is as well as how to use iSCSI on AIX.

1.4.1 iSCSI overview

iSCSI is a protocol that uses the Transmission Control Protocol and Internet Protocol (TCP/IP) to encapsulate and send SCSI commands to storage devices that are connected to a network. The detailed specification of the iSCSI standard is documented in RFC3720.

iSCSI is used to deliver SCSI commands from a client interface, which is named an iSCSI Initiator, to the server interface, which is known as the iSCSI Target. The iSCSI payload contains the SCSI CDB and, optionally, data. The target carries out the SCSI commands and sends the response back to the initiator.

An iSCSI initiator or target has globally unique iSCSI name which can be specified in two formats:

- **iSCSI qualified name (IQN)** - This is the most commonly used naming mechanism for iSCSI. It has a maximum length of 256 bytes and is prefixed with the string “iqn.”
- **iSCSI Enterprise unique identifier** - In this naming scheme the name is prefixed with “eui.” followed by 16 hexadecimal digits.
On AIX, the native MPIO enhancements in AIX 7.2 also apply to the iSCSI protocol for storage devices which have the necessary ODM definitions. For more information about the improvements to native MPIO see 1.3, “Multipath I/O” on page 10.

For more information about iSCSI you can read Introduction to iSCSI in the publication iSCSI Implementation and Best Practices on IBM Storwize Storage Systems, SG24-8327.

1.4.2 Configuring the initiator

After updating to AIX 7.2 TL3 you will notice a new device labelled iscsi0 on your AIX system, this device is the iSCSI protocol device and will be the parent adapter for all your iSCSI targets. It also holds the configuration attributes for your iSCSI software initiator. These values can be updated either through smitty using the fastpath smitty chgiscsisw.

**Initiator name**
The iSCSI initiator_name is an attribute and can be viewed by running the `lsattr -El iscsi0 -a initiator_name` command. The default name at creation time of the iscsi0 device has the syntax:

```
<hostname>.<hostid>.iscsi0
```

The values for hostname and hostid are not dynamic are only set at the time of initial configuration of the device. Hostname is the hostname of the AIX system, and hostid is the globally unique identifier for the AIX host generated by the hostid command.

This name can be updated to a name of your choosing by using the `chdev -l iscsi0 -a initiator_name="<new name>"` command. If setting the name manually ensure the name is globally unique.

**iSCSI network error recovery policy**
For the iSCSI Protocol Device you can also set the network error recovery policy by changing the attribute isw_err_recov. This determines how many attempts the iSCSI initiator will attempt to recover from network errors. We have two possible settings for the recovery policy:

- **delayed_fail**
  This is the default policy used for the iSCSI protocol device and the recommended mode for environments with single paths to an iSCSI target.

- **fast_fail**
  With this policy, time outs and retry values that are used by the iSCSI initiator are reduced to allow quick failures of paths. This allows AIX to switch over more quickly to working paths in the event of network outages to one path for an iSCSI target.

**Maximum targets allowed**
The iSCSI Protocol Device also allows you to configure the maximum allowed number of iSCSI target devices that can be configured. By reducing this number you can also reduce the amount of network memory that is pre-allocated for the driver during configuration.

**Discovery policy**
For an AIX partition to have knowledge of the targets it can access, it must first discover them. The attribute disc_policy on the iscsi0 device specifies the discovery mode. There are four different discovery modes available for the iSCSI software initiator:

- **file**
  Information about targets is specified in a configuration file which is pointed to by the disc_filename attribute on the device. This is the default discovery mode for the iSCSI initiator and the default
**disc_filename** is `/etc/iscsi/targets`. This file contains examples on defining iSCSI targets with various configurations and can be used as a reference when initially setting up your iSCSI targets.

### odm
Information about targets is specified in the ODM. If an iSCSI disk is being used as a boot disk or is part of the rootvg then ODM discovery is mandatory.

### isns
Information about targets is stored on an internet Storage Name Service (iSNS) server and is retrieved during the iSCSI initiator configuration. When using this mode you will also need to set the attributes **isns_srvports** and **isns_srvnames**. **isns_srvports** is used to define the IP addresses of your iSNS servers and **isns_srvports** defines the port they are listening on.

### slp
Information about targets is stored on a Service Location Protocol (SLP) service agent or directory agent and automatically retrieved during the iSCSI initiator configuration.

When new devices are added, you can scan for the devices by using the `cfgmgr -l iscsi0` command.

In Example 1-20 on page 18 we show the process of discovering an iSCSI target to the server where the target IQN is `iqn.1986-03.com.ibm:2145.cluster9.114.181.189.node2` with an IP address of `9.114.181.38` and port `3260`. In this example we have a CHAP secret set to `abcd1234` (we recommend more secure passwords).

#### Example 1-20  File based iSCSI discovery
```
# lsattr -El iscsi0 -a initiator_name -a disc_policy -a disc_filename
initiator_name iqn.aixlpar.hostid.092f4c6b iSCSI Initiator Name True
disc_policy    file                        Discovery Policy     True
disc_filename  /etc/iscsi/targets          Configuration file   False
# lsdev -p iscsi0
"abcd1234"' >> /etc/iscsi/targets
# cfgmgr -l iscsi0
# lsdev -p iscsi0
hdisk3 Available  MPIO IBM 2076 iSCSI Disk
# lsmpio -l hdisk3
name    path_id  status   path_status  parent  connection
hdisk3  0        Enabled  Clo          iscsi0
iqn.1986-03.com.ibm:2145.cluster9.114.181.189.node2,9.114.181.38,0xcbc,0x0
```

If discovery mode is set to odm, the configuration for the targets can be listed, removed, and added using the `lsisci`, `rmiscsi` and `mkiscsi` commands. In Example 1-21 we demonstrate adding a target with the discovery policy set to odm and using the `mkiscsi` command.

#### Example 1-21  ODM based iSCSI discovery
```
# lsdev -p iscsi0
# lsattr -El iscsi0 -a disc_policy
# chdev -l iscsi0 -a disc_policy=odm
iscsi0 changed
# lsattr -El iscsi0 -a disc_policy
disc_policy odm Discovery Policy True
```
If your environment had multiple paths to the iSCSI target, you would simply repeat the process to discover for every additional path.

1.5 Network Installation Manager

Network Installation Manager (NIM) is an application which can run on AIX partitions which provides an environment for installation and management AIX filesets. This application works on a client/server model and can be used to manage a NIM clients filesets as well as perform base operating installs on systems. The fundamental concepts of NIM and its usage are described in the publication *NIM from A to Z in AIX 5L*, SG24-7296. This section will cover the key enhancements that have been made available since its release up to AIX 7.2 TL3 SP3.

1.5.1 Object Classes

The Network Installation Manager (NIM) database is stored in the AIX Object Data Management (ODM) repository. The resources are broken into four classes with the default installation of AIX 7.2 and is further extended to a fifth class named management with the installation of the Distributed System Management (DSM).

Table 1-2 lists the available object classes within NIM along with a description of its purpose.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>Class of objects that represents a group of machines or other resources</td>
</tr>
<tr>
<td>machines</td>
<td>Class of objects that represents machines</td>
</tr>
<tr>
<td>management(^a)</td>
<td>Class of objects that represents machine control points</td>
</tr>
<tr>
<td>networks</td>
<td>Class of objects that represents networks</td>
</tr>
<tr>
<td>resources</td>
<td>Class of object that represents install resources</td>
</tr>
</tbody>
</table>

\(^a\) This class will only be visible on a Network Installation Manager master if the Distributed System Management (DSM) fileset dsm.core is installed. This fileset is available on the AIX Installation Base Media on AIX 7.1 and above.
Management object class

With the installation of the Distributed System Management (DSM) dsm.core fileset an additional set of resources may be defined which will allow you to define control points for your logical partitions. The management class objects available are the following:

Table 1-3 Management object types and description

<table>
<thead>
<tr>
<th>Object name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nas_filer</td>
<td>Defines a object that represents a Network Attached File system (NAS) device.</td>
</tr>
<tr>
<td>hmc</td>
<td>Defines a object that represents a Hardware Management Console (HMC) device.</td>
</tr>
<tr>
<td>cec</td>
<td>Defines a object that represents a Central Electronic Complex (CEC).</td>
</tr>
<tr>
<td>vios</td>
<td>Defines an object that represents a Virtual I/O Server (VIOS).</td>
</tr>
<tr>
<td>ivm</td>
<td>Defines an object that represents the Integrated Virtualization Manager (IVM).</td>
</tr>
<tr>
<td>bcmm</td>
<td>Defines an object that represents a Blade Center Management Module.</td>
</tr>
<tr>
<td>powervc</td>
<td>Defines an object that represents a PowerVC management console.</td>
</tr>
</tbody>
</table>

The hmc and powervc management class objects also require credentials, these credentials must be stored in a encrypted password file. This file is generated using the `dpasswd -f EncryptedPasswordFilePath -U UserID -P Password` command. The `-P` flag may be omitted to avoid having the password displayed as clear text in your shell history.

An example of this command is shown in Example 1-24 on page 23 when creating an HMC object.

The objects in the management class help represent the association of between a cec and the management device used to manage it. For example, to associate a AIX partition with a CEC, and associate the CEC with an HMC you would perform the following steps:

1. Create the HMC object as shown in Example 1-24 on page 23.
2. Define the CEC object as shown in Example 1-25 on page 24.
3. Define the AIX partition as shown in Example 1-27 on page 25.
4. Associate the AIX partition with the CEC it resides on and set its LPAR id as shown in Example 1-28 on page 25.

By defining this association, you can perform operations on a machine such as a live kernel update which would require authentication to a management source or control over the partition at the hypervisor level.
1.5.2 HTTP Service

As of AIX 7.2 a new service handler that provides HTTP access to NIM resources has been available. The nimhttp service is defined in the /etc/services and the nimhttp daemon, which listen for requests over 4901 port. When the nimhttp service is active, NIM clients will attempt to access supported resources on the NIM server using the HTTP service on the port listed for the nimhttp service in the /etc/services file. If HTTP access fails or if the access is denied, access failover attempt to the NFS client occurs. The NIM resources that support HTTP access are the following resources:

- file_res
- fix_bundle
- installp_bundle
- lpp_source
- script

Enabling HTTP service

To enable the NIM HTTP service, you would utilize the `nimconfig -h` command as shown in Example 1-22.

**Example 1-22  Enabling nimhttp subsystem**

```
# nimconfig -h
0513-077 Subsystem has been changed.
0513-059 The nimhttp Subsystem has been started. Subsystem PID is 10354954.
# lssrc -s nimhttp
Subsystem                  Group     PID         Status
nimhttp                      -     10354954 active
```

Once the service has started for the first time the configuration file for the daemon will be created in /httpd.conf. The log file for the HTTP service is located in /var/adm/ras/nimhttp.log.

If NIM is not configured for using cryptographic authentication the configured nimhttp service will be accessible directly through http://<ip>:4901 as well as all resources that support the HTTP service as shown in Example 1-23. This can be prevented by enabling cryptographic authentication in NIM.

**Example 1-23  Using nimhttp without the use of cryptographic authentication**

```
# lsnim -l nimexample_script
nimexample_script:
    class       = resources
    type        = script
    Rstate      = ready for use
    prev_state  = unavailable for use
    location    = /export/resources/nim_example.sh
    alloc_count = 0
    server      = master
# curl http://9.47.76.98:4901/export/resources/nim_example.sh
#!/bin/sh

echo "This is a an example of nimhttp in use."

exit 0
```
If nimsh is in use and configured to use Secure Socket Layer (SSL) communication, control traffic will be encrypted when using the NIM HTTP service as well. After enabling the cryptographic authentication as described in section 1.5.3, “Secure Socket Layer (SSL)” you will need to stop and start the nimhttp service if it is already running `stopsrc -s nimhttp` and then restarting the service using `startssrc -s nimhttp` for the changes to take affect.

### 1.5.3 Secure Socket Layer (SSL)

The nimsh service handler supports the use of OpenSSL cryptographic authentication between NIM master and NIM clients. To use this feature with NIM you will need to ensure you have OpenSSL installed on both NIM server and clients. We recommend installing the latest version of OpenSSL available from the AIX Web Download pack:


To determine if your NIM master is already configured for SSL support you can run the `lsnim` command on your NIM master. If the resultant output is either empty or displays `ssl_support=no` this means your environment has not been configured for cryptographic authentication.

To enable SSL encryption communication between SSL support can be enabled by first running `nimconfig -c` on the NIM master, this will generate the following directory structure containing the SSL configuration:

- `/ssl_nimsh` Parent directory for NIM SSL configuration
- `/ssl_nimsh/configs` Contains configuration files and scripts used for SSL creation
- `/ssl_nimsh/certs` Contains SSL certificates used during host authentication
- `/ssl_nimsh/keys` Contains SSL keys used during SSL protocol communication

**Note:** If the nimhttp service is running and you issue a `nimconfig -c` you will not be able to regenerate the certificates as the files will be in use, you will need to first stop the nimhttp service by using the `stopsrc -s nimhttp` command and then restart it after regenerating the certificates using the `startssrc -s nimhttp` command.

To configure the NIM client, you will also need to run the `nimclient -c` command on the NIM client and ensure the communication method on the client is configured for `nimsh`.

### 1.5.4 Live Kernel Update

With the introduction of Live Kernel Update (LKU) as discussed in 1.1, “Live Kernel Update” on page 2, Network Installation Manager (NIM) also supports performing updates using Live Kernel Updates. This section introduces how to prepare your environment to perform NIM updates using Live Kernel Update.

**Preparing for live kernel update**

To prepare your Network Installation Manager (NIM) server to allow you to perform live kernel updates a number of NIM resources will need to be defined:

- An object representing the Hardware Management Console (HMC)
- An object representing the Central Electronic Complex (CEC)
- An object representing the AIX partition
- An object representing the Live Kernel Update configuration
Defining the Hardware Management Console

The first step is creating a Hardware Management Console (HMC) object in Network Installation Manager (NIM) to generate an encrypted password file to access the HMC by using the `dpasswd -f EncryptedPasswordFilePath -U hmcLogin -P hmcPassword` command.

After generating the encrypted password file, you can define the HMC object using command:

```
# nim -o define -t hmc -a passwd_file=EncryptedPasswordFilePath \\
   -a if1=InterfaceDescription \\
   -a net_definition=DefinitionName \\
   HMCName
```

After creating the HMC object the next step is to utilize the `dkeyexch -f password_file -I hmc -H host` command to exchange keys between the NIM master server and the HMC.

After exchanging the keys using `dkeyexch` you should be able to ssh to the HMC from the NIM master without being prompted for a password.

In Example 1-24 we demonstrate creating a new HMC object for a device with the following information:

- Hostname: hmc01
- Password: abc123
- Gateway: 9.47.79.254
- Subnet Mask: 255.255.240.0
- Encrypted password file path: /export/hmc/hmc_passwd

Example 1-24 Creating a Hardware Management Console resource

```
# mkdir /export/hmc
# /usr/bin/dpasswd -f /export/hmc/hmc_passwd -U hscroot
Password file is /export/hmc/hmc_passwd
Warning! Password will be echoed to the screen as it is typed.
Password:abc123
Re-enter password:abc123
Password file created.
# nim -o define -t hmc \
>   -a passwd_file=/export/hmc/hmc_passwd \
>   -a if1="find_net hmc01 0" \
>   -a net_definition="ent 255.255.240.0 9.47.79.254" hmc01
# lsnim -l hmc01
hmc01:
   class       = management
```
Defining the Central Electronic Complex

The Central Electronic Complex (CEC) is a device that represents your managed system. To define this object you must already have defined the Hardware Management Console (HMC) object. To generate the CEC object you can either use the `nimquery` command to automatically create the object for you or obtain the CEC name, hardware type, hardware model and serial number manually.

**Using nimquery to define object**

To define the Central Electronic Complex (CEC) object in one step, you can utilize the `nimquery -a hmc=hcmObjectName -d` command.

This command will discover all managed systems attached to the given HMC and register them as CEC management class objects and automatically named. In Example 1-25 we demonstrate this functionality and the resultant objects defined in NIM.

**Example 1-25   Using nimquery to define Central Electronic Complex object**

```bash
# nimquery -a hmc=hmc01 -d
... Output truncated...

/usr/sbin/nim -o define -t cec -a hw_serial=214423W -a hw_type=8286 -a hw_model=41A -a mgmt_source=hmc01 -a comments="object defined using nimquery -d" 8286-41A_214423W

/usr/sbin/nim -o define -t cec -a hw_serial=213C93A -a hw_type=8247 -a hw_model=21L -a mgmt_source=hmc01 -a comments="object defined using nimquery -d" 8247-21L_213C93A

/usr/sbin/nim -o define -t cec -a hw_serial=212B8BW -a hw_type=8286 -a hw_model=41A -a mgmt_source=hmc01 -a comments="object defined using nimquery -d" 8286-41A_212B8BW

# lsnim -t cec
8286-41A_214423W     management       cec
8247-21L_213C93A     management       cec
8286-41A_212B8BW     management       cec
```

**Manually defining objects**

An alternative to using the `nimquery` command to define the Central Electronic Complex (CEC) management object is to manually define the object by using the command:

```bash
# nim -o define -t cec -a hw_serial=cecSerialNumber \   -a hw_type=cecType -a hw_model=cecModel \   -a mgmt_source=hmcObject cecName
```
To define the object manually, you will need to obtain the CEC serial number, the hardware type, hardware model, name of the Hardware Management Console (HMC) object you created previously as well as the object name of the CEC object you want to define.

You can also use the `nimquery` command to obtain the details of the CEC’s you want to add to the system by omitting the `-d` flag. This will print the details of the CEC’s but not automatically add them to NIM. Alternatively you can obtain the details directly from the HMC by utilizing the `lssyscfg -r sys -F name,type_model,serial_num` command.

Defining a CEC object manually gives the benefit of allowing you to select your own name for the CEC as shown in Example 1-26. In this example the HMC resource name is hmc01 and we have named the TESTSYS.

**Example 1-26  Manually defining a Central Electronic Complex**

```
# nim -o define -t cec -a hw_serial=214423W \
  -a hw_type=8286 -a hw_model=41A \
  -a mgmt_source=hmc01 TESTSYS
# lsnim -l TESTSYS
TESTSYS:
  class      = management
  type       = cec
  serial     = 8286-41A*214423W
  hmc        = hmc01
  Cstate     = ready for a NIM operation
  prev_state =
```

**Defining the AIX partitions**

After defining the Central Electronic Complex (CEC) we can now define the Network Installation Manager (NIM) client. Registering standalone clients in NIM is covered in detail in the *NIM from A to Z in AIX 5L*, SG24-7296 publication. If the AIX server you want to update is already running we can perform this step by logging into the client partition and using the `niminit` command. In Example 1-27 we use the `niminit` command to register and define our client to the NIM master with the hostname nimlpar using the nimsh protocol.

**Example 1-27  Registering Network Installation Manager client using niminit on client partition**

```
# niminit -a name=aixlpar -a master=nimlpar \
  -a pif_name=en0 -a platform=chrp -a netboot_kernel=up \
  -a cable_type1=bnc
nimsh:2:wait:/usr/bin/startsrc -g nimclient >/dev/console 2>&1
0513-059 The nimsh Subsystem has been started. Subsystem PID is 5702086.
```

After defining the NIM client we need to also update the mgmt_source and identity attributes for the standalone machine object. The mgmt_source is the CEC management object that the partition resides on and the identity is the partition id of the AIX client partition. In Example 1-28 we login to the NIM master and update an existing NIM clients mgmt_source and identity attributes. In this case, the mgmt_source is 8286-41A_212B8BW and the partition id is 5. The changes are reflected in the mgmt_profile1 attribute in the example.

**Example 1-28  Adding management source and LPAR id to an AIX machine**

```
# nim -o change -a mgmt_source=8286-41A_212B8BW -a identity=5 aixlpar
# lsnim -l aixlpar
aixlpar:
  class = machines
  type  = standalone
```
Defining the Live Kernel Update configuration resource

To utilize Live Kernel Update, you will also need a lvupdate.data file. For this, we have two options, this file can either exist on the virtual client or be defined as a resource in Network Installation Manager (NIM) and used when triggering a Live Kernel Update. A template of this file can be found in /var/adm/ras/liveupdate/lvupdate.template and can be used as a basis for configuration of your lvupdate.data for the AIX LPAR you wish to Live Kernel Update. Details regarding Live Kernel Update configuration are available in 1.1, “Live Kernel Update” on page 2. After you have created your lvupdate.data file you can create the NIM resource by using the nim -o define -t live_update_data -a location=<path to file> -a server=<server that hosts resource> command. Example 1-29 demonstrates creation of the Live Kernel Update data resource using a configured lvupdate.data file named aixlpar_lvupdate.data.

Example 1-29  Creation of Live Kernel Update data resource

```nim
# nim -o define -t live_update_data
>     -a location=/export/resources/aixlpar_lvupdate.data
>     -a server=master aixlpar_lvupdate_data
# lsnim -l aixlpar_lvupdate_data
aixlpar_lvupdate_data:
  class       = resources
  type        = live_update_data
  Rstate      = ready for use
  prev_state  = unavailable for use
  location    = /export/resources/aixlpar_lvupdate.data
  alloc_count = 0
  server      = master
```

Performing the Live Kernel Update

Once the required resources related to your virtual partition are defined in your Network Installation Manager (NIM) master server you will be able to utilize your lpp_source resources to perform live kernel updates on your AIX partitions. These updates can be initiated either from the NIM client or from the NIM master based on your preference.

When initiating a customization from the Network Installation Manager (NIM) master using nim -o cust -a live_update=yes this will trigger the NIM client to initiate a Live Kernel Update based installp. If you have created a live_update_data resource and want to use this for your NIM installation you would also need to specify nim -o cust -a live_update=yes live_update_data=lvupdate_data_resource, in the Example 1-30 on page 27 we perform a Live Kernel Update using a live_update_data resource named aixlpar_live_update_data and perform a update_all customization to a server. In the event that you omit the live_update_data value NIM will use the lvupdate.data located locally on the client itself.
Example 1-30  Live Kernel Update initiated from Network Installation Manager master

```
# nim -o cust -Y -a fixes=update_all \
>    -a live_update=yes \ 
>    -a live_update_data=aixlpar_lvupdate_data \ 
>    -a lpp_source=aix72t13sp2_lpp aixlpar

install_all_updates: Initializing system parameters.
install_all_updates: Log file is /var/adm/ras/install_all_updates.log
install_all_updates: Checking for updated install utilities on media.
install_all_updates: Updating install utilities to latest level on media.
+-----------------------------------------------------------------------------+
Pre-installation Verification...
+-----------------------------------------------------------------------------+
Verifying selections...done
Verifying requisites...done

... Output truncated ...

Non-interruptable Live Kernel Update operation begins in 10 seconds.

Initializing Live Kernel Update on original LPAR.

Validating original LPAR environment.

Beginning Live Kernel Update operation on original LPAR.

Requesting resources required for Live Kernel Update.

Notifying applications of impending Live Kernel Update.

Creating rootvg for boot of surrogate.

Starting the surrogate LPAR.

Creating mirror of original LPAR's rootvg.

Moving workload to surrogate LPAR.

Blackout Time started.

Blackout Time end.

Workload is running on surrogate LPAR.

Shutting down the Original LPAR.

The Live Kernel Update operation succeeded.

File /etc/inittab has been modified.

One or more of the files listed in /etc/check_config.files have changed.

See /var/adm/ras/config.diff for details.
```
The process to initiate a Live Kernel Update from the Network Installation Manager (NIM) client is very similar to that of one initiated by the NIM master, the only difference is that instead of the `nim` command you utilize the `nimclient` command.

### 1.5.5 nimadm support for MultiBOS environments

The `multibos` command allows the root level administrator to create multiple instances of AIX on the same rootvg. The multibos setup operation creates a standby Base Operating System (BOS) that boots from a distinct boot logical volume (BLV). This creates two bootable sets of BOS on a given rootvg.

Prior to the release of AIX 7.2 nimadm did not support migration of AIX instances which were configured for multibos. With AIX 7.2 nimadm will successfully be able to perform preservation or migration installations of AIX instances to the next level of AIX that will be available.

### 1.6 Logical Volume Manager

In this section we discuss the recent enhancement to the Logical Volume Manager (LVM) in AIX.

#### 1.6.1 LVM reclamation support

In AIX 7.2 with the 7200-01 Technology Level, or later, the Logical Volume Manager (LVM) supports space reclamation for physical volumes that are capable of reclaiming space.

LVM informs the disk driver, which in-turn informs the storage subsystem that the partition space is no longer in use and the storage subsystem can reclaim the allocated space. The disk driver helps LVM detect the space reclaim capability of physical volume. LVM and file system configuration commands such as the `rmlv` command, `rmlvcopy` command, and the `chfs` (shrink fs) command initiate the space reclamation for the partitions after they are freed. LVM detects the physical volume's space reclaim capability when it opens the volume during the execution of the `varyonvg` or the `extendvg` command. LVM also tries to detect it while the volume group is online. If the state change detection requires physical volume to be reopened, administrator must run the `varyoffvg` command and then run the `varyonvg` command for the volume group.

Volume groups that are created before AIX 7.2 Technology Level 1 might have free partition space, which is not eligible for automatic reclamation. Administrator can create and delete a temporary logical volume on those free partitions to reclaim this space. Space will be automatically reclaimed for the partitions which are freed after installation of AIX 7.2 Technology Level 1.

The LVM process to reclaim the space runs in the background after a command such as `rmlv` completes execution. If the system crashes before the LVM process completes the reclamation process for all the partitions, then the partitions are freed but the space is not reclaimed for the pending partitions. If this scenario occurs, you can create and delete dummy logical volume to reclaim the space from the remaining partitions.

The LVM process does not delay the processing of the `varyoffvg` command or the `reducevg` command even if the space reclamation process is pending. The space reclamation process is discarded instead of waiting for the process to finish.
The space reclamation functionality is available from the storage subsystem to reclaim the freed space from a physical volume. Each storage subsystem expects the reclaim request to be aligned on specific number of physical blocks, and the number of physical blocks varies according to the storage subsystem. So sometimes reclaiming blocks (all or some) from partition is not possible because reclaim size is not aligned with the physical blocks of the partition. Some storage subsystems support reclaim of block size that is more than the LVM partition size, and partial block reclamation is not possible. In this scenario, LVM might not be able to accumulate enough contiguous free partitions to generate even a single reclaim request. Therefore, when you delete multiple LVM partitions you might not reclaim the equivalent amount of space in the storage subsystem. You can use the `lvmstat` command with the `-r` option to get information about the space reclamation requests generated by the LVM.

### 1.7 JFS2

This section discusses the recent enhancements developed in JFS2 file system.

#### 1.7.1 JFS2 defragger

If a file system has been created with a fragment size smaller than 4 KB, it becomes necessary after a period of time to query the amount of scattered unusable fragments. If many small fragments are scattered, it is difficult to find available contiguous space.

To recover these small, scattered spaces, use either `smitty dejfs2` command or the `defragfs` command as shown in Example 1-31. Some free space must be available for the defragmentation procedure to be used. The file system must be mounted for read-write.

The `defragfs` command attempts to coalesce free space in the file system in order to increase the file system's contiguous free space.

A new option `-f` is added to also coalesce internal fragmentation of all files in the file system.

This reorganization may result in improved IO performance when accessing the files.

What it actually does is relocating and combining data extents for each file in the file system. This process prioritizes file organization over file system free space contiguity.

**Example 1-31  defragfs command**

```bash
# smitty dejfs2
..
# defragfs -r /usr
Total allocation groups : 104
Allocation groups skipped - entirely free : 31
Allocation groups skipped - too few free blocks : 64
Allocation groups that are candidates for defragmenting : 9
Average number of free runs in candidate allocation groups : 60

# defragfs -f /usr
Warning: Filesystem /usr shares a log with other filesystems. It is recommended that the filesystem be remounted with its own dedicated log volume to avoid performance issues. Continue anyway? (y/n): y
..
527 files defragged.
```
1.7.2 JFS2 space reclaim

JFS2 space reclaim is used to return space to the disk in order to enact a thin provisioning scheme. Under a thin provisioning scheme, space is not allocated until a write is issued by the host to the disk. But as a file system grows, more and more space is allocated, and even if files are deleted later, this space is not given back.

In AIX 7.2 TL 03, a new `chfs` command option is introduced to reclaim unused space in the file system. With this new `chfs` command option, we can reclaim most of this space from the underlying disk, allowing it to be reused without shrinking the size of the file system. The file system layer only provides a user command to execute this option. The actual work of the reclamation is done by the Logical Volume Manager (LVM). This option is only supported on JFS2.

There is a new attribute for the `chfs` command to reclaim unused space in JFS2 with `-a` option `chfs -a reclaim=normal | fast` If the normal mode is chosen, this command will pack the file system as much as possible. Then it will look for the biggest contiguous chunk of free space and then reclaim as much of it as it can. If the fast mode is chosen, this command will look for the biggest contiguous chunk of free space (without first packing the file system) and then reclaim as much of it as it can. The `chfs` command will not be able to tell exactly how much free space is recovered.

To get an estimate, the user should run, `lvmstat -v VNAME -r` after running the `chfs` command to provide a rough estimate of how much space was reclaimed.

Points to be taken into consideration

- Although file system space reclaim frees disk partitions for reuse, this does not alter the actual file system size.
- File system space reclaim is recommended during periods of low/O activity.
- File system space reclaim can only be run on a file system whose disks all support the reclaim operation. If all the disks on a volume group support the reclaim operation, the `lvmstat -v VNAME -r` command will display "on" on the reclaim column. See Example 1-32 on page 30

Example 1-32  Checking file system space reclaim support

```
# lvmstat -v testvg -r

PV_name  reclaim Mb_freed Mb_pending Mb_success Mb_failed Mb_reused
hdisk5    on  0  0  0  0  0
hdisk6    on  0  0  0  0  0
```

- This cannot be run while Live Kernel Update is running, nor will Live Kernel Update start if this command is in progress.
- It cannot be used on a file system that is read-only.
- It cannot be used concurrently with any file system resize operation like `shrinkfs` or `extendfs`.
- The amount of free space reclaimed depends on the number of free partitions in the file system as well as the underlying storage's reclaim block size.

Please see `lvmstat` manual page for more information

1.8 Multiple alternative disk clones

You can always clone AIX running on rootvg to an alternate disk on the same system, install a user-defined software bundle, and run a user-defined script to customize the AIX image on the alternate disk.

Cloning the rootvg to an alternate disk has many advantages. One advantage is having an online backup available, in case of a disk crash. Keeping an online backup requires an extra disk or disks to be available on the system.

Another benefit of rootvg cloning occurs when applying new maintenance or technology level updates. A copy of the rootvg is made to an alternate disk, then updates are applied to that copy. The system runs uninterrupted during this time. When it is rebooted, the system boots from the newly updated rootvg for testing. If updates cause problems, the old_rootvg can be retrieved by resetting the bootlist and then rebooting.

1.8.1 Cloning concepts

The alt_disk_copy command allows users to copy the current rootvg to an alternate disk and to update the operating system to the next maintenance or technology level, without taking the machine down for an extended period of time and mitigating outage risk. This can be done by creating a copy of the current rootvg on an alternate disk and simultaneously applying software updates. If needed, the bootlist command can be run after the new disk has been booted, and the bootlist can be changed to boot back to the older maintenance or technology level of the operating system.

If booting from the new alternate disk, the former rootvg volume group shows up in a lspv command listing as old_rootvg, and it includes all disks in the original rootvg. This former rootvg volume group is set to not vary-on at reboot, and it should only be removed with the alt_rootvg_op -X old_rootvg or alt_disk_install -X old_rootvg commands.

Cloning operation procedure

Attempting the cloning operation will go through the following steps.

- Creates an /image.data file based on the current rootvg's configuration. A customized image.data file can be used.
- Creates an alternate rootvg (altinst_rootvg).
- Creates logical volumes and file systems with the alt_inst prefix.
- Generates a backup file list from the rootvg, and if an exclude.list file is given, those files are excluded from the list.
- Copies the final list to the altinst_rootvg's file systems.
- If specified, the installp command installs updates, fixes, or new filesets into the alternate file system.
- The file systems are then unmounted, and the logical volumes and file systems are renamed.
- The logical volume definitions are exported from the system to avoid confusion with identical ODM names, but the altinst_rootvg definition is left as an ODM place holder.

Cloning scenario for multiple images into different disks

The cloning process can clone from the current source rootvg, or it can clone from a source mksysb image.

Cloning from current rootvg is very common, and it uses the alt_disk_copy command to do the job.
Cloning from mksysb image is performed by `alt_disk_mksysb` command.

**Creating a single clone from current rootvg**

In this part we discuss the procedure of cloning the rootvg into another disk.

Firstly we will check the system physical volumes as shown in Example 1-33.

**Example 1-33  Checking the system physical volume**

```bash
# lspv
hdisk0  00f6db0a6c7aece5  rootvg    active
hdisk1  00fa2b8bfff7af30e  None
hdisk2  00fa2b8bfff6b56b   None
hdisk3  00fa2b8bfff8b600   None
```

The next step is to initiate the cloning process using `alt_disk_copy` command as shown in Example 1-34 to the target hdisk1.

**Example 1-34  Initiating the cloning process to hdisk1**

```bash
# alt_disk_copy -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.
[..]
Fixing file system superblocks...
Bootlist is set to the boot disk: hdisk1 blv=hd5
```

Checking the system after the clone operation, we can see a new volume group created named as `altinst_rootvg` as shown in Example 1-35.

**Example 1-35  Check the new volume group created after the clone**

```bash
# lspv
hdisk0  00f6db0a6c7aece5  rootvg    active
hdisk1  00fa2b8bfff7af30e  altinst_rootvg
hdisk2  00fa2b8bfff6b56b   None
hdisk3  00fa2b8bfff8b600   None
```

The `altinst_rootvg` is not active, and should not be activated by normal `varyon` commands. The only way to access the data in the new `altinst_rootvg` volume group is through a wake-up operation. A volume group wake-up can be accomplished, on the non-booted volume group.

The wake-up puts the volume group in a post `alt_disk_install phase1` state, which means the its current root file system will remain `/alt_inst` and the same applies for the rest of the file systems in `altinst_rootvg`. 

32 IBM AIX Market Enhancements and Modernization
The following Example 1-36 shows how to wake up the new clone and how to put it in a sleep. For wake-up we use `alt_rootvg_op` command with `-W` flag.

Example 1-36   Check how to wake up the clone and put it in a sleep

```
# lspv
hdisk0          00f6db0a6c7aece5                    rootvg          active
hdisk1          00fa2b8bfff7af30e                    altinst_rootvg  active
hdisk2          00fa2b8bfff8b56b                    None
hdisk3          00fa2b8bfff8b600                    None

# alt_rootvg_op -W -d hdisk1
Waking up altinst_rootvg volume group ...

# lspv
hdisk0          00f6db0a6c7aece5                    rootvg          active
hdisk1          00fa2b8bfff7af30e                    altinst_rootvg  active
hdisk2          00fa2b8bfff8b56b                    None
hdisk3          00fa2b8bfff8b600                    None
```

```
# df -g | grep alt_inst
/dev/alt_hd4       2.00   1.96    3%     3747    1% /alt_inst
/dev/alt_hd11admin 0.12   0.12    1%        5    1% /alt_inst/admin
/dev/alt_hd1       0.50   0.50    1%        7    1% /alt_inst/home
/dev/alt_hd10opt   0.50   0.13   75%    14432   32% /alt_inst/opt
/dev/alt_hd2       4.00   2.12   48%    40625    8% /alt_inst/usr
/dev/alt_hd9var    2.00   1.00   1%       61    1% /alt_inst/var
/dev/alt_livedump  0.25   0.25    1%        4    1% /alt_inst/var/adm/ras/livedump
```

If we changed to /alt_inst file systems, we can see the data of the altinst_rootvg clone, however we cannot run a command to reflect on the altinst_rootvg unless we change the root directory temporarily using `chroot` command as shown in Example 1-37.

Example 1-37   Change root directory to access altinst_rootvg clone

```
# chroot /alt_inst /usr/bin/ksh
# lsuser -a ALL
root
daemon
bin
guest
esaadmin

# ps
    PID    TTY  TIME CMD
 4587858 pts/0  0:00 -ksh
 7799194 pts/0  0:00 /usr/bin/ksh
12124530 pts/0  0:00 ps
```
To confirm this, we can see a new shell process in Example 1-37 for the new `chroot` command that gave the access to `altinst_rootvg`.

Also once we accessed the `altinst_rootvg`, we did not find the user `newuser` that has been created in the original `rootvg`. Which means it has been created after the clone.

Important notes to take in consideration:

- The clone install will not allow a wake-up to occur on a volume group with a greater operating system version, unless the `FORCE` environment variable is set to yes.
- If a `FORCE` wake-up is attempted on a volume group that contains a more recent version of the running operating system, and the waking volume group has been a system `rootvg`, several errors will occur.
- When data access is no longer needed to the clone, the volume group should be put to sleep state.
- The volume group that experiences a wake-up must be put to sleep before it can be booted and used as the `rootvg`, because booting while the volume group is awake, will mess the AIX ODM.

Setting the volume into sleep state can be done by `alt_rootvg_op` command with `-S` flag as shown in Example 1-38 on page 34.

Example 1-38  Put the `altinst_rootvg` into sleep state

```
# lspv
hdisk0          00f6db0a6c7aece5   rootvg       active
hdisk1          00fa2b8bff7af30e   altinst_rootvg active
hdisk2          00fa2b8bff8b56b   None
hdisk3          00fa2b8bff8b600   None

# alt_rootvg_op -S -d hdisk1
Putting volume group altinst_rootvg to sleep ...
forced unmount of /alt_inst/var/adm/ras/livedump
forced unmount of /alt_inst/var/adm/ras/livedump
forced unmount of /alt_inst/var
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
```
forced unmount of /alt_inst/home
forced unmount of /alt_inst/admin
forced unmount of /alt_inst/admin
forced unmount of /alt_inst
forced unmount of /alt_inst
Fixing LV control blocks...
Fixing file system superblocks...

# lv
hdisk0          00f6db0a6c7aece5                    rootvg          active
hdisk1          00fa2b8bffb7af30e                    altinst_rootvg
hdisk2          00fa2b8bff8b56b                    None
hdisk3          00fa2b8bff8b600                    None
# df -g | grep alt_inst
#

Create another clone from mksysb image

We can always create another clone either from current rootvg or from mksysb image, however the default operation of cloning will create same name of altinst_rootvg, so if we attempted either from current rootvg or from mksysb image, the process will fail as shown in Example 1-39.

Example 1-39   Attempting to create another clone with error from mksysb image

# alt_disk_mksysb -m /dev/rmt0 -d hdisk2
Restoring /image.data from mksysb image.
Checking disk sizes.
Creating cloned rootvg volume group and associated logical volumes.
0505-102 alt_disk_install: mkvg has returned an error.
0516-360 /usr/sbin/mkvg: The device name is already used; choose a different name.

The solution for this is to rename the current altinst_rootvg clone, and just create new ones as much as desired.

We will use alt_rootvg_op command with -v flag shown in Example 1-40.

Example 1-40   Renaming the first altinst_rootvg clone

# lv
hdisk0          00f6db0a6c7aece5                    rootvg          active
hdisk1          00fa2b8bffb7af30e                    altinst_rootvg
hdisk2          00fa2b8bff8b56b                    None
hdisk3          00fa2b8bff8b600                    None

# alt_rootvg_op -v 1st_clone -d hdisk1

# lv
hdisk0          00f6db0a6c7aece5                    rootvg          active
hdisk1          00fa2b8bffb7af30e                    1st_clone
hdisk2          00fa2b8bff8b56b                    None
hdisk3          00fa2b8bff8b600                    None

Now we can start creating a second clone and rename it as well as shown in Example 1-41.
Example 1-41  Create a second clone and rename it

# alt_disk_mksysb -m /dev/rmt0 -d hdisk2
Restoring /image.data from mksysb image.
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.
Restoring mksysb image to alternate disk(s).
Linking to 64bit kernel.
Fixing file system superblocks...
Bootlist is set to the boot disk: hdisk2 blv=hd5

# lspv
hdisk0  00f6db0a6c7aece5     rootvg      active
hdisk1  00fa2b8bfff8b56b     1st_clone
hdisk2  00fa2b8bfff8b600     None
hdisk3  00fa2b8bfff8b56b     None

# alt_rootvg_op -v 2nd_clone -d hdisk2
# lspv
hdisk0  00f6db0a6c7aece5     rootvg      active
hdisk1  00fa2b8bfff8b56b     2nd_clone
hdisk2  00fa2b8bfff8b600     None
hdisk3  00fa2b8bfff8b56b     None

Now the second clone has been successfully created, so no limitation of number of instances or a condition of which source the clone created from.

Upon doing the multiple clones, please mind the following points

► We cannot wake up 2 rootvgs at the same time, however we can create as much as we need.
► The running operating system must be a higher version or equal to the operating system version of the volume group that undergoes the wake up.

Please see the following two links for more information

https://www.ibm.com/support/pages/multiple-alternate-rootvg-criteria

1.9 Active Memory Expansion

Active Memory Expansion (AME) is a feature available on IBM Power systems that allows you to expand the amount of memory available to an AIX LPAR beyond the limits specified in the partition profile. AME compresses memory pages thus providing additional memory for a partition.
AME is available starting with IBM Power7 systems and AIX V6.1 with Technology Level 4 with Service Pack 2. At the time of writing this publication AIX is the only operating system that can use this feature. In-memory data compression is managed by the operating system and this compression is transparent to applications and users.

Compression and decompression of memory content can allow memory expansion with percentages than can exceed 100%. This can allow a partition to do significantly more work or support more users with the same amount of physical memory. Similarly, it can allow a server to run more partitions and do more work for the same amount of physical memory.

AME uses the CPU resources allocated to the partition to compress and decompress the memory contents of the partition. AME provides a trade off of memory capacity for CPU cycles. The degree of compression varies and depends on how compressible the memory content is. AME tends to have better results with workloads that access smaller amount of memory. AIX features a command named amepat (AME Planning and Advisory Tool) that can be used to estimate the compression rate for an individual workload.

Additional details about amepat can be found at

https://www.ibm.com/support/knowledgecenter/ssw_aix_72/a_commands/amepat.html

With initial implementation of AME only 4K and 16M memory pages are subject to compression. The following type of memory page are not compressed:

- pinned memory pages
- file pages cached in memory
- pages that have been already compressed

A new unrestricted vmo tunable has been added to enable all supported page sizes.

vmo -ro ame_mpsize_support=1

Starting with AIX 7.2 with Technology Level 1 compression of 64k memory pages is available on IBM POWER8® systems with firmware FW860, or later.

Additional details about AME can be found in IBM PowerVM Virtualization Managing and Monitoring, SG24-7590, IBM PowerVM Virtualization Introduction and Configuration, SG24-7940 and at


1.10 Unicode support

To be written....

Figure 1-1 Mind blown emoji
1.11 AIX Toolbox for Linux Applications

The AIX Toolbox for Linux Applications contains a wide variety of third party open source tools that can be installed on AIX through the RPM package manager. It also contains popular open source libraries and software commonly used in Linux based distributions. It also comes with GNU tools required to build and package Linux applications for use on AIX. The software provided in this repository comes as-is and without support.

For an overview of the AIX Toolbox for Linux Applications as well as licensing and installation details see:


**Note:** Using the AIX Toolbox for Linux repositories as well as other repositories may result in dependency issues or other unexpected errors.

To simply the installation and use of the packages available on the AIX Toolbox for Linux, AIX can now also use of the YUM package management utility to access the toolbox. The YUM package manager handles management of dependencies and installs requisite packages for you so you do not need to resolve them manually. Details regarding how to install and configure YUM on AIX to access the AIX Toolbox for Linux Applications is available at:


If your AIX partition is connected to the internet the process involves simply downloading and running the yum.sh script available on the AIX Toolbox as shown in Example 1-42.

**Example 1-42 Using yum.sh to install and configure AIX Toolbox repositories**

```
# ls -l yum.sh
-rwxr-xr-x 1 root system 9489 Oct 30 10:49 yum.sh
# ./yum.sh
Attempting download of rpm.rte & yum_bundle.tar ...
Installing rpm.rte at the latest version ...
This may take several minutes depending on the number of rpms installed...

... Output truncated ...

15:yum-3.4.3-7
16:python-devel-2.7.10-1
17:python-tools-2.7.10-1

Yum installed successfully.
Please run 'yum update' to update packages to the latest level.
```

```
# yum repolist
repo id           repo name                          status
AIX_Toolbox      AIX generic repository             2,229
AIX_Toolbox_72   AIX 7.2 specific repository        98
AIX_Toolbox_noarch AIX noarch repository            142
repolist: 2,469
```

Once the YUM repositories are configured and installed you can quickly install any package available on the repository by simply running `yum install <package name>`. 
Security enhancements

This chapter describes recent IBM AIX security enhancements which include the following topics:

- “AIX Trusted Execution” on page 40
- “AIX Secure boot” on page 43
- “AIX Secure install” on page 43
- “Multi factor authentication” on page 43
- “Cryptolibs” on page 48
- “Address Space Layout Randomization” on page 48
2.1 AIX Trusted Execution

Trusted Execution (TE) consists of a collection of features that are used to verify the integrity of the system and implement advanced security policies, which can be used together to enhance the trust level of the complete system.

Using TE mechanism, the system can be configured to check the integrity of the trusted object like a command, a binary, a library, a configuration file or a shell script in run time.

The usual way for a malicious user to harm the system is to get access to the system and then install Trojans, rootkits or tamper some security critical files, resulting in the system becoming vulnerable and exploitable. The central idea behind the set of features under Trusted Execution is prevention of such activities or in worst case be able to identify if any such incident happens to the system. Using the functionality provided by Trusted Execution, the system administrator can decide upon the actual set of executables that are allowed to execute or the set of kernel extensions that are allowed to be loaded. It can also be used to audit the security state of the system and identify files that have changed, thereby increasing the trusted level of the system and making it more difficult for the malicious user to do harm to the system.

The set of features under TE can be grouped into the following items

- Managing Trusted Signature Database
- Auditing integrity of the Trusted Signature Database
- Configuring Security Policies
- Trusted Execution Path and Trusted Library Path

Similar to that of Trusted Computing Base (TCB) there exists a database which is used to store critical security parameters of trusted files present on the system. This database, named Trusted Signature Database (TSD), resides in the /etc/security/tsd/tsd.dat.

A trusted file is a file that is critical from the security perspective of the system, and if compromised, can jeopardize the security of the entire system. Typically the files that match this description are the following items

- Kernel (operating system)
- All setuid root programs
- All setgid root programs
- Any program that is exclusively run by the root user or by a member of the system group
- Any program that must be run by the administrator while on the trusted communication path (for example, the ls command)
- The configuration files that control system operation
- Any program that is run with the privilege or access rights to alter the kernel or the system configuration files

Every trusted file should ideally have an associated stanza or a file definition stored in the Trusted Signature Database (TSD). A file can be marked as trusted by adding its definition in the TSD using the trustchk command. The trustchk command can be used to add, delete, or list entries from the TSD.

When a file is marked as trusted (by adding its definition to Trusted Signature Database), the TE feature can be made to monitor its integrity on every access. TE can continuously monitor the system and is capable of detecting tampering of any trusted file (by a malicious user or application) present on the system at run-time (for example, at load time). If the file is found to be tampered, TE can take corrective actions based on pre-configured policies, such as disallow execution, access to the file, or logging error.
To enable or disable different security policies that are used with the Trusted Execution mechanism, use the \texttt{trustchk} command. You can specify the following different policies.

\begin{table}[h]
\centering
\caption{Trusted Execution different policies}
\begin{tabular}{|l|l|l|}
\hline
Policy & STOP\_UNTRUSTED & STOP\_ON\_CHKFAIL \\
\hline
CHKEXEC & Stop loading of executables that do not belong to TSD & Stop loading of executables whose hash values do not match the TSD values \\
\hline
CHKSHELBS & Stop loading of shared libraries that do not belong to TSD & Stop loading of shared libraries whose hash values do not match the TSD values \\
\hline
CHKSCRIPTS & Stop loading of shell scripts that do not belong to TSD & Stop loading of shell scripts whose hash values do not match the TSD values \\
\hline
CHKKERNEKX & Stop loading of kernel extensions that do not belong to TSD & Stop loading of kernel extensions whose hash values do not match the TSD values \\
\hline
\end{tabular}
\end{table}

\textbf{Note}: A policy can be always enabled or disabled at any time until the TE is turned on to bring the policies into effect. Once a policy is in effect, disabling that policy becomes effective only on next boot cycle. All the information messages are logged into syslog.

The trusted Execution Path (TEP) defines a list of directories that contain the trusted executables. Once TEP verification is enabled, the system loader allows only binaries in the specified paths to execute.

Trusted Library Path (TLP) has the same functionality, except that it is used to define the directories that contain trusted libraries of the system. Once TLP is enabled, the system loader allows only the libraries from this path to be linked to the binaries.

The \texttt{trustchk} command can be used to enable or disable the TEP or TLP, as well as set the colon separated path list for both, using TEP and TLP command line attributes of the \texttt{trustchk} command.

In order for TE to work, the CryptoLight for C library (CLiC) and kernel extension need to be installed and loaded on your system. These files sets are included on the AIX Expansion Pack and are provided for free.

To check whether they are installed on your system and loaded into the kernel, you can run \texttt{lslpp} command to check and also \texttt{genkex} command or \texttt{lke} internal kdb command to make sure clickeext kernel extention is currently loaded into the system as shown in Example 2-1.

\begin{example}
\texttt{Example 2-1} Checking CLiC library and kernel extention availability in the system
\end{example}

\begin{verbatim}
# lslpp -l "clic*"
Fileset                      Level  State      Description
----------------------------------------------------------------------------
Path: /usr/lib/objrepos     clic.rte.kernext          4.10.0.2  COMMITTED  CryptoLite for C Kernel  
Path: /etc/objrepos         clic.rte.kernext          4.10.0.2  COMMITTED  CryptoLite for C Kernel
\end{verbatim}
Let's look at one example of how to turn on the run-time integrity check by configuring the policy TE, CHKEXEC, and STOP_ON_CHKFAIL.

First, we are going to check the current status of the policies, turn ON these policies, then check the status again as per the following Example 2-2.

**Example 2-2 Checking TE status and enable it**

```bash
# /usr/sbin/trustchk -p TE CHKEXEC STOP_ON_CHKFAIL
TE=OFF
CHKEXEC=OFF
STOP_ON_CHKFAIL=OFF

# /usr/sbin/trustchk -p TE=ON CHKEXEC=ON STOP_ON_CHKFAIL=ON

# /usr/sbin/trustchk -p TE CHKEXEC STOP_ON_CHKFAIL
TE=ON
CHKEXEC=ON
STOP_ON_CHKFAIL=ON
```

If we run `ls` command, it should run without problems since the entry is in TSD database and no change happened to its checksum completely, so the command execution is permitted as shown in Example 2-3.

**Example 2-3 Permitting ls command as per TSD database**

```bash
# ls
.SPOT                  bin                    mnt
.Xdefaults             bosinst.data           ok2
.bash_history          bxiup.sh              opt
.bash_profile          cd0                    perzlimages
.bashrc                core                   proc
.dbxhist               data                   sample.sh
.kshrc                 db                     sbin
.mwmrc                 dev                    smit.log
```

Let's replace the `/usr/bin/ls` binary with the `/usr/bin/chmod` binary. Then we try to run the `/usr/bin/ls` command. This time the execution should fail as it should not be trusted any more. See Example 2-5 below.

**Example 2-4 New ls untrusted command not running**

```bash
# cp /usr/bin/ls /usr/bin/ls.orig
# cp /usr/bin/chmod /usr/bin/ls

# ls
ksh: ls: 0403-006 Execute permission denied
```
Hence, the run-time integrity check is working on the system. If there are any Trojans or other viruses who changed any binary, then its execution will be denied.

To disable the run time integrity check we have enabled earlier, we can run the same command with OFF value. See Example 2-5.

**Example 2-5  Disabling the TE integrity check**

```
# /usr/sbin/trustchk -p TE=OFF CHKEXEC=OFF STOP_ON_CHKFAIL=OFF

# /usr/sbin/trustchk -p TE CHKEXEC STOP_ON_CHKFAIL
TE=OFF
CHKEXEC=OFF
STOP_ON_CHKFAIL=OFF

# ls
Usage: chmod [-R] [-f] [-h] {u|g|o|a ...} {+|-|=} {r|w|x|X|s|t ...} File ...
        chmod [-R] [-f] [-h] OctalNumber File ...
        Changes the permission codes for files or directories.

# cp /usr/bin/ls.orig /usr/bin/ls

# ls
.SPOT   bin    mnt
.Xdefaults   bosinst.data  ok2
.bash_history   bxiup.sh  opt
.bash_profile   cd0    perzlimages
.bashrc   core  proc
.dbxhist   data  sample.sh
.kshrc   db    sbin
.mwmrc   dev    smit.log
```

The Trusted Execution can be exercised in the real-time execution of the trusted commands. Whenever a trusted command is entered at the command prompt, first of all a hash is calculated by the operating system and the actual hash of the trusted command is extracted from the TSD database. Only when both the hash values are matched, the execution of the command is allowed.

### 2.2 AIX Secure boot

This section..............

### 2.3 AIX Secure install

This section....................

### 2.4 Multi factor authentication

This section introduces solutions for Multi Factor Authentication (MFA) on AIX and explains how different components of these solutions interact.
In general the process of authentication means establishing with certainty that a person is indeed who claims to be before granting access to a service or a resource.

Examples of authentication in daily life include opening a door using a badge, showing a driver license, an identity card or a passport, or simply typing a user id and a password to login to a computer. In all these cases the person who has to authenticate uses just one evidence of his/her identity.

In IT providing a user name and password is perhaps the most common used method of authentication. This method however has several disadvantages: user names can be easily disclosed or shared with other persons and password can be shared or subject to brute-force or other types of attacks. Good passwords tend to have a bigger number of alpha-numeric characters and usually include special characters which makes them more difficult to crack but also more difficult to remember. This naturally led to development of other solutions to complement or extend security degree provided by authentication using just passwords. More complex solutions for authentication employ other authentication factors such as cryptographic tokens or fingerprints.

2.4.1 Authentication factors

Items used during the authentication are named factors and they can be divided into the following categories:

- **Something you know**  This is an information you know such as password, pass phrase, confirmation number, or Personal Identification Number (PIN).

- **Something you have**  This is an item you have such as badge, picture, QR code, access card, identity card, key, or a cryptographic token (a card-style, key fob or a software token).

- **Something you are**  This could be a fingerprint, retina scan, or other biometric data such as data stored on chips located on a biometric passport.

Multi factor authentication implies the usage of at least two different authentication factors from different categories. They may belong to any of the three categories mentioned before. Authentication that uses retina scan or fingerprint may seem very sophisticated but it is still single factor authentication. Similarly, authentication that uses multiple passwords does not represent a multi factor authentication.

Multi factor authentication is employed in various activity domains where enhanced security is a requirement. There are practices and standards that govern access to government building or personal data which explicitly require the usage of MFA to establish a person identity.

One such example could be using a password and token-generated code to get access to personal financial data. Another example could be using a password and the fingerprint to get access to a computer.

2.4.2 Authentication methods

When it comes to authentication methods they can be divided into following two types:

- **In-band Authentication**  When using this method authentication credentials are provided using the same communication channel used to access the target service.

- **Out-of-band authentication**  When using this method authentication credentials are provided using an alternative communication channel which is different from the one used to access the target service.
2.4.3 In-band MFA

This section describes how MFA provides support for in-band authentication using various authentication factors. When using in-band authentication method the user who wants to login must first get a token. The token can be generated using IBM PowerSC MFA in conjunction with various authentication factors. Once generated the token can be used directly for log in.

**RSA SecurID Tokens**

RSA SecurID Tokens can be hardware devices or software-based. In this case MFA collaborates with RSA Authentication Manager to decide whether credentials provided are valid. This solution uses the following factors:

- **Something you have** The RSA SecurID token, be it hardware or software.
- **Something you know** The RSA SecurID PIN and something you know.

**Certificate-based authentication options**

In this case MFA uses client identity certificate to authenticate the user and relies on the validity of the certificate. This solution uses the following factors:

- **Something you have** A valid certificate, usually stored on a Personal Identity Verification (PIV), Common Access Card (CAC) or any other type of a cryptographic smart card.
- **Something you know** The personal PIN.

For in-band solutions the PIV/CAC card readers can be directly connected to select IBM POWER® systems (models S812, S814, S822, S824, S914, S922, and S924) using USB interfaces. The systems must run AIX 7.1 Technology Level 5 Service Pack 1, or AIX 7.2 Technology Level 2 Service Pack 1, or later. The following devices are supported:

- **smart card readers**
  - Identiv/SCM SCR3310v2
  - Identiv/SCM SCR3500
  - Gemalto IDBridge CT30
- **keyboard embedded smart card readers**
  - ACS ACR38K-E1
- **smart cards**
  - PIVKey C910 PKI Smart Card
  - NIST Test PIV Cards
  - Oberthur Technologies Smart Cards
- **Authentication token**
  - Yubikey 4

Other hardware devices may also work, but have not been tested at the time at writing this publications. Additional details regarding supported devices and PIV/CAC are available at


**IBM TouchToken**

In this case MFA uses a combination of Touch ID fingerprint biometric technology with a hashed Timed One-Time Password (TOTP) generated by IBM TouchToken. This solution uses the following factors:

- **Something you have** A valid IBM TouchToken account and IBM TouchToken, IBM Verify or other compatible app provisioned on a mobile cell-phone.
- **Something you are** The personal fingerprint.
Additional details regarding IBM TouchToken solution are available at

**Radius-based solutions**

In this case MFA collaborates with a Radius server to decide whether credentials provided are valid. Radius server can be Gemalto SafeNet RADIUS, RSA SecurID RADIUS or any generic RADIUS server that can return a simple yes/no response. This type of solutions employs a client-server architecture which uses a challenge/response authentication protocol.

### 2.4.4 Out-of-band MFA

When using out-of-band authentication method the user who wants to login has to login first to an out-of-band specific web page using one or more authentication factors to retrieve an authentication item named Cache Token Credential (CTC). The CTC is valid for a specific period of time during which can be used for directly logging in.

Starting with MFA version 1.2 out-of-band authentication is supported for HMC V9R1.921.

### 2.4.5 Authentication on AIX systems using RSA SecureID

This sections describes how RSA SecureID-based can be deployed to control access to AIX systems. RSA SecurID is a two-factor authentication solution and comprises the following components:

- **Authentication manager** This is the core component of the solution which is ultimately responsible for granting or denying access to a service or a resource.
- **Authentication agent** The agent handles user requests to access a service or a resource. It verifies the data provided and initiates an authentication session with the authentication manager. Depending on the answer from authentication manager it grants or denies the request.
- **Authenticator** The authenticator handles the data used in the process of authentication. This could be a hardware or software RSA SecurID token.
- **PAM framework** Pluggable Authentication Modules (PAM) provide the ability to incorporate various authentication mechanisms into an existing system through the use of individual pluggable modules. AIX implementations consists of a library, pluggable modules, and a configuration file.

PAM framework provides an AIX system with the following capabilities:

- Select any authentication service on the system for an application.
- Use multiple authentication mechanisms for a given service.
- Add new authentication service modules without modifying existing applications.
- Use a previously entered password for authentication with multiple modules.

The RSA Authentication Agent for PAM provides the means for RSA SecurID authentication. The PAM agent uses RSA customized shared libraries, and supports several forms of RSA SecurID authenticators for access control.
The workflow for authentication typically includes the following steps:

1. A user attempts to access a service on an AIX system where PAM agent is running, either locally or remotely.
2. The AIX system administrator configures the AIX system to use PAM by editing the auth_type attribute in the usw stanza of the /etc/security/login.cfg file. Setting auth_type = PAM AUTH instructs PAM-enabled commands to invoke the PAM API directly for authentication rather than using the traditional AIX authentication routines. The following native AIX commands and applications have been modified to recognize the auth_type attribute and enabled for PAM authentication:
   - login
   - passwd
   - su
   - ftp
   - telnet
   - rlogin
   - rexec
   - rsh
   - snappd
   - imapd
   - dtaction
   - dtlogin
   - dtsession
3. Depending on the service requested the corresponding access request is invoked. For example login is used for local console access whereas telnet or ssh can be used for remote logon.
4. PAM module intercepts all logon requests, and using PAM configuration files, directs the requests to the RSA module. If the user requesting access has been configured not to use RSA SecurID, the RSA PAM module simply allows the request to proceed. If the user requesting access has been configured to use RSA SecurID, the RSA PAM module proceeds with the authentication process. RSA PAM agent on AIX provides RSA SecurID support for the following:
   - login (console)
   - su
   - ssh
   - sudo
   - rlogin
   - telnet
   - ftp (limitations apply)

Additional details regarding RSA SecurID access authentication agent for AIX versions 7.1 and 7.2 are available at the following links:

https://community.rsa.com/docs/DOC-62752
https://community.rsa.com/docs/DOC-77540

5. The agent prompts the user for the user name.
6. The agent requests the user the access code.
7. The agent sends the user name and access code to Authentication Manager in a secure manner. If Authentication Manager approves the request, the agent grants access to the user. If Authentication Manager does not approve the request, the agent denies the access request.
2.5 Cryptolibs

This section introduces the cryptographic libraries supported on AIX.

2.5.1 OpenSSL

OpenSSL is a comprehensive set of cryptographic libraries, tools and functions which provides the basic support cryptographic functions and very useful utility programs that are used in most of the modern day IT environments.

OpenSSL is free to use for commercial purposes, however there are terms and conditions included in the license that have to be accepted before using it. Additionally, legislation regarding the use of cryptographic software varies from one country to another and the usage of OpenSSL as well as any other cryptographic software is subject to laws and regulation specific to each country.

OpenSSL is used by various software products that employ cryptographic functions, such as OpenSSH used for secure remote login to AIX systems. OpenSSH can also be used in management and deployment of keys and certificates used by PowerSC MFA.

Additional details regarding OpenSSL can be found at

https://www.openssl.org/

OpenSSL for AIX can be obtained from the following location


2.5.2 CryptoLite in C library

CliC library provides application programmers with the cryptographic primitives required for applications that contain cryptographic features. The level of abstraction of this library makes it usable for application developers who are not necessarily cryptography specialists.

AIX operating system relies on CliC library for features such as Encrypted File Systems (EFS), NFS version 4 and Trusted Execution.

2.6 Address Space Layout Randomization

This section discusses......
Networking enhancements

This chapter discusses networking enhancements available on AIX including:

- “Redundant EtherChannel” on page 50
- “Shared Memory Communication over RDMA (SMC-R)” on page 50
- “100 GbE support enhancements” on page 50
- “NLP” on page 50
3.1 Redundant EtherChannel
   To be written

3.2 Shared Memory Communication over RDMA (SMC-R)
   To be written

3.3 100 GbE support enhancements
   To be written

3.4 NLP
   To be written
Virtualization and cloud capabilities

This chapter describes the virtualization capabilities on AIX and options available for automating your AIX deployments and management, including:

- “AIX on Public Cloud” on page 52
- “PowerVC” on page 52
- “IBM Cloud Automation Manager” on page 56
- “Ansible Automation and AIX” on page 70
- “Chef Infra client on AIX” on page 73
- “Puppet Enterprise on AIX” on page 73
- “AIX container technology and Kubernetes” on page 74
4.1 AIX on Public Cloud

Traditionally IBM AIX systems had to be hosted on-premise however you can now host your IBM AIX workloads on several different public infrastructure-as-a-service (IaaS) public cloud providers. You can use the IaaS services to provision IBM AIX as well as IBM i virtual servers off-site. These servers can be used for many purposes, such as:

- Providing a rapidly deployable development or test environment
- Providing off-site disaster recovery environment
- Scaling workloads

AIX is available on several cloud providers:

- IBM Cloud™
  [https://www.ibm.com/cloud/power-virtual-server](https://www.ibm.com/cloud/power-virtual-server)
- Google Cloud
  [https://cloud.google.com/ibm](https://cloud.google.com/ibm)
- Skytap
  [https://www.skytap.com/product/cloud-platform/power-systems-aix-linux-ibmi](https://www.skytap.com/product/cloud-platform/power-systems-aix-linux-ibmi)

4.2 PowerVC

This section provides a general presentation of PowerVC and introduces briefly latest PowerVC capabilities.

4.2.1 Introduction

IBM Power Virtualization Center (IBM PowerVC) is an advanced virtualization and cloud management offering for Power Systems servers based on OpenStack technology. It is simple to install and use, and enables virtual machine (VM) setup and management. IBM PowerVC significantly simplifies the management of the virtualization for IBM Power systems that run IBM AIX, IBM i, and supported Linux operating systems.

PowerVC can manage multiple hypervisors. A single instance of PowerVC can manage any combination of Power system hypervisors:

- PowerVM with traditional VIOS using either HMC or NovaLink
- PowerVM with Software Defined I/O using NovaLink
- KVM on Power with Software Defined I/O

PowerVC simplifies overall management of IBM Power environments and provides the following capabilities:

- Create VMs.
- Resize cores and memory allocated to VMs.
- Attach storage LUNs to VMs.
- Define templates for VMs and deploy new VMs based on templates.
- Define storage providers and templates for VMs and deploy new LUNs based on templates.
- Import existing VMs and volumes so that they can be managed by IBM PowerVC.
- Monitor the use of resources across the entire environment.
- Migrate running VMs from one IBM Power system to another using LPM.
- Use Dynamic Resource Optimizer (DRO).
- Create VM snapshots.
- Automate VM remote restart.
- Suspend, resume, and restart VMs.
- Schedule and activate VMs.
- Define policies for VMs placement across multiple IBM Power systems.
- Inject SSH keys into newly deployed VMs.
- Import and export VMs between clouds.

PowerVC provides cloud users a self-service portal that includes the following capabilities:

- One-click deploy templates
- Project-based resource isolation (multi-tenancy)
- Role-based access control
- Approvals and expirations
- Definition of policies
- Metering data and e-mail notifications

Because IBM PowerVC is based on the OpenStack IBM Power systems can be managed by any tool that is compatible with OpenStack standards. When a system is controlled by IBM PowerVC it can be managed in one of the following ways:

- using native IBM PowerVC GUI or CLI.
- using scripts or applications that leverage on IBM PowerVC Representational State Transfer (REST) APIs.
- using any other application than can invoke IBM PowerVC using standard OpenStack API.

**PowerVM NovaLink**

PowerVM NovaLink is a highly-scalable software virtualization management interface for IBM POWER8 and POWER9™ servers using PowerVM virtualization. While it represents an alternative to the HMC, it does not offer hardware or service management.

PowerVM NovaLink can be used in conjunction with one or more HMCs in a co-management mode in which either the HMC or NovaLink is designated as master.

PowerVM NovaLink is deployed in a Linux VM, using either Ubuntu or RHEL. PowerVM NovaLink also features an installer that simplifies the deployment by automatically creating the VMs used for VIO servers and NovaLink. The installer also deploys VIOS as well as NovaLink software, and optionally Open vSwitch when using Software Defined Networking (SDN) thus significantly reducing deployment time for complex environments.

PowerVM NovaLink integrates with PowerVC or other OpenStack solutions to manage IBM Power systems. It enables SDN to allow PowerVC or OpenStack software to virtualize networks by using industry standard Open vSwitch and Software Defined Storage (SDS) to allow PowerVC or OpenStack software to virtualize storage by using IBM Spectrum® Scale (formerly known as GPFS).

### 4.2.2 PowerVC in the virtualization and cloud software stack

PowerVC is part of a comprehensive Power based virtualization and cloud software stack. PowerVC offerings are positioned within the available solutions for Power Systems cloud as follows:

- IBM PowerVC Standard Edition: Advanced Virtualization Management
- IBM Cloud PowerVC Manager: Basic Cloud
- IBM Cloud Orchestrator: Advanced Cloud
- VMware Vrealize: Advanced Cloud
PowerVC integration into the architecture of HMC-managed Power system environments is depicted in Figure 4-1 on page 54.
PowerVC integration into the architecture of NovaLink-managed Power system environments is depicted in Figure 4-2 on page 55.

![Figure 4-2: PowerVC integration into NovaLink-managed Power system environments]

PowerVC APIs are built on OpenStack and provide Power system customers an integration point for multi-cloud ecosystem.

At the time of writing this publication POWER9 virtualization and cloud software stack includes the following elements:

- Hypervisor: PowerVM 3.1.1
- Virtualized I/O: VIOS 3.1.1 based on AIX 7.2 TL4
- Virtualization Management: NovaLink 1.0.0.16
- Hardware Management: HMC/vHMC V9R1M940
- Private Cloud Management: PowerVC 1.4.4
- Multicloud Management: vRealize Operations v7.5

4.2.3 Latest PowerVC features

At the time of writing this publication PowerVC version 1.4.4 is the latest release which introduces the following features:

- IBM i license key injection
- Initiator storage tagging
- Enables ‘pinning’ VM to specific host
- Inactive (“cold”) migration
- Image sharing between PowerVC projects
- LPM VMs back to original system after evacuation
4.3 IBM Cloud Automation Manager

IBM Cloud Automation Manager (CAM) is a self service management platform which runs on IBM Cloud Private. This product can be used to deploy cloud infrastructure on multiple cloud providers allowing you to automate and standardize delivery of your infrastructure and application stacks consistently across multiple clouds.

The key features of Cloud Automation Manager are:

- Cloud infrastructure templates utilize Terraform.
- The ability to use template parameters to describe how each Terraform variable is to be displayed and consumed, for example, if a parameter is a password or not.
- Cloud Orchestration Manager includes an orchestration engine which provides a language to describe the orchestration flow needed to deploy and manage services.
- Cloud connections that describe the necessary connection parameters to connect to supported clouds.
- The content runtime, a collection of Chef, Software Repository and the Pattern Manager which run on a series of docker containers. The Content Runtime may be deployed from a Terraform template.
- Once deployed, CAM provides an interface to manage day 2 operations using the Terraform Plan/Apply feature.

In this section we will discuss how to utilize Cloud Automation Manager to deploy AIX partitions through IBM PowerVC utilizing Terraform templates providing you with the building blocks to support the life cycle of your AIX workloads.

4.3.1 Terraform

Terraform is open source software that Cloud Automation Manager uses to deploy infrastructure as code. The template shown in Example 4-2 on page 63 can be used to deploy an AIX partition to IBM PowerVC using Terraform directly from the command line interface after configuring an OpenStack provider in your Terraform templates, to configure the OpenStack provider you will need to utilize the same details as described in Table 4-2 on page 58.

A sample Terraform provider configuration is shown in Example 4-1 where we authenticate to a PowerVC server with IP address 9.47.76.108 using the user camsvc with password abcd1234.

Example 4-1 Sample provider.tf for Terraform deployment to PowerVC

```terraform
provider "openstack" {
  user_name = "camsvc"
  tenant_name = "ibm-default"
  password  = "abcd1234"
  auth_url  = "https://9.47.76.108:5000/v3"
  domain_name = "Default"
}
```
To deploy AIX utilizing the Terraform engine directly you will need to obtain and install the Terraform command line interface on an system with connectivity to your IBM PowerVC environment.

You can download and learn more about Terraform at:

https://www.terraform.io

### 4.3.2 Configuring cloud connection

To enable IBM Cloud Automation Manager (CAM) to communicate with IBM PowerVC you will need first create a cloud connection to IBM PowerVC. Cloud connections specify the credentials and configuration in order to deploy workloads to a cloud provider. Creating the cloud connection is a one time step per IBM PowerVC environment you will be deploying to.

**Table 4-1  Sample IBM PowerVC details**

<table>
<thead>
<tr>
<th>Configuration item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address</td>
<td>9.47.76.108</td>
</tr>
<tr>
<td>Domain</td>
<td>Default</td>
</tr>
<tr>
<td>Region</td>
<td>RegionOne</td>
</tr>
<tr>
<td>Project Name</td>
<td>ibm-default</td>
</tr>
<tr>
<td>User ID</td>
<td>camsvc</td>
</tr>
<tr>
<td>Password</td>
<td>abcd1234</td>
</tr>
</tbody>
</table>

In Table 4-1 we show sample information for configuration of your cloud connection in Cloud Automation Manager. The values for domain, region, and project name are the default values for the default project. In this example we have pre-created a user named camsvc to utilize as our IBM PowerVC user for the cloud connection rather then a user with full administrative privileges. Detailed information regarding configuration and setup of IBM PowerVC as well as useful OpenStack commands are available in *IBM PowerVC Version 1.3.2 Introduction and Configuration*, SG24-8199.
Using these details after logging into Cloud Automation Manager you can select **Manage → Cloud Connections** as shown in then select **Create Connection** on the proceeding page.

![Figure 4-3 Creating a cloud connection](image)

On the following page, you will be prompted for your cloud connection details to allow Terraform to communicate with the instance of IBM PowerVC. Based on your IBM PowerVC installation details, you will need to fill in the configuration page to suit your environment ensuring you update the authentication URLs IP address to your IBM PowerVC instances IP address including the port and v3 as this is where the API for keystone identity API resides. In Table 4-2 on page 58 we show the required values you would enter in when using the configuration details of the IBM PowerVC instance.

The default domain in IBM PowerVC is **Default**, the default region is **RegionOne** and the default project name is **ibm-default**.

<table>
<thead>
<tr>
<th>Field</th>
<th>Connection configuration input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Provider</td>
<td>OpenStack</td>
</tr>
<tr>
<td>Connection Name</td>
<td>&lt;Name of IBM PowerVC connection&gt;</td>
</tr>
<tr>
<td>Connection Description</td>
<td>&lt;Description of connection&gt;</td>
</tr>
<tr>
<td>Authentication URL</td>
<td>https://&lt;IP address of PowerVC instance&gt;:5000/v3</td>
</tr>
<tr>
<td>Username</td>
<td>&lt;User in IBM PowerVC to login as&gt;a</td>
</tr>
<tr>
<td>Password</td>
<td>&lt;Password of IBM PowerVC user&gt;</td>
</tr>
<tr>
<td>Domain Name</td>
<td>Default</td>
</tr>
<tr>
<td>Region</td>
<td>RegionOne</td>
</tr>
<tr>
<td>Project Name</td>
<td>ibm-default</td>
</tr>
</tbody>
</table>

a. This userid will need to have appropriate privileges within PowerVC to perform the deployment.

When configuring the cloud connection the first step is to select the Cloud Provider and then entering a connection name description as shown in Figure 4-4 on page 59.
Figure 4-4   Selecting your cloud provider for cloud connection

After performing this step, this will display the remainder of the dialogues you will need to fill in to finish configuring your connection based on your environments configuration. A OpenStack connection configuration page being filled out is shown in Figure 4-5.

Figure 4-5   Entering configuration details for OpenStack connection

1 - Select “OpenStack.”
2 - Enter a name for your connection.
3 - Enter a description for your connection.
4 - Enter the OpenStack domain.
5 - Enter the OpenStack region.
6 - Enter the OpenStack project name.
7 - Click “Create.”
After clicking **Create** you will be prompted for a final confirmation window as seen in to save the connection.

![Success window](image)

**Figure 4-6  Saving your cloud connection**

### 4.3.3 Creating template for AIX deployment

The fundamental component for the deployment of an image within Cloud Automation Manager (CAM) is a template. A template defines a set of resources that should be provisioned during deployment. We can use Terraform templates to define how we want new logical partitions deployed through IBM PowerVC with information such as:

- Storage template to utilize
- Compute template to utilize
- Host group to deploy to
- Network adapters to attach
- Additional disks to allocate

Terraform templates can also be used to trigger provisioners such as the `remote-exec` provisioner to perform additional tasks after the initial partition initiation to run commands on the newly built partition remotely.

Cloud Automation Manager has an integrated interface for designing templates named the Cloud Automation Manager Template Designer. Information on how to use this tool can be viewed at:


In this section we will explore creating templates from scratch and defining them within Cloud Automation Manager. In Cloud Automation Manager you can also import templates sources such as:

- GitHub
- GitLab
- BitBucket Server
- From URL
To create the template after logging into Cloud Automation Manager you will need to open the menu bar from the top left hand side of the interface and then select **Library → Templates** and then select **Import Template** as illustrated in Figure 4-7 on page 61.

![Figure 4-7 Creating a template from scratch in Cloud Automation Manager](image)

After clicking **Import Template** the import template dialogue will appear. On this page you will need to select the import source for the template, name, description and type of cloud provider this template will be used for. As we want to use this template to create an AIX partition using IBM PowerVC and we want to start with an empty template we need to ensure we select the template source as **From Scratch** and select the Cloud provider as OpenStack as shown in Figure 4-8.

![Figure 4-8 Entering new template metadata](image)
After clicking **Import** your new template will be loaded and you will be on a new template page. From this page you will be able to select **Manage Template** which will allow you to add to view the current versions of templates, modify parameters as well as access an in-line editor to perform updates through the web interface. Once at the **Manage Template** page we can edit our template and add initial content, to do the initial edit you must select the sub menu for the **v1.0.0** template and select **Edit** in the menu as shown in Figure 4-9 on page 62.

**Figure 4-9 Opening the in-line editor page**
On the **Edit Version** page you will see three fields allowing you to update the name of the template version, change where to import the data from and enter your actual Terraform template. To create a basic template we will be entering our source directly into the form on this page as shown in Figure 4-10. You can also import from file and upload the file from your browser or provide a URL for Cloud Automation Manager to pull the file from.

![Edit Version](image)

To add a source to this template, please import your template either choosing from a file or a URL.

* Indicates a required field

**Display Name**

v1.0.0

**Import Type**

- Import Source File
- Type or copy & paste a URL here

Upload

Enter your Terraform template here.

![Figure 4-10 Adding your Terraform template](image)

A basic Terraform template to deploy a logical partition will utilize the `openstack_compute_instance_v2` resource. A sample Terraform all in one template is shown in Example 4-2 which creates a single logical partition with a single user defined network and a data volume. A provisioner to execute a local-exec to sleep for 600 seconds is included in this template to provide the new partition to establish an RMC connection prior to allocating its data volumes. By utilizing variables in Terraform we are able to allow for user input within Cloud Automation Manager.

**Note:** If you are using self signed certificates on your IBM PowerVC server you will need to ensure you add the `insecure=true` option as shown in Example 4-2 or alternatively configure your cloud connection to utilise the IBM PowerVC certificate when initializing the cloud connection.

To do this, you will need to obtain the certificate file from the IBM PowerVC server and then upload it to your Terraform containers in a location under a persistent volume and then specify the path to the certificate when configuring the cloud connection.

**Example 4-2  Terraform template for creation of a IBM PowerVC partition**

```terraform
provider "openstack" {
```
insecure = true
}

# Create the AIX partition
resource "openstack_compute_instance_v2" "powervc_lpar" {
  name = "${var.hostname}"
  image_name = "${var.image_name}"
  flavor_name = "${var.flavor_name}"
  availability_zone = "${var.availability_zone}"

  network {
    name = "${var.network_name}"
  }

  provisioner "local-exec" {
    command = "sleep 600;"
  }
}

# Create a volume on the specified storage template
resource "openstack_blockstorage_volume_v3" "datavg" {
  size = "${var.datavg_size}"
  description = "${var.hostname} datavg_1"
  name = "${var.hostname}_datavg_1"
  volume_type = "${var.volume_type}"
  multiattach = true
  enable_online_resize = true
}

# Attach the datavg volume to the partition
resource "openstack_compute_volume_attach_v2" "datavg_attach" {
  instance_id = "${openstack_compute_instance_v2.powervc_lpar.id}"
  volume_id = "${openstack_blockstorage_volume_v3.datavg.id}"
}

# Variables for deployment
variable "hostname" {
  type = "string"
  description = "Hostname of server"
}

variable "image_name" {
  type = "string"
  description = "IBM PowerVC Image to deploy"
}

variable "flavor_name" {
  type = "string"
  description = "IBM PowerVC Compute Template (size)"
}

variable "network_name" {
  type = "string"
  description = "IBM PowerVC Network Name"
variable "availability_zone" {
    type = "string"
    description = "IBM PowerVC availability zone"
}

variable "datavg_size" {
    type = "string"
    description = "IBM Size of datavg in gigabytes"
}

variable "volume_type" {
    type = "string"
    description = "Storage template for datavg"
}

---

**Note:** In Example 4-2 on page 63, we have the Terraform template represented as a single file due to the in-line editor we are utilizing to enter the information into Cloud Automation Manager. This template could be split into separate files if being imported from another source. Refer to for details on standard Cloud Automation Manager structure:


After inserting the template in Example 4-2 on page 63 into the field shown in Figure 4-10 on page 63 and clicking the **Update** button you will need to enable the parameter inputs within Cloud Automation Manager for this template. To do this, on the **Manage Template** page you would need to navigate to the **Edit Version** page and then select **Update Parameters** as shown in Figure 4-11.
On the **Update Parameters** page, you will need to select **From Template Source** as the import source. This will automatically generate your parameter listing based on the variables in your Terraform template. Ensure you click **Update** before leaving this page as shown in Figure 4-12 on page 66.

![Figure 4-12](image)

After you have generated the parameters, you now have all the requirements for deploying a template in Cloud Automation Manager. If required, you can edit your parameters and set default values, update the field display names and mark whether a field is sensitive or not. After a new template has been created it can be used in services or deployed as is. Services can be used to add in additional business logic required for deployments.
4.3.4 Deploying AIX template

To deploy an AIX partition you first navigate to your newly created template by selecting **Library → Templates** and selecting the template you wish to deploy as shown in Figure 4-13.

![Figure 4-13   Selecting template to deploy](image)

After selecting the template to deploy you will need to click on the **Deploy** button located on the bottom right hand side of the page. Here you will enter the details of the deployment. Here you will be prompted for the **Namespace** you want to deploy to in Cloud Automation Manager and an **Instance Name** you want the specific deployment of the template to be known as. In this example we use the **services** namespace and choose **AIXbasic** as the instance name as shown in Figure 4-14 on page 67. For the **Cloud Connection** field, select the connection matching the name you created when adding the IBM PowerVC OpenStack provider.

![Figure 4-14   Entering basic details for instance deployment](image)
After doing this you will be able to see which cloud connections are available to the namespace you selected and you will get the list of additional parameters required by the template, from the template in Example 4-2 on page 63 the following fields will need to be filled:

- availability_zone - This will reference the **Host Group** within IBM PowerVC to deploy to.
- datavg_size - Size of the additional disk to allocate in gigabytes.
- flavor_name - Name of the **Compute Template** to deploy.
- hostname - Hostname of the partition to deploy.
- image_name - The image name from IBM PowerVC for the image you wish to deploy.
- network_name - The name of the network in IBM PowerVC you wish to deploy to.
- volume_type - The **Storage Template** in IBM PowerVC you wish to use for your data volume.

**Note:** The openstack provider in Terraform does not allow you to specify a storage connectivity group to deploy to. To use a specific storage connectivity group for a deployment you must create a flavor to map to that storage connectivity group. To update the storage connectivity group you will need to login to your IBM PowerVC server and set the `powervm:storage_connectivity_group` key by using the command:

```
nova flavor-key medium-scg1 set powervm:storage_connectivity_group="<ID of storage connectivity group>"
```
After entering all the details and pressing the **Deploy** button as shown in Figure 4-15.

### 4. Additional Options

- **availability_zone**
  - Default Group

- **datavg_size**
  - 10

- **flavor_name**
  - tiny

- **hostname**
  - aixpar

- **image_name**
  - AIX_7.2

- **network_name**
  - VLAN2230

- **volume_type**
  - storwize-v7000 base template

---

**Figure 4-15** Entering additional details for instance deployment

1. Fill in all additional options.
2. Click “Deploy.”
After initiating the template deploy the build process will be initiated. If the build encounters any errors they will be shown in the Log File pane. On successful deployment you will be presented with a window similar to Figure 4-16 on page 70. This page will show details of your deployment including which IBM PowerVC management device manages the partition, the last state the partition was placed into by Cloud Automation Manager, template version used for the deployment as well as the IP address of the newly created partition.

After deploying the workload you also have the ability to destroy the workload when it is no longer required, power on, and power off the workload.

The provisioning process for your workloads can be further extended by adding provisioners to your Terraform templates or by utilizing Cloud Automation Managers content runtimes.

A list of Terraform supported provisioners can be viewed on the Terraform website: [https://www.terraform.io/docs/provisioners/index.html](https://www.terraform.io/docs/provisioners/index.html)

Cloud Automation Manager (CAM) also comes with a selection of CAM content providers, for information related to these providers see: [https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.1.0/content/cam_content_terraform_provider.html](https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.1.0/content/cam_content_terraform_provider.html)

### 4.4 Ansible Automation and AIX

Ansible is an open source third party tool that can be used for configuration management and automation of repetitive tasks. Ansible is agent-less and performs actions on a set of servers from a Ansible control node. The Ansible engine only needs to be installed on the system that will be operating as the control node.

Ansible communicates with your AIX based system securely using the OpenSSH protocol and can utilize either password based authentication or SSH key based authentication.
With Ansible you define the desired state of a system and allow Ansible to make changes to match the desired state, for example to ensure a certain fileset is installed or to ensure a attribute of a particular AIX device is set to the desired value.

To learn more about Ansible see “Get Started” at:

https://www.ansible.com

In this section we will briefly describe how to install Ansible on AIX and show the current state of AIX support in Ansible.

### 4.4.1 Installing Ansible on an AIX control node

The Ansible engine can be installed on a variety of platforms including AIX

For AIX based systems the Ansible engine is available as part of the “AIX Toolbox for Linux Applications” located at:


A simpler alternative for installation, if you have configured YUM package manage for AIX, is to simply run `yum install ansible` as shown in Example 4-3

**Example 4-3  Installing Ansible using YUM on AIX**

```bash
# yum install ansible
AIX_Toolbox
  | 2.9 kB  00:00:00
AIX_Toolbox/primary_db
  | 1.4 MB  00:00:00

... Output truncated ...

Total download size: 33 M
Installed size: 165 M
Is this ok [y/N]: y
Downloading Packages:
  (1/13): ansible-2.7.0-1.aix6.1.ppc.rpm
  | 13 MB  00:00:02
  (2/13): gmp-6.1.2-1.aix6.1.ppc.rpm
  | 1.6 MB  00:00:00
  (3/13): libgcc-8.1.0-2.aix7.2.ppc.rpm
  | 976 kB  00:00:00

... Output truncated ...

Installing: python-httplib2-0.9.2-1.noarch
12/13
Installing: ansible-2.7.0-1.ppc
13/13

Installed:
  ansible.ppc 0:2.7.0-1

Dependency Installed:
```
Complete!

```bash
# ansible --version
ansible 2.7.0
    config file = /etc/ansible/ansible.cfg
    configured module search path = ['/.ansible/plugins/modules',
        '/usr/share/ansible/plugins/modules']
    ansible python module location =
        /opt/freeware/lib/python2.7/site-packages/ansible
    executable location = /usr/bin/ansible
    python version = 2.7.10 (default, Jun 22 2016, 05:57:59) [C]
```

For details regarding how to install and configure YUM on AIX see

### 4.4.2 AIX specific Ansible modules

As of Ansible version 2.8, Ansible now has the modules shown in Table 4-3 available for use specific to AIX. These modules are not the complete subset of modules available in Ansible but only a small portion which are AIX specific.

<table>
<thead>
<tr>
<th>Module</th>
<th>Minimum Ansible version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aix_devices</td>
<td>2.8</td>
<td>Used to res-can a device, remove a device or modify a devices attributes or state</td>
</tr>
<tr>
<td>aix_filesystem</td>
<td>2.8</td>
<td>Used for creating, removing or extending AIX file systems</td>
</tr>
<tr>
<td>aix_inittab</td>
<td>2.3</td>
<td>Used for creating, removing or changing AIX inittab entries</td>
</tr>
<tr>
<td>aix_lvg</td>
<td>2.8</td>
<td>Used for creating, removing or extending AIX volume groups</td>
</tr>
<tr>
<td>aix_lvol</td>
<td>2.4</td>
<td>Used for creating, removing or resizing AIX logical volumes</td>
</tr>
<tr>
<td>installp</td>
<td>2.8</td>
<td>Used for installing and removing AIX packages</td>
</tr>
<tr>
<td>mksysb</td>
<td>2.5</td>
<td>Creates a mksysb image of an AIX system</td>
</tr>
</tbody>
</table>

Additionally, third-party custom modules created by the open source community for AIX are available on the AIXOSS GitHub project located at:
[https://github.com/aixoss/ansible-playbooks](https://github.com/aixoss/ansible-playbooks)

These custom Ansible modules can be used to perform additional AIX and VIOS maintenance tasks such as but not limited to:

- Performing automatic downloads of fix packs using Service Update Management Assistant (SUMA)
- Performing updates, rebooting and checking the status of Network Installation Manager (NIM) clients
- Check for vulnerabilities on AIX systems using the Fix Level Recommendation Tool (FLRT)
- Performing updateios tasks on VIOS

### 4.5 Chef Infra client on AIX

Chef Infra is a third party configuration management tool which operates on a server-client model. Chef Infra client agents are available for AIX 7.1 as well as AIX 7.2. Chef Infra allows for you to turn your infrastructure into code and allows you to deploy environments in a testable and repeatable manner. Chef Infra can also be used for ongoing maintenance of your environment running cookbooks periodically against your AIX systems to apply required changes on your environment including tasks such as patch management and security configuration.

The Chef Infra client communicates with the Chef Infra server securely using the HTTPS protocol to and a pull model requiring clients to authenticate with key based authentication with the Chef Infra server. Push jobs can also be initiated on Chef Infra clients from the Chef Infra server if required.

Along with a large number of built in resources that support AIX, Chef also has a large number of community contributed modules in the Chef supermarket. In relation to AIX, the open source community has contributed many custom resources that can be used during Chef cookbook development simplifying your code and management of your AIX environment. These custom resources are available on the Chef supermarket and can readily be used as part of your development efforts. Custom resources currently exist for most aspects of management of AIX, information regarding the AIX cookbook with custom resources is available on the Chef Supermarket at:

https://supermarket.chef.io/cookbooks/aix

To learn more about Chef you see the documentation at:

https://docs.chef.io/chef_overview.html

### 4.6 Puppet Enterprise on AIX

Puppet is a third party configuration management tool and IT automation software that defines and enforces the state of your infrastructure throughout your software development cycle. Puppet is written in a declarative language in which you specify the desired state and Puppet performs the steps required to meet that state.

Puppet operates on a client-server model however you can also operate a client in standalone mode. Puppet Enterprise supports running the Puppet agent on AIX 6.1/7.1/7.2.

For information on Puppet Enterprise visit:

https://puppet.com
4.7 AIX container technology and Kubernetes

This section discusses the ability to use AIX as a worker node utilizing AIX Workload Partitions (WPARs).

4.7.1 AIX Workload Partitions

Workload partitions (WPARs) are virtualized operating system environments within a single instance of the AIX operating system. WPARs secure and isolate the environment for the processes and signals that are used by enterprise applications.

The workload partition (WPAR) environment is different from the standard AIX operating system environment. Various aspects of the system, such as networking and resource controls, function differently in the WPAR environment.

The WPAR information describes how to install applications in a WPAR environment using various applications such as Apache, DB2® and WAS. These examples are not intended to imply that they are the only supported versions or configurations of these applications.

While the WPAR environment is similar to the environment for an application on a stand-alone system, there are differences that some applications can recognize. Applications might recognize differences in the following items:

**Kernel and privileges**

For applications that provide kernel extensions, the user-level application execution environment is compartmentalized between WPAR instances. However, the shared kernel recognizes all WPAR instances and must maintain the compartmentalization.

The WPAR root user does not have the same level of access as the root user in the global environment by default. Some devices are accessible within a WPAR by default. Storage devices might be mounted as file systems from the global environment into the WPAR or they can be exported to the WPAR so that file systems can be created, mounted, or removed from within the WPAR. Storage device adapters might also be exported to a system WPAR which is not a versioned WPAR, giving the WPAR complete control over configuring devices for that adapter.

**Statistics and tuning**

While several AIX system utilities have been enabled to work either fully or partially in a WPAR, there are some utilities that work only in the global environment (for example, the mpstat utility and the sar utility). Because all WPAR instances share the same kernel and certain kernel subsystems are optimized for system-wide usage, it is not possible to tune all kernel subsystem parameters from a WPAR.

You cannot modify the network configuration (for example, addresses and routes) inside a WPAR. The default configuration of a WPAR prevents applications from using any raw sockets, but system administrators can enable them.

Please see the following links for more information on AIX Workload Partitions setup and configurations:

https://www.ibm.com/support/pages/aix-wpars-how
4.7.2 AIX worker nodes with IBM Cloud Private

With recent introductions of packages to the AIX Toolbox for Linux Applications you can now run AIX worker nodes on an IBM Cloud Private cluster. This allows you to capitalize on the benefits of running in a Kubernetes environment with your AIX workloads. You can deploy native AIX containers by deploying native AIX containers running AIX workloads. For information regarding support and licensing you can refer to the AIX Toolbox for Linux Applications website for details at:


This section discusses the basic concept of Kubernetes and how to utilize AIX WPARs in a IBM Cloud Private environment.

Using AIX worker nodes with Kubernetes

Workload Partitions provide an isolated application environment that minimizes potential interactions with system objects or processes outside the WPAR by running in a virtualized OS environment within a single instance of AIX.

Even though a WPAR uses the same operating system instance as other WPARs or the global environment, the ability to view or interact with processes or devices outside the partition is limited. These WPARs own a private root file system, users and groups, login, network space and an administrative domain.

In a typical Kubernetes cluster, the kubelet service uses the Container Runtime Interface (CRI) to interface with a container runtime environment, the most common of which is Docker. Utilizing packages available in the AIX Toolbox for Linux, IBM Cloud Private can also use the CRI to interface with AIX and WPARs through a criwpard service. The criwpard service implements the Container Runtime Interface of Kubernetes. It runs on every AIX worker node. It is the interface between the kubelet service that interacts with the Kubernetes API server running on the master nodes and the Workload Partitions.

The crihelper process is part of criwpar package that is the helper process which runs in every Workload Partition. The kube-proxy service manages virtual IP addresses of Kubernetes services. It runs in a pod on every worker node.

Containers on AIX are implemented using shared Workload Partitions (WPARs). Consequently, they have their /usr and /opt file systems mounted from the host in read-only mode. This must be considered when building container images for AIX. Applications running in AIX containers that need to write to a specific subdirectory under the /usr directory or /opt directory can overlay it using a namefs mount, provided that the directory already exists on the host. It is important to note, this will mask the global definition of the directory you overlay.

When utilizing AIX as a worker node, we utilize WPAR backup files created using savewpar. These container images can be pushed and pulled from a container registry, such as the image manager provided by IBM Cloud Private using the pushwpari and pullwpari commands provided in the criwpar RPM package from the AIX Toolbox for Linux. The packages required for installation and use of WPARs as worker nodes on AIX are described in Table 4-4 on page 76.
Containers can be built in a similar manner to traditional Docker images using a Dockerfile compliant file and the `mkwpari` command. This command can be used to create new AIX container images either from scratch or based on a pre-existing container image you have previously created. For more information regarding image management refer to the user guide located on the AIX Open Source software community site at:


When using AIX worker nodes on IBM Cloud Private you will need to take note of the following:

- Container images must be deployed on the same AIX level they were created on. If the Kubernetes cluster contains AIX nodes that are at different levels, you can use Kubernetes labels and node selectors to make sure images are deployed on supported nodes.
- Kubernetes pods deployed on AIX cannot have more than one container. The deployment will fail if the pod has more than one container. You can however, use separate pods or run multiple applications in the same container.
- AIX containers can attach to a limited set of volume types. These include FC, iSCSI, NFS and GPFS (the latter through hostPath mounts only).
- Only volumes with JFS or JFS2 file systems can be mounted.

### Installation of the IBM Cloud Private AIX worker node packages

To install manually, the RPM packages listed in Table 4-4 on page 76 can be downloaded, along with their pre-requisites, from the AIX Toolbox for Linux Applications.

**Table 4-4   Package list required for Kubernetes**

<table>
<thead>
<tr>
<th>Kubernetes packages list</th>
</tr>
</thead>
<tbody>
<tr>
<td>calico-cni-3.3.1</td>
</tr>
<tr>
<td>calico-node-3.3.1</td>
</tr>
<tr>
<td>calicoctl-3.3.1</td>
</tr>
<tr>
<td>criwpar-0.2.0</td>
</tr>
<tr>
<td>icp-worker-3.1.2</td>
</tr>
<tr>
<td>kubectl-1.12.4</td>
</tr>
<tr>
<td>kubernetes-node-img-1.12.4</td>
</tr>
<tr>
<td>kubelet-1.12.4</td>
</tr>
<tr>
<td>cloudctl-3.1.2.0</td>
</tr>
<tr>
<td>helm-2.9.1</td>
</tr>
</tbody>
</table>

See AIX Toolbox for Linux Applications link


Alternatively, you can configure `yum` repository and just run `yum install icp-worker` command as described in Example 4-4

**Example 4-4   Running yum to install Kubernetes dependencies on AIX**

```bash
# yum install icp-worker
Setting up Install Process
Resolving Dependencies
```
--> Running transaction check
--- Checking packages
(1) Package icp-worker.ppc 0:3.1.2-1 will be installed
(2) Processing Dependency: kubernetes-node-img = 1.12.4 for package: icp-worker-3.1.2-1.ppc
(3) Processing Dependency: criwpar = 0.2.0 for package: icp-worker-3.1.2-1.ppc
(4) Processing Dependency: calico-node = 3.3.1 for package: icp-worker-3.1.2-1.ppc
(5) Processing Dependency: kubectl = 1.12.4 for package: icp-worker-3.1.2-1.ppc
(6) Processing Dependency: cloudctl = 3.1.2.0 for package: icp-worker-3.1.2-1.ppc
(7) Processing Dependency: calico-cni = 3.3.1 for package: icp-worker-3.1.2-1.ppc
(8) ...
Total download size: 200 M
Installed size: 891 M
Is this ok [y/N]:
...
Installing : kubelet-1.12.4-1.ppc 3/8
Installing : calico-cni-3.3.1-1.ppc 4/8
...
0513-071 The kubelet Subsystem has been added.

To configure yum on AIX, please see Section 3.1, “Redundant EtherChannel” on page 50 or refer to
the following link for more information:

https://developer.ibm.com/articles/configure-yum-on-aix

The packages list provides a set of services to manage containers and networks on AIX
nodes. It also provides a set of commands to create and manipulate container images on AIX.

After installing the packages, you can verify the installation by running the rpm -qa command
or by filtering the rpm -qa command output as shown in Example 4-5.

Example 4-5 Checking packages installed

```
# rpm -qa | egrep -i 'wpar|kub|cali|cloud|helm|icp'
cloud-init-0.7.5-4.3.ppc
calico-node-3.3.1-1.ppc
calico-cni-3.3.1-1.ppc
cloudctl-3.1.2.0-1.ppc
icp-worker-3.1.2-1.ppc
criwpar-0.2.0-2.ppc
kubelet-1.12.4-1.ppc
kubernetes-node-img-1.12.4-1.ppc
cloudctl-1.12.4-1.ppc
```

Note: The IBM Cloud Private worker binaries are installed under /opt/freeware/bin. It is
recommended to add this directory to the AIX PATH environment variable

Configuration of an AIX worker node for IBM Cloud Private

To configure the worker node on AIX, please refer to the Kubernetes on AIX user guide
available in the AIX Open Source Software community site at:


Understanding Pods

A pod is the basic execution unit of a Kubernetes application, the smallest and simplest unit in
the Kubernetes object model that you create or deploy. It represents processes running on
your Cluster. A Pod encapsulates an application’s container (or, in some cases, multiple containers), storage resources, a unique network IP, and options that govern how the containers should run.

A Pod represents a unit of deployment: a single instance of an application in Kubernetes, which might consist of either a single container or a small number of containers that are tightly coupled and that share resources.

Pods in a Kubernetes cluster can be used in two main ways.

**Pods that run a single container**
The one-container-per-Pod model is the most common Kubernetes use case; in this case, you can think of a Pod as a wrapper around a single container, and Kubernetes manages the Pods rather than the containers directly.

**Pods that run multiple containers that need to work together**
A Pod might encapsulate an application composed of multiple co-located containers that are tightly coupled and need to share resources. These co-located containers might form a single cohesive unit of service—one container serving files from a shared volume to the public, while a separate “sidecar” container refreshes or updates those files. The Pod wraps these containers and storage resources together as a single manageable entity. Pods are designed to support multiple cooperating processes (as containers) that form a cohesive unit of service. The containers in a Pod are automatically co-located and co-scheduled on the same physical or virtual machine in the cluster. The containers can share resources and dependencies, communicate with one another, and coordinate when and how they are terminated.
AIX and PowerVM features

This chapter provides an short overview of the wide range of features of IBM Power systems, PowerVM Hypervisor, and VIO server and explains briefly how they are supported and integrated with AIX.

This chapter discusses the following sections:

- “Storage access” on page 80
- “Network access” on page 80
- “Dynamic LPAR support” on page 81
- “Virtual processors” on page 82
- “Simultaneous multi-threading and logical processors” on page 82
- “Dynamic Platform Optimizer” on page 82
- “Active System Optimizer and Dynamic System Optimizer” on page 83
- “Shared Storage Pools” on page 83
- “PowerVM NovaLink” on page 87
- “Power Enterprise Pools” on page 90
- “Linux on Power” on page 95
- “Virtual I/O Server enhancements” on page 95
5.1 Storage access

This section includes a short description of virtualization features, protocols, adapters, access methods, and devices available to access to storage devices.

AIX has native access to the following physical storage devices:

Direct Attached Storage Devices (DASD) These are physical storage devices directly attached to IBM Power systems using a wide range of disk adapters, controllers, enclosures, and I/O drawers. They can be SCSI, SAS, SATA, SSD, or Flash devices. Some of these devices exceed 1 TB in size, have I/O latencies smaller than 1 ms, and provide hundreds of thousands of IOPS.

Storage Area Network (SAN) Storage These are physical storage devices located in external storage systems manufactured by IBM or other storage manufacturers. Access is possible using various protocols such as Fibre Channel, Fibre Channel over Ethernet or iSCSI. FC adapters used to get access to SAN storage systems can have 4 ports and speed of 32 Gb/s.

Tape devices These are tape libraries manufactured by IBM or other third party manufacturers. Access is possible using various protocols such as Fibre Channel or SAS.

Optical devices These are devices such as DVD-RAM, DVD-ROM and CD-ROM. Access is possible using SCSI or USB protocol.

AIX can have access to virtualized storage using the following methods:

Virtual SCSI This method provides standard SCSI compliant access using virtual SCSI interfaces. This method can be used to get access to various types of disk storage systems, tape libraries, and optical devices.

Virtual Fibre Channel This method provides standard FC compliant access using virtual Fibre Channel interfaces. This method can be used to get access to various types of disk storage systems and tape libraries.

iSCSI This method provides standard SCSI compliant access over Ethernet interfaces. This method can be used to get access to various types of IP-based disk storage systems.

Both Virtual SCSI and Virtual Fibre Channel implementations use a client/server model in which VIOS server adapters control and mediate access to real physical devices. VIOS, AIX virtual device drivers, and POWER Hypervisor work together to ensure that each logical partition has access only to its own data. VIOS controls the data flow and performs DMA operations to transfer data between logical memory ranges assigned to different partition; the POWER Hypervisor controls the mapping of logical to physical memory ranges.

5.2 Network access

This section includes a short description of virtualization features, protocols, adapters, access methods, and devices available to get network access.

5.2.1 Dedicated adapters

AIX has native support for physical Ethernet network adapters that can have 4 ports and speeds of 1, 10, 25, 40, and 100Gb.
5.2.2 Virtual Ethernet adapters

This type of virtual adapter allows AIX LPARs to send and receive network traffic without having a physical Ethernet adapter. The POWER Hypervisor provides a software implementation of an Ethernet switch that is compliant with standard IEEE 802.1Q. This switch allows operating systems running in logical partitions to communicate using standard networking protocols.

Logical partitions can have multiple Virtual Ethernet adapters. Each Virtual Ethernet adapter is connected to a POWER Hypervisor software switch. The POWER Hypervisor is invoked for transmission of each Ethernet frame and copies the data between logical partition memory areas. Virtual Ethernet adapters and POWER Hypervisor switch provide the means for efficient inter-partition communication at memory speed. Because the virtual switch functions are provided by the POWER Hypervisor, communication between logical partitions does not require configuration of a VIOS.

5.2.3 VIOS Shared Ethernet Adapter

VIOS Shared Ethernet Adapter (SEA) function provides connectivity to external networks for logical partitions that have only Virtual Ethernet adapters. The SEA acts like a layer 2 bridge to the physical adapters.

5.2.4 vNIC adapters

This type of virtual adapter allows AIX LPARs to send and receive network traffic using virtual functions incorporated in SR-IOV capable physical network adapters. This advanced type of virtual adapter supports QoS, and definition of up to 6 backing physical adapters.

AIX can have redundant network access using the following methods:

5.2.5 Network Interface Backup (NIB)

This method combines one primary and one backup adapter. When the primary interface fails the traffic is rerouted to the backup adapter. The adapters used for NIB can be physical, virtual, or a combination of both.

5.2.6 Ether Channel and IEEE 802.3ad Link Aggregation

These network port aggregation technologies allow multiple Ethernet adapters to be grouped to form a single pseudo Ethernet device. The pseudo Ethernet device will have the network bandwidth of all aggregated adapters and will continue to be operational in case of failure of individual adapters.

5.3 Dynamic LPAR support

Dynamic LPAR allows IBM Power systems to dynamically add and delete selected system resources from AIX LPARs while they are running. Resources subject to dynamic LPAR operations include processor cores, memory, physical I/O adapters, and virtual adapters such as virtual SCSI adapters or virtual Fibre Channel adapters. Adding and removing resources from AIX LPARs can be performed without restarting the LPARs or the AIX operating systems running in LPARs.
The minimum amount of memory that can be added or removed depends on the Logical Memory Block size used by the IBM Power system hosting the LPAR.

Virtual I/O adapters that can be added or removed can be virtual SCSI, virtual Fibre Channel, and virtual Ethernet adapters.

5.4 Virtual processors

Virtual processors introduce an abstract layer between the operating system and the physical cores. The operating systems perceive only virtual processors and act like they have physical cores available. Processes run by the operating system are assigned to virtual processors, which are dispatched by the POWER Hypervisor on actual physical cores. The POWER Hypervisor uses virtual processors to encapsulate all data relevant to logical partitions state from a processor core perspective at a specific point in time.

Virtual processors represent a key component of IBM PowerVM that enables IBM Power Systems to support processor core sharing and micro partitioning. Both dedicated and micro partitions use virtual processors.

POWER Hypervisor uses a para virtualization strategy. This means that operating systems are enhanced to become hypervisor-aware and use a well-defined set of interfaces. AIX operating system has been significantly enhanced to benefit from all capabilities provided by the Hypervisor.

5.5 Simultaneous multi-threading and logical processors

IBM POWER systems can run more than one thread per core. This feature is named Simultaneous Multi-threading (SMT). AIX supports changing the number of threads dynamically without restarting the operating system using `smtctl` command. You can change the number of threads per core to balance between throughput and performance.

In general, more threads per core improve overall system throughput whereas lesser threads per core lead to faster thread execution which leads to improved application response time.

For IBM POWER9 systems running AIX 7.2 with Technology Level 3 default SMT has been changed to SMT8. This leads to up to 1536 threads available in full-system LPAR.

5.6 Dynamic Platform Optimizer

The Dynamic Platform Optimizer (DPO) is a PowerVM virtualization feature allows you to assess and improve partition memory and processor placement (the degree of *affinity*) on IBM Power systems running firmware level 760 or later.

DPO determines an optimal resource placement strategy for IBM Power systems based on the partitions’ configuration and hardware topology on the system. It then performs a sequence of memory and processor relocations to LPARs’ affinity and overall system affinity. This process occurs dynamically while the partitions are running.

Starting with versions AIX 6.1 Technology Level 8 and AIX 7.1 Technology Level 2 AIX is a DPO-aware operating system.
5.7 Active System Optimizer and Dynamic System Optimizer

The Dynamic Platform Optimizer (DPO) is a PowerVM virtualization feature allows you to assess and improve partition memory and processor placement (the degree of affinity) on IBM Power systems running firmware level 760 or later.

DPO determines an optimal resource placement strategy for IBM Power systems based on the partitions’ configuration and hardware topology on the system. It then performs a sequence of memory and processor relocations to LPARs’ affinity and overall system affinity. This process occurs dynamically while the partitions are running.

Starting with versions AIX 6.1 Technology Level 8 and AIX 7.1 Technology Level 2 AIX is a DPO-aware operating system.

Additional details regarding DPO can be found in IBM PowerVM Virtualization Managing and Monitoring, SG24-7590.

5.8 Shared Storage Pools

A shared storage pool is a pool of storage area network (SAN) storage devices that can be used among Virtual I/O Servers. It is based on a cluster of Virtual I/O Servers and a distributed data object repository with a global namespace. Each VIOS that is part of a cluster represents a cluster node.

It provides a better usage of the available storage by using thin provisioning. The thinly provisioned device is not fully backed by physical storage if the data block is not in actual use.

Shared storage pools provide the following benefits

- Improve the usage of available storage.
- Simplify administration tasks.
- Simplify the aggregation of large numbers of disks among the Virtual I/O Servers.

On the Virtual I/O Server (VIOS) Version 2.2.0.11, Fix Pack 24, Service Pack 1, and later, you can create shared storage pools. Shared storage pools provide distributed storage access to all VIOS logical partitions in a cluster.

Note the following considerations for SSP against the VIOS version

- On VIOS Version 2.2.0.11, Fix Pack 24, Service Pack 1, a cluster consists of only one VIOS partition. VIOS Version 2.2.1.0 supports only one cluster in a VIOS partition.
- On VIOS Version 2.2.1.3, or later, a cluster consists of up to four networked VIOS partitions.
- On VIOS Version 2.2.2.0, or later, a cluster consists of up to 16 networked VIOS partitions. You can create a cluster with an Internet Protocol version 6 (IPv6) address that is configured on the VIOS logical partition.

Thus, a cluster consists of up to 16 VIOS logical partitions with a shared storage pool that provides distributed storage access to the VIOS logical partitions in the cluster. Each cluster requires a separate repository disk and shared storage pool disks. The shared storage pool
can be accessed by all the VIOS logical partitions in the cluster. All the VIOS logical partitions within a cluster must have access to all the physical volumes in a shared storage pool.

It simplifies cloud management and efficiency of storage usage. Starting PowerVM 2.2.6 it includes a VIOS SSP performance enhancement known as flash acceleration. This feature can transparently increase a client's workload performance by utilizing SSP flash storage caching on the VIOS.

In VIOS version 3.1, Shared Storage Pool (SSP) Management data is stored in the PostgreSQL database. All data files of the database are stored in the file system of the SSP cluster pool. If the VIOS node that manages the SSP database is unable to access the file system of the SSP cluster pool, while the PostgreSQL database process is performing an I/O operation, the PostgreSQL database aborts all operations and generates the core memory dump. The PostgreSQL database also generates the pool file system errors and stores them in the system error log file. The SSP database automatically recovers when the VIOS node that manages the SSP database regains access to the file system of the SSP cluster pool.

5.8.1 SSP consideration and procedures

The following section discusses all the considerations to take into account and the procedures to walk through a shared storage pool setup.

Before you create shared storage pools, ensure that all logical partitions are preconfigured by using the Hardware Management Console (HMC) as described in this topic. The following are the supported number of characters for the names:

- Cluster: 63
- Storage pool: 127
- Failure group: 63
- Logical unit: 127

**SSP VIOS partitions considerations**

If you plan to build 16 VIOS logical partitions, you will have to consider the following items:

- There must be at least one CPU and one physical CPU of entitlement.
- The VIOS logical partitions must be configured as a VIOS logical partitions.
- The VIOS logical partitions must consist of at least 4 GB of memory.
- The VIOS logical partitions must consist of at least one physical Fibre Channel adapter.
- The rootvg device for a VIOS logical partition cannot be included in storage pool provisioning.
- The associated rootvg device must be installed with VIOS Version 2.2.2.0, or later.
- The VIOS logical partition must be configured with sufficient number of virtual server Small Computer Serial Interface (SCSI) adapter connections required for the client logical partitions.
- The VIOS logical partitions in the cluster requires access to all the SAN-based physical volumes in the shared storage pool of the cluster.
- The VIOS logical partition must not be a mover service partition or a paging partition.
- You cannot use the logical units in a cluster as paging devices for PowerVM Active Memory Sharing (AMS) or Suspend/Resume features.
- In shared storage pools, the Shared Ethernet Adapter must be in threaded mode.

**Client logical partitions considerations**

The client logical partitions should meet the following considerations:

- The client logical partitions must be configured as AIX or Linux client systems with at least 1 GB of minimum memory.
Each client logical partition must be configured with enough virtual SCSI adapter connections to map to the virtual server SCSI adapter connections of the required VIOS logical partitions.

Network addressing considerations

For the networking, you have to ensure a stable connectivity among the VIOS logical partitions and the client logical partitions. See the following considerations to take into account:

- Uninterrupted network connectivity is required for shared storage pool operations. The network interface that is used for the shared storage pool configuration must be on a highly reliable network, which is not congested.
- Ensure that both the forward and reverse lookup for the host name that is used by the VIOS logical partition for clustering resolves to the same IP address, either by a native hosts file or by a reliable Domain Name Server (DNS).
- Each host name of each VIOS logical partition that belongs to the same cluster must resolve to the same IP address family, which is either Internet Protocol version 4 (IPv4) or IPv6.
- With the VIOS Version 2.2.2.0, or later, clusters support Internet Protocol version 6 (IPv6) addresses. Therefore, VIOS logical partitions in a cluster can have host names that resolve to an IPv6 address.

**Note:** In a cluster configuration, you cannot change the host name of a VIOS logical partition.

To change the host name of one VIOS logical partition node SSP cluster, you can just delete the cluster and change the host name, then recreate the cluster.

To change the host name for two or more VIOS logical partitions in the cluster, can just remove the VIOS logical partition from the cluster and change the host name. Then, re-add the VIOS logical partition to the cluster again with the new host name.

In VIOS 2.2.3.0 or later, by default, the shared storage pool cluster is created in a unicast address mode. In earlier VIOS versions, the cluster communication mode is created in a multicast address mode. When the cluster versions are upgraded to VIOS Version 2.2.3.0, the communication mode changes from multicast address mode to unicast address mode as part of rolling upgrade operation.

When a cluster is created, you must specify one physical volume for the repository physical volume and at least one physical volume for the storage pool physical volume. The storage pool physical volumes are used to provide storage to the actual data generated by the client partitions. The repository physical volume is used to perform cluster communication and store the cluster configuration. The maximum client storage capacity matches the total storage capacity of all storage pool physical volumes. The repository disk must have at least 1 GB of available storage space. The physical volumes in the storage pool must have at least 20 GB of available storage space in total.

Each of the VIOS logical partitions assigns hdisk names to all physical volumes available through the Fibre Channel ports. The VIOS logical partition might select different hdisk numbers for the same volumes to the other VIOS logical partition in the same cluster. For example, the viosA1 VIOS logical partition can have hdisk9 assigned to a specific SAN disk, whereas the viosA2 VIOS logical partition can have the hdisk3 name assigned to that same disk. For some tasks, the unique device ID (UDID) can be used to distinguish the volumes. Use the `chkdev` command to obtain the UDID for each disk.

Use any method that is available for the SAN vendor to create each physical volume with at least 20 GB of available storage space. Map the physical volume to the logical partition Fibre...
Channel adapter for each VIOS in the cluster. The physical volumes must be mapped only to
the VIOS logical partitions connected to the shared storage pool.

5.8.2 SSP set up procedure

To set up a valid SSP cluster for at least two VIOS logical partitions, you will have to have
shared disks mapped from the SAN storage side to both of the nodes (or to the VIOS logical
members nodes that will be part of SSP cluster).

In our scenario, hdisk2, hdisk3 and hdisk4 are shared disks among two VIOS logical
partitions. hdisk2 as a repository disk, and hdisk3 and hdisk4 for the shared pool.

1. The following Example 5-1 shows how to create SSP cluster using `cluster` command with
option `-create`, specifying the repository disk, pool name and the pool disks members.

```
Example 5-1   Creating SSP cluster

VIOS1$ cluster -create -clusternname ssp1 -repopvs hdisk2 -spname demosp -sppvs
              hdisk3 hdisk4 -hostname vios1
```

2. The next thing is to create the logical unit (LU) to be used for a client logical partition. The
LU size will be 10G, and the command used is `mkbdsp` as shown in Example 5-2.

```
Example 5-2   Creating a logical unit to be used for a client logical partition

VIOS1$ mkbdsp -clusternname ssp1 -sp demosp 10G -bd lparA_lu1
Lu Name:lparA_lu1
Lu Udid: ad67212f2cbb000833f5dbd282d5ae7
```

3. After creating the LU, you will map the LU created to a Virtual Host (vhost) named vhost0.
The Example 5-3 shows how to map the LU to vhost0 with a Virtual Target Device (VTD)
name.

```
Example 5-3   Mapping the LU to virtual host with a VTD name

VIOS1$ mkbdsp -clusternname ssp1 -sp demosp -bd lparA_lu1 -vadapter vhost0 -tn
              lparA_rootvg
```

The previous Example 5-3 shows `lparA_lu1` as LU name, `lparA_rootvg` as VTD name.

4. You have this SSP cluster with only one node vios1, you probably need to add at least
another node to make a use of this SSP cluster.

The `cluster` command will be used with option `-addnode` as shown in Example 5-4.

```
Example 5-4   Adding a second VIOS node

VIOS1$ cluster -addnode -clustername ssp1 -hostname vios2
```

Now the SSP cluster has two VIOS nodes, and you can carry out any other SSP commands
and tasks from the second node.

The Logical Units (LUs) are represented as VSCSI disks in the client logical partition side.

The very common scenario is that you want to increase the LU size in which it will reflects in
the logical partition VSCSI disk side.

5. In VIOS 2.2.40 and later, you can optionally increase the size of an existing SSP LU.
Example 5-4 below goes through increasing the LUN from 10G to 20G.
The LU name is lparA_lu1, which is a rootvg LUN for some specific client logical partition.

Example 5-5  Increasing dynamically the LU size

```
VIOS2$ lu -resize -lu lparA_lu1 -size 20G
Logical unit lparA_lu1 with udid 'ad67212f2cbb000833f5dbd2822d5ae7' has been successfully changed.
```

```
VIOS2$ lu -list
POOL_NAME: demosp
TIER_NAME: SYSTEM
LU_NAME         SIZE(MB)        UNUSED(MB)      UDID
lparA_lu1       20480           10240
ad67212f2cbb000833f5dbd2822d5ae7
[...]
```

In the AIX client logical partition you will have to run `chvg -g` command against the rootvg in which it has the disk that has been resized. So `chvg -g` will just examine all the disks in the volume group to see if they have grown in size. If any disks have grown in size, it will attempt to add additional PPs to physical volume and hence to the volume group.

Please see the following technote for more information about SSP setup


5.9 PowerVM NovaLink

PowerVM NovaLink is a Linux based software that provides on-host management capabilities to your Power system. PowerVM NovaLink is available on Power 8 or higher based systems. The NovaLink application can run on either a Red Hat Enterprise Linux 7.3 or Ubuntu Linux based PowerPC® 64 Little Endian (PPC64LE) based logical partition. An appliance based installation is also available for running a Ubuntu Linux based installation of PowerVM NovaLink. PowerVM NovaLinks integrated OpenStack drivers also allow it to be used as part of your existing OpenStack based management solution to deploy and manage both AIX and Linux based partitions on PowerVM environments.

In this section we will discuss:

- The components of PowerVM NovaLink
- Software Defined Infrastructure (SDI) capabilities
- Resource Monitoring Control (RMC) communication
- NovaLink and Hardware Management Consoles

For more information on PowerVM NovaLink see:

5.9.1 Components of PowerVM NovaLink

NovaLink is composed five components which fall into two categories:

- PowerVM NovaLink Core Services which provide communication between the NovaLink system and the PowerVM Hypervisor (PHYP) as well as:
  - Command-line interface (CLI) for shell interaction with PowerVM and management of your PowerVM environment
  - Representational state transfer (REST) application programming interface (API) for management of the system
- OpenStack drivers that can be utilized by OpenStack based management solutions, the following OpenStack drivers are available on NovaLink:
  - Nova compute driver for managing a PowerVM system as a compute node for deployments
  - Neutron network driver for providing network connectivity to logical partitions using Shared Ethernet Adapters
  - Ceilometer performance monitoring agents for providing metrics for compute, storage, and network

The core components of PowerVM NovaLink can be updated quickly and simply using the package manager for the respective operating system it resides on, dpkg on Ubuntu Linux based installations and RPM on Red Hat Enterprise Linux based installations. As the components of PowerVM NovaLink are simply packages on the Operating Systems they reside on, this allows you to install and manage your PowerVM NovaLink systems using monitoring and management tools of your choosing. It also allows for the PowerVM NovaLink systems to be secured to your own security requirements.

5.9.2 Software Defined Infrastructure (SDI) Capabilities

PowerVM NovaLink allows you to also abstract your network and storage capabilities through two different operating modes. Creating a software-defined environment is covered in detail in the *Building a SAN-less Private Cloud with IBM PowerVM and IBM PowerVC*, REDP-5455. NovaLink supports two modes of software defined infrastructure.

**Software Defined Network (SDN)**

Software Defined Networking (SDN) provides an abstraction of network hardware through software based solutions. PowerVM NovaLink provides this through industry standard Open vSwitch (OVS) technology. In this mode, PowerVM NovaLink can host some or all networks on a Power system.

**Software Defined Environment (SDE)**

PowerVM NovaLink can also operate in the PowerVM I/O Software Defined Environment (SDE) mode. In this mode, PowerVM NovaLink takes the place of Virtual I/O Servers and providing network and storage to the logical partitions hosted on the system.

**Note:** At the time of publication PowerVM NovaLink must be installed on Ubuntu Linux to utilize SDE functionality.
5.9.3 Resource Monitoring Control (RMC) communication

With PowerVM NovaLink you simplify configuration of RMC and therefore dynamic logical partition operations. NovaLink utilizes an internal virtual switch named MGMTSWITCH running on VLAN ID 4094 to communicate to logical partitions and the Virtual I/O Servers on the managed system using IPV6 adapters on each partition with self assigned addresses.

Inter-partition communication over the MGMTSWITCH is prevented as the switch is created in Virtual Edge Port Aggregator (VEPA) mode. VEPA is part of the IEEE 802.1Qbg standard. In this mode, although all partitions on the managed system are connected to the same MGMTSWITCH on VLAN 4094 the partitions are unable to communicate to one another as traffic can only be sent and received to the virtual network interface card configured as the trunk adapter. In NovaLink environments, the trunk adapter for the MGMTSWITCH network is assigned to the NovaLink partition and does not provide external bridging.

By communicating over the internal virtual switch, this eliminates the requirement of having firewall to open firewall rules between the management console and the logical partitions on a system.

It is important to note, communication over the IPV6 MGMTSWITCH network for RMC only provides NovaLink with connectivity to the partitions, traditional RMC connectivity requirements still apply if you wish to perform dynamic operations from your Hardware Management Console (HMC) through a physical network.

5.9.4 PowerVM NovaLink and Hardware Management Consoles

It is important to note that although some PowerVM NovaLink provides similar functionality to a Hardware Management Console (HMC) in terms of logical partition management and managing your PowerVM environment, they compliment each other as well where an environment which has them co-manage an environment provides the benefits of both. Unlike a HMC, PowerVM NovaLink servers are not appliances. Table 5-1 shows how the capabilities of NovaLink and Hardware Management Consoles differ, but this also shows how they compliment each other. By utilizing both in a co-management mode you can modernize your AIX infrastructure with the capabilities offered by PowerVM NovaLink while maintaining the hardware management abilities offered by Hardware Management Consoles. To run in co-management mode you will require an HMC of at least version 8.4.0, service pack 1, or later.

Table 5-1  PowerVM NovaLink and Hardware Management Console capabilities

<table>
<thead>
<tr>
<th>Task</th>
<th>PowerVM NovaLink</th>
<th>Hardware Management Console</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runs on system it manages</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Concurrent firmware update</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Disruptive firmware update</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Hardware monitoring</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Service agent call home</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Perform dynamic logical partition updates</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Enables OpenStack connectivity</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Software-defined Network capabilities</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
This section introduces the concepts, features and benefits of IBM Power Enterprise Pools (PEP), a technology which allows sharing of processor cores and memory among a pool of IBM Power Systems. PEP concept is the natural response to an increased demand for flexibility and responsiveness.

PEPs are available in two editions. First edition uses the concept of mobile capacity, while the second edition uses the concept of utility capacity. This section briefly introduces the components and benefits for both editions.

### 5.10.1 Power Enterprise Pools first edition

The initial version of PEP is intended to provide the means to benefit from resources located on different IBM Power systems. It is centered on the concept of resource activation.

IBM Power systems have hardware resources such processor cores and memory and are shipped with processor books and memory cards. Depending on the machine type and model, Power system can have a certain number and type of cores and memory modules physically installed. As the result of the purchasing process, some or all of this resources can be activated before the system is shipped. For example, an IBM Power system may have 32 cores and 8 TB of memory installed, but only 16 cores and 6 TB of memory can be activated. All installed resources that are inactive at the time of shipment can be activated any time later using an activation code.

Activation codes are unique to each and every IBM Power system and are based on system machine type, model and serial number. They are available on IBM CoD site at the following link:


To activate installed and inactive physical resources, the activation code must be entered into the system using either HMC or ASMI interface.

At the core of the first edition of PEP are activations, either memory or cores. Activations can be divided into the following categories:

<table>
<thead>
<tr>
<th>Task</th>
<th>PowerVM NovaLink</th>
<th>Hardware Management Console</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software-defined Environment capabilities</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Simplified resource management control through internal virtual switch</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Redundant Configuration</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can manage more than one system</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>PowerVM NovaLink</th>
<th>Hardware Management Console</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified resource management control through internal virtual switch</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Redundant Configuration</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can manage more than one system</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

At the core of the first edition of PEP are activations, either memory or cores. Activations can be divided into the following categories:

- **Static**: The resources associated to this type of activation are activated and bound to the IBM Power system where they have been activated. They cannot be moved from one system to another.

- **Mobile**: The resources associated to this type of activation, as the name implies, are mobile and can be activated on any IBM Power system that is part of the pool and has inactive resources.
can float from one IBM Power system to another and can be active only on one IBM Power system at a time. At any point in time the amount of usable resources on any individual system is the sum of static and mobile resources currently activated on the system.

dark These resources associated to this type of activation are physically installed in IBM Power systems, but are not activated. They are activated when mobile, static or CoD activation codes are used.

Each IBM Power system maintains its own history of activations. Replacement of physical parts does not affect activations. Static activations can be converted to mobile activations. There are rules which control how many static activations each system must have and what percentage of the activations can be mobile. Static and mobile activations are transparent to LPARs, at any point in time an individual LPAR can use any type of resources or a mix of them.

This edition of PEP essentially used the capability of the HMC to activate mobile resources among different Power systems members of the same pool.

First edition of PEP distinguishes between two types of pools:

Midrange This type of pool can include IBM Power 770+ (9117-MMD), IBM Power E870(9119-MME), IBM Power E870C(9080-MME), and IBM Power E880C (9080-MHE) systems. Systems with different clock rates and Power technologies can be part of the same pool.

High end This type of pool can include IBM Power 780+(9179-MHD), IBM Power 795(9119-FHB), IBM Power E880(9119-MHE), IBM Power E870C(9080-MME), and IBM Power E880C (9080-MHE) systems. Systems with different clock rates Power technologies can be part of the same pool.

Resource assignment and movement to different systems part of the same pool are controlled by an HMC which acts as a pool master. Master HMC manages the activation keys and knows at any point in time which Power systems are present in the pool, the amount of statically activated resources on each system, how many mobile activated resources exist in the pool, and on which system exactly they are activated. Movement of mobile activations among different Power systems is non-disruptive.

Power systems that are included in a pool must be HMC managed and can belong to only one pool at any point in time. The HMC does not require access to IBM and resources can be moved at any time without contacting IBM. A single HMC can manage multiple pools. The same HMC can also manage systems that are not part of any pool.

This version of PEP can coexist with the usage of Elastic (On/Off) CoD that is used to activate dark cores or memory for specific period of times.

Requirements for PEP first edition

PEP first edition has the following requirements:

- Only select IBM Power7 or IBM Power8 systems can be included in midrange or high end pools.
- There is a minimum amount of static processor activations on each system in the pool as follows:
  - minimum of 4 static processor activations for Power 770 systems
  - minimum of 4 static processor activations for Power 780 systems
- minimum of 8 static processor activations for Power E870 systems
- minimum of 8 static processor activations for Power E880 systems
- Power 795 systems require minimum of 24 or 25% of the installed processors (whichever is larger) to be statically activated.

- There are minimum amounts for memory installed and activated as follows:
  - 50% of installed memory must be active
  - 25% of the active memory must be static
- Memory activation is done in increments of 100GB.

Additional details, both technical and non-technical, about this version of PEP and its requirements can be found in *Power Enterprise Pools on IBM Power Systems*, REDP-5101.

Enablement of PEP first edition
The process of enabling PEP includes both technical and non-technical steps as follows:

1. Power systems eligible for PEP are ordered with static activations and mobile activations. Mobile activations will initially ship as static.
2. The customer signs a Power Enterprise Pool contract. This allows the definition of a pool.
3. The customer signs an addendum to the contract which allowing adding or removing Power systems from an existing pool.
4. The customer orders a MES to add Power Enterprise Pools enablement Feature Code to the systems and converts the initially-static mobile activations to full mobile.
5. The customer downloads an XML file and imports it to the pool master HMC.
6. The customer orders deactivation codes for static activations that have been converted into mobile.
7. The customer enters the deactivation codes on each Power system.

5.10.2 Power Enterprise Pools second edition

IBM Power Enterprise Pools 2.0 is a completely new model of sharing compute resources. This new model is available only for IBM Power E980 systems and, as opposed to the previous model, all installed processor cores and memory are activated and available for immediate use. As such there is no need to move mobile resources from one system to another or to convert static resources into mobile resources.

Whereas the previous edition of PEP is a resource-centric model, this new edition is a time-based and uses as “pay-as-you-go” model.

Each system in the pool has a permanent base activation. This base is a subset of the resources physically installed on the E980 systems when they are purchased.

This base capacity includes base processor activations for all operating system supported by Power E980, such as Base Linux Processor Activations, Base AIX software license entitlements, or Base IBM i software license entitlements, as well as Base 1 GB Memory activations. Base processor and memory activations are aggregated at pool level and determine the overall pool base capacity.

Resource usage in terms of processor and memory is monitored at system level every minute and then aggregated at pool level. Resource usage at pool level is metered and any usage that exceeds pool based capacity is charged accordingly. There are five types of charges, four of which are core-related and one for memory.
Power E980 systems are featured with PowerVM Enterprise edition and LPARs can run AIX, IBM i, or any of the supported Linux distributions. Depending on the operating systems installed in the LPARs there may be one or more of the following types of processor-related capacity charges:

- **Any OS core**: This type is generic and the cores may run any of the operating systems supported by Power E980 systems.
- **Linux or VIOS core**: These cores may run any supported Linux distribution or VIOS software. There are no software charges for Linux or VIOS partitions, however valid Linux license entitlement must be acquired separately to ensure that all cores or sockets used for Linux LPARs are licensed properly.
- **AIX software**: These cores may run only AIX software.
- **IBM i software**: These cores may run only IBM i software.

Core usage is metered by tracking *actual* core usage of LPARs in the pool.

Memory usage is metered by tracking memory *assignment* to active LPARs, not by memory actually used by the operating system running in the LPARs.

The metering solution leverages IBM Cloud as a single point of control and management and requires a connection to IBM Cloud Management Console (CMC). Because this solution is based on metering each master HMC must have must have Network Time Protocol (NTP) enabled and Performance and Capacity Monitoring (PCM) must be enabled for each Power E980 system included in the pool.

The customers must purchase in advance *capacity credits* that will be used when the pool usage exceeds the base capacity of the pool. Capacity credits can be purchased from IBM, IBM Business Partners or the IBM Entitled Systems Support web site. Capacity credits are charged in real time.

IBM CMC provides a wide range of features that allow for pool, usage and budget management such as:

- Pool management
- Defining thresholds for systems and partitions, budget and credit balance
- Defining alerts at pool level based upon budget and resource consumption
- Detailed analysis of usage of resources such as metered minutes, credits, system or pool level cores and memory
- Monitoring and management of capacity credit budget


**Requirements for PEP second edition**

PEP second edition has the following requirements:

- Only IBM Power E980 (9080-M9S) systems are supported.
- There can be maximum 16 BM Power E980 systems in the pool.
- There can be maximum 500 LPARs in the pool.
- All LPARs must be shared, dedicated and dedicated-donating LPARs are not supported.
Minimum 25% of installed processors must be activated using the base processor activation.

Minimum 50% of installed memory activated must use the base memory activation.

Memory activation is done in increments of 256 GB.

All Power E980 servers in the pool must have a connection through an HMC to an IBM Cloud Management Console.

Capacity credits must be purchased in advance.

Cloud Edition software is required for each base processor activation.


Enablement of PEP second edition

The process of enabling PEP includes both technical and non-technical steps as follows:

1. Power E980 systems are ordered with base processor and memory activations and all corresponding software license entitlements.
2. The customer purchases an initial amount of capacity credits.
3. The customer starts the PEP on the IBM ESS web site using the serial number of a Power E980 system and receives a pool ID.
4. The customer connects to IBM Cloud Management Console and defines a PEP using the pool ID.
5. All installed processor and memory resources are activated on all Power E980 systems included in the pool and CMC starts monitoring the pool.
6. Resources used in excess of the pool base are debited against the capacity credits by CMC in real time. ESS web site is also updated daily.

This version of PEP can not coexist with other CoD capabilities.

5.10.3 Comparison between PEP first and second edition

This section includes a comparison between the two editions of PEP. Table 5-2 on page 94 includes several criteria useful to differentiate between them.

Table 5-2   Comparison between PEP editions

<table>
<thead>
<tr>
<th>Criterion</th>
<th>PEP first edition</th>
<th>PEP second edition</th>
</tr>
</thead>
</table>
| Power systems supported        | IBM Power 795 (9119-FHB)  
IBM Power 770+ (9117-MMD)  
IBM Power 780+ (9179-MHD)  
IBM Power E870 (9119-MME)  
IBM Power E880 (9119-MHE)  
IBM Power E870C (9080-MME)  
IBM Power E880C (9080-MHE) | IBM POWER9 E980  
(9080-M9S)                  |
| Support for dedicated LPARs    | Yes                                                                              | No                               |
| Maximum number of systems in the pool | No limit                                                                | 16                               |
| Maximum number of LPARs in the pool | No limit                                                              | 500                               |
| HMC required                   | Yes                                                                              | Yes                              |
| CMC connectivity required      | No                                                                               | Yes                              |
5.11 Linux on Power

If your system is a Power8 or later system utilizing PowerVM you can deploy AIX, Linux and IBM i partitions. When deploying Linux partitions you can configure either PowerPC64 little endian partitions, or PowerPC64 big endian partitions. AIX and IBM i only support big endian, however with Linux you can select based on the distribution and your application requirements. PowerVM supports mixing of both big endian and little endian partitions on the same physical system allowing you to run your Linux workloads along side your AIX partitions.

When deploying Linux on Power (LOP) you can utilize the same automation tools mentioned in Chapter 4., “Virtualization and cloud capabilities” on page 51 to allow you to automate your deployment of your Linux environments. This allows you to create a truly heterogeneous environment and leverage the benefits of PowerVM for your Linux workloads.

For supported Linux distributions for POWER8 and POWER9 see: https://www.ibm.com/support/knowledgecenter/en/linuxonibm/liaam/liaamdistros.htm

5.12 Virtual I/O Server enhancements

Virtual I/O Servers are software appliances which can be utilized to share physical hardware resources on Power systems with logical partitions on the system. This section introduces enhancements to the Virtual I/O Server software with the release of Virtual I/O Server 3.1.
5.12.1 Key features

With the release of Virtual I/O Server 3.1 the underlying base operating system has been upgraded to AIX 7.2 TL 3. With this upgrade this brings along benefits such as:

- VIOS 3.1 has been streamlined with removal of unnecessary file sets resulting in quicker updates and a smaller installation size
- iSCSI over virtual SCSI (vSCSI) support including iSCSI multipathing I/O support (MPIO)
- Multipathing enhancements
- Simplified migrations using viosupgrade

5.12.2 N-Port ID Virtualization (NPIV)

The N_Port ID Virtualization (NPIV) is an industry-standard technology that helps you to configure an NPIV capable Fibre Channel adapter with multiple, virtual worldwide port names (WWPNs). When using 32 Gb/s Fibre Channel (FC) adapters you can now utilize 255 virtual Fibre Channel client adapters per physical port. When using 16 Gb/s or lesser adapters, the maximum limit of 64 active virtual Fibre Channel client adapters still applies.

Using virtual Fibre Channel connectivity for your client partitions provides a secure a method to securely share a physical Fibre Channel adapter and allows client partitions to have native access to the target. You can use virtual Fibre Channel adapters for connectivity to storage mapped from SAN devices or connectivity to fibre channel connected tape devices. A key benefit of this technology is that your AIX partition can access the storage using native storage drivers.

For information on how to use and configure virtual fibre channel see IBM PowerVM Getting Started Guide, REDP-4815.

5.12.3 iSCSI support

Internet Small Computer Systems Interface (iSCSI) provides block-level access to storage devices by carrying SCSI commands over a Internet Protocol network. iSCSI is used to facilitate data transfers over Internet using TCP, a reliable transport mechanism using either IPV6 or IPV4 protocols.

As of Virtual I/O Server 3.1, Virtual I/O Server can now operate as iSCSI initiators and access iSCSI target devices on both POWER8 and POWER9 systems. iSCSI support allows iSCSI disks to be presented to logical partitions as virtual disks. If you are using a POWER8 system your systems firmware level must be at FW860.20 or later. All firmware levels for POWER9 systems provide iSCSI support. Virtual I/O Server 3.1 also enables MPIO support for the iSCSI initiator.

As of Virtual I/O Server 3.1.0 the use of iSCSI devices on Virtual I/O Server has the following limitations:

- Virtual I/O Server may not use iSCSI devices as boot devices.
- File based discovery policy for iSCSI is not supported, discovery mode must be set to ODM.
- Use of iSCSI volumes to create logical volume backed devices is not supported.
- iSCSI volumes may not participate in shared storage pools as either repository disks or shared storage pool disks.
iSCSI volumes may not be used for paging devices for Active Memory Sharing (AMS) or Remote restart capabilities.

When using an iSCSI device, virtual lpars cannot use client_reserve and mirrored attributes.

iSCSI volumes can be presented to client virtual servers through virtual SCSI. When an iSCSI target can be accessed over multiple network paths you can utilise MPIO to configure multiple paths to the storage provider presenting the disk to one or multiple Virtual I/O Servers. Figure 5-1 on page 97 illustrates data flow from an iSCSI storage provider over two network paths. In this case, if iSCSI volumes were mapped to the Virtual I/O Server to be presented to the virtual partition over SCSI the client LPAR would see a single virtual disk per presented LUN with two paths. This would provide redundant connectivity at both the Virtual I/O Server level as well as the client logical partition.

The commands lsisci, mkiscsi, chiscsi, and rmisci can be used to manage your iSCSI devices. The iSCSI commands available on Virtual I/O Server are the same as on AIX 7.2.

5.12.4 Upgrading Virtual I/O Server to 3.1

With the introduction of Virtual I/O Server 3.1 a simplified upgrade procedure is available using the viosupgrade command VIOS 2.2.6.30 and above and on Network Installation Manager (NIM) servers running AIX 7.2 TL 3 SP1 and above. The traditional upgrade methods are still available however the process is simplified by using the viosupgrade command. The viosupgrade process when initiated on either a Virtual I/O Server directly or from a NIM server requires an unused disk to be available on the Virtual I/O Server as the upgrade process initiates a alt disk restore of the Virtual I/O Server 3.1 mksysb image onto the disk. It is important to note that the upgrade to Virtual I/O Server 3.1 is actually a fresh install of Virtual I/O Server 3.1 with virtual and logical mapping configuration restored. It will not restore customized configuration for files such ntp.conf, netsvc.conf, not any Virtual I/O Server security settings you have applied in the past. Third party applications will not be restored either, so if you are planning to upgrade you must take steps to ensure you backup any relevant third party application data you require to restore after the upgrade. In addition, if the rootvg of the Virtual I/O Server has any logical volumes that are mapped to the logical partitions these will need to be migrated off of the rootvg as the upgrade will result in a clean installation of Virtual I/O Server 3.1 with virtual mappings restored.

We recommend updating your Virtual I/O Server to the latest level of 2.2 prior to performing a migration to Virtual I/O Server 3.1. In addition, you should ensure you have the latest available images for Virtual I/O Server 3.1 media downloaded for use for the upgrade. As
always, it is highly recommended you take appropriate backups of the Virtual I/O Server configuration using `viosbr` before performing the upgrade.

When upgrading the Virtual I/O Server from NIM you will need to ensure you create the appropriate spot (for bosinst upgrades only) and mksysb resource, to do so you will need to obtain the mksysb from the VIOS 3.1 Base Install Flash media by loop mounting the ISO and copying the file out as demonstrated in Example 5-6.

When upgrading the VIOS directly on the VIOS, extracting the mksysb image can be done by utilizing the `viosupgrade` command itself where directoryPath is the directory you want to write the generated mksysb from the ISO files.

Example 5-6  Extracting mksysb from VIOS 3.1 Flash Media

```
# ls -l
total 9447016
-rw-r--r--  1 root    system  4836872192 Oct 18 13:28
Virtual_IO_Server_Base Install_3.1.0.21_Flash_052019.iso
# loopmount -i Virtual_IO_Server_Base Install_3.1.0.21_Flash_052019.iso -o "-V
udfs -o ro" -m /mnt
# ls -l /mnt/usr/sys/inst.images
total 1739226
dr-xr-xr-x  2 root    system         2048 Apr 27 18:16 installp
-rw-r--r--  1 root    system   3561932800 Apr 27 18:16 mksysb_image
# cp /mnt/usr/sys/inst.images/mksysb_image
/export/mksysb/VIO31/VIO_3.1_Flash.mksysb
```

Alternatively you can create the mksysb image by using the two part ISO installation media for VIOS 3.1 as demonstrated below in Example 5-7.

Example 5-7  Extracting mksysb from VIOS 3.1 DVD Installation Media

```
# mkdir /mnt/dvd1
# mkdir /mnt/dvd2
# loopmount -i Virtual_IO_Server_Base Install_3.1.0.20_DVD_1 of 2_052019.iso -o
"-V cdrfs -o ro" -m /mnt/dvd1
# loopmount -i Virtual_IO_Server_Base Install_3.1.0.20_DVD_2 of 2_052019.iso -o
"-V cdrfs -o ro" -m /mnt/dvd2
# ls -l /mnt/dvd1/usr/sys/inst.images
total 6289924
drwxr-xr-x  3 root    system         2048 Apr 11 2019 installp
-rw-r--r--  1 root    system   2146959360 Apr 11 2019 mksysb_image
-rw-r--r--  1 root    system   1073479680 Apr 11 2019 mksysb_image2
# ls -l /mnt/dvd2/usr/sys/inst.images
total 666984
drwxr-xr-x  3 root    system         2048 Apr 11 2019 installp
-rw-r--r--  1 root    system   341493760 Apr 11 2019 mksysb_image
# cat /mnt/dvd1/usr/sys/inst.images/mksysb_image
/mnt/dvd1/usr/sys/inst.images/mksysb_image2
/mnt/dvd2/usr/sys/inst.images/mksysb_image >
/export/mksysb/VIO31/VIOS_3.1.0.20.mksysb
```

To generate the mksysb from the ISO files on a VIOS itself, you can utilise the command `viosupgrade` as below, this will generate a mksysb in the /home/padmin/mksysb directory.
$ viosupgrade -I /home/padmin/Virtual_Io_Server_Base_Install_3.1.0.20_DVD_1_of_2_052019.iso:/home/padmin/Virtual_Io_Server_Base_Install_3.1.0.20_DVD_2_of_2_052019.iso -w /home/padmin/mksysb

To perform the viosupgrade -l -i <mksysb image file> -a <install disk> using a Virtual I/O Server, the example command below shows the migration of to a pair of alternative disks.

$ viosupgrade -l -i /home/padmin/mksysb/mksysb_image -a hdisk2:hdisk3

In cases where you are upgrading Virtual I/O Server that are utilizing shared storage pools you will also need to include the -c flag. You can also have particular files that you may require copied across to the upgraded cloned images by including a file list using the -g parameter. For more information you can view the viosupgrade documentation in IBM Knowledge Center.

To upgrade a Virtual I/O Server from NIM you will need to ensure you have created a mksysb resource with the Virtual I/O Server 31 mksysb image. For NIM based upgrades there are two options when using viosupgrade, you can either do a bosinst upgrade or an alternative disk upgrade. For bosinst upgrades you will also need to ensure you have created the appropriate SPOT from the VIOS 3.1 mksysb resource. To perform the upgrade using the alternative disk method once you have defined the resources, you would simply run the command viosupgrade -t altdisk -n <vio server> -m <mksysb resource> -a <disk list>. An example of the command is shown below where the vios name is vios1, hdisk2 and hdisk3 are the target disks we are cloning onto for the upgrade and the mksysb resource name is vios31_mksysb.

# viosupgrade -t altdisk -n vios1 -m vios31_mksysb -a hdisk2:hdisk3

To perform a bosinst type upgrade you would simply change the type to bosint, viosupgrade -t bosinst -n <vio server> -m <mksysb resource> -p <VIOS 3.1 spot name>. 

Chapter 5. AIX and PowerVM features  99
Disaster Recovery and High Availability

This chapter presents Disaster Recovery Solutions and High Availability available in IBM Power system environments. It introduces PowerHA and IBM VM Recovery Manager for Power Systems, which are by far the typical solutions used to provide high availability and disaster recovery capabilities for LPARs running AIX.

This chapter highlights the following:

- “IBM VM Recovery Manager for Power” on page 102
- “PowerHA” on page 107
6.1 IBM VM Recovery Manager for Power

The IBM VM Recovery Manager for Power Systems (VMR) is an enterprise grade availability solution that provides automated recovery for virtual machines (VMs) running on IBM Power systems. VMs on IBM Power systems are usually referred to as logical partitions (LPARs).

Unlike OS or application-level clustering solutions VMR is a VM-level technology. On IBM Power systems VMs are treated like containers that can host various operating systems. As such VMR is a operating system agnostic solution. Because VMR does not have any operating system or middle ware dependencies it can be used to deploy uniform HA/DR solutions in heterogeneous environments that may include AIX, IBM i and all Linux distributions supported by IBM Power systems.

There are two VMR versions:
- VM Recovery Manager HA (VMR HA).
- VM Recovery Manager DR (VMR DR), formerly known as Geographically Dispersed Resiliency (GDR).

**VMR HA**

VMR HA provides the capability to move VMs between different IBM Power systems using LPM for planned outages or VM restart for unexpected outages. VMR HA solutions can be deployed in environments where IBM Power systems are located in the same site and typically have access to the same network and storage devices.

During the fail over VMs located on an IBM Power systems are LPM-moved or just restarted on a different IBM Power system. VMR HA is an availability solution that can be very well used in IBM Power system environments where PowerHA is not used.

**VMR DR**

Unlike VMR HA, VMR DR is a disaster recovery solution that provides automated recovery for VMs running on IBM Power systems located in different sites and having access to different storage devices. The distance between primary and disaster recovery sites can range from several to thousands of kilometers. VMR DR relies on an out-of-band monitoring and management component and leverages consistently on storage replication technologies.

During the fail over from primary to secondary site VMR DR orchestrates shutting down VMs on primary site, managing storage level replication between the two site to preserve data consistency, and starting VMs in the secondary site.

VMR HA and VMR DR can be deployed in IBM Power system environments completely virtualized using PowerVM, VIO servers and managed by HMCs. At the time of writing this publication VMR solutions are supported only for IBM Power system environments that use HMCs. When both NovaLink and HMCs are used for management the HMCs must set in master mode.

6.1.1 IBM VM Recovery Manager versions and life cycle

VMR versions available at the time of writing this publication are 1.3 and 1.4.

IBM announcement letter for IBM VM Recovery Manager HA for Power Systems V1.4 and IBM VM Recovery Manager DR for Power Systems, V1.4 is available at:
6.1.2 IBM VM Recovery Manager HA components

This section provides a brief overview of VMR HA and introduces solution main components and capabilities. VMR HA solutions include the following components:

**KSYS controller**
This is the core component of the solution. KSYS acts as an overall orchestrator of the entire solution. It monitors the whole infrastructure, reacts to failures when they occur, communicates and interact with HMCs, IBM Power systems and VIO servers. KSYS runs on an AIX VM which is usually located on an IBM Power system that is not part of the VMR scope to ensure it remains operational even in the unlikely case of a failure of entire managed infrastructure.

**VM agent**
This is an agent which communicates with KSYS and monitors the health status of the VMs and optionally of applications running in the VMs.

**GUI server**
This is the graphical user interface to the KSYS and represents a convenient alternative to the KSYS CLI. It can be used to manage both HA and DR deployments.

**Hardware Management Console (HMC)**
These are the HMCs that manage IBM Power systems. All IBM Power systems included in a VMR HA solution must be managed by HMCs. All HMCs are registered with KSYS and act like an intermediary that provides KSYS access to IBM Power systems, VIO servers, and VMs included in the scope of VMR HA solution.

**Virtual I/O Server (VIOS)**
VIOS servers are used to deploy VMs on IBM Power systems and provide KSYS all required configuration details for all VMs. VIO servers will also run a special process named Host Monitor (HM) that monitors the heartbeats sent by VMs and intermediates the communications with KSYS.

VMR HA has the following key features:

- VM moves using LPM for planned outages.
- Restart-based VM moves for unplanned outages.
- Host grouping for IBM Power systems with similar network and storage connectivity.
- Definition of collocation and anti-collocation policies.
- Definition of VM relocation policies.
- Host, VM and application-level monitoring.
- Cross VM application dependency.
- Enhanced VIOS monitoring.
- GUI and CLI based management.

Additional details as well as implementation guidelines for VMR HA can be found in *Implementing IBM VM Recovery Manager for IBM Power Systems, SG24-8426*
**VMR HA requirements**

Table 6-1 on page 104 includes the requirements for various VMR HA components

<table>
<thead>
<tr>
<th>VMR HA component</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSYS controller</td>
<td>IBM AIX 7.2 with Technology Level 2 Service Pack 1 (7200-02-01), or later.</td>
</tr>
<tr>
<td>HMC</td>
<td>HMC Version 9 Release 9.1.0, or later.</td>
</tr>
<tr>
<td>VIOS</td>
<td>VIOS 3.1.0.1 or later.</td>
</tr>
<tr>
<td>VM operating system</td>
<td><img src="image" alt="list of operating systems" /></td>
</tr>
<tr>
<td>VM agent</td>
<td>At the time of writing this publications the VM agents used to monitor VMs and applications can be installed only on the following operating systems: <img src="image" alt="list of operating systems" /></td>
</tr>
</tbody>
</table>

### 6.1.3 IBM VM Recovery Manager DR components

This section provides a brief overview of VMR DR and introduces main solution components and capabilities. VMR DR solutions typically include the following components:

**Controller system (KSYS)**

This is the core component of the solution which acts as a single point of control for the entire environment managed by the VMR DR solution. KSYS handles discovery, verification, monitoring, notification, and recovery operations to support disaster recovery. The KSYS communicates with the HMC to collect configuration information of the IBM Power managed systems. It also interacts with the VIO servers through the HMC to obtain storage configuration information for the VMs. Additionally KSYS provides storage replication management and Capacity on Demand (CoD) management. KSYS is usually deployed in the DR site so that it is not affected by any issues or failure in the primary site. LPAR hosting KSYS must run IBM AIX 7.2 with Technology Level 1 Service Pack 1, or later.

**Sites**

These are the sites where IBM Power systems are located. Active VMs are located in the primary site, and are failed over to the DR site during the recovery process. The distance between the site is mainly determined by the storage replication technology.
Hosts
These are IBM Power systems located in either of the two sites. All Power systems must be managed by HMCs.

Virtual machines (VMs) or logical partitions (LPARs)
These are logical partitions (LPARs) deployed on IBM Power systems. They are failed over from primary to the DR site during the recovery process.

Storage agents
They interact with storage controllers from various storage systems located in both sites to manage storage specific operations such as starting, stopping, suspending, reversing, resynchronizing, pausing, and resuming storage replication.

Hardware Management Console (HMC)
These are the HMCs that manage IBM Power systems. All IBM Power systems included in a VMR DR must be managed by HMCs. HMCs provide KSYS data about hosts, VIO servers, and VMs. Data collected includes information such as number of processor cores, amount of memory, and worldwide port numbers (WWPNs) of the physical Fibre Channel adapter. The HMC also checks for VIO servers capability for disaster recovery operations.

Virtual I/O Server (VIOs)
VIO servers are used to deploy VMs on IBM Power systems and provide KSYS all configuration details for all VMs, most importantly storage configuration details. VIO servers are not migrated to the DR site during the recovery.

VMR DR has the following key features:

- Increased degree of automation
  - It allows for non-disruptive DR testing and fail over rehearsal.
  - It allows for administrator initiated fail over.
  - It significantly reduces human intervention.
  - It is significantly less prone to errors, especially useful in complex environments.
  - It allows for automatic discovery of changes in the entire environment such as addition or deletion of LPARs, or addition of new LUNs.
  - Host grouping for IBM Power systems.
  - Daily cross-site checks and validation.
  - It allows for automatic notifications.

- Better capacity management
  - It allows for increase and decrease of memory and CPU entitlement for VMs moved to the DR site.
  - It allows for advanced cross-site pairings.

- Single administrative interface
  - GUI and CLI based management
  - Coexistence with VMR HA
  - HADR policies, including Flex Capacity, user scripts, and network mapping for the GUI at the KSYS, site, host group, host, and VM level.

- Full integration with other PowerVM features and Power systems centric solutions
  - VMR DR uses and relies on Hardware Management Consoles (HMC), VIO server, Live Partition Mobility (LPM), Simplified Remote Restart (SRR) capabilities
  - VMR DR solutions for IBM Power systems environments can be complemented with PowerVC and PowerHA.
- Support for Power Enterprise Pools first edition
- Support for Elastic (On/Off) Capacity-on-Demand (CoD)

VMR DR requirements

Table 6-2 on page 106 includes the requirements for VMR HA components:

<table>
<thead>
<tr>
<th>VMR DR component</th>
<th>Requirements</th>
</tr>
</thead>
</table>
| KSYS controller  | - IBM AIX 7.2 with Technology Level 1 Service Pack 1 (7200-01-01), or later.  
- OpenSSL for AIX version 1.0.1.516, or later. |
| HMC              | - HMC Version 8 Release 8.7.1, or later.  
- HMC Version 9 Release 9.1.0, or later. |
| VIOS             | - VIOS Version 2.2.6.30, or later.  
- VIOS Version 3.1.0.21, or later. |
| VM operating system | - AIX Version 6.1, or later.  
- Red Hat Enterprise Linux (little endian, big endian) Version 7.2, or later.  
- SUSE Linux Enterprise Server Version 12.1, or later.  
- Ubuntu Linux distribution Version 16.04.  
- IBM i Version 7.2, or later. |

Support for storage systems

At the time of writing this publication VMR DR supports the following storage systems:

**EMC storage system**
VMR DR supports storage devices for the EMC VMAX family (VMAX1, VMAX2, and VMAX3). The EMC storage devices must be Symmetrix Remote Data Facility (SRDF)-capable. The EMC storage must have Solutions Enabler SRDF family Version 8.1.0.0 installed. Both SRDF/S (Synchronous), and SRDF/A (Asynchronous) replication are supported.

**IBM SVC and Storwize storage systems**
VMR DR supports IBM SVC Version 6.1.0, and later, as well as IBM Storwize® V7000 7.1.0, and later. Both Metro Mirror (synchronous), and Global Mirror (asynchronous) replication are supported.

**IBM System Storage DS8000 series**
VMR DR supports DS8700, or later and DS8000® storage systems with DSCLI version 7.7.51.48, and later. Only Global Mirror (asynchronous) replication is supported.

**Hitachi storage systems**
VMR DR supports the Hitachi Virtual Storage Platform (VSP) G1000, and Hitachi VSP G400 with CCI version 01-39-03/04 and model RAID-Manager/AIX. Only asynchronous replication is supported.

**IBM XIV Storage System and IBM Flash System A9000**
VMR DR supports the XIV® Storage System, and IBM Flash System A9000. Both Metro Mirror (synchronous), and Global Mirror (asynchronous) modes replication are supported.
6.2 PowerHA

PowerHA System Mirror for AIX (formerly known as HACMP and now referred to as PowerHA) is the IBM Power Systems data is the HA solution for applications running on AIX LPARs. It monitors, detects and reacts to a extensive list of events that may affect application availability. PowerHA relies on services provided by Reliable Scalable Cluster Technology (RSCT) and Cluster Aware AIX (CAA).

RSCT is a set of low-level operating system components that allow the implementation of clustering technologies. CAA is an AIX feature that was introduced in AIX 6.1 TL6 and AIX 7.1.

6.2.1 PowerHA editions

PowerHA SystemMirror® for AIX can be distributed in Standard Edition or Enterprise Edition. Standard Edition provides local clustering capabilities. Typically all cluster nodes share a common storage infrastructure and have visibility to the same storage.

Enterprise Edition provides both local and remote replication functions. In this case cluster nodes are usually located in different data centers separated by significant distances and integrate with storage level replication services such as Copy Services, IP Replication or Hyperswap.

6.2.2 PowerHA versions and life cycle

PowerHA versions and life cycle data valid at the time of writing this publication are the following:

- PowerHA 7.2 TL1 is estimated to go out of support on April 30th 2020
- PowerHA 7.2 TL2 is estimated to go out of support on April 30th 2021
- PowerHA 7.2 TL3 is estimated to go out of support on April 30th 2022

Release notes for all technology levels of IBM PowerHA SystemMirror Version 7.2 for AIX, both Standard and Enterprise editions are available at:


Full details for PowerHA SystemMirror support life cycle are available at:

http://www-01.ibm.com/support/docview.wss?uid=isg3T1023563

PowerHA SystemMirror 7.2 provide support for latest AIX and PowerVM enhancements including the following:

- Cluster Aware AIX
- AIX Live Kernel Update
- Power Enterprise Pools
- Live Partition Mobility
- Exploitation of LVM rootvg failure monitoring

Full details about improvements introduced by each PowerHA SystemMirror 7.2 technology level can be found in IBM PowerHA SystemMirror V7.2.3 for IBM AIX and V7.22 for Linux, SG24-8434
6.2.3 AIX requirements for various PowerHA levels

This section provides details for AIX level required for each PowerHA 7.2 Technology level.

PowerHA SystemMirror Version 7.2.0 for AIX is supported on the following version of the AIX operating system:
- IBM AIX 6.1 with Technology Level 9 with Service Pack 5, or later
- IBM AIX 7.1 with Technology Level 3 with Service Pack 5, or later
- IBM AIX 7.1 with Technology Level 4 with Service Pack 1, or later
- IBM AIX 7.2 with Technology Level 0 with Service Pack 1, or later

PowerHA SystemMirror Version 7.2.1 for AIX is supported on the following version of the AIX operating system:
- IBM AIX 7.1 with Technology Level 3 with Service Pack 6, or later
- IBM AIX 7.1 with Technology Level 4 with Service Pack 1, or later
- IBM AIX 7.2 with Technology Level 0 with Service Pack 1, or later

PowerHA SystemMirror Version 7.2.2 for AIX is supported on the following version of the AIX operating system:
- IBM AIX 7.1 with Technology Level 4, or later
- IBM AIX 7.1 with Technology Level 5, or later
- IBM AIX 7.2 with Technology Level 0, or later
- IBM AIX 7.2 with Technology Level 1, or later
- IBM AIX 7.2 with Technology Level 2, or later

PowerHA SystemMirror Version 7.2.3 for AIX is supported on the following version of the AIX operating system:
- IBM AIX 7.1 with Technology Level 3 with Service Pack 9, or later
- IBM AIX 7.1 with Technology Level 4 with Service Pack 4, or later
- IBM AIX 7.1 with Technology Level 5, or later
- IBM AIX 7.2 with Technology Level 3 with Service Pack 9, or later
- IBM AIX 7.2 with Technology Level 4 with Service Pack 4, or later
- IBM AIX 7.2 with Technology Level 5, or later
- IBM AIX 7.2 with Technology Level 0, or later
- IBM AIX 7.2 with Technology Level 1, or later
- IBM AIX 7.2 with Technology Level 2, or later

A full detail PowerHA compatibility matrix is available at:
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD101347

A full detailed list of PowerHA known fixes is available at:

6.2.4 PowerHA licensing

This section introduces several concepts used by PowerHA licensing.

PowerHA licensing is core-based which means that PowerHA must be licensed for each core used by cluster nodes. This is usually enforced by proper entitlement of LPARs.
Because they are core-based licenses for both PowerHA Standard and Enterprise editions depend on the IBM Power systems on which cluster nodes run and can be divided into the following categories:

**Small tier**
- This includes IBM Power systems S914/S814, S922/S822, S924/S824, S950/S850.

**Medium tier**
- This includes IBM Power systems E870/E880/E980.

Time wise, licenses for both PowerHA Standard and Enterprise editions can be divided into the following categories:

**Perpetual licenses**
- This is the traditional method of licensing. The client has the license permanently and is entitled to support for a predefined period of time. The license is packaged with an entitlement for support in the form of software maintenance (SWMA) for a specific period of time which must be at least one year. When SWMA period expires it has to be renewed. In case SWMA is not renewed the client is still entitled to use the product, but it will not be entitled to the benefits of SWMA.

**Termed licenses**
- This a time based license which may very well be used in cloud environments. This type of licensing bundles the license itself and SWMA for a specific period of time. Terms of validity can be 3, 6, 12 or 36 months. At the end of the term the client can order either another termed license or a perpetual license. In case the license is not renewed the client is no longer entitled to use the product.
AIX fundamentals

This chapter explains IBM AIX fundamental concepts and which includes the following topics:

- “Logical Volume Manager” on page 112
- “AIX JFS2” on page 124
- “Role Based Access Control” on page 125
- “Encrypted file systems” on page 135
- “AIXpert and PowerSC integration” on page 143
- “AIX Auditing subsystem and AHAFS” on page 146
- “MultiBOS” on page 152
7.1 Logical Volume Manager

IBM AIX partitioning is based on LVM. The role of LVM is to present a simple logical view of underlying physical storage space which is hard drives. LVM manages individual physical disks to be the individual partitions present on them.

7.1.1 Introduction to the LVM

This section discusses the following topics:
- LVM integration with file systems concept.
- AIX operating system calls with LVM.
- LVM IO with SAN

A logical volume may correspond to a physical partition. One volume may be composed of several partitions located on multiple physical disks. Not only that, the volumes can even be extended while the OS is running and the file system is being accessed.

Now wondering how traditional file systems like FAT or HPFS could be extended at runtime. The answer is, they cannot. To take full advantage of LVM, it is necessary to use a file system designed for it. JFS is not really tied to LVM, both LVM and JFS can exist separately, but only when working in concert both can reach their full potential.

The AIX operating system LVM calls the disks as physical volumes which are divided into fixed areas of space named physical partitions. These PPs are all uniform in size and span from the outer edge of the disk, moving inward toward the center of the spindle. As these PVs are gathered, they become known as volume groups. Within these VGs, the system creates structures named logical volumes that gather PPs into usable areas on which file systems can be created.

The PP size is fixed with the creation of the VG, and further PVs added to the VG are expected to conform the same PP size. Each PP can be assigned to only one LV at a time, and any LV must have a minimum of one PP to exist. If file systems are grown, the minimum size by which they will expand is one PP.

Several mathematics and physics based properties affect disk I/O, latency, and access across disk edges regions. Because of conservation of angular momentum, the outer edge of the disk has a faster rotational velocity than the slower inner edge. However, because not all the data on the disk will be written to the outer edge as a result of the limitations of the physical placement of the data, the fastest seek times for the hard disk heads will be in the center area of the disk, where the head is most likely to pass on average.

The AIX LVM handles the relationships among PVs, VGs, LVs, and file systems. The LVM creates the logical structures on physical media for managing data within the operating system. Relevant to disk optimization, the LVM also provides a means of customizing availability, performance, and redundancy.

The two most widely used features that LVM has to boost optimization are through mirroring and striping. LVM allows you to mirror LVs with up to three copies of the data. So, one LP can point to one or more PPs. This way, should a hardware failure occur, the data is preserved on other physical devices. In smaller systems that use internal storage especially, it is crucial that all data be mirrored to prevent unplanned outages.

Striping places data across multiple hard disks so that multiple read and write operations can occur simultaneously across a wide number of disks. You enable striping by changing the
inter-physical volume allocation policy property for each LV. By setting the policy to the minimum which is the default, there is greater reliability, because no one disk failure could affect the LV. By setting the policy to the maximum, LVM stripes the data across the maximum number of disks within the VG possible, maximizing the number of I/O operations that can occur at once.

Several times we may face disk failure, however it's never a matter of if a disk failure will occur but rather a question of when it will happen. No disk has ever been guaranteed to work indefinitely. The goal of any good systems administrator is to avoid being a victim of the mean time between failure value of hardware and find a way to mitigate the risk of a disk failure. The three main objectives for any AIX administrator are to maximize availability, performance, and redundancy. You'll want your storage environment to be available both to ensure that the data can be accessed on demand and to ensure that there is sufficient disk space to contain all the data. The disk has to have good performance so that applications do not get held up by any I/O wait. The disk needs to have redundancy so that a failure of resources does not impair the server's ability to function.

### 7.1.2 LVM components

LVM has four components forming the LVM structure.

- **Volume group**
- **Physical volume**
- **Logical volume**
- **File systems**

These topics are discussed in the following sections.

**Volume group**

A volume group can be created in different three types.

**Small or normal volume group**

Is a collection of 1 to 32 physical volumes of varying sizes and types.

**Big volume group**

A big can have from 1 to 128 physical volumes.

**Scalable volume group**

A scalable volume group can have up to 1024 physical volumes. A physical volume can belong to only one volume group per system; there can be up to 255 active volume groups.

When a physical volume is assigned to a volume group, the physical blocks of storage media on it are organized into physical partitions of a size you specify when you create the volume group.

When you install the system, one volume group (the root volume group, named rootvg) is automatically created that contains the base set of logical volumes required to start the system, as well as any other logical volumes you specify to the installation script. The rootvg includes paging space, the journal log, boot data, and dump storage, each in its own separate logical volume. The rootvg has attributes that differ from user-defined volume groups. For example, the rootvg cannot be imported or exported. When performing a command or procedure on the rootvg, you must be familiar with its unique characteristics.
You create a volume group with the `mkvg` command. You add a physical volume to a volume group with the `extendvg` command, make use of the changed size of a physical volume with the `chvg` command, and remove a physical volume from a volume group with the `reducevg` command. Some of the other commands that you use on volume groups include: list (`lsvg`), remove (exportvg), install (importvg), reorganize (reorgvg), synchronize (syncvg), make available for use (varyonvg), and make unavailable for use (varyoffvg).

Small systems might require only one volume group to contain all the physical volumes attached to the system. You might want to create separate volume groups, however, for security reasons, because each volume group can have its own security permissions. Separate volume groups also make maintenance easier because groups other than the one being serviced can remain active. Because the rootvg must always be online, it contains only the minimum number of physical volumes necessary for system operation.

You can move data from one physical volume to other physical volumes in the same volume group with the `migratepv` command. This command allows you to free a physical volume so it can be removed from the volume group. For example, you could move data from a physical volume that is to be replaced.

A volume group that is created with smaller physical and logical volume limits can be converted to a format which can hold more physical volumes and more logical volumes. This operation requires that there be enough free partitions on every physical volume in the volume group for the volume group descriptor area (VGDA) expansion. The number of free partitions required depends on the size of the current VGDA and the physical partition size. Because the VGDA resides on the edge of the disk and it requires contiguous space, the free partitions are required on the edge of the disk. If those partitions are allocated for a user's use, they are migrated to other free partitions on the same disk. The rest of the physical partitions are renumbered to reflect the loss of the partitions for VGDA usage. This renumbering changes the mappings of the logical to physical partitions in all the physical volumes of this volume group. If you have saved the mappings of the logical volumes for a potential recovery operation, generate the maps again after the completion of the conversion operation. Also, if the backup of the volume group is taken with map option and you plan to restore using those maps, the restore operation might fail because the partition number might no longer exist (due to reduction). It is recommended that backup is taken before the conversion and right after the conversion if the map option is used. Because the VGDA space has been increased substantially, every VGDA update operation (creating a logical volume, changing a logical volume, adding a physical volume, and so on) might take considerably longer to run.

**Physical volume**

A disk must be designated as a physical volume and be put into an available state before it can be assigned to a volume group.

A physical volume has certain configuration and identification information written on it. This information includes a physical volume identifier that is unique to the system.

The LVM can make use of the additional space that a redundant array of identical disks (RAID) can add to a logical unit number (LUN), by adding physical partitions to the physical volume associated with the LUN.

**Logical volume**

After you create a volume group, you can create logical volumes within that volume group.

A logical volume, although it can reside on noncontiguous physical partitions or even on more than one physical volume, appears to users and applications as a single, contiguous,
extensible disk volume. You can create additional logical volumes with the `mklv` command. This command allows you to specify the name of the logical volume and define its characteristics, including the number and location of logical partitions to allocate for it.

After you create a logical volume, you can change its name and characteristics with the `chlv` command, and you can increase the number of logical partitions allocated to it with the `extendlv` command. The default maximum size for a logical volume at creation is 512 logical partitions, unless specified to be larger. The `chlv` command is used to override this limitation.

**Logical Partition**

When you create a logical volume, you specify the number of logical partitions for the logical volume.

A logical partition is one, two, or three physical partitions, depending on the number of instances of your data you want maintained. Specifying one instance means there is only one copy of the logical volume (the default). In this case, there is a direct mapping of one logical partition to one physical partition. Each instance, including the first, is termed a copy. Where physical partitions are located (that is, how near each other physically) is determined by options you specify when you create the logical volume.

**File system**

The logical volume defines allocation of disk space down to the physical-partition level. Finer levels of data management are accomplished by higher-level software components such as the Virtual Memory Manager or the file system. Therefore, the final step in the evolution of a disk is the creation of file systems.

You can create one file system per logical volume. To create a file system, use the `crfs` command.

### 7.1.3 Principles to optimize LVM disks

Combining these concepts of how AIX physically structures hard disks with how LVM can impose logical structures onto disks, several principles are to help optimize LVM disks

**Make LVs contiguous as much as possible**

When file systems are strewn across a disk, the hard disk head takes longer to find any wanted data. If the LV exists in one continuous area, the seek time is minimized and the files can be found more quickly.

**Place LVs with high I/O or sequential read or write operations**

Because of the speed of the outer edge of the disk, LVs that need faster reads or writes with data in long sequences will benefit from the higher rotational velocity at the outer edge.

**Place LVs with high activity toward the center**

If you have a file system that has a great deal of reads and writes and needs quick responsiveness, because of averages, the hard disk head will most likely be near the center at any point in time. By placing those file systems in this area, there will be a higher likelihood that the head will be in that general vicinity. This configuration reduces seek time and maintains good I/O.
Place LVs with low usage near the inner edge
If you have any file systems that are rarely used or accessed, get them out of the way by putting them in the part of the disk with the lowest I/O speeds—near the spindle at the inner edge. For example, logical volumes that are seldom used fit well here.

Use only one paging space LV per disk
The purpose of paging space is to serve as a temporary place to swap pages in and out of memory to an area of physical storage, thus allowing the CPU to perform more operations and catch up on things. Defining multiple paging space LVs on the same disk defeats the purpose of trying to remedy performance shortfalls by causing more I/O, as the disk head has to go to several areas of the platters instead of just one.

No need to mirror paging space
Mirroring paging space is not recommended because the purpose of paging space is to use physical storage to offset resource shortfalls, it makes no sense to write the same set of data twice in a mirrored configuration. Instead, if two disks have two separate paging space LVs, each can address memory swapping and double the effectiveness.

Keep paging space sizes uniform
If an AIX server has two 80GB disks and one has a 1GB paging space LV while the other one has a 4GB paging space LV, one of these two will likely experience more wear. Depending on how the paging space was added to the server or shrunk in size, one of the LVs may become full and adversely affect the system. Plus, keeping sizes the same makes the `lsps` command output looks cleaner and more accurate.

Keep some free space on the disks
This is not just relative to having one or two large gaps in the inner or outer middle regions, but having space between the LVs in case anything needs to be grown and a little free space in the file systems themselves.

Keep track of newly created file systems on a mirrored environment
One common mistake administrators make is to create a new file system on a mirrored VG, then forget to create a copy onto the other disk. Just because the VG is mirrored does not mean that any subsequently added LVs that support the file systems will also be mirrored. Always make sure that new LVs get a copy on a separate PV.

Distribute I/O load as much as possible
If you have a larger system with multiple drawers, leverage them by striping the data across sets of disk packs, and if you choose to mirror your PVs, it is recommended to do it across different drawers so that if a backplane fails, redundancy is not limited to one drawer.

7.1.4 LVM strategies of various storage types
This section shows the LVM strategies of various types of storage.

Internal hard drives
The most common form of storage in AIX, internal hard drives are typically used for root volume group disks and servers with smaller foot prints. When using internal hard drives, the first step should always be to have at least two disks per volume group and to mirror the rootvg hard drives.
Small SAN storage
Smaller storage subsystems, such as direct-attached or small SAN technology, are affordable solutions for environments in which more than internal disk space is needed to hold larger amounts of data. For these situations, it is important to manage intimately the configuration of the environment, because there may be some single points of failure along the way. The storage should be optimized with a proper RAID configuration, such as a RAID 5 setup with a hot spare disks.

Large SAN storage
In larger SAN storage environments, where multiple servers are accessing a number of storage devices, such as IBM System Storage™ DS devices, through SAN switches, there are typically dedicated SAN administrators who manage disk resources. But from an AIX perspective, systems administrators can help by doing things like choosing multiple dual-port Fibre Channel cards to communicate with different fabrics and improve throughput.

7.1.5 LVM configuration

With Logical Volume Manager we can mirror volume groups, define a logical volume, and remove a disk with the system running.

Mirroring a logical volume
The following scenarios explain how to mirror a normal volume group.

The following instructions show you how to mirror a root volume group using the System Management Interface Tool (SMIT).

(Select a volume group in the Volumes container, then choose Mirror from the Selected menu). Experienced administrators can use the `mirrorvg` command.

With root authority, add a disk to the volume group using SMIT fast path, see Example 7-1

**Example 7-1   Extending the volume group to have a new disk for mirror**

```
# smitty extendvg
Add a Physical Volume to a Volume Group

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

[Entry Fields]

FORCE the creation of volume group? no
+
* VOLUME GROUP name []
+
* PHYSICAL VOLUME names []
+
```

Then we can mirror the volume group onto the new disk by typing SMIT fast path as shown in Example 7-2.

**Example 7-2   getting into mirrorvg SMIT menu**

```
# smit mirrorvg
Mirror a Volume Group
```
In the first panel, select a volume group for mirroring.

In the second panel, you can define mirroring options or accept defaults.

**Synchronizing LVM mirror and direct reads to a certain mirror**

With LVM mirror, you can always synchronize and control the sync operation.

**Control sync operation**

To synchronize the physical partitions, which are copies of the original physical partition, `syncvg` command can be used. The command can be used with logical volumes, physical volumes, or volume groups, with the Name parameter representing the logical volume name, physical volume name, or volume group name. The synchronization process can be time consuming, depending on the hardware characteristics and the amount of data.

When the `-f` flag is used, a good physical copy is chosen and propagated to all other copies of the logical partition, whether or not they are stale. Using this flag is necessary in cases where the logical volume does not have the mirror write consistency recovery.

The synchronization process can always be paused, resumed or terminated, this depends on the flag passed to the command. The good thing is we can query the sync operation which will report a verbose list of sync operation Process Identifiers (PIDs) along with the sync rate of the operation.

A new flag has just been introduced which is `-T` *SyncRate* to throttle the sync rate of the current sync operation or throttles one or more sync operations that are in progress. The *SyncRate* parameter specifies the sync rate, in MB/sec, to throttle. The syncvg command synchronizes one Logical Track Group (LTG) at a time. This parameter must be specified in multiples of the LTG size of the volume group. If the *SyncRate* parameter is not specified in the multiples of the LTG size, the syncvg command rounds up to the nearest LTG size of the volume group.

**Direct specific reads**

There is a new LVM mirror policy to direct all reads to a specific mirror, which will allow LVM mirror to be used as fast copy for improved read performance.

This will off load the primary storage resources, and improve the overall performance of the storage environment.

This can be implemented by using `-R` flag in `mklv` or `chlv` commands. The `-R` option sets read preference to the copy of the logical volume. If the `-R` flag is specified and if the preferred copy is available, the read operation occurs from the preferred copy. If the preferred copy is not available, the read operations follow the scheduling policy of the logical volume.

The `PreferredRead` variable can be set to a value in the range 0-3. The default value is 0, which disables the preferred read copy of the logical volume.

Example 7-3 shows how to set the preferred copy in creation of the LV or after the creation.
Example 7-3 Setting the preferred read copy by mklv or change it by chlv

```
# mklv -y mirroredlv -t jfs2 -c 2 -R 2 testvg 10
mirroredlv

# lslv mirroredlv | grep PREFERRED
INFINITE RETRY:   no                     PREFERRED READ: 2

# chlv -R 1 mirroredlv

# lslv mirroredlv | grep PREFERRED
INFINITE RETRY:   no                     PREFERRED READ: 1
```

Defining a raw logical volume

A raw logical volume is an area of physical and logical disk space that is under the direct control of an application, such as a database or a partition, rather than under the direct control of the operating system or a file system.

Bypassing the file system can yield better performance from the controlling application, especially from database applications. The amount of improvement, however, depends on factors such as the size of a database or the application's driver. You will need to provide the application with the character or block special device file for the new raw logical volume, as appropriate. The application will link to this device file when it attempts opens, reads, writes, and so on.

You have to keep in mind that each logical volume has a logical volume control block (LVCB) located in the first block. The size of LVCB is the block size of the physical volumes within the volume group. Data begins in the second block of the physical volume. In a raw logical volume, the LVCB is not protected. If an application overwrites the LVCB, commands that normally update the LVCB will fail and generate a message. Although the logical volume might continue to operate correctly and the overwrite can be an allowable event, overwriting the LVCB is not recommended.

The following instructions use SMIT and the command line interface to define a raw logical volume.

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

Find the disk

With root authority, find the free physical partitions on which you can create the raw logical volume by typing the following SMIT fast path as shown in Example 7-4.

Example 7-4 Checking free pps

```
# smitty lspv
```

Select the disk

Select the disk and accept the default in the second dialog status and click OK.

Get the total available space

Multiply the value in the FREE PPs field by the value in the PP SIZE field to get the total number of megabytes available for a raw logical volume on the selected disk.
If the amount of free space is not adequate, select a different disk until you find one that has enough available free space, and exit SMIT menu.

**Create raw logical volume**

Use the mklv command to create the raw logical volume.

The following Example 7-5 creates a raw logical volume named lvdb2003 in the db2vg volume group using 38 4-MB physical partitions

*Example 7-5   Using mklv command to create raw LV*

```
mklv -y lvdb2003 db2vg 38
```

At this point, the raw logical volume is created. If you list the contents of your volume group, a raw logical volume is shown with the default type, which is JFS. This type entry for a logical volume is simply a label. It does not indicate a file system is mounted for your raw logical volume.

Consult your application's instructions on how to open /dev/rawLVname and how to use this raw space.

**Removing a disk from a live up and running system**

The following procedure describes how to remove a disk using the hot-removability feature, which lets you remove the disk without turning the system off. Hot removability is useful when you want to

- Remove a disk that contains data in a separate non-rootvg volume group for security or maintenance purposes.
- Permanently remove a disk from a volume group.
- Correct a disk failure.

**Removing a disk without data**

The following procedure describes how to remove a disk that contains either no data or no data that you want to keep.

**Unmount the file systems**

Unmounting file systems can be either `umount` command as below example

*Example 7-6   Unmounting a file system from command line*

```
# umount /filesystem
```

Or you can use SMIT menu as below example

*Example 7-7   Unmounting a file system using SMIT*

```
# smitty umountfs
Unmount a File System

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

Unmount ALL mounted file systems? no [Entry Fields]
+
(except /, /tmp, /usr)
```
Deactivate the volume group
Deactivating the volume group is done by varyoffvg command as Example 7-8

Example 7-8   Deactivating the volume group using varyoffvg command

```
# varyoffvg VGNAME
```

Delete the volume group information
Deleting the volume group information referenced as metadata is done by exportvg command as shown in Example 7-9.

Example 7-9   Performing exportvg command

```
# exportvg VGNAME
```

Remove the disk definition from the AIX server
Deleting the disk definition will remove all information related to this disk from the ODM. see the following Example 7-10

Example 7-10   Deleting the disk by rmdev command

```
# rmdev -Rdl hdiskX
```

Physically remove the disk
Now you will remove the disk physically, whether it is internal disk or just mapped as a LUN from a SAN mapping.

Removing a disk with data
The disk you are removing must be in a separate non-rootvg volume group. Use this procedure when you want to move a disk to another system, this will remove a disk that contains data without turning the system off.

List the volume group associated with the disk you want to remove
To list the volume group associated with the disk you want to remove, type lspv command as shown in Example 7-11.

Example 7-11   Listing hdisk2 information using lspv hdisk2

```
# lspv hdisk2
PHYSICAL VOLUME:    hdisk2                    VOLUME GROUP:     imagesvg
PV IDENTIFIER:      00083772caa7896e VG IDENTIFIER 0004234500004c00000000e9b5cac262
PV STATE:           active                      STALE PARTITIONS:  0
PP SIZE:            16 megabyte(s)             ALLOCATABLE:      yes
TOTAL PPs:          542 (8672 megabytes)       LOGICAL VOLUMES:  5
FREE PPs:           19 (304 megabytes)         VG DESCRIPTORS:   2
                      HOT SPARE:        no
```
The name of the volume group is listed in the VOLUME GROUP field. In this example, the volume group is imagesvg.

**Make sure the disk really is in non-rootvg**

To verify that the disk is in a separate non-rootvg volume group we use lsvg command as shown in Example 7-12.

**Example 7-12  Listing volume group information**

```bash
# lsvg imagesvg

VOLUME GROUP: imagesvg                 VG IDENTIFIER:
000423450004c000000000e9b5cac262
VG STATE: active                    PP SIZE: 16 megabyte(s)
VG PERMISSION: read/write          TOTAL PPs: 542 (8672 megabytes)
MAX LVs: 256                        FREE PPs: 19 (304 megabytes)
LVs: 5                              USED PPs: 523 (8368 megabytes)
OPEN LVs: 4                         QUORUM: 2
TOTAL PVs: 1                        VG DESCRIPTORS: 2
STALE PVs: 0                        MAX LVs per PV: 1016
ACTIVE PVs: 1                       MAX PVs: 32
MAX PPs per PV: 1016                LTG size: 128 kilobyte(s)
HOT SPARE: no                      AUTO ON: yes
AUTO SYNC: no
```

In the example, the TOTAL PVs field indicates there is only one physical volume associated with imagesvg. Because all data in this volume group is contained on hdisk2, hdisk2 can be removed using this procedure

**Unmount the file system, deactivate the volume group and export it**

Now we will unmount the file system, deactivate the VG and export it as shown in Example 7-13.

**Example 7-13  Exporting the volume group**

```bash
# umount /filesystem
# varyoffvg imagesvg
# exportvg imagesvg
```

The above commands will unmount the file system, then it will varyoff the volume group so we can export it which is a removal of its information from AIX ODM.

**Remove the disk**

Last step is to remove the disk from the operating system by rmdev command as shown in Example 7-14.

**Example 7-14  Removing the disk using rmdev command**

```bash
# rmdev -Rdl hdisk2
```
Resizing a RAID volume group

On systems that use a Redundant Array of Independent Disks (RAID), `chvg` and `chpv` command options provide the ability to add a disk to the RAID group and grow the size of the physical volume that LVM uses without interruptions to the use or availability of the system.

The size of all disks in a volume group is automatically examined when the volume group is activated (varyon). If growth is detected, the system generates an informational message.

The following procedure describes how to grow disks in a RAID environment

Keep in mind the following points:

- This feature is not available while the volume group is activated in classic or in enhanced concurrent mode.
- The rootvg volume group cannot be resized using the following procedure.
- A volume group with an active paging space cannot be resized using the following procedure.

**Check for disk growth**

To check for disk growth and resize if needed, we use `chvg -g` command as below example

```
Example 7-15   Checking disk growth
# chvg -g VGname
```

Where `VGname` is the name of your volume group. This command examines all disks in the volume group. If any have grown in size, it attempts to add physical partitions to the physical volume. If necessary, it will determine the appropriate 1016 limit multiplier and convert the volume group to a big volume group.

**Turn off the bad block policy**

To turn off LVM bad block relocation policy for the volume group, we use `chvg -b` command as in Example 7-16

```
Example 7-16   Turning off bad block relocation policy
# chvg -b ny VGname
```

Where `VGname` is the name of your volume group.

**When to create separate volume groups**

There are several reasons why you might want to organize physical volumes into volume groups separate from rootvg.

**For safer and easier maintenance.**

Operating system updates, reinstallations, and crash recoveries are safer because you can separate user file systems from the operating system so that user files are not jeopardized during these operations.

Maintenance is easier because you can update or reinstall the operating system without having to restore user data. For example, before updating, you can remove a user-defined volume group from the system by unmounting its file systems. Deactivate it using the `varyoffvg` command, then export the group using the `exportvg` command. After updating the system software, you can reintroduce the user-defined volume group using the `importvg` command, then remount its file systems.
**For different physical-partition sizes**

All physical volumes within the same volume group must have the same physical partition size. To have physical volumes with different physical partition sizes, place each size in a separate volume group.

**If different quorum characteristics are required**

If you have a file system for which you want to create a nonquorum volume group, maintain a separate volume group for that data; all of the other file systems should remain in volume groups operating under a quorum.

**To switch physical volumes between systems**

If you create a separate volume group for each system on an adapter that is accessible from more than one system, you can switch the physical volumes between the systems that are accessible on that adapter without interrupting the normal operation of either.

### 7.2 AIX JFS2

AIX journaling file system is a hierarchical structure of files and directories.

This type of structure resembles an inverted tree with the roots at the top and branches at the bottom. This file tree uses directories to organize data and programs into groups, allowing the management of several directories and files at one time.

A file system resides on a single logical volume. Every file and directory belongs to a file system within a logical volume. Because of its structure, some tasks are performed more efficiently on a file system than on each directory within the file system. For example, you can back up, move, or secure an entire file system. You can make a point-in-time image of a file system named a snapshot.

#### 7.2.1 JFS2 functions

Enhanced Journaled File System (JFS2) is a file system that provides the capability to store much larger files than the existing Journaled File System (JFS).

You can choose to implement either JFS or JFS2. JFS2 is the default file system in AIX, and the most improved and enhanced one in AIX when it comes to device management and security options.

For the file system to work, a logical volume should be created in the first place or let the AIX creates the logical volume and the file system on it. To be accessible, a file system must be mounted onto a directory mount point. When multiple file systems are mounted, a directory structure is created that presents the image of a single file system.

It is a hierarchical structure with a single root. This structure includes the base file systems and any file systems you create. You can access both local and remote file systems using the mount command.

The most important feature of JFS2 file system that JFS file system do not have is the ability to shrink the file system online, unlike the normal JFS file system which can only be increased online but cannot be shrunk, and the solution was to create a new JFS file system and copy/backup the old data to the new resized file system as a solution of saving space.
There are several improvements and differences for JFS2 against JFS file system, and differences, see the following link for more information


7.2.2 JFS2 features

jfs2 has several features implemented recently which makes jfs2 file system is more reliable and secure.

**JFS2 performance**
Enhanced journaling file system (jfs2) allows for quick file system recovery after a crash occurs by logging the metadata of files. By enabling file system logging, the system records every change in the metadata of the file into a reserved area of the file system. The actual write operations are performed after the logging of changes to the metadata has been completed.

Support for data sets is integrated into JFS2 as part of the AIX operating system. A data set is a unit of data administration. It consists of a directory tree with at least one single root directory. Administration might include creating new data sets, creating and maintaining full copies (replicas) of data sets across a collection of servers, or moving a data set to another server. A data set might exist as a portion of mounted file system. That is, a mounted file system instance might contain multiple data sets. Data set support is enabled in JFS2 by using the `mkfs -o dm=on` command. By default, data set support is not enabled.

Enhanced JFS (jfs2) does not allow you to disable metadata logging. However, the implementation of journaling on Enhanced JFS makes it more suitable to handle metadata intensive applications. Thus, the performance penalty is not as high under Enhanced JFS as it is under JFS.

The main performance advantage of using Enhanced JFS over JFS is scaling, enhanced JFS provides the capability to store much larger files than the existing JFS. The maximum size of a file under JFS is 64 gigabytes. Under Enhanced JFS, AIX currently supports files up to 16 terabytes in size, although the file system architecture is set up to eventually handle file sizes of up to 4 petabytes.

**JFS2 security**
Encrypted File System (EFS) is only supported on JFS2 file systems. EFS allows you to encrypt your data and control access to the data through keyed protection.

A key is associated with each user and is stored in a cryptographically protected key store. Upon successful login, the user's keys are loaded into the kernel and associated with the process credentials. To open an EFS-protected file, the process credentials are tested. If the process finds a key that matches the file protection, the process decrypts the file key and the file content.

This is covered in details in 7.4, “Encrypted file systems” on page 135

7.3 Role Based Access Control

Traditionally, there is a single user, root, that controls the security mechanism of the system. The root user decides who can log in, who can access the data, which process has the
privileges to get into the kernel mode, and so on. But, the drawback of a single root user is that the system becomes vulnerable to attack if an unauthorized person takes control of the root user.

To avoid this problem, the AIX introduces a very powerful security features like Role Based Authentication Control (RBAC) and Multi level security (MLS) along with various other features like Trusted Execution (TE), Encrypted File System (EFS) and more, in addition to the existing traditional ROOT user based authentication.

AIX system administration is an important aspect of daily operations, and security is an inherent part of most system administration functions. Also, in addition to securing the operating environment, it is necessary to closely monitor daily system activities.

Most environments require that different users manage different system administration duties. It is necessary to maintain separation of these duties so that no single system management user can accidentally or maliciously bypass system security. While traditional UNIX system administration cannot achieve these goals, role-based access control (RBAC) can do.

RBAC allows the creation of roles for system administration and the delegation of administrative tasks across a set of trusted system users. In AIX, RBAC provides a mechanism through which the administrative functions typically reserved for the root user can be assigned to regular system users.

RBAC accomplishes this by defining job functions (roles) within an organization and assigning those roles to specific users. RBAC is essentially a framework that allows for system administration through the use of roles. Roles are typically defined with the scope of managing one or more administrative aspects of the environment. Assigning a role to a user effectively confers a set of permissions or privileges and powers to the user. For example, one management role might be to manage the file systems, while another role might be to enable the creation of user accounts.

RBAC administration has the following advantages as compared to traditional UNIX administration:

1. System administration can be performed by multiple users without sharing account access.
2. Security isolation through granular administration since each administrator does not need to be granted more power than is required.
3. Allows for enforcing a least-privilege security model. Users and applications are only granted necessary privileges when required, thereby reducing the impact a system attacker can have.
4. Allows for implementing and enforcing company-wide security policies consistently in regard to system management and access control.
5. A role definition can be created once and then assigned to users or removed as needed when users change job functions.

7.3.1 RBAC elements

The RBAC framework is centered on the following three core concepts

**Authorizations**
An authorization is a text string associated with security-related functions or commands. Authorizations provide a mechanism to grant rights to users to perform privileged actions and to provide different levels of functionality to different classes of users.
When a command governed by an authorization is run, access is granted only if the invoking user has the required authorization. An authorization can be thought of as a key that is able to unlock access to one or more commands. Authorizations are not directly assigned to users. Users are assigned roles, which are a collection of authorizations.

Note: aix.ras.error.errpt and aix.ras.trace.trcrptch0 access authorizations have been introduced in AIX 7.2 to manage system error report, and to administer system trace report.

Roles
Roles allow a set of management functions in the system to be grouped together. Using the analogy that an authorization is a key, a role can be thought of as a key ring that can hold multiple authorizations. Authorizations may be directly assigned to a role or indirectly assigned through a sub-role. A sub-role is simply another role that a given role inherits the authorizations from.

A role by itself does not grant the user any additional powers, but instead serves as a collection mechanism for authorizations and a facility for assigning authorizations to a user. Defining a role and assigning the role to a user determines the system administration tasks that can be performed by the user. After a role has been defined, the role administrator can assign the role to one or more users to manage the privileged operations that are represented by the role. Additionally, a user can be assigned multiple roles. Once a role has been assigned to a user, the user can use the authorizations assigned to the role to unlock access to administrative commands on the system.

Organizational policies and procedures determine how to allocate roles to users. Do not assign too many authorizations to a role or assign a role to too many users. Most roles should only be assigned to members of the administrative staff. Just as the powers of root have historically only been given to trusted users, roles should only be assigned to trusted users. Grant roles only to users with legitimate needs and only for the duration of the need. This practice reduces the chances that an unauthorized user can acquire or abuse authorizations.

Privileges
A privilege is a process attribute that allows the process to bypass specific system restrictions and limitations.

The privilege mechanism provides trusted applications with capabilities that are not permitted to untrusted applications. For example, privileges can be used to override security constraints, to permit the expanded use of certain system resources such as memory and disk space, and to adjust the performance and priority of a process. A privilege can be thought of as an ability that allows a process to overcome a specific security constraint in the system.

Authorizations and roles are user-level tools that configure a user’s ability to access privileged operations. On the other hand, privileges are the restriction mechanism used in the kernel to determine if a process is allowed to perform a particular action.

Privileges are associated with a process and are typically acquired through the invocation of a privileged command. Because of these associated privileges, the process is eligible to perform the related privileged operation. For example, if a user uses a role that has an authorization to run a command, a set of privileges is assigned to the process when the command is run.

Some operations are privileged and permission to perform these operations is restricted to authorized users. These privileged operations usually include tasks such as rebooting the
system, adding and modifying file systems, adding and deleting users, and modifying the system date and time.

### 7.3.2 RBAC Kernel Security Table and exact files

Kernel Security Tables are tables reside in the AIX kernel, which provide an extra layer of security. See Table 7-1.

**Table 7-1 RBAC Kernel Security Tables**

<table>
<thead>
<tr>
<th>Kernel Security Table Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>For authorizations</td>
</tr>
<tr>
<td>role</td>
<td>For roles</td>
</tr>
<tr>
<td>cmd</td>
<td>For privileged commands</td>
</tr>
<tr>
<td>dev</td>
<td>For privileged devices</td>
</tr>
<tr>
<td>dom</td>
<td>For domains</td>
</tr>
<tr>
<td>domobj</td>
<td>For domain objects</td>
</tr>
</tbody>
</table>

As for the RBAC definition files, they all are listed under `/etc/security` path. Table 7-2 shows RBAC definition files.

**Table 7-2 RBAC definition file**

<table>
<thead>
<tr>
<th>RBAC definition file</th>
<th>File function</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/security/privcmds</td>
<td>To execute AIX commands</td>
</tr>
<tr>
<td>/etc/security/privfiles</td>
<td>To edit AIX files with special edit command</td>
</tr>
<tr>
<td>/etc/security/privdevs</td>
<td>To read/write devices</td>
</tr>
<tr>
<td>/etc/security/domains</td>
<td>To domain names</td>
</tr>
<tr>
<td>/etc/security/domobjs</td>
<td>To objects protected by domains</td>
</tr>
</tbody>
</table>

The RBAC can always be customized to serve the administration needs in which who can do exactly what commands with certain options.

RBAC has also ready made roles to directly assign to users for certain jobs like backup, shutdown and file systems managements.

### 7.3.3 Customizing RBAC role for certain tasks

This section discusses a real example of a customized role by RBAC.

**Customizing role for a command associated to an authorization**

The following topic discusses a customization of a role for any specific command whether this command has an associated authorization included to it in `/etc/security/privcmds` file.

1. If you want to create a custom role for `mklv` command for example, you would have first to make sure of its full path and if it really exists in the privileged command as shown below in the following
Example 7-17  Checking mklv command full path and check if it has security attributes

```
# which mklv
/usr/sbin/mklv

# lssecattr -Fc /usr/sbin/mklv
/usr/sbin/mklv:
  euid=0
egid=0
  accessauths=aix.lvm.manage.create
  innateprivs=PV_AZ_ROOT, PV_DAC_O, PV_DAC_R, PV_DAC_X
  inheritprivs=PV_AZ_CHECK, PV_DAC_R, PV_DAC_W
  secflags=FSF_EPS
```

2. The next step is to check if there is any role associated to the command authorization mentioned in the previous example as accessauth.

The following Example 7-18 will check if any role is associated to the access authorization.

**Example 7-18  Checking if any role associated to the access authorization**

```
# lsrole ALL | grep aix.lvm.manage.create
```

In the previous Example 7-18, it came back with nothing so no current role exists with that specific authorization.

3. Now we will create the custom role which will have the authorization we got, by running `mkrole` command. See the following Example 7-19.

**Example 7-19  Creating a role using mkrole command**

```
# mkrole dfltmsg="LVs Creation" authorizations="aix.lvm.manage.create"
makelvrole
```

4. Once the role created, it is mandatory to update the kernel security table using `setkst` command as shown in the following Example 7-20.

**Example 7-20  Using setkst command to update kernel security tables**

```
# setkst
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.
Successfully updated the Kernel Object Domain Table.
Successfully updated the Kernel Domains Table.
```

5. To attach this makelvrole to a user, we have to use `chuser` to assign the role either in the roles database or just in the default_roles database.

If we assigned the role to the roles database, the use will need to switch to it using `swrole` command, however if we assigned it to the default_roles, it will be applied automatically applied upon user login. See Example 7-21 for assigning the new role to user bob.

**Example 7-21  Using chuser command to add the new role**

```
# chuser roles=makelvrole default_roles=makelvrole bob
```

6. If the user bob logged in, he will see that his makelvrole is automatically active as the following Example 7-22.
Example 7-22  Listing roles assigned to the current logged in user

$id
uid=217(bob) gid=1(staff)

$ rolelist -a
lvcreate   aix.lvm.manage.create

Customizing a role for a command not associated to any authorization
This topic discusses customizing a new role to any command privileges in AIX, and this also applies on any other non-AIX command, like HTTP, apachectl application commands.

First check is to see that enhanced_RBAC is enabled
enhanced_RBAC kernel attribute should be enabled in the system to do the RBAC customization.

The following command in Example 7-23 shows how to check enhanced_RBAC

Example 7-23  Checking if enhanced_RBAC is enabled

# lsattr -El sys0 -a enhanced_RBAC
    enhanced_RBAC true Enhanced RBAC Mode True

We see from the above Example 7-23 that enhanced_RBAC is enabled as per the word true.

If the enhanced_RBAC is not enabled, then we must enable it and reboot the AIX server as shown in the following Example 7-24.

Example 7-24  Enabling enhanced_RBAC and reboot

# chdev -l sys0 -a enhanced_RBAC=true
sys0 changed
# shutdown -Fr

Create a user defined authorization for the new role
We use mkauth command to create authorization as the following Example 7-25 mentions

Example 7-25  Using mkauth command to create authorization

# mkauth dfltmsg='My Custom Auth App' my_app_auth

This authorization has been added to /etc/security/authorizations file and can be viewed by lsauth command as shown in the following Example 7-26.

Example 7-26  Listing the authorization created

# lsauth my_app_auth
my_app_auth id=10020 dfltmsg=My Custom Auth App

Trace privileges for the command or binary needs for a successful run
You should trace the privileges for the command you want for a successful run and to grab the security attributes that you will associate after with the authorization. If the command like /usr/local/bin/myapp will be traced, it should be written completely good with its full path and arguments, otherwise it will not get you the required privileges. If the command has been provided with no arguments where it should require, no outputs will be shown as a result of the trace.
The following Example 7-27 shows how to trace a certain command.

**Example 7-27 Using tracepriv for a certain command**

```bash
# tracepriv -e /usr/local/bin/myapp
```

18678226: Used privileges for /usr/local/bin/myapp:
- PV_AU_ADD
- PV_AU_PROC
- PV_DAC_R
- PV_DAC_W
- PV_FS_CHOWN

The output from the previous tracepriv example shows five security attributes used while running the command /usr/local/bin/myapp. You need to take a note of them.

**Set the security attributes for the binary object**

You must associate now the security attributes you got from previous example to the authorization created earlier. You will use `setsecattr` command as the following Example 7-28 mentions.

**Example 7-28 Using setsecattr command to bind the security attributes to the authorization**

```bash
# setsecattr -c innateprivs=PV_AU_ADD,PV_DAC_R,PV_FS_CHOWN,PV_AU_PROC,PV_DAC_W
  accessauths=my_app_auth
  euid=0 /usr/local/bin/myapp
```

Now this complete authorization and security attributes attached to it have been added to /etc/security/privcmds file and can be viewed by `lssecattr` command as the below Example 7-29.

**Example 7-29 Listing the complete authorization associated with the command**

```bash
# lssecattr -Fc /usr/local/bin/myapp
```

**Create a role and associate to the authorization**

Now you create the role using `mkrole` command and mention the authorization name that you just created. Use the `mkrole` command as shown in Example 7-30.

**Example 7-30 Using mkrole command**

```bash
# mkrole authorizations=my_app_auth dfltmsg="My_Role" exec_app
```

**Add the role to the user**

Add the role to the user roles or default_roles using `chuser` command as shown in Example 7-31.

**Example 7-31 Adding the new role to the user**

```bash
# chuser roles=exec_app testuser
```

**Update the kernel security table**

Update the kernel security table using `setkst` command as below shown in below example.

**Example 7-32 Updating the kernel security table using setkst command**

```bash
# setkst
Successfully updated the Kernel Authorization Table.
```
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.
Successfully updated the Kernel Object Domain Table.
Successfully updated the Kernel Domains Table.

7.3.4 Domain RBAC

The domain feature for RBAC is used to restrict access to authorized users. The users and resources of the system are labeled by attaching tags named domains, and the specific access rules determine access to resources by the users.

Definition and access rules
Domain RBAC has key definition concepts which are related to access rules.

RBAC Domain Subject
A subject is an entity that requests access to an object. An example of a subject is a process.

RBAC Domain Object
An object is an entity that holds information of value. Examples of objects are files, devices, and network ports.

RBAC Domain
A domain is defined as a category to which an entity belongs. When an entity belongs to a domain, access control to the entity is governed by the access rules.

Access Rules
A subject can access an object when it has all the domains to which the object belongs. This specifies that the list of domains the subject belongs to is a super set of an object’s domains. This is the default behavior.

A subject can access an object when it has at least one domain of the object. That is, the subject and object have one domain in common. This behavior depends on the security flags of the object.

An object can deny access to certain domains. If an object defines a set of domains named conflict sets and if one of the domains of the subject is part of the conflict set, the object can deny access to the subject.

7.3.5 Domain RBAC implementation scenario

This section goes through domain RBAC implementation in a scenario that a non root user ends up with having a role of mounting the file systems, however the need was to separate his privileges not to mount specific file systems.

1. Make sure enhanced_RBAC is enabled as the following Example 7-33

```
Example 7-33  Checking enhanced_RBAC status

# lsattr -El sys0 -a enhanced_RBAC
enhanced_RBAC true Enhanced RBAC Mode True
```

2. Create a user for the job of file systems mounting as below Example 7-34
Example 7-34  Creating a user for the job of mounting the file systems

# mkuser test
# passwd test
Changing password for "test"
test's New password:
Enter the new password again:

3. Check security authorizations of specific mount command as shown in Example 7-35.

Example 7-35  Checking authorization associated with mount command

# lssecattr -c -a accessauths /usr/sbin/mount
/usr/sbin/mount accessauths=aix.fs.manage.mount

The previous Example 7-36 shows there is aix.fs.manage.mount authorization.

4. Check security authorizations of specific umount command as shown in Example 7-36.

Example 7-36  Checking authorization associated with umount command

# lssecattr -c -a accessauths /usr/sbin/umount
/usr/sbin/mount accessauths=aix.fs.manage.unmount

The previous Example 7-36 shows there is aix.fs.manage.unmount authorization.

5. Create a role and assign both authorizations to it as in Example 7-37 then list it to make sure

Example 7-37  Creating a role with both authorizations and list it

# mkrole authorizations=aix.fs.manage.mount,aix.fs.manage.unmount
fs_manage

# lsrole fs_manage
fs_manage
authorizations=aix.fs.manage.mount,aix.fs.manage.unmount
rolelist= groups= visibility=1 screens= auth_mode=INVOKER id=18

6. Assign the newly created role to the designated user as shown in the following Example 7-38.

Example 7-38  Assigning the new role created to the designated user

# chuser roles=fs_manage test
# lsuser -a roles test

7. Update the kernel security table as shown in Example 7-39.

Example 7-39  Updating kernel security table

# setkst
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.
Successfully updated the Kernel Object Domain Table.
Successfully updated the Kernel Domains Table.

8. Test the role by the user as shown in the following Example 7-40.
Example 7-40   Testing the role by the new user

```
# su - test
$ swrole ALL
```  

`test's Password:
$ mount /testfs1
$ mount /testfs2
$ df -g | grep testfs
/dev/fslv11 1.00 1.00 1% 4 1% /testfs1
/dev/fslv12 1.00 1.00 1% 4 1% /testfs2
$ umount /testfs1
$ umount /testfs2
```

Now user test can mount/umount any file system. Following steps will show how to restrict specific file systems mount using domain RBAC objects.

9. Create two RBAC domains and list them as shown in Example 7-41.

   Example 7-41   Creating RBAC domains

   ```
   # mkdom dom1
   # mkdom dom2
   # lsdom ALL
   dom1 id=1
   dom1 id=1
   ```

10. Check the file systems which we will apply the domains on them. See Example 7-42.

   Example 7-42   Checking the file system

   ```
   # lsfs | grep testfs
   /dev/fslv11 -- /testfs1 jfs2 2097152 -- yes no
   /dev/fslv12 -- /testfs2 jfs2 2097152 -- yes no
   ```

11. Assign each domain with a device object to one file system as shown in the following Example 7-43 on page 134.

   Example 7-43   Assigning a domain to exact file system

   ```
   # setsecattr -o domains=dom1 objtype=device secflags=FSF_DOM_ANY
   /dev/fslv11
   # setsecattr -o domains=dom2 objtype=device secflags=FSF_DOM_ANY
   /dev/fslv12
   # lssecattr -o /dev/fslv11
   /dev/fslv11 domains=dom1 objtype=device secflags=FSF_DOM_ANY
   # lssecattr -o /dev/fslv12
   /dev/fslv12 domains=dom2 objtype=device secflags=FSF_DOM_ANY
   ```

12. Assign only dom1 domain to test user. See Example 7-44.

   Example 7-44   Assigning a domain to a user

   ```
   # chuser domains=dom1 test
   # lsuser -a roles domains test
   test roles=FSAdmin domains=dom1
   ```

13. Update the kernel security table as shown in Example 7-45.
Example 7-45  Updating the kernel security table

```
# setkst
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.
Successfully updated the Kernel Object Domain Table.
Successfully updated the Kernel Domains Table.
```

14. Attempt to mount both file systems. Only the one associated to dom1 domain will be mounted. See Example 7-46

Example 7-46  Attempting to mount the file systems

```
# su - test
$ swrole ALL
$ mount /testfs1
$ mount /testfs2
mount: 0506-324 Cannot mount /dev/fslv12 on /testfs2: Operation not permitted.
```

This is very expected in Example 7-46, since the user can only mount any file system with the security attribute of the domain assigned to him. He cannot mount the file systems in a different domain.

7.4 Encrypted file systems

The Encrypted File System (EFS) is a J2 file system-level encryption through individual key stores. This allows for file encryption in order to protect confidential data from attackers with physical access to the computer.

User authentication and access control lists can protect files from unauthorized access while the operating system is running; however, it’s easy to circumvent the control lists if an attacker gains physical access to the computer.

One solution is to store the encrypted files on the disks of the computer. In EFS, a key is associated to each user. These keys are stored in a cryptographically protected key store and upon successful login, and the user’s keys are loaded into the kernel and associated with the process credentials. When the process needs to open an EFS-protected file, the system tests the credentials. If the system finds a key matching the file protection, the process is able to decrypt the file key and file content. The cryptographic information is kept in the extended attributes for each file.

EFS uses extended attribute Version 2, and each file is encrypted before being written on the disk. The files are decrypted when they are read from the disk into memory so that the file data kept in memory is in clear format. The data is decrypted only once, which is a major advantage. When another user requires access to the file, their security credentials are verified before being granted access to the data even though the file data is already in memory and in clear format. If the user is not entitled to access the file, the access is refused. File encryption does not eliminate the role of traditional access permissions, but it does add more granularity and flexibility.
In order to be able to create and use the EFS-enabled file system on a system, the following prerequisites must be met:

- Install the CryptoLite in C (CliC) cryptographic library.
- Enable the RBAC.
- Enable the system to use the EFS file system.

### 7.4.1 EFS commands used

This section discusses the Encrypted File Systems (EFS) essential commands used.

**efsenable**
The efsenable command activates the EFS capability on a system. It creates the EFS administration keystore, the user keystore, and the security group keystore.

Keystore is a key repository that contains EFS security information. The access key to the EFS administration keystore is stored in the user keystore and the security group keystore. The efsenable command creates the /var/efs directory. The /etc/security/user and /etc/security/group files are updated with new EFS attributes on execution of this command.

**efskeymgr**
The efskeymgr command is dedicated to all key management operations needed by an EFS. The initial password of a user keystore is the user login password. Group keystores and admin keystores are not protected by a password but by an access key.

Access keys are stored inside all user keystores that belong to this group. When you open a keystore (at login or explicitly with the efskeymgr command), the private keys contained in this keystore are pushed to the kernel and associated with the process. If access keys are found in the keystore, the corresponding keystores are also opened and the keys are automatically pushed into their kernel.

**efsmgr**
The efsmgr command is dedicated to the files encryption and decryption management inside EFS. Encrypted files can only be created on the EFS-enabled JFS2 file systems. Inheritance is set on the file system or the directory where the file is being created using this command. When inheritance is set on a directory, all new files created in this directory are encrypted by default. The cipher used to encrypt files is the inherited cipher. New directories also inherit the same cipher.

If inheritance is disabled on a subdirectory, the new files created in this subdirectory will not be encrypted.

Setting or removing inheritance on a directory or a file system has no effect on the existing files. The efsmgr command must be used explicitly to encrypt or decrypt files.

### 7.4.2 Sample scenario of EFS

We have a sample scenario of a company that has three departments, namely sales, marketing, and finance. These three departments share the same AIX machine to store their confidential content. If EFS is not enabled, the potential of having the data exposed between the three departments is extremely high.
Enable EFS
To enable the EFS we use efsenable command as the following example

**Example 7-47  Enabling EFS using efsenable**

```
# efsenable -a
Enter password to protect your initial keystore:
Enter the same password again:
```

We can check the EFS directories have been created under /var/efs which will facilitate the EFS operation. See below example.

**Example 7-48  Checking EFS directories**

```
# cd /var/efs
# ls
  efs_admin efsenabled groups users
```

Create the encrypted file system
All of the EFS capabilities should now be enabled. You are now going to create a separate file system for all the three departments.

The creation of an EFS is similar to the creation of a normal file system. The only difference is that you have to enable the EA2 efs = yes attribute. The below example shows how to create the encrypted file systems.

**Example 7-49  Creating encrypted file systems**

```
# crfs -v jfs2 -g rootvg -m /sales -a size=100M -a efs=yes
# crfs -v jfs2 -g rootvg -m /marketing -a size=100M -a efs=yes
# crfs -v jfs2 -g rootvg -m /finance -a size=100M -a efs=yes
```

You have now successfully created three separate file systems for these three departments with enabling efs option.

Create the users and check the keystores
To create the users, you will use `mkuser` command as the below example, and check the keystore.

**Example 7-50  Creating the users example**

```
# mkuser salesman
# passwd salesman
# mkuser marketingman
# passwd marketingman
# mkuser financeman
# passwd financeman

# ls /var/efs/users
  .lock salesman marketingman financeman root
```

Create EFS directories and set the EFS properties
In order to create EFS directories, you need the EFS file system to be mounted. The below example shows creating EFS directories and setting the inheritance.
Example 7-51  Creating the EFS directory and set the inheritance

```bash
# mount /finance
# cd /finance
# mkdir yearlyreport
# efsmgr -E yearlyreport
# efsmgr -L yearlyreport
EFS inheritance is set with algorithm: AES_128_CBC
```

The yearlyreport directory is now set for inheritance. It indicates that a file or directory inherits both the property of encryption and all encryption parameters from its parent directory.

There are various options with efsmgr, which facilitates you to set the type of cipher to be used on this directory, enable and disable inheritance, and add or remove users and groups from the EFS access list of this directory.

Now the encryption is on, and if you tried to create files in the directory yearlyreport, it should be giving an error as described in Example 7-52.

Example 7-52  Checking the EFS enablement

```bash
# cd yearlyreport
# ls
# touch apr_report
  touch: 0652-046 Cannot create apr_report.
```

Load EFS keystore into the shell

To perform the previous touch command activity, you should have the EFS keystore loaded into the shell. The following example show how to load the EFS keystore.

Example 7-53  Loading the EFS keystore into the shell

```bash
# efsmgr -o ksh
financeman's EFS password:
# touch enc_file
```

Now that you have loaded the keystore, any information that is added to this file is encrypted at the file system level.

Check the encryption of files and the ability to encrypt single file

You can always check the encryption flag of the files created in the encrypted file systems and even to encrypt individual files.

Check the encryption

Checking the encryption of files can be done by using ls command as described in Example 7-54.

Example 7-54  Listing encrypted file

```bash
# ls -U enc_file
-rw-r--r--e 1 financeman system 0 Oct 21 09:14 enc_file
```
The -U option in ls command will print extra bit which is the 11th character, it can be either one of the following:

- **E** indicates a file has extended attributes information. The EA of a file is displayed by using the `getea` command.
- **-** indicates a file does not have extended attributes information.
- **e** indicates a file is encrypted.

You can also list the encrypted file attributes using the efsmgr command as described in Example 7-55.

**Example 7-55  Listing encrypted file attributes**
```
# efsmgr -l enc_file
EFS File information:
Algorithm: AES_128_CBC
List of keys that can open the file:
Key #1:
Algorithm : RSA_1024
Who : uid 0
Key fingerprint : 4b6c5f5f:63cb8c6f:752b37c3:6bc818e1:7b4961f9
```

**Encrypt an individual file**

If you need to encrypt an individual file, you can use efsmgr command as the following example.

**Example 7-56  Encrypting individual file**
```
# cd /finance
# touch salarylist
# ls -U
total 16
-rw-r--r--- 1 root system 8 Nov 28 06:21 salarylist
# efsmgr -c AES_192_ECB -e companylist
# ls -U companylist
-rw-r--r--e 1 root system 8 Nov 28 06:24 companylist
```

### 7.4.3 Integrate EFS keystore with OpenSSH key authentication

This section discusses how to configure EFS keystore access while openssh public key authentication is used. It explains the procedure for automatic opening of EFS keystore when SSH public key authentication is used to log on to remote system.

**SSH configuration**

This part sets up a public key authentication first in openssh.

**Enable public authentication on both client and server**

Create a user on the client side and generate keys for this user. Public-private key pairs can be generated using the ssh-keygen command after that.

So just enable the public key authentication on both client and servers by `/etc/ssh/ssh_config` file and set PubKeyAuthentication to yes as the following example.
Example 7-57  Enabling public key authentication

```
# hostname
client.ibm.com
# grep PubkeyAuthentication /etc/ssh/ssh_config
PubkeyAuthentication yes
# hostname
server.ibm.com
# grep PubkeyAuthentication /etc/ssh/ssh_config
PubkeyAuthentication yes
```

Configure openssh client and server to use EFS login

To enable EFS login for both client and server you have to set AllowPKCS12keystoreAutoOpen to yes in /etc/ssh/ssh_config as the following Example 7-58.

Example 7-58  Enabling EFS login for client and server

```
# hostname
client.ibm.com
# grep AllowPKCS12keystoreAutoOpen /etc/ssh/ssh_config
AllowPKCS12keystoreAutoOpen yes
# hostname
server.ibm.com
# grep AllowPKCS12keystoreAutoOpen /etc/ssh/ssh_config
AllowPKCS12keystoreAutoOpen yes
```

Restart ssh daemon in both client and server

You have to restart sshd in both client and server to take effect as the following example.

Example 7-59  Restarting sshd on both client and server machines

```
# hostname
client.ibm.com
# stopsrc -s sshd
0513-044 The sshd Subsystem was requested to stop
# startsrc -s sshd
0513-059 The sshd Subsystem has been started. Subsystem PID is 209040.
# hostname
server.ibm.com
# stopsrc -s sshd
0513-044 The sshd Subsystem was requested to stop
# startsrc -s sshd
0513-059 The sshd Subsystem has been started. Subsystem PID is 206378
```

Create a user in the client side and generate the key by the user

Generate the key by a newly created user from the client machine. The below example shows how to create the user and create the key.

Example 7-60  Creating a client user and generate a public key

```
# hostname
client.ibm.com
# mkuser bob
# su - bob
$ ssh-keygen -t rsa -b 2048
```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bob/.ssh/id_rsa):
Created directory '/home/bob/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/bob/.ssh/id_rsa.
Your public key has been saved in /home/bob/.ssh/id_rsa.pub.
The key fingerprint is
The key's randomart image is
+---[ RSA 2048]-+
 | .......o+=ooo+. |
 | 0.........+o. = |
 | ............+ |
 | ............+ |
 | S . . . . . . |
 | . . . . . . . |
 | o= . . . . . |
+-----------------+

In the previous ssh-keygen command, it prompts for passphrase. This passphrase will be used to encrypt the private-key file on the client side, even ssh-keygen command will accept the empty passphrase, in which case, private-key file will not be encrypted.

Copy the public keys on to the server in the file ~/.ssh/authorized_keys as the following Example 7-61 shows.

Example 7-61 Copying public keys on to the server in authorized_keys

# hostname
server.ibm.com
# cat id_rsa.pub > /home/laxman/.ssh/authorized_keys
# cat /home/laxman/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABAIAwAAQEAqYK16NpoJ1Nq1/ccb1Ftu2fGkQd2T4H74d1c6Ogs
kRHGo7eQyQtg58yFJ05h7Zr8g1eQLo09H6CVA7h17EKwfg7fPpGWUdpL6
Ag8g4wRkhJ0yptczeRuj
jSC17hyvT2DhLx7svZ0x47K1PfHFTNPrujKZ1lyPscTs2XwqAdVnPQPV0T14agRFqB81d/gXm2vfS
VUP+PJD0VVub/DMY928FRv6fYEfGZybyMOR1
4kbuQoJFrnoGZAcg4maiPi5fKLiXYOW1+

Upon doing that, any number of a client user's public key can be copied in the file ~/.ssh/authorized_keys file on server user account.

EFS has to be enabled on the server side using the efsenable command. This creates an adminkeystore. The keystore gets created for a user whenever a new user is created, and when password is assigned to the user or when user logs in.

The path where user keystore gets created on the system is /var/efs/users/<userlogin>/keystore, and the format of user keystore is in PKCS#12 which contains public and private objects.

Private objects are protected by user access key. This user access key is nothing but a hash of a user defined password, either login password or another password specific to EFS.
**Insert the public key into server keystore**

Now the public key cookie needs to be created and inserted into the keystore on server side. User invokes the efskeymgr command to insert the cookie. A public key cookie is the `passwd` encrypted with users public key.

The following example shows how to create a keystore for a user and insert the public key cookies as per Example 7-62.

**Example 7-62  Creating keystore and insert the public key cookies**

```
# hostname
server.ibm.com
# passwd laxman
laxman's New password:
Enter the new password again:
# ls -l /var/efs/users/laxman
  total 8
  -rw------- 1 root system 0 Oct 21 15:40 .lock
  -rw------- 1 root system 1914 Oct 21 15:40 keystore
# su - laxman
$ cd .ssh
$ ls
authorized_keys id_rsa id_rsa.pub
$ efskeymgr -P authorized_keys
laxman's EFS password:
# ls -l /var/efs/users/laxman
  total 8
  -rw------- 1 root system 0 Oct 21 15:40 .lock
  -rw------- 1 root system 2252 Oct 21 15:42 keystore
```

Once all the previous configuration setting are complete, run the ssh to log onto the remote machine using the public key authentication

**Verify EFS login authentication**

The openssh client user bob is all set for Public Key Authentication to user laxman on the openssh server with EFS login. You have just to verify the same with ssh login from client as shown in Example 7-63.

**Example 7-63  Verifying login authentication**

```
# hostname
client.ibm.com
# su - bob
$ ssh -vvv laxman@server.ibm.com
*************************************************************
**  Welcome to AIX Version 7.2!
*  
** Please see the README file in /usr/lpp/bos for information
**  pertinent to this release of the AIX Operating System.
*************************************************************
$ efskeymgr -V
List of keys loaded in the current process:
Key #0:
  Kind ................. User key
  Id (uid / gid) ....... 216
```
This setup can be used along with different applications including database engines for which OpenSSH public key authentication can be used.

A good example for this is IBM DB2. If you are running a DB2 system on the AIX operating system, you have the option to set up an encrypted database by using AIX encrypted file system. For more information on DB2 encryption using EFS, please see the link below:


### 7.5 AIXpert and PowerSC integration

AIXpert is a security hardening AIX tool which is a part of the bos.aixpert fileset. It uses a policy file to automatically lock down AIX. It has various settings like low, medium, high, and SOX/Cobit.

The policy has well over a hundred rules and tools that it runs in all the regular areas like password aging and lockout on failed logins, SUID bit programs, disabling services, blocking port scanning, and so on.

AIX Security Expert can be used to implement the appropriate level of security, without the necessity of reading a large number of papers on security hardening and then individually implementing each security element. It can also be used to take a security configuration snapshot. This snapshot can be used to set up the same security configuration on other systems. This saves time and ensures that all systems have the proper security configuration in an enterprise environment.

#### 7.5.1 Using AIXpert

The essence of AIXpert is that it is a network and security hardening tool, which incorporates many functions into one system.

Prior to AIXpert, you needed to remember many different commands, AIXpert incorporates over 300 security configuration settings, while still providing control over each element.

In other words, it provides a center for all security settings, including TCP, IPSEC, and auditing.

Essentially, there are four different levels that can be defined as part of the system: high, medium, low, and advanced.

**High level**

It should be used in environments where security and controls are of paramount importance. Note that applications such as telnet, rlogin, and FTP, and other systems that transmit non-encrypted passwords will not work, so be careful when turning this high level on.
Understand that most ports will be blocked in this scenario. Systems that may be connected directly to the Internet that have sensitive data are good examples of a system that would be run with this level.

**Medium level**
Which is appropriate for systems that reside behind a firewall, where users need to use services such as telnet, but still want protection. This setting also provides port scan and password-setting protections.

**Low level**
Which is configured usually when the system is in an isolated and secured type of LAN. This is used where system administrators need to be careful not to interrupt any services to the environment. This article shows you how to configure these settings.

**Advanced level**
Which is for customization. It allows you to use different rules from different levels, the rules themselves being mutually exclusive of one another. It does not in itself provide a higher level of security.

More than any specific feature, it is the consolidation of security that really adds the value to AIXPert. For example, AIXPert incorporates the File Permissions Manager in the form of the fpm command to also help manage SUID programs. This feature is demonstrated later in the article. Another important feature is the ability to take snapshots of your system and reproduce these settings through the enterprise. This allows you to clone your security features across the organization. This feature includes the ability to undo settings and also checks the overall security health of the systems to report back on settings that may have been changed.

The ease with which you can undo security levels cannot be overstated. It's as simple as running this command from the command line: `aixpert - u undo.xml`. This undo feature is one of the enhancements that were made.

Setting specific AIXpert security level can be done using AIXpert command as shown in Example 7-64.

```
Example 7-64   Setting specific AIXpert security level

# aixpert l <LEVEL>
```

Where the level can be as the following options in Table 7-3.

<table>
<thead>
<tr>
<th>AIXpert security level</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>Specifies high level security options</td>
</tr>
<tr>
<td>medium</td>
<td>Specifies medium level security options</td>
</tr>
<tr>
<td>low</td>
<td>Specifies low level security options</td>
</tr>
<tr>
<td>default</td>
<td>Specifies AIX standard level security options</td>
</tr>
<tr>
<td>sox-cobit</td>
<td>Specifies SOX-COBIT best practices levels security options</td>
</tr>
</tbody>
</table>

More than any specific feature, it is the consolidation of security that really adds the value to AIXPert. For example, AIXPert incorporates the File Permissions Manager in the form of the fpm command to also help manage SUID programs. This feature is demonstrated later in the article. Another important feature is the ability to take snapshots of your system and reproduce these settings through the enterprise. This allows you to clone your security features across the organization. This feature includes the ability to undo settings and also checks the overall security health of the systems to report back on settings that may have been changed.

The ease with which you can undo security levels cannot be overstated. It's as simple as running this command from the command line: `aixpert - u undo.xml`. This undo feature is one of the enhancements that were made.

Setting specific AIXpert security level can be done using AIXpert command as shown in Example 7-64.

```
Example 7-64   Setting specific AIXpert security level

# aixpert l <LEVEL>
```

Where the level can be as the following options in Table 7-3.
7.5.2 Using AIXpert to generate a compliance report by pscxpert PowerSC

Since AIXpert can be implemented and used to generate a compliance reports of the security level imposed, we can always get a compliance report generated by pscxpert command which is a part of PowerSC fileset.

In the following scenario, we will apply the low level security by AIXpert.

1. To implement a low level AIXpert security, we will use aixpert command as shown in the following Example 7-65.

Example 7-65 Implementing AIXpert low level security in the AIX

```bash
# aixpert -l low
```

The command takes time to process several rules, and will get eventually the failing rules. The things, if you need a verbose running which is preferred, you will have to use -p option as Example 7-66 shows below.

Example 7-66 Applying low level security with verbose mode

```bash
# aixpert -l low -p
Processing lls_maxage_97D38C75 :done.
Processing lls_maxexpired_97D38C75 :done.
Processing lls_minlen_97D38C75 : failed.
Processing lls_minalpha_97D38C75 :done.
Processing lls_minother_97D38C75 :done.
[...]
Processing lls_sockthresh_97D38C75 :done.
Processing lls_crontabperm_97D38C75 :done.
Processing lls_rmdotfrmpathroot_97D38C75 :done.
Processedrules=44 Passedrules=43 Failedrules=1 Level=AllRules
Input file=/etc/security/aixpert/core/appliedaixpert.xml
```

As shown in previous b Example 7-66 example, we have one only failed rule. This means one rule that does not comply with the low level security of AIXpert which is the lls_minlen:

Processing lls_minlen_97D38C75 : failed.

The low level security has taken the instructions of an XML file to apply it. By the end of the command shown in previous Example 7-66, it lists the XML input file which is /etc/security/aixpert/core/appliedaixpert.xml file.

2. As per the output, there must be a discrepancy between the minlen rule from XML file, and what is currently implemented in the AIX system at the moment.

The following Example 7-67 shows the minlen attribute in both the XML input file and the current implemented in /etc/security/user file

Example 7-67 Checking the rule in XML file and in /etc/security/user

```bash
# grep minlen /etc/security/aixpert/core/appliedaixpert.xml
</AIXPertEntry name="lls_minlen_97D38C75" function="minlen">
</AIXPertArgs>minlen=8 ALL lls_minlen</AIXPertArgs>

# grep "minlen =" /etc/security/user
minlen = 0
```
3. Based on the difference shown in the previous Example 7-67, we will comply to the minlen of 8 and will change it in /etc/security/user.

After doing so, we can check the security settings against the previously applied set of rules using aixpert command with -p option for verbosity as shown in the following Example 7-68 on page 146.

```
Example 7-68  Checking security settings against the previously applied set of rules

# aixpert cp | grep lls_minlen
Processing lls_minlen_97D38C75 :done.
```

4. Since we are sure now we are compliance with the security rule level we set earlier, we can generate a compliance report using pscxpert which is a part of powerscStd.ice files.

Using pscxpert command to generate a compliance report will generate .txt and .csv files. Below Example 7-69 shows how to generate it.

```
Example 7-69   Generating a compliance report by pscxpert

# pscxpert -c -r
Processed rules=44 Passed rules=44 Failed rules=0 Level=LLS
Input file=/etc/security/aixpert/core/appliedaixpert.xml
```

5. We can see a sample of the .txt file showing the compliance report. See Example 7-70.

```
Example 7-70   Checking the .txt version compliance report

# cat /etc/security/aixpert/check_report.txt
...
server.ibm.com,10.10.10.10,Minimum length for password: Specifies the minimum length of a password to 8,/etc/security/aixpert/bin/chusrattr minlen=8 ALL lls_minlen,PASS
...
```

The other version of the security compliance report is generated into a .csv file /etc/security/aixpert/check_report.csv which can be loaded into a spreadsheet and the results can be filtered.

### 7.6 AIX Auditing subsystem and AHAFS

IBM AIX has different ways to audit and monitor the system for any unknown action happening on the AIX system. The most powerful options to monitor the system or to monitor specific action is to enable either the AIX audit subsystem or the AHAFS monitoring.

#### 7.6.1 AIX Auditing subsystem

The auditing subsystem provides the means to record security-related information and to alert system administrators of potential and actual violations of the system security policy. The information collected by auditing includes: the name of the auditable event, the status (success or failure) of the event, and any additional event-specific information related to security auditing.

While dealing with AIX auditing subsystem, we have to understand the following five items.
Mode
BIN mode, STREAM mode, or both at the same time can be chosen while setting up the audit.
BIN mode is useful when you plan to store records on a long term basis.
However STREAM mode is useful when you want to do real-time information processing.

Events
Events are particular security related activities that have been defined by the system. The following are examples of events:
FILE_Open (file is opened)
FILE_Read (file is read)
FILE_Write (file is written to standard output)
PROC_Create (process creation for more OR cat)
PROC_Execute (command execution)
PROC_Delete (process completion)
Check all available events in the following link

Classes
Classes define groups of events. There are one or more events in a class.
For example, both USER_SU and PASSWORD_Change are grouped in a general class.
Class names are arbitrary, and you can define any class name for a given group of events.

User
You can define what classes you want to audit for a specific user. You can audit one or more classes per user.

Object
Object means files, so auditing objects means auditing files. file read, write, and execute can be audited using audit objects, and the file itself can be an executable command or just a certain configuration file.

7.6.2 Implementing AIX audit subsystem for exact event
We can define what classes we want to audit for a specific user, and we can audit one or more classes per user.
The following scenario goes through auditing setup against PROC_Execute event which means when a user runs any program.

Apply the mode
What we will do first is to edit /etc/security/audit/config file and apply the desired mode.
See Example 7-71 that shows enabling the stream mode.
Example 7-71   Changing the audit mode in /etc/security/audit/config

```bash
# vi /etc/security/audit/
start:
  binmode = off
  streammode = on
```

Define a class name
We will define our class that will include the exact event we need.

So at the end of the classes section in /etc/security/audit/config file on the line before the users, enter a new line as the following Example 7-72 on page 148.

Example 7-72   Define a class that has the desired event name in audit config file

```bash
classes:
  ...
  my_class = PROC_Execute
```

Add the newly defined class under a specific user
Under the user stanza in /etc/security/audit/config file, you may delete all the entries and add an entry for the class you just defined, or just add it by the very end of the users section as shown in Example 7-73.

Example 7-73   Adding the class to a specific user

```bash
users:
  ...
  my_user = my_class
```

Note: If you have any other users you want to monitor him for any PROC_Execute then add an entry for that user name.

Adjust audit stream commands file to have auditstream invoked
You may adjust /etc/security/audit/streamcmds file to include /usr/sbin/auditstream command redirected to a certain output file, so it gets invoked when the audit system is initialized. The path name of this file is defined in the stream stanza of the /etc/security/audit/config file.

You can just backup /etc/security/audit/streamcmds file and edit the contents as Example 7-74 shows below.

Example 7-74   Adding auditstream command redirected to an output file

```bash
# cp /etc/security/audit/streamcmds /etc/security/audit/streamcmds.orig
# vi /etc/security/audit/streamcmds
/usr/sbin/auditstream | auditpr -herlRtcpP -v -t1 > /audit/stream.out &
```

Optionally you can a monitoring action upon specific action
You can always monitor exact object, which is a certain file or a certain command by monitoring its read, write or execution.
What we will do is to setup an audit on the execution of date command object, which has the full path of /usr/bin/date.

The first step is to define this object by a name in /etc/security/audit/object and define what action we would need to audit, like r for read, w for write and x for execution. See Example 7-75 on page 149 below for defining an execution action against the date command.

**Example 7-75  Defining an audit object to monitor its execution**

```
/usr/bin/date:
  x = "DATE_EXEC"
```

We have now defined an event named DATE_EXEC that will be reported every time the date command is executed.

Next step is to format or to define the format string of the DATE_EXEC event which will be reported. This can be defined in /etc/security/audit/events file as shown in Example 7-76.

**Example 7-76  Defining an event format**

```
* /usr/bin/dateS_FILE_EXECUTE = printf "%s"
```

%ss means the event will be formatted as a text string.

See the following link for events file for more information about the text formats


**Do not forget to recycle the audit**

For the configuration to work and take effect, you will have to stop and start the audit subsystem as shown in Example 7-77.

**Example 7-77  Recycling audit subsystem**

```
# audit shutdown
# audit start
```

Then you will wait until the issue you set the audit for happens. Once happened, stop the audit and read the file stream.out as shown in Example 7-78.

**Example 7-78  Stopping the audit and monitor the events**

```
# audit shutdown
# cd /audit
# more stream.out
```

See the following best practices link for more information


### 7.6.3 AHAFS

The Autonomic Health Advisor File System (AHAFS) provides event monitoring framework in IBM AIX. Users can monitor predefined system events and get notified about them.
Each event is represented as a file in a pseudo file system named AHAFS. The standard file APIs such as open, read, write, close, and select can be used by applications to monitor the files and retrieve the event data. The instance of the event infrastructure file system must be mounted to monitor the events.

Each type of event, which may be monitored, is associated with an event producer. Event producers are simply sections of code which can detect an event as it happens and notify the AIX event infrastructure of event occurrences.

The following Table 7-4 represents examples of available event producers.

<table>
<thead>
<tr>
<th>Predefined event producer</th>
<th>When it notifies consumer</th>
</tr>
</thead>
<tbody>
<tr>
<td>utilFs</td>
<td>The utilization of a monitored file system crosses the user specified threshold</td>
</tr>
<tr>
<td>modFile</td>
<td>The contents of a monitored file are modified</td>
</tr>
<tr>
<td>modFileAttr</td>
<td>The attributes like modebits, access control list, ownership of a monitored file are modified</td>
</tr>
<tr>
<td>modDir</td>
<td>A file or subdirectory is created, renamed or deleted under a directory</td>
</tr>
<tr>
<td>schedo</td>
<td>The value of a monitored scheduler tunable has been changed</td>
</tr>
<tr>
<td>vmo</td>
<td>The value of a monitored VMM tunable has been changed</td>
</tr>
<tr>
<td>waitTmCPU</td>
<td>The average wait time of all the threads waiting for CPU resources exceeds the user specified threshold</td>
</tr>
<tr>
<td>waitersFreePg</td>
<td>The number of waiters to get a free page frame in the last second exceeds the user specified threshold</td>
</tr>
<tr>
<td>waitTmPginOut</td>
<td>The average wait time of all the threads waiting for page in or page out operations to complete exceeds the user specified threshold</td>
</tr>
<tr>
<td>processMon or pidProcessMon</td>
<td>The monitored process exits</td>
</tr>
</tbody>
</table>

Events are represented as files within a pseudo file system. Existing file system interfaces read(), write(), select(), and so on are used to specify how and when monitoring applications (also named consumers) should be notified and to wait on and read data about event occurrences. The path name to a monitor file within the AIX event infrastructure file system is used to determine which event a consumer wishes to monitor.

**AHAFS setup sample**

The AHAFS is shipped with many useful sample programs and scripts under the directory /usr/samples/ahafs/ designed to make AHAFS very easy to use. The directory /usr/samples/ahafs/bin/ contains the script aha.pl and its input file aha-pl.inp which can be used to monitor events without writing any code.

All you need to do is make a copy of the aha-pl.inp file and modify it by uncommenting lines in the file for things you want to monitor; or modify some key values if you want to use values other than the defaults, then invoke the aha.pl script specifying your tailored .inp file, and optionally specifying the list of email IDs to which you want the monitoring reports to go to.
By default, the AHAFS fileset bos.ahafs comes with AIX. If it is installed, you can type the following commands as a root user to create /aha as a mount point and then mount the AHAFS file system as shown in Example 7-79.

Example 7-79  Mounting /aha file system

```
# mkdir /aha
# mount -v ahafs /aha /aha
```

Mounting the AHAFS will create the following items.

**Event produce list**

/aha/evProds.list file which contains the list of predefined and user defined event producers available to this AIX instance. This is a special file in which its content can be viewed by listing the file contents.

**Monitor factory group components**

The components for grouping the monitor factories, they are the subdirectories under /aha like mem/, cpu/, fs/ and so on.

**Monitor factories**

The subdirectories under the component subdirectories with the file type of .monFactory, each of which corresponds to an event producer for AHAFS.

We will discuss an example below to show a procedure of setup.

- Copy aha-pl.inp file to another directory and edit it to have the monitoring line as shown in Example 7-80

Example 7-80  Editing aha-pl.inp file to monitor /tmp file system

```
# cp /usr/samples/ahafs/bin/aha-pl.inp /myfiles
# vi /myfiles/aha-pl.inp
...
/aha/fs/modDir.monFactory/tmp.mon YES -- -- 2 2 -- 00:00:05:00
```

This previous Example 7-80 monitors the /tmp directory for any changes, and send an email alert after the second change, then it will re-arm the alert to start watching in the next 5 minutes. That way they aren't overrun with messages.

- Mount the aha file system and start the monitoring perl script as shown in Example 7-81.

Example 7-81  Mounting aha file system and start monitoring

```
# mkdir /aha
# mount -v ahafs /aha /aha
# /usr/samples/ahafs/bin/aha.pl -i /myfiles/inputfile -e bob
```

bob is the user who get an email notification of the event. The above scenario will do the following things

1. Notify about the event match on the stdout as well as in the email.
2. The notification will match any new files/subdirectories created, renamed, copied or deleted under a directory /tmp.
3. The event match will not include changes of the files content, because it is modDir and Not a modFile event.
4. The event match will not include any metadata modification (ownships, permissions and so on), because it is not modFileAttr event.

7.7 MultiBOS

The multibos utility allows you, as root, to create multiple instances of AIX on the same current root volume group (rootvg).

Which is totally different than altinst_rootvg cloning, in which it processes the clones into different disks.

So the two main differences here are:
- The instances will be inside the rootvg itself.
- Only two instances of BOS are supported per rootvg.

The multibos setup operation creates a standby base operating system (BOS) that boots from a distinct Boot Logical Volume (BLV). This creates two bootable instances of base operating systems on a given rootvg.

You can boot from either instance of a BOS by specifying the respective Boot Logical Volume (BLV) as an argument to the bootlist command, or using system firmware boot operations.

You can simultaneously maintain two bootable instances of a BOS. The instance of a BOS associated with the booted BLV will be mentioned as the active BOS.

The instance of a BOS associated with the BLV that has not been booted is mentioned as the standby BOS. Only two instances of BOS are supported per rootvg.

The multibos allows you to access, install, maintain, update, and customize the standby BOS either during setup or during any subsequent customization operations. Installing technology level updates to the standby BOS does not change system files on the active BOS. This allows for concurrent update of the standby BOS, while the active BOS remains in production.

General requirements for multibos:
- The current rootvg must have enough space for each BOS object copy. BOS object copies are placed on the same disk or disks as the original.
- The total number of copied logical volumes cannot exceed 128. The total number of copied logical volumes and shared logical volumes are subject to volume group limits.

7.7.1 Standby BOS setup

The following section discusses the MultiBOS standby operation and procedures.

Standby BOS setup operation
The standby BOS setup creates the standby base operating system instance. The following actions are performed once the standby BOS setup initiated:
- The multibos methods are initialized.
- If you provide a customized image.data file, it is used for the logical volume attributes. Otherwise, a new one is generated. You can use the customized image.data file to change BOS object (logical volume or file systems) attributes. You cannot use the customized image.data file to add or delete BOS logical volumes or file systems.
The standby logical volumes are created based on image.data attributes. The active and standby logical volumes are marked with unique tags in the logical volume control block. The multibos utility uses these tags to identify copied logical volumes. If the active logical volume names are classic names, such as hd2, hd4, hd5, and so on, then the bos_prefix is prepended to create a new standby name. If the active logical volume names have the bos_prefix, the prefix is removed to create a new standby name.

The standby file systems are created based on image.data attributes. The active and standby file systems are marked with unique tags in the hosting logical volume control block and /etc/filesystems. The multibos utility uses these tags to identify copied logical volumes. The /bos_inst prefix is prepended to the original active file system name to create the standby file system name. The standby file system name may not exceed the system’s PATH_MAX limit. The standby file systems appear as standard entries in the active BOS /etc/filesystems.

The standby file systems are mounted.

A list of files that will be copied from the active BOS is generated. This list is comprised of the current files in copied active BOS file systems, less any files that you excluded with the optional exclude list.

The list of files generated in the previous step is copied to the standby BOS file systems using the backup and restore utilities.

Any optional customization is performed. This can include installation of fileset updates or other software.

The standby boot image is created and written to the standby BLV using the AIX bosboot command. You can block this step with the -N flag. Only use the -N flag if you are an experienced administrator and have a good understanding the AIX boot process.

The standby BLV is set as the first boot device, and the active BLV is set as the second boot device. You can skip this step using the -t flag.

**Note:** The Logical Volume Manager (LVM) limits the maximum length of a logical volume name to 15 characters. This means that any logical volume classic name may not exceed 11 characters. You can rename logical volumes that have classic names that exceed 11 characters using the chlv command. If the active logical volume name already has the bos_prefix, then the prefix is removed in the standby name.

**Standby BOS setup scenario**

The following scenario and examples go through the standby BOS commands.

The multibos -s command creates the multibos instance, it would be always better if you run the command in preview mode beforehand by adding -p flag, and also you can include -X flag to allow for automatic file system expansion if space is needed during the instance setup operation.

Once you got "Return Status = SUCCESS", then you are good to go without -p. The log file for every multibos operation is /etc/multibos/logs/op.alog. See Example 7-82.

**Example 7-82  Previewing the standby BOS instance creation**

```
# multibos -sXp
Initializing multibos methods ...
Initializing log /etc/multibos/logs/op.alog ...
Gathering system information ...

+-----------------------------------------------------------------------------+
Preview
+-----------------------------------------------------------------------------+

Verifying operation parameters ...```
Processing preview information ...

ACTIVE LV:        hd4
STANDBY LV:       bos_hd4
TYPE:             jfs2
ACTIVE FS:        /
STANDBY FS:       /bos_inst
ACTION:           Setup
STATE:            mounted

ACTIVE LV:        hd2
STANDBY LV:       bos_hd2
TYPE:             jfs2
ACTIVE FS:        /usr
STANDBY FS:       /bos_inst/usr
ACTION:           Setup
STATE:            mounted

ACTIVE LV:        hd9var
STANDBY LV:       bos_hd9var
TYPE:             jfs2
ACTIVE FS:        /var
STANDBY FS:       /bos_inst/var
ACTION:           Setup
STATE:            mounted

ACTIVE LV:        hd10opt
STANDBY LV:       bos_hd10opt
TYPE:             jfs2
ACTIVE FS:        /opt
STANDBY FS:       /bos_inst/opt
ACTION:           Setup
STATE:            mounted

ACTIVE LV:        hd5
STANDBY LV:       bos_hd5
TYPE:             boot
ACTIVE FS:        None
STANDBY FS:       None
ACTION:           Setup
STATE:            closed

Log file is /etc/multibos/logs/op.alog
Return Status = SUCCESS

Now we can run the normal command without a preview. See Example 7-83 that describes the multbos stages it take to create the instance.

Example 7-83 Creating multbos instance

# multbos -sX
Initializing multbos methods ...
Initializing log /etc/multibos/logs/op.alog ...
Gathering system information ...
Setup Operation

Verifying operation parameters ...
Creating image.data file ...

Logical Volumes

Creating standby BOS logical volume bos_hd5
Creating standby BOS logical volume bos_hd4

[......]

Bootlist Processing

Verifying operation parameters ...
Setting bootlist to logical volume bos_hd5 on hdisk0
Log file is /etc/multibos/logs/op.alog

Checking the file systems and logical volumes created by `lsvg` and `lsfs` command as shown in Example 7-84.

**Example 7-84  Checking the standby logical volumes**

```
# lsvg -l rootvg | grep bos_
bos_hd5  boot  1  1  1  closed/syncd  N/A
bos_hd4  jfs2  16  16  1  closed/syncd  /bos_inst
bos_hd2  jfs2  32  32  1  closed/syncd  /bos_inst/usr
bos_hd9var jfs2  16  16  1  closed/syncd  /bos_inst/var
bos_hd10opt jfs2  4  4  1  closed/syncd  /bos_inst/opt

# lsfs | grep bos_
/dev/bos_hd4  -- /bos_inst  jfs2  4194304 --  no  no
/dev/bos_hd2  -- /bos_inst/usr  jfs2  8388608 --  no  no
/dev/bos_hd9var  -- /bos_inst/var  jfs2  4194304 --  no  no
/dev/bos_hd10opt  -- /bos_inst/opt  jfs2  1048576 --  no  no
```

Since both active and standby instances are in same hdisk0, we can set the boot list to check first the standby instance then second in the list to check the active instance.

The current active instance is the normal AIX logical volumes and mount points, however the standby instance would be the logical volumes having `bos_` prefix as shown in Example 7-84.

Setting the boot list to have the standby instance first then the current active instance requires specifying the boot logical volume name as shown in the following Example 7-85.

**Example 7-85  Setting the boot list to point to standby instance first**

```
# bootlist -m normal hdisk0 blv=bos_hd5 hdisk0 blv=hd5
# bootlist -om normal hdisk0 blv=bos_hd5 pathid=0
hdisk0 blv=hd5 pathid=0
```
Let us reboot, and see how the instance is being activated and figure out and confirm which instance we have booted off. See Example 7-86.

**Example 7-86  Rebooting the system and check which instance we booted off**

```bash
# shutdown -r now

SHUTDOWN PROGRAM
Tue Oct 29 15:34:16 CDT 2019

Broadcast message from root@aixlpar (tty) at 15:34:16 ...
[...]
# bootinfo -b
hdisk0

# bootinfo -v
bos_hd5
```

```bash
# lsvg -l rootvg | grep "open/syncd"
hd6     paging  17  17  1   open/syncd  N/A
hd8     jfs2log  1   1   1   open/syncd  N/A
hd3     jfs2     8   8  1    open/syncd  /tmp
hd1     jfs2     1   1  1    open/syncd  /home
hd11admin jfs2     1   1  1    open/syncd  /admin
lg_dumplv sysdump  8   8  1    open/syncd  N/A
livedump jfs2     2   2  1    open/syncd  /var/adm/ras/livedump
bos_hd4  jfs2     16  16  1   open/syncd  /
bos_hd2  jfs2     32  32  1   open/syncd  /usr
bos_hd9var jfs2     16  16  1   open/syncd  /var
bos_hd10opt jfs2     4   4  1    open/syncd  /opt
```

```bash
# df -g | grep bos_
/dev/bos_hd4   2.00   1.95   3%   3770   1%  /
/dev/bos_hd2   4.00   2.12  48%  40627   8%  /usr
/dev/bos_hd9var 2.00   1.92   5%  1163   1%  /var
/dev/bos_hd10opt 0.50   0.13  75% 14434  32%  /opt
```

As shown in previous Example 7-86, the system booted off the standby image which now became the active one. The active one has the same mount points, but different logical volume names which is correct.

Important thing to know, the `bosboot -a` command will always update the boot image of the current boot logical volume we booted from. In our case, the current boot logical volume is `bos_hd5`. So running `bosboot -a` command will update the active boot image (`bos_hd5`) as per our current situation. To update the other standby image (`hd5`) you will always run a `multibos` command with `-B` option to do it for you.

The following Example 7-87 shows how to update the current active and the standby boot images.

**Example 7-87  Updating the current active and standby boot image**

```bash
# bootinfo -b
hdisk0
# bootinfo -v
bos_hd5
```

---

IBM AIX Market Enhancements and Modernization

156
# bosboot -ad hdisk0
bosboot: Boot image is 57372 512 byte blocks.

# multibos -B
Initializing multibos methods ...
Initializing log /etc/multibos/logs/op.alog ...
Gathering system information ...
[...]
+-----------------------------------------------------------------------------+
Active boot logical volume is bos_hd5.
Standby boot logical volume is hd5.
Creating standby BOS boot image on boot logical volume hd5
bosboot: Boot image is 57372 512 byte blocks.Setting bootlist to logical volume hd5 on hdisk0.
Log file is /etc/multibos/logs/op.alog
Return Status = SUCCESS

The most important thing in multibos that make the instances unique with their logical volumes and file systems is the tag number printed on top of each logical volume and attached to each file system information inside /etc/filesystems file. See Example 7-88 for multibos unique tagging.

Example 7-88  Checking multibos unique tagging

# cat /etc/multibos/data/acttag
2C7A38F29C1DD074
# cat /etc/multibos/data/sbytag
2C7A38F19C2D9238

# getlvcb -f bos_hd5
mb=2C7A38F29C1DD074:mbverify=0:mbs=true
# getlvcb -f hd5
mb=2C7A38F19C2D9238:mbverify=1:mbs=true

# getlvcb -f bos_hd4
vfs=jfs2:log=/dev/hd8:mb=2C7A38F29C1DD074:account=false:mbs=true
# getlvcb -f hd4
vfs=jfs2:log=/dev/hd8:mount=automatic:type=bootfs:mb=2C7A38F19C2D9238:mbs=true

# grep -wp "mb" /etc/filesystems | egrep ':|/dev|mb' | grep -v "log"
/: 
  dev = /dev/bos_hd4
  mb = 2C7A38F29C1DD074
/bos_inst:
  dev = /dev/hd4
  mb = 2C7A38F19C2D9238
[.....]

If you wants to remove an instance, either the current active one or the other standby one, you will have to boot off the instance that you actually want to keep, then issue the multibos command with option -R which will remove the other inactive instance. See Example 7-89.

Example 7-89  Removing the multibos instance

# multibos -RX
Initializing multibos methods ...
Initializing log /etc/multibos/logs/op.alog ...
Gathering system information ...

+-----------------------------------------------------------------------------+
Remove Operation
+-----------------------------------------------------------------------------+
Verifying operation parameters ...

[....]

+-----------------------------------------------------------------------------+
Boot Partition Processing
+-----------------------------------------------------------------------------+
Active boot logical volume is bos_hd5.
Standby boot logical volume is hd5.
Log file is /etc/multibos/logs/op.alog
Return Status = SUCCESS

# lsvg -l rootvg
rootvg:
 LV NAME  TYPE       LPs  PPs  PVs  LV STATE    MOUNT POINT       
hd6      paging    17   17   1  open/syncd  N/A
hd8      jfs2log    1    1    1  open/syncd  N/A
hd3      jfs2       8    8    1  open/syncd  /tmp
hd1      jfs2       4    4    1  open/syncd  /home
hd1admin jfs2       1    1    1  open/syncd  /admin
lg_dumplv sysdump    8    8    1  open/syncd  N/A
livedump  jfs2       2    2    1  open/syncd  /var/adm/ras/livedump
bos_hd5   boot       1    1    1  closed/syncd N/A
bos_hd4   jfs2       16   16   1  open/syncd  /
bos_hd2   jfs2       32   32   1  open/syncd  /usr
bos_hd9var jfs2       16   16   1  open/syncd  /var
bos_hd10opt jfs2      4    4    1  open/syncd  /opt

The previous example shows the removal process of the inactive instance, and since the current instance is the bos_LVNAME, this will be kept with no issues with the normal regular mount points as shown in Example 7-89.

If you just wanted to keep the normal names, you will simply have to boot off the normal hd5 boot logical volume to remove the other bos_LVNAME instance.
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.

(0.1" spine)
0.1"<->0.169"
53<->89 pages

(0.2" spine)
0.17"<->0.473"
90<->249 pages

(0.5" spine)
0.475"<->0.873"
250<->459 pages

(1.0" spine)
0.875"<->1.498"
460<->788 pages

(1.5" spine)
1.5"<->1.998"
789<->1051 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.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- NIM from A to Z in AIX 5L, SG24-7296
- IBM AIX Version 7.1 Differences Guide, SG24-7910
- IBM PowerVM Virtualization Managing and Monitoring, SG24-7590
- IBM PowerVM Virtualization Introduction and Configuration, SG24-7940
- IBM PowerVC Version 1.3.2 Introduction and Configuration, SG24-8199
- Implementing IBM VM Recovery Manager for IBM Power Systems, SG24-8426
- IBM PowerHA SystemMirror V7.2.3 for IBM AIX and V7.22 for Linux, SG24-8434
- iSCSI Implementation and Best Practices on IBM Storwize Storage Systems, SG24-8327

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- JFS2 and JFS differences
- lvmstat manual page
- SDDPCM to AIXPCM migration guide
- AIX Web Download Pack
- amepat command documentation
  https://www.ibm.com/support/knowledgecenter/ssw_aix_72/a_commands/amepat.html
- Introduction to Active Memory Expansion
- DB2 encryption using EFS
- Audit event listings
- Audit events file format
- AIX auditing best practices
- PowerSC Multifactory Authentication components and requirements
- IBM TouchToken concepts
- RSA SecurID Access Authentication Agent Implementation Guide for AIX 7.1
  https://community.rsa.com/docs/DOC-62752
- RSA SecurID Access Authentication Agent for PAM 7.1.0.2 announcement
  https://community.rsa.com/docs/DOC-77540
- OpenSSL cryptographic library home documentation
  https://www.openssl.org/
- Multiple alternative rootvg criteria documentation
  https://www.ibm.com/support/pages/multiple-alternate-rootvg-criteria
- Managing multiple instances of altinst_rootvg and applying fixes to them
- IBM Cloud
  https://www.ibm.com/cloud/power-virtual-server
- Google Cloud
  https://cloud.google.com/ibm
- Skytap
  https://www.skytap.com/product/cloud-platform/power-systems-aix-linux-ibmi
- Terraform documentation
  https://www.terraform.io
- Managing Cloud Automation Manager templates from Template Designer

- Structure of Cloud Automation Manager templates

- Terraform supported provisioner listing
  https://www.terraform.io/docs/provisioners/index.html

- Cloud Automation Manager content providers
  https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.1.0/content/cam_content_terraform_provider.html

- Ansible home page
  https://www.ansible.com

- Configuring YUM on AIX

- Ansible playbooks developed by the AIX open source community
  https://github.com/aixoss/ansible-playbooks

- Chef Supermarket
  https://supermarket.chef.io/cookbooks/aix

- Chef Infra Overview
  https://docs.chef.io/chef_overview.html

- Puppet Enterprise homepage
  https://puppet.com

- IBM AIX Workload Partition documentation

- AIX WPARs - How to guide
  https://www.ibm.com/support/pages/aix-wpars-how

- Versioned WPAR creation guide

- Managing WPAR clients

- Creating an application WPAR

- Configuring YUM on AIX
  https://developer.ibm.com/articles/configure-yum-on-aix/
- AIX Open Source Software community forum Kubernetes on AIX user guide
- Shared Storage Pool setup
- PowerVM NovaLink Knowledge Center documentation
- Capacity on Demand activation code lookup
  http://www-912.ibm.com/pod/pod
- CMC Enterprise Pool 2.0 documentation
- CMC FAQ
- Software requirements for Enterprise Pool 2.0 LPARs
- Linux on Power supported distributions
- IBM VM Recovery Manager HA and DR V1.4 announcement letter
- IBM PowerHA SystemMirror release notes
- IBM PowerHA SystemMirror life cycle
  http://www-01.ibm.com/support/docview.wss?uid=isi3T1023563
- IBM PowerHA compatibility matrix
  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD101347
- IBM PowerHA known fixes
- AIX Toolbox for Linux Applications Overview

### Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services